Kai-Min Chung
Yu Sasaki (Eds.)

LNCS 15490

Advances in Cryptology – ASIACRYPT 2024

30th International Conference on the Theory
and Application of Cryptology and Information Security
Kolkata, India, December 9–13, 2024
Proceedings, Part VII

7 Part VII

INTERNATIONAL ASSOCIATION FOR CRYPTOLOGIC RESEARCH

iacr

Springer

Kai-Min Chung · Yu Sasaki
Editors

Advances in Cryptology – ASIACRYPT 2024

30th International Conference on the Theory
and Application of Cryptology and Information Security
Kolkata, India, December 9–13, 2024
Proceedings, Part VII

Springer

Editors
Kai-Min Chung
Academia Sinica
Taipei, Taiwan

Yu Sasaki
NTT Social Informatics Laboratories
Tokyo, Japan

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-981-96-0940-6 ISBN 978-981-96-0941-3 (eBook)
https://doi.org/10.1007/978-981-96-0941-3

Preface

The 30th Annual International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt 2024) was held in Kolkata, India, on December 9–13, 2024. The conference covered all technical aspects of cryptology and was sponsored by the International Association for Cryptologic Research (IACR).

We received a record 433 paper submissions for Asiacrypt from around the world. The Program Committee (PC) selected 127 papers for publication in the proceedings of the conference. As in the previous year, the Asiacrypt 2024 program had three tracks.

The two program chairs are greatly indebted to the six area chairs for their great contributions throughout the paper selection process. The area chairs were Siyao Guo for Information-Theoretic and Complexity-Theoretic Cryptography, Bo-Yin Yang for Efficient and Secure Implementations, Goichiro Hanaoka for Public-Key Cryptography Algorithms and Protocols, Arpita Patra for Multi-Party Computation and Zero-Knowledge, Prabhanjan Ananth for Public-Key Primitives with Advanced Functionalities, and Tetsu Iwata for Symmetric-Key Cryptography. The area chairs helped suggest candidates to form a strong program committee, foster and moderate discussions together with the PC members assigned as paper discussion leads to form consensus, suggest decisions on submissions in their areas, and nominate outstanding PC members. We are sincerely grateful for the invaluable contributions of the area chairs.

To review and evaluate the submissions, while keeping the load per PC member manageable, we selected the PC members consisting of 105 leading experts from all over the world, in all six topic areas of cryptology, and we also had approximately 468 external reviewers, whose input was critical to the selection of papers. The review process was conducted using double-blind peer review. The conference operated a two-round review system with a rebuttal phase. This year, we continued the interactive rebuttal from Asiacrypt 2023. After the reviews and first-round discussions, PC members and area chairs selected 264 submissions to proceed to the second round. The remaining 169 papers were rejected, including two desk-rejects. Then, the authors were invited to participate in a two-step interactive rebuttal phase, where the authors needed to submit a rebuttal in five days and then interact with the reviewers to address questions and concerns the following week. We believe the interactive form of the rebuttal encouraged discussions between the authors and the reviewers to clarify the concerns and contributions of the submissions and improved the review process. Then, after several weeks of second-round discussions, the committee selected the final 127 papers to appear in these proceedings. This year, we received seven resubmissions from the revise-and-resubmit experiment from Crypto 2024, of which five were accepted. The nine volumes of the conference proceedings contain the revised versions of the 127 papers that were selected. The final revised versions of papers were not reviewed again and the authors are responsible for their contents.

The PC nominated and voted for three papers to receive the Best Paper Awards. The Best Paper Awards went to Mariya Georgieva Belorgey, Sergiu Carpov, Nicolas Gama, Sandra Guasch and Dimitar Jetchev for their paper "Revisiting Key Decomposition Techniques for FHE: Simpler, Faster and More Generic", Xiaoyang Dong, Yingxin Li, Fukang Liu, Siwei Sun and Gaoli Wang for their paper "The First Practical Collision for 31-Step SHA-256", and Valerio Cini and Hoeteck Wee for their paper "Unbounded ABE for Circuits from LWE, Revisited". The authors of those three papers were invited to submit extended versions of their papers to the Journal of Cryptology.

The program of Asiacrypt 2024 also featured the 2024 IACR Distinguished Lecture, delivered by Paul Kocher, as well as an invited talk by Dakshita Khurana. Following Eurocrypt 2024, we selected seven PC members for the Distinguished PC Members Awards, nominated by the area chairs and program chairs. The Distinguished PC Members Awards went to Sherman S. M. Chow, Elizabeth Crites, Matthias J. Kannwischer, Mustafa Khairallah, Ruben Niederhagen, Maciej Obremski and Keita Xagawa.

Following Crypto 2024, Asiacrypt 2024 included an artifact evaluation process for the first time. Authors of accepted papers were invited to submit associated artifacts, such as software or datasets, for archiving alongside their papers; 14 artifacts were submitted. Rei Ueno was the Artifact Chair and led an artifact evaluation committee of 10 members listed below. In the interactive review process between authors and reviewers, the goal was not just to evaluate artifacts but also to improve them. Artifacts that passed successfully through the artifact review process were publicly archived by the IACR at https://artifacts.iacr.org/.

Numerous people contributed to the success of Asiacrypt 2024. We would like to thank all the authors, including those whose submissions were not accepted, for submitting their research results to the conference. We are very grateful to the area chairs, PC members, and external reviewers for contributing their knowledge and expertise, and for the tremendous amount of work that was done with reading papers and contributing to the discussions. We are greatly indebted to Bimal Kumar Roy, the General Chairs, for their efforts in organizing the event, to Kevin McCurley and Kay McKelly for their help with the website and review system, and to Jhih-Wei Shih for the assistance with the use of the review system. We thank the Asiacrypt 2024 advisory committee members Bart Preneel, Huaxiong Wang, Bo-Yin Yang, Goichiro Hanaoka, Jian Guo, Ron Steinfeld, and Michel Abdalla for their valuable suggestions. We are also grateful for the helpful advice and organizational material provided to us by Crypto 2024 PC co-chairs Leonid Reyzin and Douglas Stebila, Eurocrypt 2024 PC co-chairs Marc Joye and Gregor Leander, and TCC 2023 chair Hoeteck Wee. We also thank the team at Springer for handling the publication of these conference proceedings.

December 2024

Kai-Min Chung
Yu Sasaki

Organization

General Chair

Bimal Kumar Roy	TCG CREST Kolkata, India

Program Committee Chairs

Kai-Min Chung	Academia Sinica, Taiwan
Yu Sasaki	NTT Social Informatics Laboratories Tokyo (Japan) and National Institute of Standards and Technology, USA

Area Chairs

Prabhanjan Ananth	University of California, Santa Barbara, USA
Siyao Guo	NYU Shanghai, China
Goichiro Hanaoka	National Institute of Advanced Industrial Science and Technology, Japan
Tetsu Iwata	Nagoya University, Japan
Arpita Patra	Indian Institute of Science Bangalore, India
Bo-Yin Yang	Academia Sinica, Taiwan

Program Committee

Akshima	NYU Shanghai, China
Bar Alon	Ben-Gurion University, Israel
Elena Andreeva	TU Wien, Austria
Nuttapong Attrapadung	AIST, Japan
Subhadeep Banik	University of Lugano, Switzerland
Zhenzhen Bao	Tsinghua University, China
James Bartusek	University of California, Berkeley, USA
Hanno Becker	Amazon Web Services, UK
Sonia Belaïd	CryptoExperts, France
Ward Beullens	IBM Research, Switzerland
Andrej Bogdanov	University of Ottawa, Canada

Pedro Branco	Max Planck Institute for Security and Privacy, Germany
Gaëtan Cassiers	UCLouvain, Belgium
Céline Chevalier	CRED, Université Paris-Panthéon-Assas, and DIENS, France
Avik Chakraborti	Institute for Advancing Intelligence TCG CREST, India
Nishanth Chandran	Microsoft Research India, India
Jie Chen	East China Normal University, China
Yu Long Chen	KU Leuven and National Institute of Standards and Technology, Belgium
Mahdi Cheraghchi	University of Michigan, USA
Nai-Hui Chia	Rice University, USA
Wonseok Choi	Purdue University, USA
Tung Chou	Academia Sinica, Taiwan
Arka Rai Choudhuri	NTT Research, USA
Sherman S. M. Chow	Chinese University of Hong Kong, China
Chitchanok Chuengsatiansup	University of Melbourne, Australia
Michele Ciampi	University of Edinburgh, UK
Valerio Cini	NTT Research, USA
Elizabeth Crites	Web3 Foundation, Switzerland
Nico Döttling	CISPA Helmholtz Center, Germany
Avijit Dutta	Institute for Advancing Intelligence TCG CREST, India
Daniel Escudero	JP Morgan AlgoCRYPT CoE and JP Morgan AI Research, USA
Thomas Espitau	PQShield, France
Jun Furukawa	NEC Corporation, Japan
Rosario Gennaro	CUNY, USA
Junqing Gong	East China Normal University, China
Rishab Goyal	University of Wisconsin-Madison, USA
Julia Hesse	IBM Research Europe, Switzerland
Akinori Hosoyamada	NTT Social Informatics Laboratories, Japan
Michael Hutter	PQShield, Austria
Takanori Isobe	University of Hyogo, Japan
Joseph Jaeger	Georgia Institute of Technology, USA
Matthias J. Kannwischer	Chelpis Quantum Corp, Taiwan
Bhavana Kanukurthi	Indian Institute of Science, India
Shuichi Katsumata	PQShield and AIST, Japan
Jonathan Katz	Google and University of Maryland, USA
Mustafa Khairallah	Lund University, Sweden
Fuyuki Kitagawa	NTT Social Informatics Laboratories, Japan

Katerina Sotiraki	Yale University, USA
Akshayaram Srinivasan	University of Toronto, Canada
Marc Stöttinger	Hochschule RheinMain, Germany
Akira Takahashi	J.P. Morgan AI Research and AlgoCRYPT CoE, USA
Qiang Tang	University of Sydney, Australia
Aishwarya Thiruvengadam	IIT Madras, India
Emmanuel Thomé	Inria Nancy, France
Junichi Tomida	NTT Social Informatics Laboratories, Japan
Monika Trimoska	Eindhoven University of Technology, Netherlands
Huaxiong Wang	Nanyang Technological University, Singapore
Meiqin Wang	Shandong University, China
Qingju Wang	Telecom Paris, Institut Polytechnique de Paris, France
David Wu	UT Austin, USA
Keita Xagawa	Technology Innovation Institute, United Arab Emirates
Chaoping Xing	Shanghai Jiaotong University, China
Shiyuan Xu	University of Hong Kong, China
Anshu Yadav	IST, Austria
Shota Yamada	AIST, Japan
Yu Yu	Shanghai Jiao Tong University, China
Mark Zhandry	NTT Research, USA
Hong-Sheng Zhou	Virginia Commonwealth University, USA

Additional Reviewers

Hugo Aaronson
Damiano Abram
Hamza Abusalah
Abtin Afshar
Siddharth Agarwal
Navid Alamati
Miguel Ambrona
Parisa Amiri Eliasi
Ravi Anand
Saikrishna Badrinarayanan
Chen Bai
David Balbás
Brieuc Balon
Gustavo Banegas
Laasya Bangalore
Jiawei Bao
Jyotirmoy Basak
Nirupam Basak
Gabrielle Beck
Hugo Beguinet
Amit Behera
Mihir Bellare
Tamar Ben David
Aner Moshe Ben Efraim
Fabrice Benhamouda
Tyler Besselman
Tim Beyne
Rishabh Bhadauria
Divyanshu Bhardwaj
Shivam Bhasin

Amit Singh Bhati
Loïc Bidoux
Alexander Bienstock
Jan Bobolz
Alexandra Boldyreva
Maxime Bombar
Nicolas Bon
Carl Bootland
Jonathan Bootle
Giacomo Borin
Cecilia Boschini
Jean-Philippe Bossuat
Mariana Botelho da Gama
Christina Boura
Pierre Briaud
Jeffrey Burdges
Fabio Campos
Yibo Cao
Pedro Capitão
Ignacio Cascudo
David Cash
Wouter Castryck
Anirban Chakrabarthi
Debasmita Chakraborty
Suvradip Chakraborty
Kanad Chakravarti
Ayantika Chatterjee
Rohit Chatterjee
Jorge Chavez-Saab
Binyi Chen
Bohang Chen
Long Chen
Mingjie Chen
Shiyao Chen
Xue Chen
Yu-Chi Chen
Chen-Mou Cheng
Jiaqi Cheng
Ashish Choudhury
Miranda Christ
Qiaohan Chu
Eldon Chung
Hao Chung
Léo Colisson
Daniel Collins
Jolijn Cottaar
Murilo Coutinho
Eric Crockett
Bibhas Chandra Das
Nayana Das
Pratish Datta
Alex Davidson
Hannah Davis
Leo de Castro
Luca De Feo
Thomas Decru
Giovanni Deligios
Ning Ding
Fangqi Dong
Minxin Du
Qiuyan Du
Jesko Dujmovic
Moumita Dutta
Pranjal Dutta
Duyen
Marius Eggert
Solane El Hirch
Andre Esser
Hülya Evkan
Sebastian Faller
Yanchen Fan
Niklas Fassbender
Hanwen Feng
Xiutao Feng
Dario Fiore
Scott Fluhrer
Danilo Francati
Shiuan Fu
Georg Fuchsbauer
Shang Gao
Rachit Garg
Gayathri Garimella
Pierrick Gaudry
François Gérard
Paul Gerhart
Riddhi Ghosal
Shibam Ghosh
Ashrujit Ghoshal
Shane Gibbons
Valerie Gilchrist

Xinxin Gong
Lorenzo Grassi
Scott Griffy
Chaowen Guan
Aurore Guillevic
Sam Gunn
Felix Günther
Kanav Gupta
Shreyas Gupta
Kamil Doruk Gur
Jincheol Ha
Hossein Hadipour
Tovohery Hajatiana Randrianarisoa
Shai Halevi
Shuai Han
Tobias Handirk
Yonglin Hao
Zihan Hao
Keisuke Hara
Keitaro Hashimoto
Aditya Hegde
Andreas Hellenbrand
Paul Hermouet
Minki Hhan
Hilder Lima
Taiga Hiroka
Ryo Hiromasa
Viet Tung Hoang
Charlotte Hoffmann
Clément Hoffmann
Man Hou Hong
Wei-Chih Hong
Alexander Hoover
Fumitaka Hoshino
Patrick Hough
Yao-Ching Hsieh
Chengcong Hu
David Hu
Kai Hu
Zihan Hu
Hai Huang
Mi-Ying Huang
Yu-Hsuan Huang
Zhicong Huang
Shih-Han Hung
Yuval Ishai
Ryoma Ito
Amit Jana
Ashwin Jha
Xiaoyu Ji
Yanxue Jia
Mingming Jiang
Lin Jiao
Haoxiang Jin
Zhengzhong Jin
Chris Jones
Eliran Kachlon
Giannis Kaklamanis
Chethan Kamath
Soumya Kanti Saha
Sabyasachi Karati
Harish Karthikeyan
Andes Y. L. Kei
Jean Kieffer
Jiseung Kim
Seongkwang Kim
Sebastian Kolby
Sreehari Kollath
Dimitris Kolonelos
Venkata Koppula
Abhiram Kothapalli
Stanislav Kruglik
Anup Kumar Kundu
Péter Kutas
Norman Lahr
Qiqi Lai
Yi-Fu Lai
Abel Laval
Guirec Lebrun
Byeonghak Lee
Changmin Lee
Hyung Tae Lee
Joohee Lee
Keewoo Lee
Yeongmin Lee
Yongwoo Lee
Andrea Lesavourey
Baiyu Li
Jiangtao Li
Jianqiang Li

Junru Li
Liran Li
Minzhang Li
Shun Li
Songsong Li
Weihan Li
Wenzhong Li
Yamin Li
Yanan Li
Yu Li
Yun Li
Zeyong Li
Zhe Li
Chuanwei Lin
Fuchun Lin
Yao-Ting Lin
Yunhao Ling
Eik List
Fengrun Liu
Fukang Liu
Hanlin Liu
Hongqing Liu
Rui Liu
Tianren Liu
Xiang Liu
Xiangyu Liu
Zeyu Liu
Paul Lou
George Lu
Zhenghao Lu
Ting-Gian Lua
You Lyu
Jack P. K. Ma
Yiping Ma
Varun Madathil
Lorenzo Magliocco
Avishek Majumder
Nikolaos Makriyannis
Varun Maram
Chloe Martindale
Elisaweta Masserova
Jake Massimo
Loïc Masure
Takahiro Matsuda
Christian Matt
Subhra Mazumdar
Nikolas Melissaris
Michael Meyer
Ankit Kumar Misra
Anuja Modi
Deep Inder Mohan
Charles Momin
Johannes Mono
Hart Montgomery
Ethan Mook
Thorben Moos
Tomoyuki Morimae
Hiraku Morita
Tomoki Moriya
Aditya Morolia
Christian Mouchet
Nicky Mouha
Tamer Mour
Changrui Mu
Arindam Mukherjee
Pratyay Mukherjee
Anne Müller
Alice Murphy
Shyam Murthy
Kohei Nakagawa
Barak Nehoran
Patrick Neumann
Lucien K. L. Ng
Duy Nguyen
Ky Nguyen
Olga Nissenbaum
Anca Nitulescu
Julian Nowakowski
Frederique Oggier
Jean-Baptiste Orfila
Emmanuela Orsini
Tapas Pal
Ying-yu Pan
Roberto Parisella
Aditi Partap
Alain Passelègue
Alice Pellet-Mary
Zachary Pepin
Octavio Perez Kempner
Edoardo Perichetti

Léo Perrin
Naty Peter
Richard Petri
Rafael del Pino
Federico Pintore
Erik Pohle
Simon Pohmann
Guru Vamsi Policharla
Daniel Pollman
Yuriy Polyakov
Alexander Poremba
Eamonn Postlethwaite
Sihang Pu
Luowen Qian
Tian Qiu
Rajeev Raghunath
Srinivasan Raghuraman
Mostafizar Rahman
Mahesh Rajasree
Somindu Chaya Ramanna
Simon Rastikian
Anik Raychaudhuri
Martin Rehberg
Michael Reichle
Krijn Reijnders
Doreen Riepel
Guilherme Rito
Matthieu Rivain
Bhaskar Roberts
Marc Roeschlin
Michael Rosenberg
Paul Rösler
Arnab Roy
Lawrence Roy
Luigi Russo
Keegan Ryan
Markku-Juhani Saarinen
Éric Sageloli
Dhiman Saha
Sayandeep Saha
Yusuke Sakai
Kosei Sakamoto
Subhabrata Samajder
Simona Samardjiska
Maria Corte-Real Santos
Sina Schaeffler
André Schrottenloher
Jacob Schuldt
Mark Schultz
Mahdi Sedaghat
Jae Hong Seo
Yannick Seurin
Aein Shahmirzadi
Girisha Shankar
Yixin Shen
Rentaro Shiba
Ardeshir Shojaeinasab
Jun Jie Sim
Mark Simkin
Jaspal Singh
Benjamin Smith
Yongha Son
Fang Song
Yongsoo Song
Pratik Soni
Pierre-Jean Spaenlehauer
Matthias Johann Steiner
Lukas Stennes
Roy Stracovsky
Takeshi Sugawara
Adam Suhl
Siwei Sun
Elias Suvanto
Koutarou Suzuki
Erkan Tairi
Atsushi Takayasu
Kaoru Takemure
Abdullah Talayhan
Quan Quan Tan
Gang Tang
Khai Hanh Tang
Tianxin Tang
Yi Tang
Stefano Tessaro
Sri AravindaKrishnan Thyagarajan
Yan Bo Ti
Jean-Pierre Tillich
Toi Tomita
Aleksei Udovenko
Arunachalaeswaran V.

Aron van Baarsen
Wessel van Woerden
Michiel Verbauwhede
Corentin Verhamme
Quoc-Huy Vu
Benedikt Wagner
Julian Wälde
Hendrik Waldner
Judy Walker
Alexandre Wallet
Han Wang
Haoyang Wang
Jiabo Wang
Jiafan Wang
Liping Wang
Mingyuan Wang
Peng Wang
Weihao Wang
Yunhao Wang
Zhedong Wang
Yohei Watanabe
Chenkai Weng
Andreas Weninger
Stella Wohnig
Harry W. H. Wong
Ivy K. Y. Woo
Tiger Wu
Yu Xia
Zejun Xiang
Yuting Xiao
Ning Xie
Zhiye Xie
Lei Xu
Yanhong Xu
Haiyang Xue
Aayush Yadav
Saikumar Yadugiri
Kyosuke Yamashita
Jiayun Yan
Yingfei Yan
Qianqian Yang
Rupeng Yang
Xinrui Yang
Yibin Yang
Zhaomin Yang
Yizhou Yao
Kevin Yeo
Eylon Yogev
Yusuke Yoshida
Aaram Yun
Gabriel Zaid
Riccardo Zanotto
Shang Zehua
Hadas Zeilberger
Runzhi Zeng
Bin Zhang
Cong Zhang
Liu Zhang
Tianwei Zhang
Tianyu Zhang
Xiangyang Zhang
Yijian Zhang
Yinuo Zhang
Yuxin Zhang
Chang-an Zhao
Tianyu Zhao
Yu Zhou
Yunxiao Zhou
Zhelei Zhou
Zibo Zhou
Chenzhi Zhu
Ziqi Zhu
Cong Zuo

Artifact Chair

Rei Ueno	Kyoto University, Japan

Artifact Evaluation Committee

Julien Béguinot	LTCI, Télécom Paris, Institut Polytechnique de Paris, France
Aron Gohr	Independent Researcher
Hosein Hadipour	Graz University of Technology, Austria
Akira Ito	NTT Social Informatics Laboratories, Japan
Haruto Kimura	University of Melbourne, Australia and Waseda University, Japan
Kotaro Matsuoka	Kyoto University, Japan
Florian Mendel	Infineon Technologies, Germany
Hiraku Morita	Aarhus University, University of Copenhagen, Denmark
Prasanna Ravi	Nanyang Technological University, Singapore
Élise Tasso	Tohoku University, Japan

Contents – Part VII

Information-Theoretic Cryptography

On the Complexity of Cryptographic Groups and Generic Group Models

Keyu Ji[1,2], Cong Zhang[1,2(✉)], Taiyu Wang[1,2], Bingsheng Zhang[1,2(✉)], Hong-Sheng Zhou[3(✉)], Xin Wang[4], and Kui Ren[1,2]

[1] the State Key Laboratory of Blockchain and Data Security, Zhejiang University, Hangzhou, China
{jikeyu,congresearch,taiyuwang,bingsheng,kuiren}@zju.edu.cn
[2] Hangzhou High-Tech Zone (Binjiang) Blockchain and Data Security Research Institute, Hangzhou, China
[3] Virginia Commonwealth University, Richmond, USA
hszhou@vcu.edu
[4] Digital Technologies, Ant Group, Hangzhou, China
wx352699@antgroup.com

Abstract. Ever since the seminal work of Diffie and Hellman, cryptographic (cyclic) groups have served as a fundamental building block for constructing cryptographic schemes and protocols. The security of these constructions can often be based on the hardness of (cyclic) group-based computational assumptions. Then, the generic group model (GGM) has been studied as an idealized model (Shoup, EuroCrypt 1997), which justifies the hardness of many (cyclic) group-based assumptions and shows the limits of some group-based cryptosystems. We stress that, the importance of the *length* of group encoding, either in a concrete group-based construction or assumption, or in the GGM, has not been studied.

In this work, we initiate a systematic study on the complexity of cryptographic groups and generic group models, varying in different lengths of group encodings, and demonstrate evidences that "the length matters". More concretely, we have the following results:

– We show that there is no black-box/relativizing reduction from the CDH-secure groups (i.e., over such groups, the computational Diffie-Hellman assumption holds) with shorter encodings, to the CDH-secure groups with longer encodings, within the same security parameter. More specifically, given any arbitrary longer CDH-secure group, it is impossible to generically shorten the group encoding and obtain a shorter CDH-secure group within the same group order.
– We show that there is a strict hierarchy of the GGMs with different lengths of encodings. That is, in the framework of indifferentiability, the shorter GGM is *strictly stronger* than the longer ones, even in the presence of computationally *bounded* adversaries.

The work was mainly supported by National Key Research and Development Program of China, Grant No. 2023YFB3106000. Cong Zhang is the co-first author.

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15490, pp. 3–35, 2025.
https://doi.org/10.1007/978-981-96-0941-3_1

1 Introduction

Provable Security and Black-Box Reduction. In the past decades, *provable security* becomes one of the cornerstones of modern cryptography. As the main technique of provable security, reductions are involved to justify the security of a scheme based on a cryptographic primitive. Essentially, given an allegedly successful adversary that breaks the scheme, one can convert it into another successful adversary against the underlying primitive. To a large extent, we study the reductions that are in a black-box manner, in the sense that reductions consider the primitive and/or the adversary against the scheme only via the input-output behavior, without exploring the internal code of the primitive or of the adversary.

In the realm of group-based cryptography (initiated by Diffie and Hellman in their seminal work [DH76]), reductions are established based on the security of cryptographic groups. Serving as the foundation, the community is motivated to study cryptographic groups from various perspectives.

From an Efficiency Perspective. In the literature, with few exceptions, group-based cryptosystems are often built on cryptographic groups in an abstract and black-box manner, which means the underlying groups can be instantiated by any concrete ones as long as the desired security properties are fulfilled. For instance, the well-known public key encryption (PKE) scheme, the ElGamal encryption [ElG85], is chosen-plaintext attack secure (IND-CPA) w.r.t. any concrete prime-order cyclic group in which the decisional Diffie-Hellman (DDH) assumption holds.

In practice, when it comes to instantiating cryptosystems for better efficiency, we typically prefer concrete groups with shorter descriptions. Specifically, the ElGamal encryption utilizes the prime-order subgroup of $\mathbb{Z}_p^*$, for prime p, where the typical bit-length of a group element is 3072 (for 128-bit security) [Bar20]; an alternative approach involves elliptic curves, an increasingly popular choice, and NIST SP 800-186 [CMR+23] provides a list of recommended curves for 128-bit security, such as Curve25519 (with a 255-bit prime modulus). With classic point compression technique, each group element of Curve25519 can be encoded in 256 bits.

This highlights a critical yet subtle issue that has long been overlooked by the community. That is, *the bit-length of a group element is not explicitly taken into account* when the group is utilized in a black-box manner. Note, in real-world applications, groups with shorter descriptions are often preferred to minimize communication and computation overhead. Hereby, we ask the following questions: Does the length of the group description matter when using it in a black-box manner? Is it possible to construct a group with a shorter description generically from groups with longer descriptions? For notation simplicity, throughout this work, we will use *shorter groups* and *longer groups*, to denote "groups with shorter descriptions" and "groups with longer descriptions," respectively.

From a Security Perspective. Unfortunately, despite the advancement of modern cryptography, to the best of our knowledge, there is a fundamental limitation in provable security—the inability of establishing *unconditional hardness* with respect to a *concrete* group. In the past decades, researchers have made significant efforts to explore various ways to demonstrate the hardness of those group-based problems, and one approach is through the class of generic algorithms.

In essence, generic algorithms do not explore the specific encoding of group elements, but instead treat them in a generic manner. Studying this class of algorithms is highly motivated, since several well-known algorithms such as the baby-step/giant-step algorithm [PH78] and Pollard's rho algorithm [Pol78] fall within this classification. To formally describe generic algorithms, ever since the initial work by Nechaev [Nec94], variants of generic group models (GGMs) have been proposed. In Shoup's GGM [Sho97], the group is conceptualized as a random injective encoding from the additive group $\mathbb{Z}_N$ into bit strings uniformly sampled from a set S, where algorithms are allowed to retrieve group encodings and perform group operations, through oracle access. In Maurer's GGM [Mau05], the group is modeled as pointers with respect to a stateful register, where group encodings are the handles (or the indexes) of the register. Within both models, we can establish the unconditional hardness, affirming the justification of the security of cryptographic groups.

When it comes to the study of the lengths of group encodings in the GGMs, Maurer, Portmann, and Zhu [MPZ20] initiate the models varying in the length of the group encoding, and illustrate a *partial hierarchy* of the GGMs, wherein any adversary within the GGM with a longer group encoding (below we denote it as "the longer GGM" for simplicity) can be converted into an adversary within the GGM with a shorter group encoding (below we denote it as "the shorter GGM" for simplicity). Despite the partial hierarchy, the connection and distinction between the longer GGM and the shorter GGM remains unexplored, which hinders a comprehensive interpretation and comparison of the numerous positive and negative results in the GGM.

To deepen our understanding on cryptographic groups, we ask the following question:

Will the longer group/GGM and the shorter group/GGM yield the same complexity?

1.1 Our Results

In this work, we initiate a fine-grained study of cryptographic groups and generic group models with different lengths of group encodings. Specifically, we give evidences that:

- There is a black-box separation between the shorter CDH-secure groups and the longer CDH-secure groups with the same security parameter; in other words, given longer CDH-secure groups, one cannot build a shorter CDH-secure group with the same group order from any standard techniques;
- The shorter GGMs are strictly stronger than the longer GGMs, even in the presence of *computationally bounded* adversaries.

To illustrate our findings, we first formalize the notions of groups/GGMs, parameterized by (N, m)[1], where N and m denote the order of the group and the length of the group encodings[2], respectively. More concretely, we respectively denote the (parameterized) groups and GGMs as $\mathcal{P}^{\mathsf{CDH}}_{N,m}$ and $\mathcal{G}_{N,m}$.

To establish the black-box separation between $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ and $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$ where m_2 is much larger than m_1, we apply the common technique, namely, the relativizing separation. Concretely, we identify an idealized oracle $\mathcal{O}$ and prove that the longer CDH-secure groups exist relative to $\mathcal{O}$, but the shorter one does not exist. In our strategy, we set this oracle to be the GGM with longer group encodings, namely $\mathcal{G}_{N,m_2}$. At the first glance, this seems impossible, because the GGM is designed as the idealized model for cryptographic groups, and the GGM justifies the CDH by having the unconditional lower bound of the hardness. Fortunately, we observe that the analysis becomes subtle when the "length" is involved.

Theorem 1 (Main Theorem, informal). *Consider $m_1 < m_2$. The shorter CDH-secure groups $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ are black-box separated from the longer CDH-secure groups $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$. Concretely,*

- *$\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ does not exist in the generic group model $\mathcal{G}_{N,m_2}$;*
- *$\mathcal{G}_{N,m_2}$ implies $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$.*

Remark 1. Careful readers might wonder what is the relationship between the longer groups and the shorter groups in which *discrete logarithm* is assumed to be hard. We emphasize that, due to technical challenges[3], the relationship between the longer and shorter groups remains unknown—neither positively nor negatively established. We leave it as an open problem.

Next, we turn our attention to understanding the relationship between the GGMs with different lengths of encoding. Based on the trivial observation that "$\mathcal{G}_{N,m_1}$ implies $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$", we immediately note that the shorter GGM, $\mathcal{G}_{N,m_1}$, and the longer GGM, $\mathcal{G}_{N,m_2}$ do not yield the same black-box complexity. However, when attempting to grasp the relationship between two idealized models, solely relying on black-box complexity might not provide us a comprehensive understanding. Essentially, the black-box complexity of a model only demonstrates the limit of standard-model cryptographic systems it implies and considers the computationally unbounded adversary.

To supplement this, Zhang and Zhandry [ZZ23] propose an orthogonal perspective to the black-box complexity, namely the heuristic complexity. It considers computationally bounded adversaries, thereby excluding the impact of all standard-model cryptosystems on the complexity. We investigate "the length matters" of GGMs within this new perspective, showing that:

[1] Typically N is sufficiently large, and $2^m \geq N$.

[2] By the length of group encoding, we mean that the binary length of the longest canonical representation for all group elements. For instance, let $\mathbb{G}$'s be a group such that the order is 3 and the canonical representation of the group elements is $\{00, 111, 0101\}$, then the length of $\mathbb{G}$, denoted as $\mathsf{len}_{\mathbb{G}}$, is 4.

[3] It is still unclear that whether discrete logarithm implies key agreement or not yet.

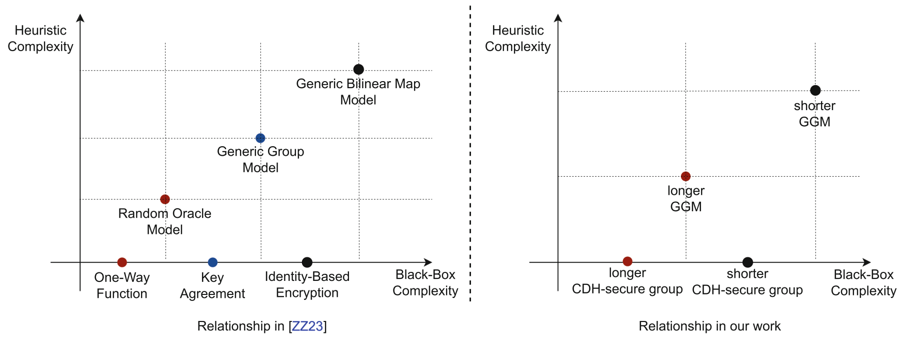


Fig. 1. Relationship between idealized models.

Theorem 2 (Hierarchy of GGMs, Informal). *In the framework of indifferentiability, the GGM with shorter encodings is strictly stronger than the GGM with longer ones, even against computational bounded adversaries.*

To make it clearer, we show our results in Fig. 1. Following the notions in [ZZ23], we give evidence that the shorter GGM statistically implies the longer ones, but the existence of longer GGM's does not computationally imply the existence of a shorter one. More concretely, there exists an indifferentiable construction of a longer generic group with oracle access to shorter generic group without any computational assumption; whereas, as long as the difference in encoding lengths is sufficiently large, there does not exist an indifferentiable construction of a shorter generic group from a long generic group, even with any additional computational assumption.

1.2 Interpretation

Below, we offer interpretations of our findings.

From the Perspective of Black-Box Separation. Our results will bring the research community a better understanding of the cryptographic groups and the generic group models, from the perspective of the black-box reduction/separation[4]. In literature, generic group models have been frequently used to show the impossibility of constructing advanced group-based cryptosystems. Examples include identity-based encryption (IBE) [PRV12, SGS21, Zha22], indistinguishable obfuscation (iO) [MMN16], registration-based encryption (RBE) [HMQS23], accumulators [SGS20], order revealing encryption (ORE) [ZZ18], verifiable delay functions (VDF) [RSS20], and digital signature [DHH+21]. Most of the separation results (e.g., [MMN16, ZZ18, RSS20, SGS20]) are established in Maurer's GGM. Meanwhile, Zhandry [Zha22] illustrates the limits of Maurer's GGM by proving that there are many natural

[4] In this work, when talking about the black-box reduction/separation, we mean that the *fully* black-box reduction/separation that is explicitly defined in [RTV04].

group-based cryptographic schemes (e.g., efficient IND-CPA secure PKE) cannot be modeled by Maurer's GGM, and motivates the line of research, i.e., separations in Shoup's GGM (e.g., IBE in [Zha22] and RBE in [HMQS23]).

Our results demonstrate the *first* evidence that the generic group model can also be used to show the impossibility of constructing plain cryptographic groups, varying in distinct length of group encodings. Speaking of the "lengths" in cryptographic primitives, prior to our work, Garg et.al. [GMM17] prove that there is no iO construction from the single-key functional encryption (FE), if the output length of the functions is much shorter than the length of the ciphertexts[5]. Therefore, we believe that, our result would motivate the community to study the "lengths" in fundamental primitives (e.g., PKE).

However, when delving deeper into our analysis, we stress that our separation results have a limitation. That is, we only establish the separations between the shorter CDH-secure groups and the longer CDH-secure groups under the condition that those groups yield the same security-parameter, which indicates that our separations are somehow security-parameter *dependent*.

For readability, we now explain the limitation through a concrete example. Let λ and λ' be two security parameters. Let p and p' be two primes where $\lfloor \log p \rfloor = \lambda$ and $\lfloor \log p' \rfloor = \lambda'$. Let $\mathbb{G}_1$ be a CDH-secure cryptographic group where the group order is p and the length is 2λ. Let $\mathbb{G}_2$ be another CDH-secure cryptographic group where the group order is p' and the length is $4\lambda'$. Apparently, $\mathbb{G}_1$ is the shorter group and $\mathbb{G}_2$ is the longer one. According to our findings, if $\lambda' \geq \lambda$, then one cannot generically build $\mathbb{G}_1$ from $\mathbb{G}_2$. Unfortunately, if $\lambda' < \lambda$ (say, $\lambda' = \frac{1}{3}\lambda$, indicating that $4\lambda' = \frac{4}{3}\lambda < 2\lambda$), then the relationship between $\mathbb{G}_1$ and $\mathbb{G}_2$ becomes unclear.

In contrast, most known separations are security-parameter *independent*. Take the separation of IBE in Shoup's GGM [Zha22] for instance; according to Zhandry's analysis, we have that for any sufficiently large λ and λ', one cannot generically build an IBE along with security-parameter λ', in Shoup's GGM with security-parameter λ. In order to establish a complete black-box separation (i.e., in the sense of security-parameter independent) between shorter groups and longer groups, novel techniques must be developed to resolve the limitation; we leave this as an important open problem.

Next, we justify that despite of the limitation, our results are interesting and important. First, when it comes to the problem that building a cryptographic group (say, $\mathbb{G}_1$) from another one (say, $\mathbb{G}_2$), it is natural to study the cases that: (1) $\mathbb{G}_1$ and $\mathbb{G}_2$ are with the same group order; (2) the order of $\mathbb{G}_1$ is a factor of $\mathbb{G}_2$[6]. Second, to the best of our knowledge, we are aware of no technique that can be used to generically build $\mathbb{G}_1$ from $\mathbb{G}_2$ if the group orders of $\mathbb{G}_1$ and $\mathbb{G}_2$ are distinct and co-prime. Therefore, we stress that our separations do capture the *natural* settings.

[5] The separation is established under the condition that one-way functions (OWFs) exist and $\mathbf{NP} \not\subseteq \mathbf{coAM}$.

[6] This case indicates that the security parameter of $\mathbb{G}_2$ is bigger than $\mathbb{G}_1$'s, and fortunately our analysis does capture such a case.

Moreover, our results serve as the first attempt to pin down the "lengths" problem for a fundamental primitive (i.e., cryptographic groups), which might open up new research directions (say, the "lengths" problem for other fundamental primitives). Below, for the ease of exposition, when we say the black-box separation between groups, we always mean the one with the same security parameter.

From a Heuristic Perspective. Our results will deepen our understanding of the generic group models from the perspective of heuristic complexity. Inspired by [MRH04,ZZ23], an idealized model can be interpreted through two orthogonal perspectives: the black-box complexity and the heuristic complexity, as depicted in Fig. 1.

For the heuristic aspect, initiated by Maurer et.al. [MRH04] and explicitly studied by Zhang and Zhandry [ZZ23], we consider the framework of indifferentiability against computationally bounded adversaries, where all cryptosystems that exist in the standard model are incorporated. Therefore, the perspective of heuristic is orthogonal to the one of black-box reduction/separation, and understanding the heuristic aspect of various idealized models is important for the relative security of cryptosystems based on idealized models. We establish a strict hierarchy of GGMs from this perspective and prove that the shorter GGM is strictly stronger than the longer one.

In the following, we will give an overview of our approach to comparing the various primitives/models, varying in different lengths of encodings, and our solutions for separating them.

1.3 Technical Overview

Separation Between Cryptographic Groups. Given two cryptographic primitives $\mathcal{P}$ and $\mathcal{Q}$, the typical technique to establish the black-box separation is "relativizing separation" [IR89]. That is, we find a proper oracle $\mathcal{O}$ and prove that the primitive $\mathcal{P}$ exists relative to $\mathcal{O}$ but $\mathcal{Q}$ does not. In our setting, we consider the primitives $\mathcal{P}$ and $\mathcal{Q}$ to be the longer CDH-secure group and shorter one, respectively.

Compared to prior works, the main technical challenge is that, we need to show the gap between two primitives within the same security game (i.e., the CDH game), rather than within different games[7]. The first obstacle is to find a proper oracle. Apparently, the random oracle does not serve our purpose, because the random oracle is weak and there is no construction for CDH-secure groups in the random oracle model.

Our idea is to use a *stronger* oracle, the generic group model. At the first glance, this is impossible, because GGM implies CDH trivially! Fortunately, the GGMs varying in length of group encodings might also yield different levels of complexity, and thus we set this oracle to be the longer GGM within the same security parameter. Concretely, we denote the shorter groups, the longer groups

[7] Games in [IR89] are one-wayness and key recovery attack, respectively.

and the longer GGM as $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}, \mathcal{P}^{\mathsf{CDH}}_{N,m_2}, \mathcal{G}_{N,m_2}$, respectively; recall that $m_2 > m_1$; and we prove that:

– $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$ exists relative to $\mathcal{G}_{N,m_2}$;
– $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ does not exist relative to $\mathcal{G}_{N,m_2}$.

As the former statement is trivial, below we only explain the latter one. To show that $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ does not exist in $\mathcal{G}_{N,m_2}$, it suffices to construct an adversary $\mathcal{A}$ that breaks the CDH game for any construction of shorter group relative to $\mathcal{G}_{N,m_2}$. Due to technical difficulties, we switch to an alternative path. First we pin down a new primitive—non-interactive key exchange (NIKE) with shorter public key, denoted as $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$ [8]. Then we prove that:

1. $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ implies $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$;
2. $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$ does not exist relative to $\mathcal{G}_{N,m_2}$.

As the first statement is straightforward, we will prove the second one. Essentially, $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$, in and of itself, is a key agreement scheme. Next, we give a brief explanation of the separation between NIKE and the random oracle [BKSY11] and then demonstrate how to incorporate the ideas into our analysis. Let $\mathcal{H}$ be a random oracle and $\Pi^{\mathcal{H}} := (\mathsf{KGen}^{\mathcal{H}}, \mathsf{SHK}^{\mathcal{H}})$ be an NIKE scheme with *perfect* correctness. Assuming that the algorithms KGen and SHK make at most q queries, we then construct the adversary $\mathcal{A}$ as follows[9]. Let Alice and Bob be two honest parties. Given the transcript of an execution between Alice and Bob, i.e., pk_A and pk_B, in the present of $\mathcal{H}$, the adversary $\mathcal{A}$ maintains a set $S_{\mathsf{que\text{-}res}}$ of query/response pairs of $\mathcal{H}$, and a multiset S_{key} of candidate keys, both initialized to be $\emptyset$. The adversary $\mathcal{A}$ then runs $4q+1$ iterations of the following attack:

– *Simulation Phase.* The adversary $\mathcal{A}$ searches a proper view of Alice that is consistent with pk_A and $S_{\mathsf{que\text{-}res}}$. Specifically, this view contains the randomness r^*_A used by KGen and SHK, as well as a set of oracle queries/responses $\hat{S}_{\mathcal{A}}$ made by KGen and SHK. The set $\hat{S}_{\mathcal{A}}$ is chosen to be consistent with $S_{\mathsf{que\text{-}res}}$, but it is unnecessary to be consistent with the true oracle $\mathcal{H}$. Let key be the value computed by $\mathsf{SHK}(r^*_A, \mathsf{pk}_B)$. Now, $\mathcal{A}$ adds key into S_{key}.
– *Update Phase.* The adversary $\mathcal{A}$ makes all queries in $\hat{S}_{\mathcal{A}} \setminus S_{\mathsf{que\text{-}res}}$ to the oracle $\mathcal{H}$, and adds the corresponding pairs into $S_{\mathsf{que\text{-}res}}$.

Finally, $\mathcal{A}$ outputs the majority of the keys in S_{key}. Next, we explain why $\mathcal{A}$ recovers the key. Let S_B denote the queries made by Bob in the real execution of the key exchange protocol. In a given iteration, there are two events:

– Event 1 (**Bad event**): $\exists(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_B$ s.t. $\mathsf{que}_A = \mathsf{que}_B$ but $\mathsf{res}_A \neq \mathsf{res}_B$.

[8] Here, m_1 means the length of the public key; please find the formal definition in Sect. 2.1.
[9] The adversary here is computational unbounded but query-efficient.

– Event 2 (**Good event**): $\forall(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_B$, we have that if $\mathsf{que}_A = \mathsf{que}_B$, then $\mathsf{res}_A = \mathsf{res}_B$.

Note that event 1 only occurs in at most $2q$ iterations because $|S_B| \leq 2q$ and once it happens, the update phase would absorb at least one pair $(\mathsf{que}_B, \mathsf{res}_B) \in S_B$ into $S_{\mathsf{que\text{-}res}}$. For event 2, we observe that, when it occurs, there is another oracle $\tilde{\mathcal{H}}$ that is consistent with both $\hat{S}_{\mathcal{A}}$ and S_B. Based on the perfect correctness, we have that the shared key computed in that iteration is valid. Moreover, event 2 occurs in at least 2q+1 iterations, indicating that the majority in S_{key} is valid.

However, when it comes to the GGM, the attack fails. Comparing to ROM, there are two kinds of queries in GGM, namely the labeling query $(x, \mathcal{G}^{\mathsf{label}}_{N,m_2}(x))$ and the addition query $(\mathcal{G}^{\mathsf{label}}_{N,m_2}(x), \mathcal{G}^{\mathsf{label}}_{N,m_2}(y), \mathcal{G}^{\mathsf{label}}_{N,m_2}(x+y))$. Therefore, we should define S_B that covers all the group encodings that appear in the queries (both labeling and addition) with the discrete logarithms (Bob might not know the value). Then, in a given iteration, there are three events:

– Event 1 (**Bad event**): $\exists(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_B$ s.t. $\mathsf{que}_A = \mathsf{que}_B$ but $\mathsf{res}_A \neq \mathsf{res}_B$.
– Event 2 (**Bad event**): $\exists(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_B$ s.t. $\mathsf{que}_A \neq \mathsf{que}_B$ but $\mathsf{res}_A = \mathsf{res}_B$.
– Event 3 (**Good event**): $\forall(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}, (\mathsf{que}_B, \mathsf{res}_B) \in S_B$, we have that if $\mathsf{que}_A = \mathsf{que}_B$, then $\mathsf{res}_A = \mathsf{res}_B$.

Note that event 1 and event 3 can be handled similarly as above. However, the fatal problem is that event 2 might always happen. In other words, we cannot find a GGM that is consistent with both $\hat{S}_{\mathcal{A}}$ and S_B, indicating that the above attack fails immediately.

More specifically, we note that the reason why event 2 happens is that, given pk_A and pk_B, algorithms can obtain valid group encoding without making labeling query[10]. Moreover, if algorithms *cannot* obtain valid group encodings without making labeling queries, then the GGM can be simulated by a stateful oracle that only provides labeling queries, as the addition queries can be easily converted into labeling queries. Such an oracle is close to the random oracle model and thus our goal is to design a mechanism that prevent extracting valid group elements without knowing the corresponding discrete logarithms.

Here we introduce our *length tool*, intuitively, if the length of the public key is much shorter than the group encoding (say, the length gap is at least $\omega(\log \lambda)$), then the public key would not carry enough information to recover the group encodings. This also explains why we choose NIKE other than general key agreement (say, multi-round KA), because the adversary only obtains two public keys in the setting of NIKE.

[10] This in fact is natural in group-based cryptosystem, take the ElGamal encryption scheme [ElG85] for instance, the public key itself is a valid group element.

Concretely, let Q_{sk_A} and Q_{sk_B} be the set of the query/response pairs (only labeling queries[11]) made when running $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}_A)$ and $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}_B)$, respectively. Let h be the valid group encoding that an algorithm outputs, by having pk_A and pk_B, we then consider the following four cases:

– Case 1: (**Independent**) $h \notin Q_{\mathsf{sk}_A} \cup Q_{\mathsf{sk}_B}$.
– Case 2: (**Frequent**) $h \in Q_{\mathsf{sk}_A} \cap Q_{\mathsf{sk}_B}$.
– Case 3: (**Dependent but hard to extract**) $h \in Q_{\mathsf{sk}_A} \setminus Q_{\mathsf{sk}_B}$.
– Case 4: (**Dependent but hard to extract**) $h \in Q_{\mathsf{sk}_B} \setminus Q_{\mathsf{sk}_A}$.

For case 1, h is independent of pk_A and pk_B. Due to the sparseness of the group encodings in $\mathcal{G}_{N,m_2}$, no algorithm can output h except for negligible probability.

For case 2, it is apparent that pk_A and pk_B together might carry enough information for h. Fortunately, with high probability h is a frequent query, therefore the discrete logarithm of h can be easily obtained by repeatedly running $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\cdot)$ on sufficiently many random inputs.

For case 3 (or case 4), note that h is independent of pk_B, which means that only pk_A carries the information of h. Note that the length of pk_A is m_1 but length of h is m_2. Moreover, h is uniformly distributed over the probability of GGM. Therefore, conditioned on that $m_2 - m_1$ is sufficiently large, no algorithm can extract such an h except for negligible probability.

The above sketch is not precise; please find low-level details, in Sect. 3.

Hierarchy of GGMs. To establish the hierarchy of the generic group models against computational bounded adversaries, we formalize our goal in the framework of indifferentiability. Specifically, we prove that the shorter GGM (denoted as $\mathcal{G}_{N,m_1}$) statistically implies the longer one (denoted as $\mathcal{G}_{N,m_2}$), but the longer GGM does not computationally imply the shorter one.

$\mathcal{G}_{N,m_1}$ statistically implies $\mathcal{G}_{N,m_2}$. We first explain how $\mathcal{G}_{N,m_1}$ implies $\mathcal{G}_{N,m_2}$ against computationally unbounded adversaries. Let $\mathcal{H}$ be a random oracle that maps $\{0,1\}^* \rightarrow \{0,1\}^{m_2-m_1}$; as the first attempt, it is natural to design the labeling function as:

$$L^{\mathcal{G}_{N,m_1},\mathcal{H}}(x) := \mathcal{G}^{\mathsf{label}}_{N,m_1}(x)||\mathcal{H}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x)).$$

However, there always exists an efficient distinguisher that breaks the indifferentiability w.r.t. the above scheme. Specifically, in the ideal world, the distinguisher uniformly samples $x \in \mathbb{Z}_N$, makes a labeling query with x, and obtains $\mathcal{G}^{\mathsf{label}}_{N,m_2}(x)$. Let str and str' be the first m_1 bits and the last $m_2 - m_1$ bits of $\mathcal{G}^{\mathsf{label}}_{N,m_2}(x)$, respectively. Then the distinguisher makes a query to the simulator with input str and checks whether the response matches str'. Note that, without knowing x, the

[11] We stress that, for the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}$, without loss of generality, it only makes labeling queries. Essentially, the group encodings of $\mathcal{G}_{N,m_2}$ are sparse, which means any algorithm with inputs that are independent of $\mathcal{G}_{N,m_2}$ cannot obtain a valid group encoding without making labeling query, indicating that any addition query can be absorbed by the corresponding labeling query.

simulator cannot answer this query properly except for a negligible probability. To prevent the attack above, we enhance the power of the simulator. We involve an additional oracle, the random permutation oracle $\mathcal{E}$, that permutes $\{0,1\}^{m_2}$ with its inverse[12] $\mathcal{E}^{-1}$, and design the labeling function as:

$$L^{\mathcal{G}_{N,m_1},\mathcal{H},\mathcal{E}}(x) := \mathcal{E}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x)||\mathcal{H}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x))).$$

Careful readers may wonder why it works. Note that both $\mathcal{E}$ and $\mathcal{E}^{-1}$ are under full control of the simulator, which means that the distinguisher is independent of the value $\mathcal{H}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x))$ without making queries to the simulator. This extra information gained from these queries is exactly what the simulator requires for the proof to go through. In fact, with the aid of $\mathcal{E}$, we can even simplify the construction by replacing $\mathcal{H}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x))$ with a fixed string, say $0\cdots 0$, concretely:

$$L^{\mathcal{G}_{N,m_1},\mathcal{E}}(x) := \mathcal{E}(\mathcal{G}^{\mathsf{label}}_{N,m_1}(x)||\underbrace{0\cdots 0}_{m_2-m_1}).$$

The addition algorithm can be easily constructed by applying the inverse oracle $\mathcal{E}^{-1}$. While the additional oracle $\mathcal{E}$ and its inverse $\mathcal{E}^{-1}$ have protected against certain natural attacks, we need to argue indifferentiability against all possible attacks. To do so, we use a careful simulation strategy for $\mathcal{G}^{\mathsf{label}}_{N,m_1}, \mathcal{G}^{\mathsf{add}}_{N,m_1}, \mathcal{E}$, and $\mathcal{E}^{-1}$, and prove indifferentiability through a careful sequence of hybrids. Due to the space limit, we omit the formal descriptions of our simulation, and we refer interesting readers to see it in the full version of this paper [ZJW+24].

Remark 2. Careful readers might note that the building blocks of construction above contain both the shorter GGM $\mathcal{G}_{N,m_1}$ and an *additional* independent randome oracle, rather than the shorter GGM solely. Although we have that GGM implies ROM statistically [ZZ23], it is unclear to us that how to build an indifferentiable GGM plus an independent ROM from a single GGM. Therefore, we stress that our hierarchy of the GGM is established with the aid of an additional independent random oracle.

Moreover, this even motivates an interesting research question that whether one single GGM implies multiple independent GGMs, comparing to the fact that the random oracle does.

$\mathcal{G}_{N,m_2}$ *does not computationally imply* $\mathcal{G}_{N,m_1}$. Suppose we have a purported construction $\Pi^{\mathcal{G}_{N,m_2}} := (L^{\mathcal{G}_{N,m_2}}, A^{\mathcal{G}_{N,m_2}})$ of a shorter group from a longer GGM. How could we prove that $\Pi^{\mathcal{G}_{N,m_2}}$ can be differentiated from $\mathcal{G}_{N,m_1}$ by a computationally bounded distinguisher?

Following the strategy in [ZZ23], we should find some security property P that holds for $\mathcal{G}_{N,m_1}$ but fails for *any* $\Pi^{\mathcal{G}_{N,m_2}}$. As explained in [ZZ23], any standard model assumption cannot serve as the property, and thus, this property P is set to be a variant of discrete logarithm problem, called *discrete log identification*

[12] According to [HKT11], the random oracle and random permutation oracle with inverse are equivalent, therefore we take $\mathcal{E}$ and $\mathcal{E}^{-1}$ for granted.

(DLI). Intuitively, DLI is defined as: given $h := L(x)$, construct a (probabilistic, efficient, and query-free) circuit C such that $C(x)$ accepts with a high probability, but $C(x')$ rejects with a overwhelming probability on all $x' \neq x$. Apparently, the DLI problem is easy on any standard-model group: for any y, set $C(y)$ to be 1 if and only if $L(y) = h$, where $L(y) := g^y$ is computed as part of the circuit[13]. To establish the separation between GGM and ROM, Zhandry and Zhang prove that the DLI problem is also easy on any group built within the random oracle model. Intuitively, they "compile out" the random oracle $\mathcal{H}$ and design an attacker that can easily construct an oracle-aided circuit $C^{\mathcal{H}}(\cdot)$, breaking the DLI problem by computing $L^{\mathcal{H}}(\cdot)$. The subtlety is to anticipate the oracle queries that C will make to the random oracle model and have the attacker make the corresponding queries for itself. Concretely, given input $L^{\mathcal{H}}(x)$, the attacker runs the addition algorithm $A^{\mathcal{H}}(L^{\mathcal{H}}(y), L^{\mathcal{H}}(x-y))$ and $L^{\mathcal{H}}(\cdot)$ on several random inputs, records all queries/responses that were made, and hardcodes the queries/responses into the C to obtain an oracle-free circuit, which C outputs.

Below, we outline our method for integrating the aforementioned technique into the analysis within the longer GGM. The difficulty is that, our goal seems to conflict with the results in [ZZ23], as they have proven that the DLI problem is hard with respect to the generic group model. To bypass the obstacle, we here leverage the *length tool* again.

Consider computing $L^{\mathcal{G}_{N,m_2}}(x)$ from x, which in turn makes queries to the longer GGM, $\mathcal{G}_{N,m_2}$. Let Q_x be the set of query/response pairs made during the procedure of computing $L^{\mathcal{G}_{N,m_2}}(x)$. Similarly as above, we assume that, without loss of generality, each query/response pair $(\mathsf{que}, \mathsf{res}) \in Q_x$ is a labeling query. Consider computing $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$ where y and z are random, conditioned on $y + z = x$. The output of this addition is $L^{\mathcal{G}_{N,m_2}}(y+z) = L^{\mathcal{G}_{N,m_2}}(x)$. For each query/response pair $(\mathsf{que}, \mathsf{res}) \in Q_x$, there are roughly four possible cases:

- Case 1: The label $L^{\mathcal{G}_{N,m_2}}(x)$ does *not* depend on the response res at all;
- Case 2: The label $L^{\mathcal{G}_{N,m_2}}(x)$ depends on the response res, but with a overwhelming probability over the choice of y and z, res does not appear when computing $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$;
- Case 3: The label $L^{\mathcal{G}_{N,m_2}}(x)$ depends on the response res, and with a non-negligible probability over the choice of y and z, $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$ makes a "labeling" query to $\mathcal{G}_{N,m_2}$ on input que;
- Case 4: The label $L^{\mathcal{G}_{N,m_2}}(x)$ depends on the response res, and with a non-negligible probability over the choice of y and z, an "addition" query $(\mathsf{que}_1, \mathsf{que}_2, \mathsf{res})$ occurs when computing $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$.

Now we explain our approach of building the oracle-free circuit C. We collect queries into a list, denoted as $S_{\mathsf{que\text{-}res}}$, and hardcode $S_{\mathsf{que\text{-}res}}$ into the circuit C to

[13] Note that for standard-model groups, $L(y)$ denotes the value g^y for the fixed generator g, and here y is the discrete logarithm of h with respect to g.

make sure that $C(x)$ will be able to reconstruct $L^{\mathcal{G}_{N,m_2}}(x)$ without making any query to the oracle at all:

- In case 1 (**Non-sensitive query**), same as in [ZZ23], since $L^{\mathcal{G}_{N,m_2}}(x)$ does not depend on res, when computing $L^{\mathcal{G}_{N,m_2}}(x)$ we can just replace the response with a random string without affecting the ultimate labeling. Therefore, for any query not in $S_{\mathsf{que\text{-}res}}$, we will have C respond with a uniformly random string.
- In case 2 (**Sensitive but frequent query**), same as in [ZZ23], since $L^{\mathcal{G}_{N,m_2}}(x)$ does depend on res, this query is a sensitive query for the ultimate labeling. In this case, it must be that $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$ be able to extract res from the inputs, i.e., $L^{\mathcal{G}_{N,m_2}}(y)$ and $L^{\mathcal{G}_{N,m_2}}(z)$, which indicates that, with a high probability, $(\mathsf{que}, \mathsf{res}) \in Q_x \cap (Q_y \cup Q_z)$. On the other hand, x, y and z are pairwise independent, which means that $Q_x \cap Q_y$ and $Q_x \cap Q_z$ *only* contains "frequent" queries. Therefore, this query/response pair can be collected by running $L^{\mathcal{G}_{N,m_2}}(\cdot)$ on sufficiently many random inputs.
- In case 3 (**Sensitive labeling query**), same as in [ZZ23], $S_{\mathsf{que\text{-}res}}$ collects all the labeling queries that occur when running $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$. We know that, with a non-negligible probability $(\mathsf{que}, \mathsf{res})$ will be amongst the queries in $S_{\mathsf{que\text{-}res}}$. By repeating several times, we have that $(\mathsf{que}, \mathsf{res}) \in S_{\mathsf{que\text{-}res}}$ with a high probability.
- In Case 4 (**Sensitive addition query**), different from [ZZ23], the addition query, i.e., $(\mathsf{que}_1, \mathsf{que}_2, \mathsf{res})$ occurs, where que_1 and que_2 are two valid group encodings of $\mathcal{G}_{N,m_2}$. Although res appears in this query, collecting this kind of query is not usually useful for our purpose. Specifically, when running $L^{\mathcal{G}_{N,m_2}}(x)$, the algorithm might make labeling queries on points $(x_1, \ldots, x_q)$, whereas $S_{\mathsf{que\text{-}res}}$ might only store query/response pairs in the form of addition, i.e., $(\mathcal{G}_{N,m_2}(y_i), \mathcal{G}_{N,m_2}(z_i), \mathcal{G}_{N,m_2}(y_i + z_i))$, without explicitly knowing either y_i or z_i. As a result, $C(x)$ might fail to reconstruct $L^{\mathcal{G}_{N,m_2}}(x)$: when running $C(x) := L^{S_{\mathsf{que\text{-}res}}}(x)$, although C knows that $\mathcal{G}_{N,m_2}(x_i)$ exists in the database $S_{\mathsf{que\text{-}res}}$, it does *not* know which tuple corresponds to the correct one.

  To resolve the problem, we need to transform this addition query into a labeling query. Observe that if the discrete logarithms of que_1 and que_2 are known, then the transformation is trivial. Exploring deeper, during the procedure of computing $A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(y), L^{\mathcal{G}_{N,m_2}}(z))$, the algorithm $A^{\mathcal{G}_{N,m_2}}$ can only extract valid group encodings in $Q_y \cup Q_z$[14]. Moreover, we have that $L^{\mathcal{G}_{N,m_2}}(y)$ and $L^{\mathcal{G}_{N,m_2}}(z)$ are independent of the responses that are in $Q_z \setminus Q_y$ and $Q_y \setminus Q_z$, respectively.

  Now, we *leverage the length tool.* Concretely, from $A^{\mathcal{G}_{N,m_2}}$'s perspective, $L^{\mathcal{G}_{N,m_2}}(y)$ is the only string that carries information of the valid group encodings $\in Q_y \setminus Q_z$. If $m_2 - m_1$ is sufficiently large, say $m_2 - m_1 \geq \omega(\log \lambda)$, where λ is the security parameter, then it is impossible for $A^{\mathcal{G}_{N,m_2}}$ to extract a valid group encoding from $Q_y \setminus Q_z$ except for a negligible probability, indicating that the valid group encodings that $A^{\mathcal{G}_{N,m_2}}$ can extract are in $Q_y \cap Q_z$. Having

[14] Other group encodings in $\mathcal{G}_{N,m_2}$ are independent of $L^{\mathcal{G}_{N,m_2}}(y)$ and $L^{\mathcal{G}_{N,m_2}}(z)$.

that x, y and z are pairwise independent, we know that queries in $Q_y \cap Q_z$ are frequent with a high probability, which can be easily captured as in case 2.

Next, we consider the value of $C(x')$ for $x' \neq x$. If we are lucky and $S_{\mathsf{que\text{-}res}}$ contains all sensitive queries of $Q_{x'}$, then $C(x') = L^{\mathcal{G}_{N,m_2}}(x') \neq L^{\mathcal{G}_{N,m_2}}(x)$, indicating that $C(x')$ rejects as desired. Otherwise, if $S_{\mathsf{que\text{-}res}}$ does not contain all the sensitive queries of $Q_{x'}$, then $S_{\mathsf{que\text{-}res}}$ would respond to the query with random value, which means that $C(x')$ computes an invalid label for x'. As explained in [ZZ23], the random response would only serve to inject further randomness into the label, and the invalid label would be unequal to $L^{\mathcal{G}_{N,m_2}}(x)$ with a high probability. Combining the above together, we build an oracle-free circuit that *only* accepts the discrete logarithm x.

The above sketch is not precise; please find the low-level details in Sect. 4.

The Hierarchy is Tight. To complement our results of the hierarchy, we next show that if $m_2 - m_1$ is *small*, then $\mathcal{G}_{N,m_1}$ and $\mathcal{G}_{N,m_2}$ are equivalent under the indifferentiability framework. To explain our idea, we illustrate the simplest case, where $m_2 - m_1 = 1$[15]. Let Trunc be the function that chops off the last bit of the input, we build an indifferentiable group in $\mathcal{G}_{N,m_2}$ as follows:

$$L^{\mathcal{G}_{N,m_2}}(x) := \mathsf{Trunc}(\mathcal{G}^{\mathsf{label}}_{N,m_2}(x));$$

$$A^{\mathcal{G}_{N,m_2}}(\mathsf{str}_0, \mathsf{str}_1) := \begin{cases} \mathsf{Trunc}(\mathcal{G}^{\mathsf{add}}_{N,m_2}(\mathsf{str}_0||b_0, \mathsf{str}_1||b_1)), & \text{if } \mathsf{str}_0||b_0 \text{ and } \mathsf{str}_1||b_1 \text{ are valid;} \\ \perp & \text{otherwise.} \end{cases}$$

For clarity, if there exist $b_0, b_1 \in \{0, 1\}$ such that both $\mathsf{str}_0||b_0$ and $\mathsf{str}_1||b_1$ are valid, then the addition algorithm outputs $\mathsf{Trunc}(\mathcal{G}^{\mathsf{add}}_{N,m_2}(\mathsf{str}_0||b_0, \mathsf{str}_1||b_1))$, otherwise it aborts. Based on the fact that the group encodings of $\mathcal{G}_{N,m_2}$ are sparse, we know that for any string str, the probability that both $\mathsf{str}||0$ and $\mathsf{str}||1$ are valid is negligible, which indicates that the addition algorithm is well defined. Moreover, we prove that the construction above is indifferentiable from $\mathcal{G}_{N,m_1}$. Due to the space limit, we leave the proof in the full version of this paper [ZJW+24].

Due to the composition of indifferentiability, our results can be easily extended to the case that $m_2 - m_1 \leq \Theta(\log \lambda)$, which completes the entire picture of the hierarchy asymptotically.

1.4 Organization

In Sect. 2, we present the necessary notations, concepts, and definitions. We establish a separation between two CDH-secure groups with sufficiently large

[15] We also require that group encodings in $\mathcal{G}_{N,m_2}$ are sparse, say $m_2 - \log N \geq \omega(\log \lambda)$.

encoding length difference in Sect. 3. We then establish a hierarchy among GGMs with different encoding lengths in Sect. 4. All formal proofs can be found in the full version of this paper [ZJW+24] due to the space limitation.

2 Preliminaries

Notation. For a finite set S, we denote a random sample s from S according to the uniform distribution as $s \xleftarrow{\$} S$. We say a positive function $\mathsf{negl}(\cdot)$ is negligible, if for all positive polynomial $p(\cdot)$, there exists a constant $\lambda_0 > 0$ such that for all $\lambda > \lambda_0$, it holds that $\mathsf{negl}(\lambda) < 1/p(\lambda)$. We say a function $\rho(\cdot)$ is noticeable in λ, if the inverse $1/\rho(\lambda)$ is polynomial in λ. We write $y \xleftarrow{\$} \mathsf{Alg}(I)$ to denote variable y that is obtained by running a randomized algorithm Alg on input I (which may consist of a tuple $I := (I_1, ..., I_n)$). If Alg is deterministic, we write "$\leftarrow$" instead of "$\xleftarrow{\$}$". By $x||y$, we mean the concatenation of strings x and y.

Algorithms. Denote $\lambda \in \mathbb{N}$ as the security parameter. Here we use a non-uniform circuit to formalize the model of computation. An algorithm Alg is a collection of circuits $\{C_\lambda\}_{\lambda\in\mathbb{N}}$ with domain Dom_λ and range Ran_λ, respectively. When considering interactive algorithms $(\mathsf{Alg}_1, \ldots, \mathsf{Alg}_n)$, algorithms are treated as a sequence of circuits $C_\lambda^{(1)}, C_\lambda^{(2)}, \ldots$, where the domain of $C_\lambda^{(i)}$ is denoted as $\mathsf{Dom}_\lambda^{(i)} = \mathsf{stat}_\lambda^{(i)} \times \mathsf{input}_\lambda^{(i-1)}$, the range of $C_\lambda^{(i)}$ is denoted as $\mathsf{Ran}_\lambda^{(i)} = \mathsf{stat}_\lambda^{(i+1)} \times \mathsf{output}_\lambda^{(i)}$. Here, $\mathsf{stat}_\lambda^{(i)}$ ($\mathsf{input}_\lambda^{(i)}$, $\mathsf{output}_\lambda^{(i)}$) is the space of the state (inputs, outputs) that $C_\lambda^{(i)}$ sends to $C_\lambda^{(i+1)}$, respectively.

Games. A game is initiated by a probabilistic interactive algorithm C, called a challenger, and a predicate function $\mathsf{pf} : \{0,1\}^* \to [0,1]$. The challenger takes the security parameter as input and interacts with k communicating-restricted parties $(\mathsf{Alg}_1, \ldots, \mathsf{Alg}_k)$. We call $\mathcal{A} := (\mathsf{Alg}_1, \ldots, \mathsf{Alg}_k)$ the adversary. In the end of the game, the challenger C outputs a bit b; if $b = 1$ we say the adversary wins the game, otherwise we say the adversary loses. Let $\mathrm{Cl}(\mathcal{A})$ be a class of adversary. We say a game $(\mathsf{C}, \mathsf{pf})$ is hard with respect to $\mathrm{Cl}(\mathcal{A})$, if for any adversary $\mathcal{A} \in \mathrm{Cl}(\mathcal{A})$, we have $\Pr[\mathcal{A} \text{ wins}] \leq \mathsf{pf} + \mathsf{negl}(\lambda)$.

Cryptosystems. A cryptosystem Σ consists of a set of algorithms, which typically are non-interactive. Here, Σ is accessible via two interfaces Σ.hon and Σ.adv, where Σ.hon provides an honest interface through which the system can be accessed by all parties in a black-box manner, and Σ.adv models the adversarial access to the inner working part of Σ.

2.1 Primitives, Idealized Models, and Reduction Notions

In this work, we treat CDH-secure groups as cryptographic primitives, and explore black-box reduction between them with different lengths. First of all, we recall the definition of primitive formalized by [RTV04].

2.1.1 Cryptographic Primitives

Definition 1 (Cryptographic Primitive [RTV04]). *A primitive $\mathcal{P}$ is a pair $\langle \mathcal{F}_\mathcal{P}, \mathcal{R}_\mathcal{P} \rangle$, where $\mathcal{F}_\mathcal{P}$ is a set of functions $f : \{0,1\}^* \mapsto \{0,1\}^*$, and $\mathcal{R}_\mathcal{P}$ is a relation over pairs $\langle f, \mathcal{A} \rangle$ of a function $f \in \mathcal{F}_\mathcal{P}$ and an adversarial machine $\mathcal{A}$. (The set $\mathcal{F}_\mathcal{P}$ is required to contain at least one function which is computable by a* PPT *machine.)*

- Efficient implementation. *We say a function f* **implements** *$\mathcal{P}$ or is an implementation of $\mathcal{P}$ if $f \in \mathcal{F}_\mathcal{P}$. An* **efficient implementation** *of $\mathcal{P}$ is an implementation of $\mathcal{P}$ which is polynomial-time computable.*
- Secure implementation. *We say an adversarial machine $\mathcal{A}$* **$\mathcal{P}$-breaks** *$f \in \mathcal{F}_\mathcal{P}$ if $\langle f, \mathcal{A} \rangle \in \mathcal{R}_\mathcal{P}$. A* **secure implementation** *of $\mathcal{P}$ is an implementation of $\mathcal{P}$ such that no* PPT *adversarial machine $\mathcal{P}$-breaks f.*

We say the **primitive $\mathcal{P}$ exists** *if there is an efficient and secure implementation of $\mathcal{P}$.*

As mentioned before, we treat CDH-secure groups as a cryptographic primitive. Now we formalize this primitive by using the terms in [RTV04].

Definition 2 (CDH-Secure Groups). *A CDH-secure group $\mathcal{P}^{\mathsf{CDH}}$ consists of the following pair $\langle \mathcal{F}_{\mathcal{P}^{\mathsf{CDH}}}, \mathcal{R}_{\mathcal{P}^{\mathsf{CDH}}} \rangle$:*

1. *The set $\mathcal{F}_{\mathcal{P}^{\mathsf{CDH}}}$ for specifying syntax and capturing the correctness property.*
   Here, the set $\mathcal{F}_{\mathcal{P}^{\mathsf{CDH}}}$ consists of functions f, where f represents the group generation function *for generating group description of finite cycle groups. Concretely, we write $(\mathbb{G}, g, N, m) \overset{\$}{\leftarrow} f(1^\lambda)$, where $\mathbb{G}$ is a cyclic group of prime order N, g is a generator $\mathbb{G}$, and m is the length of group encoding (that is, each group element in $\mathbb{G}$ can be represented as an m-bit string).*
   We note that the correctness is guaranteed by the basic properties of the cyclic group.
2. *The relation $\mathcal{R}_{\mathcal{P}^{\mathsf{CDH}}}$ for capturing the security property.*
   For function $f \in \mathcal{F}_{\mathcal{P}^{\mathsf{CDH}}}$ and PPT *(adversarial) machine $\mathcal{A}$, we define $\langle f, \mathcal{A} \rangle \in \mathcal{R}_{\mathcal{P}^{\mathsf{CDH}}}$ if there exists a polynomial $p(\cdot)$ such that $\Pr[\mathcal{A}(\mathbb{G}, g, N, m, h_1, h_2) = g^{x_1 x_2}] > 1/p(\lambda)$ for infinitely many λ.*
   Here, $(\mathbb{G}, g, N, m) \overset{\$}{\leftarrow} f(1^\lambda)$, and $h_1, h_2 \in \mathbb{G}$ are two uniformly chosen group elements where $h_1 = g^{x_1}$, $h_2 = g^{x_2}$, and $x_1, x_2 \in \mathbb{Z}_N$.

We say CDH-secure group $\mathcal{P}^{\mathsf{CDH}}$ exists, if there exists a function $f \in \mathcal{F}_{\mathcal{P}^{\mathsf{CDH}}}$, it holds that no PPT *adversarial machine $\mathcal{A}$ such that $\langle f, \mathcal{A} \rangle \in \mathcal{R}_{\mathcal{P}^{\mathsf{CDH}}}$. Often, we make the parameters, the order N and the encoding length m, explicit, and denote the CDH-secure group as $\mathcal{P}^{\mathsf{CDH}}_{N,m}$.*

Non-interactive key exchange (NIKE) was initially studied by Diffie and Hellman in their breakthrough paper [DH76]. We now describe this primitive by using the terms in [RTV04].

Definition 3 (Non-Interactive Key Exchange). *A non-interactive key exchange protocol* $\mathcal{P}^{\mathsf{NIKE}}$ *consists of the following pair* $\langle \mathcal{F}_{\mathcal{P}^{\mathsf{NIKE}}}, \mathcal{R}_{\mathcal{P}^{\mathsf{NIKE}}} \rangle$*:*

1. *The set* $\mathcal{F}_{\mathcal{P}^{\mathsf{NIKE}}}$ *for specifying syntax and capturing the correctness property. Here, the set* $\mathcal{F}_{\mathcal{P}^{\mathsf{NIKE}}}$ *consists of functions* f*, where* $f := (\mathsf{KGen}, \mathsf{SHK})$ *represents*
   - *the* public-key message function $\mathsf{KGen} : \mathcal{SK} \mapsto \mathcal{PK}$ *for generating the public-key message based on a randomly chosen private-key, where* $\mathcal{PK}$ *and* $\mathcal{SK}$ *are public-key space and private-key space, respectively.*
   - *the* shared key generation function $\mathsf{SHK} : \mathcal{PK} \times \mathcal{SK} \mapsto \mathcal{K} \cup \{\perp\}$ *for generating the shared key, where* $\mathcal{K}$ *is shared-key space, and* $\perp$ *denotes that the computation fails.*

   Concretely, for randomly chosen $\mathsf{sk} \xleftarrow{\$} \mathcal{SK}$*, we write* $\mathsf{pk} \leftarrow \mathsf{KGen}(\mathsf{sk})$*, where* pk *is called public key. Furthermore, for randomly chosen* $\mathsf{sk}' \xleftarrow{\$} \mathcal{SK}$*, compute* $\mathsf{pk}' \leftarrow \mathsf{KGen}(\mathsf{sk}')$*. We write* $\mathsf{shk} \leftarrow \mathsf{SHK}(\mathsf{pk}', \mathsf{sk})$ *and* $\mathsf{shk}' \leftarrow \mathsf{SHK}(\mathsf{pk}, \mathsf{sk}')$*. Note that, when the shared key generation function fails, we write* $\mathsf{shk} = \perp$ *or* $\mathsf{shk}' = \perp$*.*

   We say correctness *is achieved if there exists an* $\mathsf{negl}(\cdot)$ *such that*

   $$\Pr\left[\mathsf{shk} \neq \perp \wedge \mathsf{shk}' \neq \perp \wedge \mathsf{shk} \neq \mathsf{shk}'\right] \leq \mathsf{negl}(\lambda)$$

   When $\mathsf{negl}(\lambda) = 0$*, then we say* perfect correctness *is achieved.*
2. *The relation* $\mathcal{R}_{\mathcal{P}^{\mathsf{NIKE}}}$ *for capturing the security property against key-recovery attack (KRA).*
   For function $f := (\mathsf{KGen}, \mathsf{SHK}) \in \mathcal{F}_{\mathcal{P}^{\mathsf{NIKE}}}$ *and a* PPT *(adversarial) machine* $\mathcal{A}$*, we define* $\langle f, \mathcal{A} \rangle \in \mathcal{R}_{\mathcal{P}^{\mathsf{NIKE}}}$ *if there exists a polynomial* $p(\cdot)$ *such that* $\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{pk}') = \mathsf{SHK}(\mathsf{pk}', \mathsf{sk}) = \mathsf{SHK}(\mathsf{pk}, \mathsf{sk}') \neq \perp] > 1/p(\lambda)$ *for infinitely many* λ*.*
   Here, for randomly chosen $\mathsf{sk} \xleftarrow{\$} \mathcal{SK}$ *and* $\mathsf{sk}' \xleftarrow{\$} \mathcal{SK}$*, compute* $\mathsf{pk} \leftarrow \mathsf{KGen}(\mathsf{sk})$ *and* $\mathsf{pk}' \leftarrow \mathsf{KGen}(\mathsf{sk}')$*, respectively.*

We say non-interactive key exchange protocol $\mathcal{P}^{\mathsf{NIKE}}$ *exists, if there exists a function* $f \in \mathcal{F}_{\mathcal{P}^{\mathsf{NIKE}}}$*, it holds that no* PPT *adversarial machine* $\mathcal{A}$ *such that* $\langle f, \mathcal{A} \rangle \in \mathcal{R}_{\mathcal{P}^{\mathsf{NIKE}}}$*. When* $\mathcal{SK} = \mathbb{Z}_N$ *and* $\mathcal{PK} = \mathcal{K} = \{0,1\}^m$*, we make the parameters* N *and* m *explicit and denote the non-interactive key exchange protocol as* $\mathcal{P}^{\mathsf{NIKE}}_{N,m}$*.*

2.1.2 Idealized Models In this subsection, we introduce idealized models including the Random Oracle Model (ROM) [BR93], the Random Permutation Model (RPM) [RS08], and the Generic Group Model (GGM) [Sho97]. In each idealized model, all entities including the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$, are provided with the access to the corresponding oracle. Below we will specify the behavior of the oracle in each idealized model.

Definition 4 (Random Oracle Model [BR93]). *Let* $\mathcal{I}_{*,S}$ *denote the set of functions* $h : \{0,1\}^* \rightarrow S$*, where* $S := \{0,1\}^n$ *for some integer* n*. The random oracle model* $\mathcal{H}$ *is an idealized model, sampling a random function* h *from* $\mathcal{I}_{*,S}$*. Every algorithm can query* x*, obtaining the corresponding value* $h(x) \in S$*.*

Definition 5 (Random Permutation Model [RS08]**).** *Let $\mathcal{I}_{S,S}$ denote the set of permutations $\pi : S \rightarrow S$, where $S := \{0,1\}^n$ for some integer n. The random permutation model $\mathcal{E}$ is an idealized model, sampling a random permutation π from $\mathcal{I}_{S,S}$. Every algorithm can query $x \in S$ with $\mathcal{E}$ for both π and its inverse π^{-1}, obtaining the corresponding value $\pi(x) \in S$ or $\pi^{-1}(x) \in S$.*

Definition 6 (Generic Group Model [Sho97]**).** *Denote by $\mathcal{I}_{\mathbb{Z}_N,S}$ the set of injections $\sigma : \mathbb{Z}_N \mapsto S$, where $S := \{0,1\}^m$. The generic group model $\mathcal{G}_{N,m}$ is an idealized model, sampling a random injection σ from $\mathcal{I}_{\mathbb{Z}_N,S}$, with functions $\mathcal{G}^{\mathsf{label}}_{N,m}$ and $\mathcal{G}^{\mathsf{add}}_{N,m}$. Concretely, for each query $x \in \mathbb{Z}_N$, the "labeling" function $\mathcal{G}^{\mathsf{label}}_{N,m}$ responds with a value $\sigma(x) \in S$. For a query (g_1, g_2), the "adding" function $\mathcal{G}^{\mathsf{add}}_{N,m}$ answers as follows: if $g_1 = \sigma(x_1)$ and $g_2 = \sigma(x_2)$ for some $x_1, x_2 \in \mathbb{Z}_N$, replying by $\sigma(x_1 + x_2)$, and replying by $\perp$ otherwise.*

2.1.3 Notions of Reductions

To establish separations between primitives, in this paper, we follow two notions, *fully black-box reduction* and *relativizing reduction*, as formalized by Reingold, Trevisan, and Vadhan [RTV04].

Definition 7 (Fully Black-Box Reduction [RTV04]**).** *There exists a fully black-box reduction from a primitive $\mathcal{P} := \langle \mathcal{F}_\mathcal{P}, \mathcal{R}_\mathcal{P} \rangle$ to a primitive $\mathcal{Q} := \langle \mathcal{F}_\mathcal{Q}, \mathcal{R}_\mathcal{Q} \rangle$, if there exist* PPT *oracle machines Π and $\mathcal{B}$ such that:*

Correctness *For every implementation $f \in \mathcal{F}_\mathcal{Q}$ we have that $\Pi^f \in \mathcal{F}_\mathcal{P}$.*
Security *For every implementation $f \in \mathcal{F}_\mathcal{Q}$, if there exists a* PPT *oracle machine $\mathcal{A}$ such that $\mathcal{A}^f$ $\mathcal{P}$-breaks Π^f, then there exists a* PPT *oracle machine $\mathcal{B}$ such that $\mathcal{B}^f$ $\mathcal{Q}$-breaks f.*

In literature, a typical technique for black-box separation, say for primitives $\mathcal{P}$ and $\mathcal{Q}$, is relativizing separation, which means that there is no relativizing reduction between $\mathcal{P}$ and $\mathcal{Q}$. Reingold et al. [RTV04] indicate that fully black-box reduction implies relativizing reduction, referring to that the relativizing separation from $\mathcal{P}$ to $\mathcal{Q}$ indicates the corresponding fully black-box separation.

Definition 8 (Relativizing Reduction [RTV04]**).** *There exists a relativizing reduction from a primitive $\mathcal{P} := \langle \mathcal{F}_\mathcal{P}, \mathcal{R}_\mathcal{P} \rangle$ to a primitive $\mathcal{Q} := \langle \mathcal{F}_\mathcal{Q}, \mathcal{R}_\mathcal{Q} \rangle$, if for every oracle $\mathcal{O}$, the primitive $\mathcal{P}$ exists relative to $\mathcal{O}$ whenever $\mathcal{Q}$ exists relative to $\mathcal{O}$. A primitive $\mathcal{P}$ is said to exist relative to $\mathcal{O}$, if there exists $f \in \mathcal{F}_\mathcal{P}$ which has an efficient implementation when having access to the oracle $\mathcal{O}$ such that no* PPT *oracle machine with access to $\mathcal{O}$, can $\mathcal{P}$-break f.*

2.2 Indifferentiability

The framework of indifferentiability is proposed by Maurer, Renner, and Holenstein [MRH04], which formalizes a set of necessary and sufficient conditions for securely replacing one cryptosystem with another in an arbitrary environment. This framework is used to justify the structural soundness of various cryptographic primitives, including hash functions [CDMP05,DRS09], block ciphers [ABD+13, CHK+16, DSSL16, GWL23], domain extenders [CDMS10], authenticated encryption with associated data [BF18], and public key cryptosystems [ZZ20]. It can also be used to study the relationship between idealized models [ZZ23]. Within the context of the indifferentiability framework, it is customary to consider that a cryptosystem either implements certain ideal objects denoted as $\mathcal{F}$, or it is a construction denoted as $C^{\mathcal{F}'}$ that relies on underlying ideal objects $\mathcal{F}'$.

Definition 9 (Indifferentiability [MRH04]). *Let Σ_1 and Σ_2 be two cryptosystems and $\mathcal{S}$ be a simulator. The indifferentiability advantage of a distinguisher $\mathcal{D}$ against (Σ_1, Σ_2) with respect to $\mathcal{S}$ is*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Sigma_1,\Sigma_2,\mathcal{S},\mathcal{D}}(1^\lambda) := \Pr[\mathrm{Real}_{\Sigma_1,\mathcal{D}}] - \Pr[\mathrm{Ideal}_{\Sigma_2,\mathcal{S},\mathcal{D}}],$$

where games $\mathrm{Real}_{\Sigma_1,\mathcal{D}}$ and $\mathrm{Ideal}_{\Sigma_2,\mathcal{S},\mathcal{D}}$ are defined in Fig. 2. We say Σ_1 is indifferentiable from Σ_2, if there exists an efficient simulator $\mathcal{S}$ such that for any efficient distinguisher $\mathcal{D}$, the advantage above is negligible. Moreover, we say Σ_1 is statistically indifferentiable from Σ_2, if there exists an efficient simulator such that, for any unbounded distinguisher $\mathcal{D}$, the advantage above is negligible.

```
Indifferentiability

Real_{Σ1,D}:                     HonestR(X)
  $b \leftarrow \mathcal{D}^{\mathrm{HonestR},\mathrm{AdvR}}$      Return $\Sigma_1.\mathrm{hon}(X)$.
  Return $b$.                    AdvR(X)
                                 Return $\Sigma_1.\mathrm{adv}(X)$.

Ideal_{Σ2,S,D}:                  HonestI(X)
  $b \leftarrow \mathcal{D}^{\mathrm{HonestI},\mathrm{AdvI}}$      Return $\Sigma_2.\mathrm{hon}(X)$.
  Return $b$.                    AdvI(X)
                                 Return $\mathcal{S}^{\Sigma_2.\mathrm{adv}(\cdot)}(X)$.
```

Fig. 2. Indifferentiability of Σ_1 and Σ_2, where $\mathcal{S}$ is the simulator and $\mathcal{D}$ is the adversary.

Below, we also use the notations in [BF18] and consider the definition above to two systems with interfaces as:

$$(\Sigma_1.\mathrm{hon}(X), \Sigma_1.\mathrm{adv}(x)) := (\Pi^{\mathcal{F}_1}(X), \mathcal{F}_1(x)),$$
$$(\Sigma_2.\mathrm{hon}(X), \Sigma_2.\mathrm{adv}(x)) := (\mathcal{F}_2(X), \mathcal{F}_2(x)),$$

where $\mathcal{F}_1$ and $\mathcal{F}_2$ are two ideal objects sampled from their distributions and $\Pi^{\mathcal{F}_1}$ is a construction of $\mathcal{F}_2$ by calling $\mathcal{F}_1$. Maurer, Renner, and Holenstein prove the composition theorem for the framework of indifferentiability; for simplicity, we give a game-based formalization from [RSS11].

Theorem 3 (Composition Theorem [MRH04]). *Let $\Sigma_1 := (\Pi^{\mathcal{F}_1}, \mathcal{F}_1)$ and $\Sigma_2 := (\mathcal{F}_2, \mathcal{F}_2)$ be two systems that Σ_1 is indifferentiable from Σ_2 with respect to a simulator $\mathcal{S}$, then Σ_1 is as secure as Σ_2 for any single-stage game. More concretely, let* Game *be a single-stage game, then for any adversary $\mathcal{A}$, there is an adversary $\mathcal{B}$ and a distinguisher $\mathcal{D}$ such that*

$$\Pr[\mathsf{Game}_{\Pi^{\mathcal{F}_1},\mathcal{A}^{\mathcal{F}_1}}] \leq \Pr[\mathsf{Game}_{\mathcal{F}_2,\mathcal{B}^{\mathcal{F}_2}}] + \mathrm{Adv}^{\mathsf{indif}}_{\Sigma_1,\Sigma_2,\mathcal{S},\mathcal{D}}.$$

The proof of Theorem 3 is straightforward; due to space limit, we skip it here. Next, we give the formal definition of the separation between two idealized models in the framework of indifferentiability against computational adversaries.

Definition 10 (Computational Indifferentiable Separation [MRH04, ZZ23]**).** *Let Σ_1, Σ_2 be two idealized models, we say Σ_2 is computationally indifferentiably separated from Σ_1 if for any efficient algorithm Π and any efficient simulator $\mathcal{S}$, there exists an efficient distinguisher $\mathcal{D}_{\Pi,\mathcal{S}}$ and a noticeable function ρ such that*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Pi^{\Sigma_1},\Sigma_2,\mathcal{S},\mathcal{D}_{\Pi,\mathcal{S}}}(1^\lambda) := \Big|\Pr[\mathrm{Real}_{\Sigma_1,\mathcal{D}_{\Pi,\mathcal{S}}}] - \Pr[\mathrm{Ideal}_{\Sigma_2,\mathcal{S},\mathcal{D}_{\Pi,\mathcal{S}}}]\Big| \geq \rho(\lambda).$$

Observe that, if an idealized model Σ_2 is computationally indifferentiably separated from another idealized model Σ_1, it means that, we cannot build a scheme Π^{Σ_1} such that Π^{Σ_1} is indifferentiable from Σ_2, even under arbitrarily strong computational assumptions.

3 Separation Between Cryptographic Groups

In this section, we establish the separation between two CDH-secure groups, $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ and $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$, under the condition that both N and $(m_2 - m_1)$ are sufficiently large within the same security parameter.

Theorem 4 (Main Theorem). *Let $\lambda \in \mathbb{N}$ be the security parameter. Let N, m_1, m_2 be integers such that $N \geq 2^{\omega(\log \lambda)}, m_1 > \log N$ and $m_2 - m_1 \geq \omega(\log \lambda)$. Then there is no black-box reduction from $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$ to $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$.*

Proof. To establish the theorem, we apply the so-called two-oracle technique [HR04]. Let PSPACE be a PSPACE-complete oracle. Essentially, we set $\mathcal{O} := (\mathsf{PSPACE}, \mathcal{G}_{N,m_2})$ and prove the following:

1. $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$ exists relative to $\mathcal{O}$;
2. $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ does not exist relative to $\mathcal{O}$.

The former statement holds trivially as $\mathcal{G}_{N,m_2}$ implies $\mathcal{P}^{\mathsf{CDH}}_{N,m_2}$ in the canonical manner. Therefore, it suffices to prove the latter one.

Lemma 1. *$\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ does not exist relative to $\mathcal{O}$.*

To establish the proof, we first pin down an intermediary primitive, i.e., $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$ (within the same security parameter), defined in Sect. 2.1, and then prove that:

1. $\mathcal{P}^{\mathsf{CDH}}_{N,m_1}$ implies $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$;
2. $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$ does not exist relative to $\mathcal{O}$.

The first statement holds straightforwardly. Next, we establish our theorem by proving the following lemma.

Lemma 2. $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$ *does not exist relative to* $\mathcal{O}$*.*

Intuitively, to prove that $\mathcal{P}^{\mathsf{NIKE}}_{N,m_1}$ does not exist relative to $\mathcal{O}$, it is sufficient to build a PPT oracle adversary $\mathcal{A}^{\mathcal{O}}$ that breaks any construction $\Pi^{\mathcal{O}} := (\mathsf{KGen}^{\mathcal{O}}, \mathsf{SHK}^{O})$. Observe that $\mathcal{A}^{\mathcal{O}}$ has access to a PSPACE-complete oracle, which means that $\mathcal{A}^{\mathcal{O}}$ implies a computationally unbounded but query-efficient adversary that only has access to $\mathcal{G}_{N,m_2}$[16]. Therefore, it suffices to construct such an adversary $\mathcal{A}^{\mathcal{G}_{N,m_2}}$. In Fig. 3, we illustrate the description of the adversary.

We first clarify some undefined notions: Let n be a sufficiently large integer that will be specified below. By $\{(\mathsf{que}_1, \mathsf{res}_1), \dots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\mathsf{query}} \mathsf{KGen}^{\mathcal{G}_{N,m_2}}(r_i)$, we mean that when running the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(r_i)$, the algorithm makes queries $(\mathsf{que}_1, \dots, \mathsf{que}_q)$ to the oracle $\mathcal{G}_{N,m_2}$ and obtains $(\mathsf{res}_1, \dots, \mathsf{res}_q)$[17].

Next, we prove that $\mathcal{A}^{\mathcal{G}_{N,m_2}}$ outputs the valid shared key with noticeable probability. Let $S_{B\text{-}\mathsf{label}}$ be the set of the valid group elements that appear when running $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}_B)$ and $\mathsf{SHK}^{\mathcal{G}_{N,m_2}}(\mathsf{pk}_A, \mathsf{sk}_B)$; those group elements are either the responses of labeling/addition queries or the valid inputs of the addition queries. It is apparent that $|S_{B\text{-}\mathsf{label}}| \leq 6q$, due to the fact that each algorithm makes at most q queries. Now, we define:

$$S_B := \{(x, h) | h \in S_{B\text{-}\mathsf{label}},\ \mathcal{G}^{\mathsf{label}}_{N,m_2}(x) = h\}.$$

Note that, for any iteration, if the adversary successfully guesses S_B in $\hat{S}_{\mathcal{A}}$, then the shared key computed in this iteration would be valid. Specifically, in such a context, there exists an instance of the GGM that is consistent with the query views of both the adversary and the user B, and the validity of the shared key follows by the perfect correctness of $\Pi^{\mathcal{G}_{N,m_2}}$. However, without the knowledge of sk_B, $\mathcal{A}$ might not guess S_B correctly with a good probability. In fact, there are three events:

- Event 1: There exist $(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}$ and $(\mathsf{que}_B, \mathsf{res}_B) \in S_B$ such that $\mathsf{que}_A = \mathsf{que}_B$ but $\mathsf{res}_A \neq \mathsf{res}_B$.
- Event 2: There exist $(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}$ and $(\mathsf{que}_B, \mathsf{res}_B) \in S_B$ such that $\mathsf{que}_A \neq \mathsf{que}_B$ but $\mathsf{res}_A = \mathsf{res}_B$.
- Event 3: For any $(\mathsf{que}_A, \mathsf{res}_A) \in \hat{S}_{\mathcal{A}}$, $(\mathsf{que}_B, \mathsf{res}_B) \in S_B$, we have that if $\mathsf{que}_A = \mathsf{que}_B$ then $\mathsf{res}_A = \mathsf{res}_B$, and vice versa.

[16] Any computationally unbounded but query-efficient adversary can be simulated by a PPT oracle machine with access to a PSPACE-complete oracle, that is because what we need are specific labeling query-response tuples of GGM. These tuples can be picked by using a PSPACE-complete oracle. See [MM11] for more details.

[17] As explained above, we stress that $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}$ only makes labeling queries.

```
Adversary $\mathcal{A}^{\mathcal{G}_{N,m_2}}(\mathsf{pk}_A, \mathsf{pk}_B)$

$\underline{\mathcal{A}^{\mathcal{G}_{N,m_2}}(\mathsf{pk}_A, \mathsf{pk}_B)}$:
$S_{\mathsf{key}} \leftarrow \emptyset$; $S_{\mathsf{que\text{-}res}} \leftarrow \emptyset$; $r_1, \dots, r_n \xleftarrow{\$} \mathbb{Z}_N$;
Initial phase: //collecting frequent queries
    for $i = 1$ to $n$:
        $\{(\mathsf{que}_1, \mathsf{res}_1), \dots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\mathsf{query}} \mathsf{KGen}^{\mathcal{G}_{N,m_2}}(r_i)$
        $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(\mathsf{que}_1, \mathsf{res}_1), \dots, (\mathsf{que}_q, \mathsf{res}_q)\}$;
for $i = 1$ to $12q + 1$: //running $12q + 1$ iterations
    Simulation phase: //searching a proper view
        Search a secret key $\widetilde{\mathsf{sk}}_A$ and a set of query-response tuples $\hat{S}_{\mathcal{A}}$ satisfying
        the following properties:
            Property 1: $\hat{S}_{\mathcal{A}}$ is consistent with $S_{\mathsf{que\text{-}res}}$;
            Property 2: $\mathsf{pk}_A = \mathsf{KGen}^{S_{\mathsf{que\text{-}res}} \cup \hat{S}_{\mathcal{A}}}(\widetilde{\mathsf{sk}}_A)$;
            Property 3: $\hat{S}_{\mathcal{A}}$ only collects tuples of labeling queries and $|\hat{S}_{\mathcal{A}}| \leq 12q$;
            Property 4: $\hat{S}_{\mathcal{A}}$ is sufficient for SHK. That is, when running the algo-
            rithm $\widetilde{\mathsf{shk}}_i \leftarrow \mathsf{SHK}^{S_{\mathsf{que\text{-}res}} \cup \hat{S}_{\mathcal{A}}}(\mathsf{pk}_B, \widetilde{\mathsf{sk}}_A)$: (1) the set $S_{\mathsf{que\text{-}res}} \cup \hat{S}_{\mathcal{A}}$ covers
            all labeling queries; (2) the set $S_{\mathsf{que\text{-}res}} \cup \hat{S}_{\mathcal{A}}$ is able to convert any ad-
            dition query into a labeling query properly. Let $\mathsf{que} = (h_1, h_2)$ be an
            addition query when running the shared-key algorithm, if either $h_1$ or $h_2$
            is not covered in $\hat{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$, then responds with $\perp$; otherwise the set
            $S_{\mathsf{que\text{-}res}} \cup \hat{S}_{\mathcal{A}}$ must cover the following three tuples: $(x_1, h_1), (x_2, h_2)$ and
            $(x_1 + x_2, h_3)$, and responds with $h_3$.
        $S_{\mathsf{key}} \leftarrow S_{\mathsf{key}} \cup \{\widetilde{\mathsf{shk}}_i\}$;
    Update phase: //updating the guessing labeling queries with valid encodings
        for each $(\mathsf{que}, \mathsf{res}) \in \hat{S}_{\mathcal{A}} \setminus S_{\mathsf{que\text{-}res}}$: $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup (\mathsf{que}, \mathcal{G}^{\mathsf{label}}_{N,m_2}(\mathsf{que}))$;
Final phase: //outputting the guessing shared key
    return the majority value in $S_{\mathsf{key}}$.
```

Fig. 3. The description of the adversary that breaks $\Pi^{\mathcal{G}_{N,m_2}}$.

We immediately observe that event 1 occurs at most $6q$ times, because the updating phase would eliminate at least one pair in S_B. Therefore, it suffices to prove that event 2 never occurs except for negligible probability and event 3 would deduce the valid shared key with high probability. According to the description of the adversary Fig. 3, we have that in event 3, the set $\hat{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$ responds to the labeling queries perfectly and converts the addition queries into labeling queries properly. Concretely, let $\mathsf{que} := (h_1, h_2)$ be an addition query, there are two cases: (1) $\hat{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$ covers (x_1, h_1), (x_2, h_2), and $(x_1 + x_2, h_3)$; (2) either h_1 or h_2 is not stored in $\hat{S}_{\mathcal{A}} \cup S_{\mathsf{que\text{-}res}}$. For the former case, the response is valid; for latter one, the response is invalid if and only if both h_1 and h_2 are valid group encodings. Therefore, the only bad case that prevents event 3 from deducing the valid shared key is that the adversary outputs a valid group encoding h without knowing the discrete logarithm.

Moreover, in the simulation phase, $\hat{S}_{\mathcal{A}}$ must be consistent with $S_{\mathsf{que\text{-}res}}$, which indicates that when event 2 occurs, the adversary successfully outputs a valid group encoding h without making labeling query. To bound the probability, we define that, for any $\mathsf{sk} \in \mathbb{Z}_N$:

$$Q_{\mathsf{sk}} := \left\{(\mathsf{que}_1, \mathsf{res}_1), \dots, (\mathsf{que}_q, \mathsf{res}_q)\right\} \xleftarrow{\mathsf{query}} \mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}).$$

Note that the adversary only takes pk_A and pk_B as inputs, where $\mathsf{pk}_A = \mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}_A)$ and $\mathsf{pk}_B = \mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}_B)$. It is apparent that the group encoding $h \notin S_{\mathsf{que\text{-}res}}$, and we next establish our analysis by considering the following four cases:

- Case 1: (**Independent group encoding**) $h \notin Q_{\mathsf{sk}_A} \cup Q_{\mathsf{sk}_B}$
- Case 2: (**Frequent group encoding**) $h \in Q_{\mathsf{sk}_A} \cap Q_{\mathsf{sk}_B}$
- Case 3: (**Dependent but hard to extract**) $h \in Q_{\mathsf{sk}_A} \setminus Q_{\mathsf{sk}_B}$.
- Case 4: (**Dependent but hard to extract**) $h \in Q_{\mathsf{sk}_B} \setminus Q_{\mathsf{sk}_A}$.

It is apparent that, for any query-efficient adversary (might be computationally inefficient), if the probability that it outputs such an h (for all cases) is bounded, then we are done.

Case 1. We note that, h is independent of pk_A and pk_B, indicating that the probability that any adversary outputs such a h is bounded by $\frac{O(q)\cdot N}{2^{m_2}} \le \mathsf{negl}(\lambda)$.

Case 2. We first define the frequent group encodings. Specifically, let $t := 26q^2$, we say a group encoding res is frequent if

$$\Pr[(\mathsf{que}, \mathsf{res}) \in Q_z : z \xleftarrow{\$} \mathbb{Z}_N] \ge \frac{1}{t}.$$

In such a case, we also call $(\mathsf{que}, \mathsf{res})$ as a frequent query. Note that sk_A and sk_B are uniformly sampled, therefore, for any $(\mathsf{que}, \mathsf{res}) \in Q_{\mathsf{sk}_A}$, if it is not a frequent query, then $\Pr[(\mathsf{que}, \mathsf{res}) \in Q_{\mathsf{sk}_B}] \le \frac{1}{t}$, indicating that

$$\Pr[Q_{\mathsf{sk}_A} \cap Q_{\mathsf{sk}_B} \text{ are all frequent queries}] \ge 1 - \frac{q}{t} = 1 - \frac{1}{26q}.$$

Next, we bound the probability that $h \notin S_{\mathsf{que\text{-}res}}$ conditioned on that $Q_{\mathsf{sk}_A} \cap Q_{\mathsf{sk}_B}$ are all frequent queries. Let $n := t \cdot \lambda$, we then prove that, with a high probability, $Q_{r_1} \cup \cdots \cup Q_{r_n}$ contains all frequent queries. Essentially, there are at most $q_f := q \cdot t$ frequent queries, denoted as $\{(\mathsf{que}'_i, \mathsf{res}'_i)\}_{i \in [q_f]}$. For each $(\mathsf{que}'_i, \mathsf{res}'_i)$, we have that

$$\Pr[(\mathsf{que}'_i, \mathsf{res}'_i) \notin Q_{r_1} \cup \cdots \cup Q_{r_n}] \le \left(1 - \frac{1}{t}\right)^n \le e^{-\lambda},$$

which means

$$\Pr[(\mathsf{que}'_i, \mathsf{res}'_i) \in Q_{r_1} \cup \cdots \cup Q_{r_n} : \forall i \in [q_f]] \ge 1 - (q \cdot t)e^{-\lambda}.$$

Therefore,

$$\Pr[\text{Case 2}] = \Pr[h \in Q_{\mathsf{sk}_A} \cap Q_{\mathsf{sk}_B} \wedge h \notin S_{\mathsf{que\text{-}res}}] \le \frac{1}{26q} + (q \cdot t)e^{-\lambda} \le \frac{1}{26q} + \mathsf{negl}(\lambda).$$

Case 3. We immediately observe that pk_B is independent of h, which means that only pk_A carries the information of h. Note that the length of pk_A is m_1; in contrast, the length of h is m_2; this intuitively indicates that, over the probability of sampling the GGM instance, it is impossible to extract a valid group encoding in $Q_{\mathsf{sk}_A} \setminus (Q_{\mathsf{sk}_B} \cup S_{\mathsf{que\text{-}res}})$ except for negligible probability.

To establish the formal analysis, we strengthen the adversary $\mathcal{A}$ by providing $\mathcal{A}$ the unbounded computational power, and the following information: the tuple $(\mathsf{sk}_A, \mathsf{sk}_B, \mathsf{pk}_A, Q_{\mathsf{sk}_B}, S_{\mathsf{que\text{-}res}})$. It is easy to see that $\mathcal{A}$ itself can compute $\mathsf{pk}_A, \mathsf{pk}_B$ and $S_{\mathsf{que\text{-}res}}$, therefore it suffices to prove that

$$\Pr[\mathcal{A} \text{ outputs } h \in Q_{\mathsf{sk}_A} \setminus (Q_{\mathsf{sk}_B} \cup S_{\mathsf{que\text{-}res}})] \leq \mathsf{negl}(\lambda)$$

where the probability is over the sampling of $\mathsf{sk}_A, \mathsf{sk}_B$ and the GGM instance[18]. Observe that, pk_B is independent of h, which indicates that knowing sk_B would not increase $\mathcal{A}$'s winning probability. To further simplify the analysis, we prove a more general statement: for any secret key sk and any S (set of query-response tuples, poly-size),

$$\Pr[\mathcal{A}(\mathsf{sk}, \mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}), S) \to h : h \in Q_{\mathsf{sk}} \setminus S] \leq \mathsf{negl}(\lambda)$$

where the probability is *only* over the sampling of the GGM instance, conditioned on that the GGM instance $\mathcal{G}_{N,m_2}$ is consistent with S.

Note that, for any fixed poly-size S, the total number of the GGM instances (mapping from N to $\{0,1\}^{m_2}$) that are consistent with S is

$$(2^{m_2} - |S|) \cdot (2^{m_2} - (|S| + 1)) \cdots (2^{m_2} - (N - 1)).$$

Next, we introduce some notations. Note that, once the secret key sk and the GGM instance $\mathcal{G}_{N,m_2}$ are fixed, the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk})$ is deterministic (including the queries made to $\mathcal{G}_{N,m_2}$). We here define $Q_{\mathsf{sk}\text{-}\mathcal{G}}$ as the sequence of the query-response tuples, denoted as

$$Q_{\mathsf{sk}\text{-}\mathcal{G}} := \{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\}.$$

More clearly, when running the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk})$, the i-th query that the algorithm makes to $\mathcal{G}_{N,m_2}$ is que_i and the corresponding response is res_i. Besides, for each $(\mathsf{sk}, \mathcal{G}_{N,m_2})$, the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk})$ outputs a public key. Next, we categorize the public keys into two types, namely the "good public keys" and the "bad public keys", with respect to the fixed secret key sk. We denote

$$T = 2^{\frac{m_2 - m_1}{2}} \cdot (2^{m_2} - (|S| + 1)) \cdots (2^{m_2} - (N - 1)),$$

and for any public key pk we denote S_{pk} as the set of the GGM instances such that $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}) = \mathsf{pk}$. Now, we say a public key pk (with respect to sk) is bad if $|S_{\mathsf{pk}}| \leq T$, otherwise we say the public key is good. Note that, given a

[18] The instance of GGM must be consistent with $Q_{\mathsf{sk}_B} \cup S_{\mathsf{que\text{-}res}}$.

bad public key pk (e.g., $|S_{\mathsf{pk}}| = 1$), the adversary might output a valid group encoding, thus we need to prove that, over the sampling of the GGM instance,

$$\Pr[\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}) \text{ is bad}] \leq \mathsf{negl}(\lambda).$$

Note that the space of public keys is $\{0,1\}^{m_1}$, which means that there are at most 2^{m_1} public keys. Therefore, the counting of the GGM instances that induce to a bad public key is bounded by $2^{m_1} \times T$, referring to

$$\Pr[\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}) \text{ is bad}] \leq \frac{2^{m_1} \cdot 2^{\frac{m_2 - m_1}{2}}}{(2^{m_2} - |S|)} \leq \frac{1}{2^{\frac{m_2 - m_1}{2}} - |S|} \leq \mathsf{negl}(\lambda).$$

Hence, it suffices to prove that, given any good public key, any adversary $\mathcal{A}$ cannot extract a valid group encoding $h \in Q_{\mathsf{sk}} \setminus S$ except for negligible probability.

For readability, we first elaborate the analysis in the case that $S = \emptyset$, where the adversary only has knowledge of $(\mathsf{sk}, \mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk}))$. Let str be any string in $\{0,1\}^{m_2}$, we denote S_{str} as the set of GGM instances such that $\mathsf{str} \in Q_{\mathsf{sk}\text{-}\mathcal{G}}$. Therefore it is sufficient to prove that, for any $\mathsf{str} \in \{0,1\}^{m_2}$, the size of S_{str} is much smaller than T (in this special case, $|S| = 0$). Specifically, by having that

$$T > 2^{\frac{m_2 - m_1}{2}} \cdot (2^{m_2} - 1) \cdots (2^{m_2} - (N-1))$$

we prove that

$$|S_{\mathsf{str}}| \leq q \cdot (2^{m_2} - 1) \cdots (2^{m_2} - (N-1))$$

Note that, once the secret key sk and the GGM instance $\mathcal{G}_{N,m_2}$ are fixed, the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk})$ is deterministic. We next illustrate an observation about $Q_{\mathsf{sk}\text{-}\mathcal{G}}$. Let $\mathcal{G}_{N,m_2}$ and $\mathcal{G}'_{N,m_2}$ be two different instances of GGM, and we denote

$$Q_{\mathsf{sk}\text{-}\mathcal{G}} := \{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\}$$
$$Q_{\mathsf{sk}\text{-}\mathcal{G}'} := \{(\mathsf{que}'_1, \mathsf{res}'_1), \ldots, (\mathsf{que}'_q, \mathsf{res}'_q)\}$$

We claim that either $Q_{\mathsf{sk}\text{-}\mathcal{G}} = Q_{\mathsf{sk}\text{-}\mathcal{G}'}$ or $\exists i \in [q]$ such that $\mathsf{res}_i \neq \mathsf{res}'_i$. In other words, it is impossible that $Q_{\mathsf{sk}\text{-}\mathcal{G}} \neq Q_{\mathsf{sk}\text{-}\mathcal{G}'}$ but $(\mathsf{res}_1, \ldots, \mathsf{res}_q) = (\mathsf{res}'_1, \ldots, \mathsf{res}'_q)$. In fact, if such an event occurs, then there exists an index $j \in [q]$ such that (1) $\forall i < j$, $(\mathsf{que}_i, \mathsf{res}_i) = (\mathsf{que}'_i, \mathsf{res}'_i)$; (2) $\mathsf{que}_j \neq \mathsf{que}'_j$, which contradicts to that $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\mathsf{sk})$ is deterministic.

This observation illustrates that $Q_{\mathsf{sk}\text{-}\mathcal{G}}$ can be represented only by $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$; that is, once the sequence of the responses is fixed, then the corresponding sequence of the queries is also settled down. We denote

$$V = ((2^{m_2} - q) \cdots (2^{m_2} - (N-1)))$$

and note that for each response sequence $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$, there are exactly V numbers of GGM instances that would induce it.

Next, we compute the upper bound of $|S_{\mathsf{str}}|$. If str appears in the sequence $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$, then there exists an index i such that $\mathsf{res}_i = \mathsf{str}$. For the rest, we maximize the possibility and have that the number of all possible sequences that contain str is bounded by

$$q \cdot ((2^{m_2} - 1) \cdots (2^{m_2} - (q-1))).$$

Combining the above together, we have that

$$|S_{\mathsf{str}}| \leq q \cdot (2^{m_2} - 1) \cdots (2^{m_2} - (N-1)).$$

In the following, we extend our analysis into the general case, where S is poly-size and

$$T = 2^{\frac{m_2 - m_1}{2}} \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1))$$

We immediately observe that, the upper bound above does not serve our purpose any more. The reason is that the upper bound above is calculated over all possible GGM instances, while what we need to count are the ones over the GGM instances that are consistent with S.

It is apparent that $Q_{\mathsf{sk}\text{-}\mathcal{G}}$ can be still represented by the sequence of responses when $S \neq \emptyset$. To complete the analysis, we then illustrate an additional observation about $Q_{\mathsf{sk}\text{-}\mathcal{G}}$. Let $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$ and $(\mathsf{res}'_1, \ldots, \mathsf{res}'_q)$ be two different sequences. We claim it is impossible that there exists an index $j \in [q]$ such that(1) $\forall i < j$, $\mathsf{res}_i = \mathsf{res}'_i$; (2)$\mathsf{res}_j \in S$ but $\mathsf{res}'_j \notin S$[19]. More specifically, given the statement that $\forall i < j$, $\mathsf{res}_i = \mathsf{res}'_i$, it is apparent that $\mathsf{que}_j = \mathsf{que}'_j$. Moreover, by having $(\mathsf{que}_j, \mathsf{res}_j) \in S$, we claim that the response of que'_j must be res_j, because the GGM instances must be consistent with S. Based on this new observation, we next prove the upper bound by induction.

Let str be a string such that $\mathsf{str} \notin S$ (note that the adversary's goal is to output a valid group encoding without knowing the discrete logarithm), we denote $S_{\mathsf{str}\text{-}k}$ as the set of the GGM instances such that: (1) the algorithm $\mathsf{KGen}^{\mathcal{G}_{N,m_2}}(\cdot)$ makes k queries; (2) $\mathsf{str} \in Q_{\mathsf{sk}\text{-}\mathcal{G}} \setminus S$. We then prove that for any k,

$$|S_{\mathsf{str}\text{-}k}| \leq k \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)).$$

We first compute $|S_{\mathsf{str}\text{-}1}|$. Note that que_1 is always fixed, and if $\mathsf{que}_1 \in S$[20], then $|S_{\mathsf{str}\text{-}1}| = 0$ because str would never appear. On the other hand, if $\mathsf{que}_1 \notin S$, then the response must be str because str appears. Thus, the counting of the GGM instances that are consistent with $S \cup \{(\mathsf{que}_1, \mathsf{str})\}$ is

$$1 \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1))$$

Note that, the response of que_1 is str if and only if those GGM instances are sampled. Moreover, based on our second observation, we have that, either $\mathsf{res}_1 \in S$ or $\mathsf{res}_1 \notin S$. Hence,

$$|S_{\mathsf{str}\text{-}1}| \leq \max\{0, 1 \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1))\}.$$

[19] We here abuse the notation $\mathsf{res}_j \in S$ by meaning that there exists a query/response tuple in S with the response res_j.

[20] We here abuse the notation $\mathsf{que}_1 \in S$ by meaning that there exists a query/response pair in S with the query is que_1.

Next, given the assumption that

$$|S_{\textsf{str-}i}| \leq i \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)),$$

we prove

$$|S_{\textsf{str-}(i+1)}| \leq (i+1) \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)),$$

Again, $\textsf{que}_1$ is always fixed, and if $\textsf{que}_1 \in S$, then $|S_{\textsf{str-}(i+1)}|$ is bounded by $|S_{\textsf{str-}i}|$, because the response of $\textsf{que}_1$ is always fixed by S, and $\textsf{str}$ must appear in the last i queries. Thus, it suffices to prove that $|S_{\textsf{str-}(i+1)}|$ is properly bounded when $\textsf{que}_1 \notin S$. Next we consider two scenarios:

- Scenario 1: $\textsf{res}_1 = \textsf{str}$;
- Scenario 2: $\textsf{res}_1 \neq \textsf{str}$.

Observe that scenario 1 occurs if and only if the GGM instances that are consistent with $S \cup \{(\textsf{que}_1, \textsf{str})\}$ are selected. Therefore, the counting of those GGM instances is:

$$1 \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)).$$

When scenario 2 occurs, there are at most $2^{m_2} - (|S|+1)$ options for $\textsf{res}_1$. Once the response of $\textsf{que}_1$ is fixed, say $(\textsf{que}_1, \textsf{str}')$, we apply the induction. Specifically, we denote $S' = S \cup \{(\textsf{que}_1, \textsf{str}')\}$ ($|S'| = |S| + 1$). Note that scenario 2 occurs means that $\textsf{str}$ appears in the last i queries conditioned on that all the GGM instances are consistent with S'. Applying the assumption, we have that the counting of the GGM instances is bounded by

$$\begin{aligned}&(2^{m_2} - (|S|+1)) \cdot i \cdot (2^{m_2} - (|S'|+1)) \cdots (2^{m_2} - (N-1)) \\ =&i \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)).\end{aligned}$$

Now, we see that, if $\textsf{que}_1 \notin S$ (combining both scenario 1 and scenario 2), then

$$|S_{\textsf{str-}(i+1)}| \leq (i+1) \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1)).$$

Again, $\textsf{res}_1$ is either in S or not in S. We have that

$$\begin{aligned}|S_{\textsf{str-}(i+1)}| &\leq \max\{|S_{\textsf{str-}i}|, (i+1) \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1))\} \\ &= (i+1) \cdot (2^{m_2} - (|S|+1)) \cdots (2^{m_2} - (N-1))\end{aligned}$$

By setting $\textsf{sk} := \textsf{sk}_A$ and $S := S_{\textsf{que-res}}$, we have that the probability that the adversary outputs $h \in Q_{\textsf{sk}_A} \setminus S_{\textsf{que-res}}$ is bounded by $\frac{O(q^2)}{2^{\frac{m_2 - m_1}{2}}} \leq \textsf{negl}(\lambda)$.

Case 4. It is trivial that

$$\Pr[\text{Case 4}] = \Pr[\text{Case 3}].$$

Combining together, we have that

$$\begin{aligned}\Pr[\mathcal{A}^{\mathcal{G}_{N,m_2}} \text{ outputs the valid shared key}] &\geq 1 - (6q+1)(\frac{1}{26q} + \textsf{negl}(\lambda)) \\ &\geq \frac{2}{3} - \textsf{negl}(\lambda).\end{aligned}$$

4 The Hierarchy of GGMs

In this section, we establish a hierarchy among GGMs, varying in distinct lengths of group encodings and prove that the shorter GGM is strictly stronger than the longer GGM. Specifically, we show that one can construct an indifferentiable longer generic group from a shorter one plus an additional independent random oracle, but the shorter generic group model is computationally indifferentiably separated from the longer generic group (when the gap between the lengths is sufficiently large).

4.1 $\mathcal{G}_{N,m_1}$ Statistically Implies $\mathcal{G}_{N,m_2}$

In this section, we show how to build an longer indifferentiable generic group model from a shorter one plus an additional independent ROM. Here are the building blocks:

- $\mathcal{G}_{N,m_1} := (\mathcal{G}^{\mathsf{label}}_{N,m_1}, \mathcal{G}^{\mathsf{add}}_{N,m_1})$ is a generic group model that maps $\mathbb{Z}_N$ to $\{0,1\}^{m_1}$;
- $\mathcal{E} : \{0,1\}^{m_2} \to \{0,1\}^{m_2}$ is a random permutation oracle with its inverse $\mathcal{E}^{-1}$.

For simplicity, we denote $\mathcal{O}$ as the tuple $(\mathcal{G}_{N,m_1}, (\mathcal{E}, \mathcal{E}^{-1}))$. The following is the construction $\Pi^{\mathcal{O}}_{\mathsf{L\text{-}GGM}} := (L^{\mathcal{O}}_{\mathsf{L\text{-}GGM}}, A^{\mathcal{O}}_{\mathsf{L\text{-}GGM}})$, depicted in Fig. 4. Correctness easily follows, and it rests to prove the indifferentiability. Formally,

```
Construction $\Pi^{\mathcal{O}}_{\mathsf{L\text{-}GGM}}$

$L^{\mathcal{O}}_{\mathsf{L\text{-}GGM}}(x)$:
    $\tilde{h} \leftarrow \mathcal{G}^{\mathsf{label}}_{N,m_1}(x)$;
    $h \leftarrow \mathcal{E}(\tilde{h}||0^{(m_2-m_1)})$;
    return $h$.

$A^{\mathcal{O}}_{\mathsf{L\text{-}GGM}}(h_1, h_2)$:
    if $\mathcal{E}^{-1}(h_1) \neq \tilde{h}_1||0^{(m_2-m_1)}$: return $\perp$;
    if $\mathcal{E}^{-1}(h_2) \neq \tilde{h}_2||0^{(m_2-m_1)}$: return $\perp$;
    if $\mathcal{G}^{\mathsf{add}}_{N,m_1}(\tilde{h}_1, \tilde{h}_2) = \perp$: return $\perp$;
    return $\mathcal{E}(\mathcal{G}^{\mathsf{add}}_{N,m_1}(\tilde{h}_1, \tilde{h}_2)||0^{(m_2-m_1)})$.
```

Fig. 4. The construction $\Pi^{\mathcal{O}}_{\mathsf{L\text{-}GGM}}$ in the $\mathcal{G}_{N,m_1}$ and RPM.

Theorem 5. *Let m_1, m_2 be two integers that $m_2 \geq m_1$. The scheme $\Pi^{\mathcal{O}}_{\mathsf{L\text{-}GGM}}$ in Fig. 4, with access to a generic group $\mathcal{G}_{N,m_1}$, a random permutation $\mathcal{E}$ and its inverse $\mathcal{E}^{-1}$, is indifferentiable from a generic group $\mathcal{G}_{N,m_2}$. More precisely, there exists a simulator $\mathcal{S}$ such that for all $(q_{\mathcal{G}^{\mathsf{label}}_{N,m_1}}, q_{\mathcal{G}^{\mathsf{add}}_{N,m_1}}, q_{\mathcal{E}}, q_{\mathcal{E}^{-1}})$-query distinguisher $\mathcal{D}$ with $q_{\mathcal{G}^{\mathsf{label}}_{N,m_1}} + q_{\mathcal{G}^{\mathsf{add}}_{N,m_1}} + q_{\mathcal{E}} + q_{\mathcal{E}^{-1}} \leq q$, we have*

$$\mathrm{Adv}^{\mathsf{indif}}_{\Pi^{\mathcal{O}}_{\mathsf{L\text{-}GGM}}, \mathcal{G}_{N,m_2}, \mathcal{S}, \mathcal{D}} \leq \frac{6q^2}{N} + \frac{10q^2 + 4q}{2^{m_1}} + \frac{3q}{2^{\lambda}} + \frac{2q}{2^{m_1} - 2q}.$$

The simulator makes at most $3q$ queries to $\mathcal{G}_{N,m_2}$.

Due to space limit, we leave the proof in the full version of this paper [ZJW+24].

4.2 $\mathcal{G}_{N,m_2}$ Does Not Computationally Imply $\mathcal{G}_{N,m_1}$

In this section, we show that the shorter GGM is computationally indifferentiably separated from the longer one. Formally,

Theorem 6. *Let λ be the security parameter. Let $\mathcal{G}_{N,m_1}$ and $\mathcal{G}_{N,m_2}$ be two generic group models. If $(m_2 - m_1) \geq \omega(\log \lambda)$, then $\mathcal{G}_{N,m_1}$ is computationally indifferentiably separated from $\mathcal{G}_{N,m_2}$.*

To prove it, we adopt the discrete logarithm identification (DLI) problem proposed by [ZZ23]. To absorb Zhang and Zhandry 's analysis into our setting, we propose the DLI problem w.r.t the shorter groups in the longer GGM. Below, we give the proof sketch of Theorem 6 and the formal proof can be found in the full version of this paper [ZJW+24].

Proof Sketch. Suppose $\Pi^{\mathcal{G}_{N,m_2}} := (L^{\mathcal{G}_{N,m_2}}, A^{\mathcal{G}_{N,m_2}})$ is indifferentiable from $\mathcal{G}_{N,m_1}$ in the longer GGM $\mathcal{G}_{N,m_2}$. The argument goes in three steps:

1. DLI w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$ is easy.
2. If $\Pi^{\mathcal{G}_{N,m_2}}$ is indifferentiable from $\mathcal{G}_{N,m_1}$ and DLI w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$ is easy, then DLI w.r.t. $\mathcal{G}_{N,m_1}$ is also easy.
3. Yet, DLI w.r.t. the generic group $\mathcal{G}_{N,m_1}$ is hard.

The above three steps draw a contradiction, so the statement "$\Pi^{\mathcal{G}_{N,m_2}}$ is indifferentiable from $\mathcal{G}_{N,m_1}$" cannot be true, completing our proof. Note that, Step 2 is already proven in [ZZ23]; and the proof of Step 3 is straightforward according to Definition 9 for indifferentiability. Due to the space limit, we skip them here. Below, we prove Step 1.

By the definition of indifferentiability, the algorithms $L^{\mathcal{G}_{N,m_2}}$ and $A^{\mathcal{G}_{N,m_2}}$ are deterministic; and they shall support group operations correctly with high probability. We stress that $L^{\mathcal{G}_{N,m_2}}$ only makes labeling queries. Let q be an integer in $\mathsf{poly}(\lambda)$. We assume that both $L^{\mathcal{G}_{N,m_2}}$ and $A^{\mathcal{G}_{N,m_2}}$ make at most q queries to $\mathcal{G}_{N,m_2}$. Next, we prove that the DLI problem w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$ is easy by constructing an efficient adversary $\mathcal{A}$ and a query-free circuit $C_{\mathsf{G\text{-}GGM}}$ in Fig. 5. (Here, G-GGM denotes the shorter group in the longer GGM.)

We first clarify some undefined notions in Fig. 5. Let n be a sufficiently large integer to be specified below. By $\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \overset{\mathsf{query}}{\longleftarrow} L^{\mathcal{G}_{N,m_2}}(r_i)$, we denote that on input r_i, the algorithm $L^{\mathcal{G}_{N,m_2}}(r_i)$ makes queries $(\mathsf{que}_1, \ldots, \mathsf{que}_q)$ to $\mathcal{G}_{N,m_2}$ and gets responses of $(\mathsf{res}_1, \ldots, \mathsf{res}_q)$; and similar for the notation $\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \overset{\mathsf{query}}{\longleftarrow} A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x - z), L^{\mathcal{G}_{N,m_2}}(z))$.[21] Given an input $z \in \mathbb{Z}_N$, the query-free circuit $C_{\mathsf{G\text{-}GGM}}$ runs algorithm $L^{\mathcal{G}_{N,m_2}}(z)$ except for replacing the querying oracle by looking up the table $S_{\mathsf{que\text{-}res}}$ (and lazy sampling); we denote that as $L^{S_{\mathsf{que\text{-}res}}}$.

We argue that the query-free circuit $C_{\mathsf{G\text{-}GGM}}$ in Fig. 5 identifies x with a good probability, which means DLI w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$ is easy. Note that, we say $C_{\mathsf{G\text{-}GGM}}$

[21] Here, we abuse the notation $L^{\mathcal{G}_{N,m_2}}(x-z)$ as both the group element and the labeling operation on $x - z$.

```
Adversary $\mathcal{A}^{\mathcal{G}_{N,m_2}}$

$\mathcal{A}^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x))$:
$S_{\mathsf{que\text{-}res}} \leftarrow \emptyset$; $z, r_1, \ldots, r_n \xleftarrow{\$} \mathbb{Z}_N$;
for $i = 1$ to $n$: //collecting frequent queries
    $\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\mathsf{query}} L^{\mathcal{G}_{N,m_2}}(r_i)$;
    $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\}$;
compute $L^{\mathcal{G}_{N,m_2}}(z)$, $L^{\mathcal{G}_{N,m_2}}(x-z) \leftarrow A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x), L^{\mathcal{G}_{N,m_2}}(-z))$;
$\{(\mathsf{que}_1, \mathsf{res}_1), \ldots, (\mathsf{que}_q, \mathsf{res}_q)\} \xleftarrow{\mathsf{query}} A^{\mathcal{G}_{N,m_2}}(L^{\mathcal{G}_{N,m_2}}(x-z), L^{\mathcal{G}_{N,m_2}}(z))$;
for $j = 1$ to $q$: //collecting queries in group addition
    if $\mathsf{que}_j$ is an addition query: //converting addition queries into labeling queries
        parse the addition query $\mathsf{que}_j$ to two labels $h_1, h_2$;
        if $\exists(x_1, h_1), (x_2, h_2) \in S_{\mathsf{que\text{-}res}}$: $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(x_1 + x_2, \mathsf{res}_j)\}$;
    else: $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(\mathsf{que}_j, \mathsf{res}_j)\}$; //collecting labeling queries
return $C_{\mathsf{G\text{-}GGM}}(\,\cdot\,, S_{\mathsf{que\text{-}res}}, L^{\mathcal{G}_{N,m_2}}(x))$.

$C_{\mathsf{G\text{-}GGM}}(\,\cdot\,, S_{\mathsf{que\text{-}res}}, L^{\mathcal{G}_{N,m_2}}(x))$:
take $z \in \mathbb{Z}_N$ as input; run $\mathsf{str} \leftarrow L^{S_{\mathsf{que\text{-}res}}}(z)$;
when $L$ makes a labeling query $\mathsf{que} = x$: //responding to labeling queries
    if $\exists(x, h) \in S_{\mathsf{que\text{-}res}}$: respond with $h$;
    else: respond with a uniformly sampled $h$; $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(x, h)\}$;
when $L$ makes an addition query $\mathsf{que} = (h_1, h_2)$: //responding to addition queries
    if $\exists(x_1, h_1), (x_2, h_2) \in S_{\mathsf{que\text{-}res}}$:
        if $\exists(x_1 + x_2, h) \in S_{\mathsf{que\text{-}res}}$: respond with $h$;
        else: respond with a uniformly sampled $h$; $S_{\mathsf{que\text{-}res}} \leftarrow S_{\mathsf{que\text{-}res}} \cup \{(x_1 + x_2, h)\}$;
    else: respond with $\perp$; //if addition query has a new labeling, then responds with $\perp$
if $\mathsf{str} = L^{\mathcal{G}_{N,m_2}}(x)$: return 1; else: return 0.
```

Fig. 5. Efficient Adversary $\mathcal{A}^{\mathcal{G}_{N,m_2}}$ and query-free circuit $C_{\mathsf{G\text{-}GGM}}$ w.r.t. $\Pi^{\mathcal{G}_{N,m_2}}$.

identifies x with a good probability if it satisfies following properties. Due to the space limit, we leave the proof in the full version of this paper [ZJW+24].

- $\Pr[C_{\mathsf{G\text{-}GGM}}(x) = 1] \geq \frac{2}{3}$;
- for any noticeable function ρ: $\Pr_{x' \neq x}[C_{\mathsf{G\text{-}GGM}}(x') = 1] \leq \rho$.

Acknowledgment. Cong Zhang was supported by the National Key Research and Development Program of China (Grant No. 2023YFB3106000). This work was also supported by Ant Group through CCF-Ant Research Fund (Grant No. CCF-AFSG RF20230308). Bingsheng Zhang was supported by the National Natural Science Foundation of China (Grant No. 62072401 and No. 62232002) and Input Output (iohk.io). Hong-Sheng Zhou was supported in part by NSF grant CNS-1801470 and a VCU Research Quest grant.

References

ABD+13. Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger.On the indifferentiability of key-alternating ciphers.In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 531–550. Springer, Heidelberg, August 2013.

Bar20. Elaine Barker. Recommendation for key management: Part 1 – general, 2020. https://doi.org/10.6028/NIST.SP.800-57pt1r5.

BF18. Manuel Barbosa and Pooya Farshim. Indifferentiable authenticated encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 187–220. Springer, Heidelberg, August 2018.

BKSY11. Zvika Brakerski, Jonathan Katz, Gil Segev, and Arkady Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions.In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 559–578. Springer, Heidelberg, March 2011.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

CDMP05. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function.In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, August 2005.

CDMS10. Jean-Sébastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A domain extender for the ideal cipher. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 273–289. Springer, Heidelberg, February 2010.

CHK+16. Jean-Sébastien Coron, Thomas Holenstein, Robin Künzler, Jacques Patarin, Yannick Seurin, and Stefano Tessaro.How to build an ideal cipher: The indifferentiability of the Feistel construction.*Journal of Cryptology*, 29(1):61–114, January 2016.

CMR+23. Lily Chen, Dustin Moody, Karen Randall, Andrew Regenscheid, and Angela Robinson. Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters, 2023. https://doi.org/10.6028/NIST.SP.800-186.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

DHH+21. Nico Döttling, Dominik Hartmann, Dennis Hofheinz, Eike Kiltz, Sven Schäge, and Bogdan Ursu.On the impossibility of purely algebraic signatures.In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 317–349. Springer, Heidelberg, November 2021.

DRS09. Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for practical applications. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 371–388. Springer, Heidelberg, April 2009.

DSSL16. Yevgeniy Dodis, Martijn Stam, John P. Steinberger, and Tianren Liu. Indifferentiability of confusion-diffusion networks. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 679–704. Springer, Heidelberg, May 2016.

ElG85. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

GMM17. Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. When does functional encryption imply obfuscation? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 82–115. Springer, Heidelberg, November 2017.

GWL23. Chun Guo, Lei Wang, and Dongdai Lin. Impossibility of indifferentiable iterated blockciphers from 3 or less primitive calls. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *LNCS*, pages 408–439. Springer, Heidelberg, April 2023.

HKT11. Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 89–98, New York, NY, USA, 2011. Association for Computing Machinery.

HMQS23. Mohammad Hajiabadi, Mohammad Mahmoody, Wei Qi, and Sara Sarfaraz. Lower bounds on assumptions behind registration-based encryption. In Guy Rothblum and Hoeteck Wee, editors, *Theory of Cryptography*, pages 306–334, Cham, 2023. Springer Nature Switzerland.

HR04. Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 92–105. Springer, Heidelberg, August 2004.

IR89. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

MM11. Takahiro Matsuda and Kanta Matsuura. On black-box separations among injective one-way functions. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 597–614. Springer, Heidelberg, March 2011.

MMN16. Mohammad Mahmoody, Ameer Mohammed, and Soheil Nematihaji. On the impossibility of virtual black-box obfuscation in idealized models. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 18–48. Springer, Heidelberg, January 2016.

MPZ20. Ueli Maurer, Christopher Portmann, and Jiamin Zhu. Unifying generic group models. Cryptology ePrint Archive, Report 2020/996, 2020. https://eprint.iacr.org/2020/996.

MRH04. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology.In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004.

Nec94. Vassiliy Ilyich Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

PH78. Stephen Pohlig and Martin Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (Corresp.). *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.

Pol78. John M Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, 1978.

PRV12. Periklis A. Papakonstantinou, Charles W. Rackoff, and Yevgeniy Vahlis. How powerful are the DDH hard groups? Cryptology ePrint Archive, Report 2012/653, 2012. https://eprint.iacr.org/2012/653.

RS08. Phillip Rogaway and John P. Steinberger. Constructing cryptographic hash functions from fixed-key blockciphers. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 433–450. Springer, Heidelberg, August 2008.

RSS11. Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework.In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.

RSS20. Lior Rotem, Gil Segev, and Ido Shahaf. Generic-group delay functions require hidden-order groups. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 155–180. Springer, Heidelberg, May 2020.

RTV04. Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 1–20. Springer, Heidelberg, February 2004.

SGS20. Gili Schul-Ganz and Gil Segev. Accumulators in (and beyond) generic groups: Non-trivial batch verification requires interaction.In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 77–107. Springer, Heidelberg, November 2020.

SGS21. Gili Schul-Ganz and Gil Segev. Generic-group identity-based encryption: A tight impossibility result. In *Information Theoretic Cryptography*, 2021.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

Zha22. Mark Zhandry. To label, or not to label (in generic groups).In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Heidelberg, August 2022.

ZJW+24. Cong Zhang, Keyu Ji, Taiyu Wang, Bingsheng Zhang, Hong-Sheng Zhou, Xin Wang, and Kui Ren. On the complexity of cryptographic groups and generic group models. In *Cryptology ePrint Archive, Paper 2024/1452*, 2024. https://eprint.iacr.org/2024/1452.

ZZ18. Mark Zhandry and Cong Zhang. Impossibility of order-revealing encryption in idealized models. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 129–158. Springer, Heidelberg, November 2018.

ZZ20. Mark Zhandry and Cong Zhang. Indifferentiability for public key cryptosystems. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 63–93. Springer, Heidelberg, August 2020.

ZZ23. Cong Zhang and Mark Zhandry. The relationship between idealized models under computationally bounded adversaries. In *ASIACRYPT 2023*, 2023.

Randomness in Private Sequential Stateless Protocols

Hari Krishnan P. Anilkumar[1(✉)], Varun Narayanan[2], Manoj Prabhakaran[3], and Vinod M. Prabhakaran[1]

[1] TIFR, Mumbai, India
{hari.a,vinodmp}@tifr.res.in
[2] UCLA, Los Angeles, USA
[3] IIT Bombay, Mumbai, India
mp@cse.iitb.ac.in

Abstract. A significant body of work in information-theoretic cryptography has been devoted to the fundamental problem of understanding the power of randomness in private computation. This has included both in-depth study of the randomness complexity of specific functions (e.g., Couteau and Rosén, ASIACRYPT 2022, gives an upper bound of 6 for n-party AND), and results for broad classes of functions (e.g., Kushilevitz et al. STOC 1996, gives an $O(1)$ upper bound for all functions with linear-sized circuits). In this work, we make further progress on both fronts by studying randomness complexity in a new simple model of secure computation called Private Sequential Stateless (PSS) model.

We show that functions with $O(1)$ randomness complexity in the PSS model are *exactly* those with constant-width branching programs, restricting to "speak-constant-times" protocols and to "read-constant-times" branching programs.

Towards this our main construction is a novel PSS protocol for "strongly regular branching programs" (SRBP). As we show, any constant-width branching program can be converted to a constant-width SRBP, yielding one side of our characterization. The converse direction uses ideas from Kushilevitz et al. to translate randomness to communication.

Our protocols are concretely efficient, has a simple structure, covers the broad class of functions with small-width, read-once (or read-a-few-times) branching programs, and hence may be of practical interest when 1-privacy is considered adequate. Also, as a consequence of our general result for SRBPs, we obtain an improvement over the protocol of Couteau and Rosén for ANDin certain cases—not in terms of the number of bits of randomness, but in terms of a simpler protocol structure (sequential, stateless).

Keywords: Randomness complexity · Private computation · Private Sequential Stateless model · lower bound · branching programs

Hari Krishnan P. Anilkumar and Vinod M. Prabhakaran were supported by Department of Atomic Energy, Government of India, under project no. RTI4001.
V. Narayanan was supported by NSF Grants CNS-2246355, CCF-2220450, and CNS-2001096.
M. Prabhakaran was supported by IIT Bombay Trust Lab and Algorand Centre of Excellence by Algorand Foundation.

1 Introduction

In information-theoretic cryptography, randomness is the key resource available for creating secrecy (unlike in computational cryptography, where computational hardness plays an equally important role). As such, understanding the amount of randomness needed for various tasks is a problem of great theoretical and practical importance in this area.

In this paper we study randomness complexity in the context of private multi-party computation (i.e., multi-party computation secure against honest-but-curious adversary). Not surprisingly, this fundamental question has received a great amount of attention in the literature [KR94, KOR96, KM96, KOR98, BSPV99, BGP99, GR03, JLR03, BGP07, DPP16, RU19, KOP+19, CR22, GIS22].

We shall focus on the most basic setting of 1-privacy, wherein the adversary can corrupt only a single party, which itself has been studied in depth. In one of the early works in this line, it was shown that for any function which has a linear sized circuit, the the randomness complexity of 1-private computation is constant, irrespective of the number of parties holding the inputs. Several subsequent works improved on the exact constant when considering a specific function, namely the n-party AND function [KOP+19, CR22].

We introduce a simplified model of multi-party computation called the **Private Sequential Stateless** (PSS) model that is intrinsically connected to randomness complexity. Various simplified models of MPC, like *Private Simultaneous Message* (PSM) model [FKN94, IK97], *Non-Interactive Secure Computation* (NISC) [IKO+11], *Conditional Disclosure of Secrets* (CDS) [GIKM98], etc., have proven influential in shaping our understanding of private computation and information-theoretic cryptography. We propose PSS in a similar spirit, but it deals with a different aspect of complexity. While PSM, NISC, CDS all impose a "star" topology of communication, PSS considers a *chain* model of communication. In PSS, given correlated randomness, the parties communicate in a (pre-determined) sequence from one party to the next; the parties do not retain any state between rounds, except for their own input and their share of the correlated randomness.

The randomness cost of a PSS protocol – namely, the number of random bits used to prepare the correlated randomness for the parties – is a crucial factor that determines whether a function has such a protocol or not. Indeed, without any restriction on the amount of randomness used, the restrictions in the PSS model can be subverted. [1] Our focus will be on PSS protocols which have a constant randomness cost for a family of functions (with variable input length), independent of the number of parties (each with a bit of the input). Also, unless

[1] Statelessness can be subverted by each party sending out an encryption (using one-time pads) of its state as part of the communication, which will be forwarded as it is until the party is again invoked (at which point it can update the contents of the state and re-encrypt using a new one-time pad). Once state is allowed, the sequentiality requirement can be subverted by letting a sender communicate to many parties one-by-one over several rounds.

otherwise specified, we shall also restrict to *speak-constant-times* PSS protocols, in which the number of times each party speaks is at most a constant.

Our main result is an *exact characterization* of boolean functions with *constant-randomness speak-constant-times PSS protocols* in terms of Branching Programs: they are exactly those functions which have *constant-width read-constant-times branching programs* (read-constant-times refers to the condition that each input is used at most a constant number of times in evaluating the branching program). Towards proving this result we present a PSS protocol for branching programs, and also, conversely, show how to convert a PSS protocol into a branching program, respecting the cost constraints.

Our contribution can alternately be viewed as developing concretely efficient PSS protocols for a large class of interesting functions. This class includes functions like AND, inner product, and any boolean function which can be computed by a streaming algorithm with constant memory and a constant number of passes over the input sequence. [2] The converse demonstrates an optimality of this result – that one could not have constructed such protocols for a wider class of functions.

Along the way to constructing our protocol, we identify a class of branching programs that we term *strongly regular branching programs* (SRBP). Strong regularity captures the technical conditions necessary for our protocol construction to be private. While some functions naturally have branching programs that are strongly regular, we observe that any branching program can be converted into an SRBP with a polynomial blow-up in the width (keeping the other size parameters intact). SRBPs may be of independent interest as they are a restriction of (a natural generalization of) "regular branching programs" as studied in [LPV23]. [3]

For the converse, we rely on a lemma from [KOR96], who proved a similar characterization of functions with constant randomness complexity for 1-private computation in terms of circuit complexity, and adapt it to the setting of PSS protocols.

Finally, as a related result of interest, we also obtain a lower bound for the 3-party AND function. While the best known lower bound result shows that 1 bit of randomness is insufficient [KOP+19], we show that at least 3 bits are necessary for computing this function (even without the sequentiality and memorylessness restrictions). Our result uses a reduction from a recently introduced problem called 3-secret sharing [ARN+23] instantiated for a suitable set of secrets, for which the exact randomness complexity was determined to be 3 bits.

1.1 Our Results

We briefly list our contributions, and expand on them below.

[2] Understanding the exact computational power of constant-width read-m-times branching programs is an interesting problem in its own right.

[3] The regular branching programs defined in [LPV23] can be termed 2-regular; the generalization referred to here allows d-regularity for any $d \geq 2$. Strong regularity imposes additional requirements on top of d-regularity. See Sect. 5 for more details.

- We introduce a simple model of private protocols with pre-processed correlated randomness, called *Private Sequential Stateless* (PSS) model.
- Our main result is to show a tight connection between functions computable using PSS and Branching Programs, in both directions.
- As an intermediate step, we identify a new model of branching programs called *Strongly Regular Branching Programs* (SRBP) which may be of independent interest.
- We show how our PSS protocols can be adapted to an "unassisted" setting without correlated randomness. While this result applies to a broad class of functions, applying it to the function AND (which has received a significant amount of attention in the literature) yields results matching or closely matching the state-of-the-art results [CR22], but with a simpler protocol structure.
- Continuing to focus on AND, we present a new lower bound on the randomness complexity of 3-party AND for 1-private computation (even with correlated randomness).

In a PSS protocol, the parties are deterministic and stateless, except for the correlated randomness and the input that they receive at the beginning. At every round a single pre-determined party receives a message and then sends a message in the next round to a single other party, without updating its state. The last message in the protocol is sent to a special output party who produces the final output (using its share of the correlated randomness along with the message it received). A PSS protocol is defined to be a 1-private protocol – i.e., with information theoretic security against semi-honest corruption of one party. Two costs of interest to us in a PSS protocol are the number of times any party speaks in the protocol and the amount of randomness used in the protocol, that is, the number of random bits used to prepare the correlated randomness for the parties.

We show a close connection between functions computable using Private Sequential Stateless and Branching Programs.

Theorem 1 (Main Result (informal)). *An n-input boolean function has a speak-constant-times constant-randomness PSS protocol iff it has a read-constant-times constant-width branching program.*

To construct a PSS protocol from a branching program, first we convert it, if necessary, into a *strongly regular branching program* (SRBP), with a polynomial blow-up in the width (keeping the other parameters intact). SRBP is a new definition we introduce, which may be of independent interest (see Sect. 2.3 and Sect. 5 for more details). The upper bound on randomness complexity that we obtain depends on the width of the strongly regular branching program and the number of times each input is read while evaluating the branching program – *but not on the number of inputs* or the length of the branching program.

To build a PSS protocol from SRBP, we start by considering a read-once strongly regular branching program (abbreviated as 1-SRBP), in which each input is used for only one transition in the branching program. We obtain the following result:

Theorem 2. *The randomness complexity of PSS computation of n-input boolean functions which have 1-SRBPs of width w, is $O(w \log w)$. This is achieved by a speak-once PSS protocol.*

To prove the theorem, we present a concretely efficient protocol, which requires sampling just 4 uniformly random permutations over $[w]$ and 2 uniform bits in addition. This is comparable to the state-of-the-art concrete parameters obtained in prior work on randomness complexity of 1-private computation for a specific function, namely AND (which has a width-2 1-SRBP). The prior work considered a model without the restrictions of PSS model, but also without external parties to supply correlated randomness and produce the final output. To fairly compare to those results (since additional parties without input *can* help save on randomness), we show how our PSS protocol can be modified so that the pre-processing computation and the final output computation can be carried out by two of the parties with inputs. As shown in Sect. 6, the modified PSS protocol we obtain for read-once AND uses 6 or 9 bits of randomness (depending on odd or even number of inputs).

We generalize Theorem 2 to functions with an k-SRBP, in which each input is used for at most k transitions in the branching program.

Theorem 3. *The randomness complexity of PSS computation of n-input boolean functions which have k-SRBPs of width w, is $O(kw \log w)$. This is achieved by a speak-$(2k-1)$-times PSS protocol.*

We remark that the $O(k)$ factor in the result above is in fact an upper bound for the chromatic number of a "conflict graph" associated with the branching program; for specific branching programs, this factor could be lower.

Next, we show that one cannot hope to improve this result significantly in terms of the functions covered. That is, we show that constant-randomness PSS protocols exist only for functions which are computable using constant-width branching programs.

Theorem 4. *Boolean functions with constant-randomness speak-k-times PSS protocols have constant-width read-k-times branching programs.*

Note that a sequential protocol with constant communication can be naturally translated to a constant-width branching program. When the protocol is 1-private, randomness cost can be translated to communication cost, as was first shown in [KOR96]. We adapt this result to the setting of PSS protocols to establish our result above.

Finally, on the lower bound front, we obtain the following result for the 3-party AND functionality.

Theorem 5. *Randomness complexity of 1-private computation of $\mathcal{F}^*_{\mathsf{AND}}$ for 3 parties is at least 3 bits.*

1.2 Related Work

As mentioned above, the fundamental question of randomness complexity of secure computation has received much attention. Among these, Kushilevitz, Ostrovsky and Rosén [KOR96] obtained a result analogous to our main result, but in the context of circuits and unrestricted protocols (as opposed to branching programs and PSS protocols).

A line of works have focused on upper bounding the randomness complexity of AND [KOR96,KOP+19,CR22]. Some of these protocols are in the PSS model (or rather, the unassisted PSS model, as defined in Sect. 6.1), and forms the basic motivation for studying this model. However, the state-of-the-art results in [CR22] are not (which, we show, can be attained in the unassisted PSS model, when the number of parties is odd).

Lower bound results for randomness complexity of 1-private computation have been fewer, with the notable exception of [KOP+19]. Our lower bound result relies on information-theoretic techniques from [DPP16,ARN+23].

Branching programs are a well-studied model of computation [INW94, Bar86]. In particular, a notion of regular branching programs studied in [LPV23] is closely related to the notion of strongly regular branching programs we introduce.

While we have focused on 1-privacy, the question of t-privacy has also been studied in the literature [KM96,CKOR00,GIS22]. We leave it for future work to study the power of t-Private Sequential Stateless model.

2 Technical Overview

We build 1-private sequential stateless protocols for n-party functions that are computable using a family of branching programs called strongly regular branching programs (SRBPs). We first construct a PSS protocol for read-once SRBP (1-SRBP). We will then elaborate on the notion of strong regularity, and present a conversion from a general branching program to strongly regular branching program with a polynomial blow-up in width. A more complex protocol is then presented that realizes PSS for general (read-m) SRBP. As an application of our main result, we construct a 1-private protocol with a limited communication pattern for standard n-party AND functionality by modifying the PSS protocol for AND. Our construction matches the best known randomness cost for AND for odd values of n.

2.1 Branching Programs and Private Sequential Stateless Protocols

A width-w and length-ℓ branching program for an n input (alternatively n-party) function $f : \{0,1\}^n \to \{0,1\}$ is described by a layer assignment function $\sigma : [\ell] \to [n]$ mapping each layer $i \in [\ell]$ to a party $\mathsf{P}_{\sigma(i)}$; for each layer $i \in [\ell]$, a pair of transition functions $g_0^{(i)}, g_1^{(i)} : [w] \to [w]$ one of which will be chosen according to the input of $\mathsf{P}_{\sigma(i)}$ to map the incoming state to layer i to its outgoing state; and an output function ϕ that maps the final state outgoing from layer ℓ to an output bit. The branching program computes an n-party function f if, for all $(x_1, \dots, x_n) \in \{0,1\}^n$, $f(x_1, \dots, x_n) = \phi(u_\ell)$, where, for $i \in [\ell]$, the outgoing state u_i from layer i (which is the incoming state for layer $i+1$) is defined as follows: $u_i = g^{(i)}_{x_{\sigma(i)}}(u_{i-1})$ and $u_0 = 1$.

If every party is assigned at most m layers, i.e., $|\sigma^{-1}(i)| \leq m$ for each i, the branching program is said to be read-m. A branching program is said to be read-once if $\ell = n$ and σ is the identity function (without loss of generality).

In this paper, we draw a connection between branching programs and a simplified model of secure multi-party computation called the private sequential stateless (PSS) protocols. A PSS protocol $\pi = (\mathsf{Prep}_\pi, \varsigma_\pi, \mathsf{Next}_\pi, \mathsf{Out}_\pi)$ is a semi-honest 1-private protocol in the correlated randomness setting with sequential communication and stateless computation. A set of parties $\mathsf{P}_i, i \in [n]$, each P_i holding an input bit x_i want to deliver a boolean $f(x_1, \dots, x_n)$ to an external party P_{n+1} who has no input. π starts off with a preprocessing step Prep_π in which a trusted party samples correlated randomness $(r_1, \dots, r_{n+1})$ (independent of the inputs), and delivers r_i to each $\mathsf{P}_i, i \in [n]$ and r_{n+1} to P_{n+1}. The protocol then proceeds sequentially for T rounds, with $\mathsf{P}_{\varsigma_\pi(t)}$ sending a message m_t to $\mathsf{P}_{\varsigma_\pi(t+1)}$ in round $t \in [T]$. The party $\mathsf{P}_{\varsigma_\pi(t)}$ computes m_t using a stateless next message function (corresponding to round t) that takes the message m_{t-1} the party received in round-$t-1$, the party's own input, and the randomness it received during preprocessing. The receiver of the message in round T is necessarily the output party P_{n+1} who computes the output as $\mathsf{Out}_\pi(m_T, r_{n+1})$.

Our first result shows that if f is privately computed by a speak-m-times PSS protocol (where each party speaks in at most m rounds) and using constant randomness, then f is computable using a read-m branching program of constant width independent of the number of parties and rounds. Using an argument along the lines of [KOR96], we first show that for any fixing of the randomness chosen during preprocessing, the number of possible messages received in any round in the now deterministic protocol while ranging over all possible inputs is at most $2^{\rho+2}$ irrespective of the number of rounds, when ρ is the randomness cost of π. Thus, when the randomness is fixed arbitrarily, there is a map η_{t-1} that maps the set of all possible messages m_{t-1} received by the speaker of any round t to the set $[2^{\rho+2}]$. For any round t, with $\mathsf{P}_{\varsigma_\pi(t)}$ as speaker, we can set the transition function $g^{(t)}_0 : [2^{\rho+2}] \rightarrow [2^{\rho+2}]$ as a translation of the next message function invoked with 0 as user's input (and randomness fixed) induced when m_{t-1} and m_t are mapped to $[2^{\rho+2}]$ by η_{t-1} and η_t, respectively; $g^{(t)}_1$ is defined similarly. Finally, the output function for the branching program is a translation of the output function of that in π induced by the mapping of m_T to $[2^{\rho+2}]$. The resulting branching program has width $2^{\rho+2}$ and length T (number of rounds in π), and the layer assignment function is the same as the speaker assignment function of π.

2.2 A Protocol Idea for Branching Programs

Let $f : \{0,1\}^n \to \{0,1\}$ be an n-party function computable using a *read-once* branching program of width w with transition functions $g_0^{(i)}, g_1^{(i)} : [w] \to [w]$ for $i \in [n]$, and sets $S_0, S_1 \subset [w]$ such that, for any $x_1, \ldots, x_n \in \{0,1\}$,

$$g_{x_n}^{(n)} \circ g_{x_{n-1}}^{(n-1)} \circ \ldots \circ g_{x_1}^{(1)}(1) \in \begin{cases} S_0 \text{ if } f(x_1, \ldots, x_n) = 0 \\ S_1 \text{ if } f(x_1, \ldots, x_n) = 1 \end{cases} .$$

A *non-private* sequential protocol for f can be built as follows: P_1 sends the message $u_1 = g_{x_1}^{(1)}(1)$ to P_2, P_2 sends $u_2 = g_{x_2}^{(2)}(u_1) = g_{x_2}^{(2)} \circ g_{x_1}^{(1)}(1)$ to P_3 and so on with P_i sending $u_i = g_{x_i}^{(i)}(u_{i-1}) = g_{x_i}^{(i)} \circ g_{x_{i-1}}^{(i-1)} \circ \ldots \circ g_{x_1}^{(1)}(1)$ to P_{i+1}. Party P_n, who receives u_{n-1}, sends $u_n = g_{x_n}^{(n)}(u_{n-1}) = g_{x_n}^{(n)} \circ g_{x_{n-1}}^{(n-1)} \circ \ldots \circ g_{x_1}^{(1)}(1)$ to P_{n+1}, who outputs $b \in \{0,1\}$ if $u_n \in S_b$.

The above non-private protocol admits a straightforward conversion to a PSS protocol: we will arrange each P_i to compute and forward a random permutation of u_i, say $\alpha_i(u_i)$, where α_i is a random permutation, in the place of u_i. For this, the (descriptions of) functions $(\hat{g}_0^{(1)}, \hat{g}_1^{(1)}) = (\alpha_1 \circ g_0^{(1)}, \alpha_1 \circ g_1^{(1)})$ are sent to P_1 during the preprocessing phase, and $(\hat{g}_0^{(i)}, \hat{g}_1^{(i)}) = (\alpha_i \circ g_0^{(i)} \circ \alpha_{i-1}^{-1}, \alpha_i \circ g_1^{(i)} \circ \alpha_{i-1}^{-1})$ are sent to P_i for each $2 \le i \le n$. When $(\hat{g}^{(i)}, \hat{g}^{(i)})$ is replaced with $(g^{(i)}, g^{(i)})$ in the aforementioned non-private protocol, each $\mathsf{P}_i, i > 1$ receives $\alpha_{i-1}(u_{i-1})$ instead of u_i, which hides u_{i-1} as long as α_{i-1} is sampled independent of α_i. Finally, P_{n+1} who receives $\alpha_n(u_n)$ from P_n, is also sent $\{\alpha_n(j) : j \in S_0\}$ and $\{\alpha_n(j) : j \in S_1\}$ during preprocessing. Now, P_{n+1} can check whether u_n belongs to S_0 or S_1, allowing it to decode $f(x_1, \ldots, x_n)$. In the process, P_{n+1} only learns whether u_n belongs to S_0 or S_1 since α_n is a random permutation unknown to P_{n+1}. Since 1-privacy is ensured as long as each α_i is uniformly random and independent of α_{i-1}, we can set $\alpha_i = \alpha_{\mathsf{odd}}$ for all odd i, and $\alpha_i = \alpha_{\mathsf{even}}$ for all even i, where $\alpha_{\mathsf{odd}}, \alpha_{\mathsf{even}}$ are randomly sampled from $\mathrm{Sym}(w)$: the set of all permutations of $[w]$. Thus, preprocessing uses a randomness domain of size $2 \log w!$.

The approach sketched above can be shown to be 1-private if f is computed using a read-once *permutation* branching program where $g_b^{(i)}$ is a one-to-one function for each $i \in [n]$ and $b \in \{0,1\}$. However, it fails to be 1-private when $\{g_b^{(i)}\}$ are not all one-to-one. For instance, consider the n-party AND function, computable using a read-once branching program of width 2 such that, for each $i \in [n]$, $g_1^{(i)} : b \mapsto b$ and $g_0^{(i)} : b \mapsto 0$. P_i receives $\alpha_{i-1}(u_{i-1})$ from P_{i-1}, and $(\alpha_i \circ g_0^{(i)} \circ \alpha_{i-1}^{-1}, \alpha_i \circ g_1^{(i)} \circ \alpha_{i-1}^{-1})$ during preprocessing. Since $g_0^{(i)}$ maps all inputs to 0, P_i can learn $\alpha_i(0)$ from $\alpha_i \circ g_0^{(i)} \circ \alpha_{i-1}^{-1}$ which further reveals α_i. But then, α_i and $\alpha_i \circ g_1^{(i)} \circ \alpha_{i-1}^{-1}$ can be used to learn α_{i-1} since $g_1^{(i)}$ is the identity function. Thus P_i can recover $u_{i-1} = x_1 \cdot \ldots \cdot x_{i-1}$ from $\alpha_{i-1}(u_{i-1})$, breaking security of the protocol.

To get around this, the preprocessing step samples a random bit r_i and permutations $\alpha_{i,0}, \alpha_{i,1}$ for each $i \in [n]$, and sets $\alpha_{0,0}$ and $\alpha_{0,1}$ to be the identity function. To each P_i, it sends r_i, and the following functions:

$$\hat{g}^{(i)}_{0,0} = \alpha_{i,0} \circ g^{(i)}_0 \circ \alpha^{-1}_{i-1,r_{i-1}} \qquad \hat{g}^{(i)}_{0,1} = \alpha_{i,0} \circ g^{(i)}_0 \circ \alpha^{-1}_{i-1,r_{i-1}\oplus 1} \tag{1}$$

$$\hat{g}^{(i)}_{1,0} = \alpha_{i,1} \circ g^{(i)}_1 \circ \alpha^{-1}_{i-1,r_{i-1}} \qquad \hat{g}^{(i)}_{1,1} = \alpha_{i,1} \circ g^{(i)}_1 \circ \alpha^{-1}_{i-1,r_{i-1}\oplus 1} \tag{2}$$

In the protocol, we will ensure the invariant $\alpha_{i,x_i}(u_i) = \alpha_{i,x_i} \circ g^{(i)}_{x_i}(u_{i-1})$. For this, P_1 sends $x_1 \oplus r_1$ and $\alpha_{1,x_1} \circ g^{(1)}_{x_1}(0) = \alpha_{1,x_1}(u_1)$ to P_2. To propagate the invariant, it suffices to show that P_i can compute $x_i \oplus r_i$ and $\alpha_{i,x_i}(u_i) = \alpha_{i,x_i} \circ g^{(i)}_{x_i}(u_{i-1})$, assuming P_{i-1} sends $x_{i-1} \oplus r_{i-1}$ and $\alpha_{i-1,x_{i-1}}(u_{i-1})$ to P_i. Observe that

$$\hat{g}^{(i)}_{x_i,x_{i-1}\oplus r_{i-1}} = \alpha_{i,x_i} \circ g^{(i)}_{x_i} \circ \alpha^{-1}_{i-1,x_{i-1}}.$$

Hence, using $\hat{g}^{(i)}_{x_i,x_{i-1}\oplus r_{i-1}}$, r_i, $x_{i-1} \oplus r_{i-1}$ and $\alpha_{i-1,x_{i-1}}(u_{i-1})$, P_i can compute $x_i \oplus r_i$ and

$$\begin{aligned}\hat{g}^{(i)}_{x_i,x_{i-1}\oplus r_{i-1}}(\alpha_{i-1,x_{i-1}}(u_{i-1})) &= \alpha_{i,x_i} \circ g^{(i)}_{x_i} \circ \alpha^{-1}_{i-1,x_{i-1}} \circ \alpha_{i-1,x_{i-1}}(u_{i-1}) \\ &= \alpha_{i,x_i} \circ g^{(i)}_{x_i}(u_{i-1}) = \alpha_{i,x_i}(u_i)\end{aligned}$$

preserving the invariant. To allow computing the output, during preprocessing, P_{n+1} is sent

$$\hat{S}_{c,0} = \{\alpha_{n,r_n\oplus c}(j) : j \in S_0\}, \qquad \hat{S}_{c,1} = \{\alpha_{n,r_n\oplus c}(j) : j \in S_1\}, \quad c \in \{0,1\}.$$

P_{n+1} outputs b satisfying $u_n \in \hat{S}_{x_n\oplus r_n,b}$, on receiving $\alpha_{n,x_n}(u_n)$ and $x_n \oplus r_n$ from P_n. It is easy to verify that P_{n+1} outputs b such that $u_n \in S_b$.

Security against P_1 and P_{n+1} are straight-forward to argue: the former does not receive any message, and the view of the latter can be easily simulated using the output of the function. For $1 < i \le n$, P_i receives $x_{i-1} \oplus r_{i-1}$ and $\alpha_{i-1,x_{i-1}}(u_{i-1})$ from P_{i-1}, and r_i and $\{\hat{g}^{(i)}\}_{c,c'\in\{0,1\}}$. Although $x_{i-1} \oplus r_{i-1}, \alpha_{i-1,x_{i-1}}(u_{i-1})$, and r_i do not reveal any information to P_i, taken together with $\{\hat{g}^{(i)}_{c,c'}\}_{c,c'\in\{0,1\}}$, these random variables can indeed break security for certain branching programs even when $\alpha_{i,b}$ is chosen uniformly and independently for all $i \in [n]$ and $b \in \{0,1\}$. Using a careful analysis, we characterize the family of branching programs for which the protocol remain perfectly private. We refer to this family as strongly regular branching programs (discussed in the next section). We further show that 1-privacy is maintained even when randomness is reused as follows: set $(\alpha_{i,0}, \alpha_{i,1}, r_i) = (\alpha_{\mathsf{odd},0}, \alpha_{\mathsf{odd},1}, r_{\mathsf{odd}})$ for all odd i, and $(\alpha_{i,0}, \alpha_{i,1}, r_i) = (\alpha_{\mathsf{even},0}, \alpha_{\mathsf{even},1}, r_{\mathsf{even}})$ for all even i, where $\alpha_{\mathsf{odd},0}, \alpha_{\mathsf{odd},1}, \alpha_{\mathsf{even},0}, \alpha_{\mathsf{even},1}$ are randomly sampled from $\mathrm{Sym}(w)$ and $r_{\mathsf{odd}}, r_{\mathsf{even}}$ are random bits. Thus, the protocol used a randomness domain of $4 \log w! + 2$ to carry out the private computation.

2.3 Strongly Regular Branching Programs

For the PSS protocol outlined in the previous section (and its generalization to read-m branching programs) to achieve 1-security requires the branching program to satisfy a technical condition we call strong regularity. In this section, we will briefly describe this condition and sketch the intuition behind a construction that converts *any* branching program to a strongly regular branching program while incurring a quadratic blow-up in the width, but preserving the length of the branching program.

A branching program is said to be strongly regular if the pair of transition functions $g_0^{(i)}, g_1^{(i)} : [w] \to [w]$ are strongly regular for every layer i in the program. Strong regularity of $g_0^{(i)}, g_1^{(i)}$ requires that the preimages of $g_0^{(i)}$ form a partition of $[w]$ into sets of equal size (ignoring empty pre-images); similarly for $g_1^{(i)}$. Further, the intersection of these partitions created by $g_0^{(i)}$ and $g_1^{(i)}$ is also a partition into sets of equal size, say d (again ignoring empty intersections). Strong regularity requires an additional technical condition: For this define a bipartite graph H with the same set $[w]$ as both left and right vertices. There is an edge between a left vertex u and a right vertex v if the preimage of u under $g_0^{(i)}$ intersects the preimage of v under $g_0^{(i)}$ (this intersection is of size d by previous conditions). The final condition for strong regularity demands that a random automorphism of H maps every edge to a uniformly random edge in H. Here, by an automorphism of of H, we mean any permutation of the left and right vertices of H under which every edge is mapped to some edge of H. The security of our protocols depend crucially on the strong regularity of transition functions, and hence that of the branching program.

In Theorem 7, we show how to transform an arbitrary branching program into a strongly regular one while scaling the width from w to w^2. Leaving the layer assignment function σ unchanged, we define new transition functions $\{h_b^{(i)}\}_{i\in[\ell],b\in\{0,1\}}$ and output function ϕ' as follows. Given a pair of transition functions $g_0^{(i)}, g_1^{(i)} : [w] \to [w]$, we construct functions $h_0^{(i)}, h_1^{(i)} : [w]\times[w] \to [w]\times[w]$ as follows: assign $h_b^{(i)}(u,v) = (u, g_b^{(i)}(u))$ if i is odd and $h_b^{(i)}(u,v) = (g_b^{(i)}(v), v)$ if i is even for all $(u,v) \in [w]^2$, $b \in \{0,1\}$. Thus, for odd i (the case of even i is similar), every node u in $[w]$ is replaced by a set of w nodes $(u,1),\ldots,(u,w)$ and $h_b^{(i)}$ maps all of them to $(u, g_b^{(i)}(u))$. This implies strong regularity as

1. for each $u \in [w]$, the nodes $(u,1),\ldots,(u,w)$ in the domain of $h_b^{(i)}$ are all mapped to $(u, g_b^{(i)}(u))$, while all the other nodes of the form $(u,v), v \neq g_b^{(i)}(u))$ in the co-domain have an empty pre-image; therefore $|(h_b^{(i)})^{-1}(u,v)|$ is either w or 0,
2. for $u', v', u'', v'' \in [w]$, the pre-images of (u',v') under $h_0^{(i)}$ and (u'',v'') under $h_1^{(i)}$ have a non-empty intersection if and only if $u' = u''$, and $v' = v'' = g_0^{(i)}(u') = g_1^{(i)}(u'')$, and in such a case, their pre-images are both $\{(u',1),\ldots,(u',w)\}$; hence, the size of the intersection of pre-images of (u',v') and (u'',v'') is either w or 0, and
3. the H graph consists of an edge between $(u, g_0^{(i)}(u))$ and $(u, g_1^{(i)}(u))$ for each $u \in [w]$; therefore, the edge set E consists of w edges where no two of them have any common vertices, which implies that every edge is mapped to every edge with the same probability under a uniformly chosen $\mathrm{Aut}(H)$.

Furthermore, note that if (u_0, v_0) is the initial state of the program, then, for odd i,

$$(u_i, v_i) = (u_{i-1}, g^{(i)}_{x_{\sigma(i)}} \circ g^{(i-1)}_{x_{\sigma(i-1)}} \circ \cdots \circ g^{(1)}_{x_{\sigma(1)}}(u_0)),$$

and for even i,

$$(u_i, v_i) = (g^{(i)}_{x_{\sigma(i)}} \circ g^{(i-1)}_{x_{\sigma(i-1)}} \circ \cdots \circ g^{(1)}_{x_{\sigma(1)}}(u_0), v_{i-1}).$$

Hence, defining the new output function as $\phi'(u, v) = \phi(v)$ for odd ℓ and $\phi'(u, v) = \phi(u)$ for even ℓ ensures that the output is identical to that of the original branching program. Thus, for a function $f : \{0,1\}^n \to \{0,1\}$, given a length-$\ell$ and width-w branching program, we have a length-ℓ and width-w^2 strongly regular branching program.

2.4 Beyond Read-Once Branching Programs

We generalize our construction for read-once strongly regular branching programs to accommodate strongly regular branching programs in which a party may have inputs at multiple layers. Suppose $f : \{0,1\}^n \to \{0,1\}$ computable using a strongly regular branching program of width w and length ℓ. That is, there exist $\sigma : [\ell] \to [n]$; for each $t \in [\ell]$, a pair of functions $g^{(t)}_0, g^{(t)}_1 : [w] \to [w]$; and sets $S_0, S_1 \subset [w]$ such that for $(x_1, \ldots, x_n) \in \{0,1\}^n$,

$$g^{(\ell)}_{x_{\sigma(\ell)}} \circ \ldots \circ g^{(2)}_{x_{\sigma(2)}} \circ g^{(1)}_{x_{\sigma(1)}}(1) \in \begin{cases} S_0 \text{ if } f(x_1, \ldots, x_n) = 0, \\ S_1 \text{ if } f(x_1, \ldots, x_n) = 1. \end{cases}$$

The straightforward extension of the protocol sketched in Sect. 2.2 (without randomness reuse) continues to private for read-m strongly regular branching programs. That is, during preprocessing, $(\alpha_{t,0}, \alpha_{t,1})$ and $(r_{t,0}, r_{t,1})$ are sampled independently and uniformly at random for each $t \in [\ell]$, and the corresponding party $\mathsf{P}_{\sigma(t)}$ receives 'masked functions' $\{\hat{g}^{(i)}_{b,b'}\}_{b,b'} \in \{0,1\}$, as defined in Eq. (1). To aid in computing the output, P_{n+1} is sent

$$\hat{S}_{c,0} = \{\alpha_{\ell, r_\ell \oplus c}(j) : j \in S_0\}, \qquad \hat{S}_{c,1} = \{\alpha_{\ell, r_\ell \oplus c}(j) : j \in S_1\}, \quad c \in \{0,1\}.$$

This results in a protocol where the randomness cost grows with ℓ, the length of the branching program. In the read once case, we could bring the randomness cost down by using the same permutations for all odd parties, and a independently sampled set of permutations for even parties. Reusing randomness in this manner will break security if a party P_i appears more than once in the branching program.

We next describe how to reuse randomness for the read-m case. Let $z_t = x_{\sigma(t)}$ be the input of $\mathsf{P}_{\sigma(t)}$ at layer $t \in [\ell]$. We already observed that the view of $\mathsf{P}_{\sigma(t)}$ consisting of $z_{t-1} \oplus r_{t-1}, \alpha_{t-1, z_{t-1}}(u_{t-1})$, and $\{\hat{g}^{(t)}_{b,b'}\}$ is private. Hence, to ensure privacy against P_i in the protocol, it suffices to ensure that,

$\{\alpha_{t-1,0}, \alpha_{t-1,1}, \alpha_{t,0}, \alpha_{t,0} : t \in [\ell], \sigma(t) = i\}$ are sampled uniformly and independently from $\mathrm{Sym}(w)$, and $\{r_{t-1}, r_t : t \in [\ell], \sigma(t) = i\}$ are sampled independently from $\{0, 1\}$.

To satisfy these conditions, define a conflict graph $G = ([\ell], E)$ where $\{t, t'\} \in E$ if $t \neq t' \in [\ell]$ and $\{\sigma(t), \sigma(t+1)\} \cap \{\sigma(t'), \sigma(t'+1)\} \neq \emptyset$. Let $\mathsf{col} : [\ell] \to [\chi]$ be an optimal vertex coloring of G. Sample $\alpha_{j,0}$ and $\alpha_{j,1}$ uniformly from $\mathrm{Sym}(w)$, and r_j uniformly from $\{0, 1\}$ for each $j \in [\chi]$ and set

$$\alpha_{t,0} = \alpha_{\mathsf{col}(t),0} \quad \alpha_{t,1} = \alpha_{\mathsf{col}(t),1} \quad r_t = r_{\mathsf{col}(t)}, \quad t \in [\ell].$$

Such an assignment satisfies the constraints given above by the construction of the conflict graph, ensuring 1-privacy. The randomness cost of such a protocol is $O(\chi) \cdot w!$.

2.5 Private Computation of AND

As an application of our positive result, we modify our PSS protocol construction to realize with 1-privacy the AND functionality which takes a bit from each of the n parties and delivers their product to all parties. The resulting protocol achieves AND computation with 6 bits of randomness when n is odd, matching the best randomness upper bound in the literature [CR22]. When n is even we get a randomness cost of 9 bits. In this section, we outline the modified construction. To complement this upper bound, we also show that 1-private computation of AND functionality among 3 parties requires at least 3 bits of randomness even while employing non-sequential protocols. The previously best known lower bound [KOP+19] result is that 1 bit of randomness is insufficient (for any number of parties).

Our protocol for n-party AND in the PSS model consumes 6 bits of randomness. We convert this into the standard model by getting rid of the preprocessing step, and internalizing the output party P_{n+1}. For odd values of n, we effect this transformation without requiring any extra randomness; whereas, for even n, our transformation consumes 3 more bits, resulting in 9 bits of randomness. The preprocessing step can be removed from the PSS protocol for AND (or any read-once branching program in general) by letting P_1 sample and deliver the randomness supplied in the preprocessing step. Since P_1 does not receive messages during in the online step, this change does not affect security against P_1. To remove the output party P_{n+1}, we transfer the role of P_{n+1} to P_2, by redirecting the P_n's messages $\alpha_{n,x_n \oplus r_n}(u_n)$ and $x_n \oplus r_n$ to P_2 instead of P_{n+1}, and also redirecting the randomness used for computing the output to P_2. To ensure privacy against P_2 despite these extra messages, $\alpha_{n,0}, \alpha_{n,1}$ and r_n are sampled using 3 bits of fresh randomness, and the randomness needed for computing the output is sampled appropriately. When n is odd, we observe that some randomness can be recycled avoiding the need of fresh randomness for sampling $\alpha_{n,0}, \alpha_{n,1}$ and r_n, and maintaining randomness cost of 6 bits[4]. Finally, P_2 distributes the decoded output to all the parties.

[4] An involved construction can bring down the randomness cost for even values of n from 9 bits to $6 + \log 3$ bits. We do not present this construction in this work.

The protocol obtained by this transformation preserves the sequentiality except in the initial step where P_1 sends correlated randomness to all parties, and in the last step where P_2 delivers output to all parties; we refer to such a protocol as an unassisted PSS protocol or simply a uPSS protocol. We note that this results in a much sparser communication pattern compared to some of the protocols in the literature with low randomness cost [KOP+19, CR22].

Our lower bound of 3 bits for 3-party AND is obtained using a lower bound for the so-called 3-Secret Sharing (3SS) problem, recently presented in [ARN+23]. In a 3SS for secret domain M, the dealer, with input $(m_1, m_2, m_3) \in M$ wants to compute shares $s_{\{1,2\}}, s_{\{2,3\}}, s_{\{1,3\}}$ such that for any distinct $i, j, k \in [3]$, $s_{\{i,j\}}$ and $s_{\{i,k\}}$ form a secret sharing of m_i. We show that, in any 3-party 1-private AND protocol, the transcripts $T_{\{i,j\}}$ and $T_{\{i,k\}}$ between P_i and P_j, and between P_i and P_k, respectively, form a secret sharing of x_i, the input of P_i, for any distinct $i, j, k \in [3]$. Consequently, for the secret domain $D = (x_1, x_2, x_3) \in \{0,1\}^3 \setminus \{(1,1,1)\}$, the transcripts $\{T_{\{i,j\}}\}$ of the AND protocol with input (m_1, m_2, m_3) forms a 3SS of $(m_1, m_2, m_3) \in D$. At this point, we invoke the fact that randomness complexity of 3SS for D is 3 bits to obtain the desired lower bound for AND computation.

3 Preliminaries

We use the standard notion of 1-private computation and randomness complexity associated with it. By default, we shall use a model with correlated randomness generated during a pre-processing phase, and no other randomness, as this is the setting we shall use in Sect. 4 and Sect. 5; however, local (uncorrelated) randomness can be modeled as a special case of this.

For the sake of being self-contained, we summarize the standard protocol model below, with notation that will be convenient for us. For our purposes, a T-round protocol π (over private channels) with n input-parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ and an output party P_{n+1}, is specified by a correlated-randomness generation function Prep_π, a deterministic next message function Next_π, and output function Out_π which behave as follows in an execution of the protocol. A random element $R \leftarrow \mathcal{R}$ is sampled first, where $\mathcal{R}$ is a finite set representing the randomness space of the protocol. Prep_π, on input (i, R) where $i \in [n+1]$, outputs a string R_i (corresponding to the share of correlated-randomness for party P_i). Next_π takes as input $(i, \mathsf{View}^{(i)}_{\pi,t})$, where $i \in [n]$ is an index, and $\mathsf{View}^{(i)}_{\pi,t}$ (for $0 \leq t < T$) is the *view* of P_i in t rounds – consisting of its input, the string R_i (obtained from Prep_π), and all the messages received from all the other parties till then – and outputs a set of messages for P_i to send to all the other parties in round $t+1$. Out_π takes as input $(i, \mathsf{View}^{(i)}_{\pi,T})$ and produces an output for party P_i. We define the random variables $\mathsf{View}^{(i)}_\pi(x_1, \ldots, x_n)$ and $\pi(x_1, \ldots, x_n)$ to be, respectively, the view of P_i in a complete execution of π with parties using inputs $(x_1, \ldots, x_n)$, and the outputs produced by the parties at the end of such an execution.

An n-party functionality $\mathcal{F}$ is simply a function that takes n inputs, one from each party, and deterministically produces n outputs, one for each party.

We say that $\pi = (\mathsf{Prep}_\pi, \mathsf{Next}_\pi)$ is a 1-private realization of the functionality $\mathcal{F}$ (or simply, π is a protocol for $\mathcal{F}$) if the following conditions hold:

- **Correctness.** For any set of inputs $(x_1, \ldots, x_n)$, $\Pr[\pi(x_1, \ldots, x_n) = \mathcal{F}(x_1, \ldots, x_n)] = 1$ (where the probability is over the random input $R \in \mathcal{R}$ given to Prep_π).
- **1-Privacy.** For any $i \in [n]$ and any two sets of inputs $(x_1, \ldots, x_n)$ and $(x'_1, \ldots, x'_n)$ such that $x_i = x'_i$ and the i^{th} output of $\mathcal{F}$ on both are equal, $\mathsf{View}_\pi^{(i)}(x_1, \ldots, x_n)$ and $\mathsf{View}_\pi^{(i)}(x'_1, \ldots, x'_n)$ are identically distributed.

For a function $f : \{0,1\}^n \to \{0,1\}$, we define an $(n+1)$-party functionality $\mathcal{F}_f$, which takes an input bit x_i from party P_i for $i \in [n]$ (and empty input from P_{n+1}) and outputs $f(x_1, \ldots, x_n)$ to party P_{n+1} (and empty output to the other parties).

Branching Programs.

Definition 1. *A width w and length ℓ branching program for a function $f : \{0,1\}^n \to \{0,1\}$ is a collection of functions $(\sigma, \{g_b^{(t)}\}_{t \in [\ell], b \in \{0,1\}}, \phi)$ where $\sigma : [\ell] \to [n]$ encodes the order in which inputs are accessed, $g_b^{(t)} : [w] \to [w]$ denotes the transition function for each choice bit b and $t \in [\ell]$, and $\phi : [w] \to \{0,1\}$ denotes the output function, such that for all $(x_1, \ldots, x_n) \in \mathcal{C}_1, \times \ldots, \times \mathcal{C}_n$, $f(x_1, \ldots, x_n) = \phi(u_\ell)$, where u_i is defined as follows: $u_0 = 1$ and for $t \geq 1$, $u_t = g_{x_{\sigma(t)}}^{(t)}(u_{t-1})$.*

We shall refer to a length ℓ branching program as having ℓ *layers*. σ in the above definition is said to be the *input label function*, which maps each layer to an input index. Also, given a branching program as above and an input $(x_1, \ldots, x_n)$ for it, we shall refer to $x_{\sigma(t)}$ as the *choice bit* at layer t. We shall also be interested in a natural complexity measure of a branching program (apart from width and length), namely the number of layers at which the same input is used: we say that a branching program is a read-k-times branching program if for all $i \in [n]$, $|\{t : \sigma(t) = i\}| \leq k$. By default, all the branching programs we consider, unless otherwise specified, are read-constant-times branching programs; note that in this case the length $\ell = O(n)$.

4 Private Sequential Stateless Protocols

In this section we define the PSS model and further show that constant - randomness speak-constant-times PSS protocols imply constant-width read-constant-times branching programs.

A Private Sequential Stateless protocol is a 1-private protocol with certain restrictions on its communication pattern (sequential) and computation (stateless). Below we define a PSS protocol π in terms of functions $(\mathsf{Prep}_\pi, \varsigma_\pi, \mathsf{Next}_\pi, \mathsf{Out}_\pi)$, where ς_π determines which party speaks at each round, Prep_π computes the correlated randomness given to the parties in the pre-processing phase, Next_π is the next-message function used by a party (who is speaking at a round) to generate the message for the next round based solely on the message sent to it in the current round and its share of correlated randomness and input (since it does not update its state during the online phase), and Out_π is the function used by the output party (who does not participate in the protocol otherwise) to generate the final output.

Definition 2 (Private Sequential Stateless Protocol). *A T-round* Private Sequential Stateless protocol π for $f : \{0,1\}^n \to \{0,1\}$ *is a tuple* $(\mathsf{Prep}_\pi, \varsigma_\pi, \mathsf{Next}_\pi, \mathsf{Out}_\pi)$*, with* $\mathsf{Prep}_\pi : [n+1] \times \mathcal{R} \to \{0,1\}^*$, $\varsigma_\pi : [T+1] \to [n+1]$, $\mathsf{Next}_\pi : [T] \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\} \to \{0,1\}^*$*, and* $\mathsf{Out}_\pi : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$*, such that and the following is a 1-private protocol for the $n+1$-party functionality $\mathcal{F}_f$ in the pre-processing model:*

- *First, in the pre-processing phase $R \leftarrow \mathcal{R}$ is sampled and for each $i \in [n+1]$, P_i receives is $r_i := \mathsf{Prep}_\pi(i; R)$ as its share of correlated randomness;*
- *then for each $i \in [n]$, party P_i receives an input bit x_i;*
- *then, at each round $t \in [T]$, party $\mathsf{P}_{\varsigma_\pi(t)}$ receives a message m_{t-1} (m_0 is defined as the empty string) and sends the message $m_t := \mathsf{Next}_\pi(t, m_{t-1}, r_{\varsigma_\pi(t)}, x_{\varsigma_\pi(t)})$ to party $\mathsf{P}_{\varsigma_\pi(t+1)}$; it is required that $\varsigma_\pi(t) = n+1$ iff $t = T+1$.*
- *Finally, party P_{n+1} outputs the bit $\mathsf{Out}_\pi(m_{T+1}, r_{n+1})$.*

We call the PSS protocol π a speak-k-times protocol if $|\varsigma_\pi^{-1}(i)| \leq k$ for all $i \in [n]$. The randomness cost of π is defined as $\log_2 |\mathcal{R}|$ bits.

By default in the PSS model, unless otherwise specified, we always consider speak-constant-times protocols (i.e., speak-k-times protocols where k does not grow with the input size n).

It is worth emphasising that statelessness is a structural feature of a PSS protocol, and it does not alter the security model of 1-privacy: the adversary can corrupt a party at the beginning of the protocol and it can see all the messages ever sent to that party.

4.1 PSS Protocols to Branching Programs

In this section we prove the following result.

Theorem 6. *For any constant k, boolean functions over $\{0,1\}^n$ which have speak-k-times, constant-randomness-cost PSS protocols also have read-k-times, constant-width branching programs.*

Proof. The proof uses the ideas from the transformation of protocols into circuits in [KOR96]. We make the required transformation in two steps, firstly, converting the given randomized protocol into a deterministic protocol by freezing the randomnness in the system, and then, converting this deterministic protocol into a branching program by defining appropriate functions on the set of these messages. The width of the branching program will be determined by the number of different messages a party can receive at any round.

We start by bounding the number of different views a party can have under a fixed randomness $R \in \mathcal{R}$ (analogous to Lemma 4 from [KOR96]).

Lemma 1. *If π is a PSS protocol with randomness cost ρ, then for any fixed choice of randomness, over all the choices of inputs, the total number of communication transcripts seen by any party P_i is at most $2^{\rho+2}$.*

Proof. Let π be a PSS protocol for a function $f : \{0,1\}^n \to \{0,1\}$. For an input $\boldsymbol{x} \in \{0,1\}^n$, let $C_i(\boldsymbol{x})$ denote the set of communication transcripts a party P_i can see in executions of π, using all choices of the randomness $R \in \mathcal{R}$. Note that $|C_i(\boldsymbol{x})| \le |\mathcal{R}| = 2^\rho$. Secondly, for all $\boldsymbol{x}$ in an equivalence class such that x_i and $f(\boldsymbol{x})$ are equal (there are four such equivalence classes), the distribution, and hence support, of the views of P_i are identical; this follows from the 1-privacy guarantee of a PSS protocol. Since the view contains the communication (as well as the party's share of correlated randomness), for all $\boldsymbol{x}$ and $\boldsymbol{x}'$ in the same equivalence class, $C_i(\boldsymbol{x}) = C_i(\boldsymbol{x}')$. Hence, taking the union over all $\boldsymbol{x}$ in the same equivalence class, $|\bigcup_{\boldsymbol{x}} C_i(\boldsymbol{x})| \le 2^\rho$. Since there are four such equivalence classes, we have

$$\left| \bigcup_{\boldsymbol{x} \in \{0,1\}^n} C_i(\boldsymbol{x}) \right| \le 2^{\rho+2}.$$

In particular, for any fixed choice of randomness, the transcripts seen by P_i comes from this set of size $2^{\rho+2}$, as claimed. □

We now proceed to transform the given protocol into a branching program. Let π' be a deterministic protocol obtained by fixing the randomness of π to $R^* \in \mathcal{R}$. To convert π' to a branching program we shall interpret the message sent in round t from party P_i to P_j (where $i = \varsigma_\pi(t)$ and $j = \varsigma_\pi(t+1)$) as a state in the $t+1^{\text{st}}$ layer of the branching program. Since the number of different messages that P_j can receive in a round (over all inputs $\boldsymbol{x} \in \{0,1\}^n$) is upper bounded by the total number of communication transcripts, which is in turn bounded by $2^{\rho+2}$ by Lemma 1, the width of the branching program can be set to $w = 2^{\rho+2}$. The length of the branching program $\ell = T$, the number of rounds in π, and the input-reading function σ is the same as ς_π, but restricted to $[T]$ (rather than $[T+1]$).

The transition functions g_b^t from layer $t-1$ to layer t will implement $\mathsf{Next}_\pi(t, \cdot, r^*, b)$, for $b \in \{0,1\}$ and where $r^* = \mathsf{Prep}_\pi(\varsigma_\pi(t), R^*)$, under a mapping of messages to states. In more detail, for $t \in [T]$, let M^t denote the set of messages that can be sent in round t, over all possible inputs $\boldsymbol{x} \in \{0,1\}^n$ (with the randomness fixed to R^*); also let $M^0 = \{\epsilon\}$. We noted above that $|M^t| \le w$. Let $\eta^t : M^t \to [w]$ be an arbitrary injective function for each $t \in [T]$; also let $\eta^0(\epsilon) = 1$ to set the start state (in layer 1) to be 1. Then we define $g_b^t : [w] \to [w]$, for $b \in \{0,1\}$, such that if $u = \eta^{t-1}(m)$ for $m \in M^{t-1}$, let $g_b^t(u) = \eta^t(\mathsf{Next}_\pi(t, m, r^*, b))$ where $r^* = \mathsf{Prep}_\pi(\varsigma_\pi(t), R^*)$; if u is not in the image of η^t, we set $g_b^t(u)$ arbitrarily. Finally, The output function $\phi : [w] \to \{0,1\}$ is defined as follows: if $u = \eta^T(m)$, then $\phi(u) = \mathsf{Out}_\pi(m, \mathsf{Prep}_\pi(n+1, R^*))$; if u is not in the image of η^T, we set $\phi(u)$ arbitrarily.

From the perfect correctness of π, it follows that this branching program computes f. Also, since $\sigma = \varsigma_\pi$ (restricted to $[T]$), if π is a speak-k-times protocol, then the branching program constructed above will be a read-k-times branching program. Finally, as required, if π's randomness cost ρ is a constant, so is the width $w = 2^{\rho+2}$. □

5 PSS Protocols From Branching Programs

We begin by formally defining strong regularity and strongly regular branching programs. We then present Private Sequential Stateless protocols for strongly regular branching programs.

Definition 3 (Strong Regularity). *A pair of functions $g_0, g_1 : [w] \to [w]$ is* strongly regular *if the following conditions are met:*

1. *There exists c_0, c_1 such that $|g_0^{-1}(u)| \in \{c_0, 0\}$, and $|g_1^{-1}(u)| \in \{c_1, 0\}$ for all $u \in [w]$, when $g_0^{-1}(u)$ and $g_1^{-1}(u)$ denote the preimages of u under g_0 and g_1, respectively.*
2. *There exists c such that $|g_0^{-1}(u) \cap g_1^{-1}(v)| \in \{c, 0\}$ for all $u, v \in [w]$.*
3. *Define a bipartite graph $H = (L \cup R, E)$ where $L = [w]$ and $R = [w]$ (disjoint copies) are the left and right set of vertices respectively, and $E = \{(u, v) \in L \times R : g_0^{-1}(u) \cap g_1^{-1}(v) \neq \emptyset\}$ is the edge set. Let $\mathrm{Aut}(H)$ be the set of all automorphisms of H that respect the left and right parts; i.e., $\mathrm{Aut}(H) = \{(\mu, \nu) \in \mathrm{Sym}(w) \times \mathrm{Sym}(w) : (\mu(u), \nu(v)) \in E \Leftrightarrow (u, v) \in E\}$. Then,*

$$\mathrm{Prob}\left[(\mu(u), \nu(v)) = (u', v') \,|(\mu, \nu) \leftarrow \mathrm{Aut}(H)\right] = 1/|E|, \forall (u, v), (u', v') \in E. \tag{3}$$

We shall be interested in branching programs where, at all layers, the pairs of transition functions are strongly regular. We capture this in the following definition.

Definition 4 (SRBP and k-SRBP). *A branching program with input labeling function $\sigma : [\ell] \to [n]$ and transition functions $\{(g_0^{(t)}, g_1^{(t)})\}_{t \in [\ell]}$ is a strongly regular branching program (SRBP) if for every $t \in [\ell]$, the pair $(g_0^{(t)}, g_1^{(t)})$ is strongly regular. It is said to be a k-SRBP if for all $i \in [n]$, $|\{t : \sigma(t) = i\}| \leq k$.*

A special case of interest is a 1-SRBP: in this case, we may w.l.o.g. assume that $\ell = n$ (adding layers with identity functions as transition functions, if necessary), and σ is the identity function (by permuting the order of the arguments to the function evaluated by the SRBP, if necessary).

While SRBPs may appear restrictive, they are in fact quite expressive, and any branching program can be converted to one with only a polynomial blow-up in the width, and no change to the length or the input label function. We give an overview this conversion in Sect. 2.3. We state this formally here and present the proof in the extended version [ANPP24].

Theorem 7. *For any branching program of width w and length ℓ there is an SRBP computing the same function of the same length and input label function, and width w^2.*

Examples. Apart from the fact that any constant width branching program can be converted to a constant width SRBP, natural branching programs to compute some interesting functions are already constant width SRBPs. We mention three such examples of 1-SRBP below:

- $\mathsf{AND}(x_1, \ldots, x_n) = x_1 \wedge \ldots \wedge x_n$ has a width-2 1-SRBP.
- $\mathsf{IP}(x_1, y_1, \ldots, x_n, y_n) = \bigoplus_i (x_i \wedge y_i)$ has a width-4 1-SRBP.
- Every permutation branching program (in which all transition functions are permutations) is an SRBP (with $c_0 = c_1 = c = 1$ and H being a perfect matching, in Definition 3).

Strong Regularity and Regularity. It is instructive to compare SRBP with the notion of a regular branching program from [LPV23]. Let us call a pair of functions $g_0, g_1 : [w] \to [w]$ (c_1, c_2)-regular if for all $u \in [w]$, $|g_0^{-1}(u)| + |g_1^{-1}(u)| \in \{0, c_1, c_2, c_1 + c_2\}$. Note that a strongly regular pair (as a consequence of the first condition in Definition 3) is regular according to this definition. For the special case of $(1,1)$-regularity, we require $|g_0^{-1}(u)| + |g_1^{-1}(u)| \leq 2$; but since the average value of $|g_0^{-1}(u)| + |g_1^{-1}(u)|$ is 2, it must be the case that for each u, $|g_0^{-1}(u)| + |g_1^{-1}(u)| = 2$. This is the definition of regularity used in [LPV23].

Restricting to $(1,1)$-regular branching programs results in somewhat crippled computational power: even a simple function like n-input AND requires a $(1,1)$-regular branching program to have width that grows (exponentially) with n. On the other hand, AND has a width 2 branching program that is $(2,1)$-regular. As such, regular branching programs as generalized above (or possibly with the restriction that all layers use the same (c_1, c_2) – since the transformation in Theorem 7 yields a (w,w)-regular branching program) is an interesting class on its own right.

Strong regularity imposes additional constraints beyond (c_1, c_2)-regularity. One may in fact add even more constraints, and yet retain the result in Theorem 7 (e.g., require the bipartite graph H in Definition 3 to be a complete bipartite graph after pruning 0-degree nodes), but this will rule out some of the examples above (e.g., permutation branching programs).

5.1 PSS Protocols From 1-SRBP

Proof of Theorem 2. Let $\left(\{g_b^{(i)}\}_{i \in [n], b \in \{0,1\}}, \phi\right)$ be a width w 1-SRBP computing the function $f : \{0,1\}^n \to \{0,1\}$. The protocol given in Fig. 1 is a speak-once PSS protocol which computes f. All the variables used in the sequel are defined in Fig. 1. We will separately prove the correctness and privacy of the protocol.

A PSS PROTOCOL TO COMPUTE f HAVING A 1-SRBP

Let $\left(\{g_b^{(t)}\}_{t\in[n],b\in\{0,1\}},\phi\right)$ be a width-w 1-SRBP computing the function $f:\{0,1\}^n\to\{0,1\}$. For each $i\in[n]$, let $x_i\in\{0,1\}$ be the input to party P_i. and let $\mathsf{col}(i)$ be the parity of i, *i.e.*, $\mathsf{col}(i)=0$ if i is even, and 1 otherwise. Set $\varsigma_\pi:[n+1]\to[n+1]$ to be the identity map.

Preprocessing

1. $\alpha_{0,0},\alpha_{0,1},\alpha_{1,0},\alpha_{1,1},r_0,r_1\leftarrow\mathcal{R}$ is sampled where $\mathcal{R}=\mathrm{Sym}(w)\times\mathrm{Sym}(w)\times\mathrm{Sym}(w)\times\mathrm{Sym}(w)\times\{0,1\}\times\{0,1\}$.
2. P_1 receives $\mathsf{Prep}_\pi(1;\alpha_{0,0},\alpha_{0,1},\alpha_{1,0},\alpha_{1,1},r_0,r_1):=(r_1,\alpha_{1,0},\alpha_{1,1})$.
3. For each $2\le i\le n$: P_i receives $\mathsf{Prep}_\pi(i;\alpha_{0,0},\alpha_{0,1},\alpha_{1,0},\alpha_{1,1},r_0,r_1):=(r_{\mathsf{col}(i)},\hat{g}_{0,0}^{(i)},\hat{g}_{0,1}^{(i)},\hat{g}_{1,0}^{(i)},\hat{g}_{1,1}^{(i)})$ where
$$\begin{aligned}\hat{g}_{0,0}^{(i)}&=\alpha_{\mathsf{col}(i),0}\circ g_0^{(i)}\circ\alpha^{-1}_{\mathsf{col}(i-1),r_{\mathsf{col}(i-1)}}\\ \hat{g}_{0,1}^{(i)}&=\alpha_{\mathsf{col}(i),0}\circ g_0^{(i)}\circ\alpha^{-1}_{\mathsf{col}(i-1),r_{\mathsf{col}(i-1)}\oplus1}\\ \hat{g}_{1,0}^{(i)}&=\alpha_{\mathsf{col}(i),1}\circ g_1^{(i)}\circ\alpha^{-1}_{\mathsf{col}(i-1),r_{\mathsf{col}(i-1)}}\\ \hat{g}_{1,1}^{(i)}&=\alpha_{\mathsf{col}(i),1}\circ g_1^{(i)}\circ\alpha^{-1}_{\mathsf{col}(i-1),r_{\mathsf{col}(i-1)}\oplus1}.\end{aligned}$$
4. P_{n+1} receives $\mathsf{Prep}_\pi(n{+}1;\alpha_{0,0},\alpha_{0,1},\alpha_{1,0},\alpha_{1,1},r_0,r_1):=(S_{b,y})_{(b,y)\in\{0,1\}^2}$, where $S_{b,y}=\{j:\alpha^{-1}_{\mathsf{col}(n),r_{\mathsf{col}(n)}\oplus b}(j)\in\phi^{-1}(y)\}$.

Computation.

1. P_1 sends $\mathsf{Next}_\pi(1,m_0,(r_1,\alpha_{1,0},\alpha_{1,1}),x_1):=(s_1,v_1)$ to P_2, where $s_1=r_{\mathsf{col}(1)}\oplus x_1$ and $v_1=\alpha_{\mathsf{col}(1),x_1}\circ g_{x_1}^{(1)}(1)$.
2. For each $2\le i\le n$:
   P_i sends $\mathsf{Next}_\pi(i,(s_{i-1},v_{i-1}),(r_{\mathsf{col}(i)},\hat{g}_{0,0}^{(i)},\hat{g}_{0,1}^{(i)},\hat{g}_{1,0}^{(i)},\hat{g}_{1,1}^{(i)}),x_i):=(s_i,v_i)$ to P_{i+1}, where $s_i=r_{\mathsf{col}(i)}\oplus x_i$ and $v_i=\hat{g}^{(i)}_{x_i,s_{i-1}}(v_{i-1})$.
3. P_{n+1} outputs $\mathsf{Out}_\pi((s_n,v_n),\{S_{b,y}\}_{b,y\in\{0,1\}})=y$, where y is s.t. $v_n\in S_{s_n,y}$.

Fig. 1. A PSS protocol to compute f having a 1-SRBP.

Correctness. We claim,
$$v_i=\alpha_{\mathsf{col}(i),x_i}\circ g_{x_i}^{(i)}\circ g_{x_{i-1}}^{(i-1)}\circ\ldots\circ g_{x_1}^{(1)}(1),\qquad i\in[n].\tag{4}$$
Before proving this, we show that it implies correctness. We have, $s_n=r_{\mathsf{col}(n)}\oplus x_n$, and
$$S_{s_n,y}=\{j\in[w]\text{ s.t. }\alpha^{-1}_{\mathsf{col}(n),r_{\mathsf{col}(n)}\oplus s_n}(j)=\alpha^{-1}_{\mathsf{col}(n),x_n}(j)\in\phi^{-1}(y)\},\forall y\in\{0,1\}.$$
Hence, by Eq. (4) (for $i=n$), P_{n+1} outputs y such that
$$\alpha^{-1}_{\mathsf{col}(n),r_{\mathsf{col}(n)}\oplus s_n}(v_n)=\alpha^{-1}_{\mathsf{col}(n),x_n}\left(\alpha_{\mathsf{col}(n),x_n}\circ g_{x_n}^{(n)}\circ g_{x_{n-1}}^{(n-1)}\circ\ldots g_{x_1}^{(1)}(1)\right)\in\phi^{-1}(y).$$
Therefore P_{n+1} outputs y such that $\phi(g_{x_n}^{(n)}\circ g_{x_{n-1}}^{(n-1)}\circ\ldots g_{x_1}^{(1)}(1))=y$, ensuring correctness.

To conclude the proof of correctness, we prove (4) by induction. Clearly, (4) holds for $i = 1$. Assume that (4) holds for $i - 1$. Since $x_{i-1} = s_{i-1} \oplus r_{\mathsf{col}(i-1)}$,

$$\begin{aligned} v_i = \hat{g}^{(i)}_{x_i, s_{i-1}}(v_{i-1}) &= \alpha_{\mathsf{col}(i), x_i} \circ g^{(i)}_{x_i} \circ \alpha^{-1}_{\mathsf{col}(i-1), r_{\mathsf{col}(i-1)} \oplus s_{i-1}}(v_{i-1}) \\ &= \alpha_{\mathsf{col}(i), x_i} \circ g^{(i)}_{x_i} \circ \alpha^{-1}_{\mathsf{col}(i-1), x_{i-1}}(v_{i-1}) \\ &= \alpha_{\mathsf{col}(i), x_i} \circ g^{(i)}_{x_i} \circ \alpha^{-1}_{\mathsf{col}(i-1), x_{i-1}} \circ \alpha^{-1}_{\mathsf{col}(i-1), x_{i-1}} \\ &\quad \circ g^{(i-1)}_{x_{i-1}} \circ \ldots \circ g^{(1)}_{x_1}(1) \\ &= \alpha_{\mathsf{col}(i), x_i} \circ g^{(i)}_{x_i} \circ g^{(i-1)}_{x_{i-1}} \circ \ldots \circ g^{(1)}_{x_1}(1). \end{aligned}$$

Security. The view of P_1 consists of its input and the correlated randomness received during preprocessing. Hence, privacy against P_1 follows trivially.

We next show privacy against P_i for $i \in \{2, \ldots, n\}$. The view of P_i consists of its input and the messages received from P_{i-1} and the correlated randomness received during preprocessing, *viz.*, x_i, s_{i-1}, v_{i-1} and $\{\hat{g}^{(i)}_{b,b'}\}_{b,b' \in \{0,1\}}$. We first simplify the above expression. By Eq. (4), $v_{i-1} = \alpha_{\mathsf{col}(i-1), x_{i-1}}(u_{i-1})$, where $u_{i-1} = g^{(i-1)}_{x_{i-1}} \circ \ldots \circ g^{(1)}_{x_1}(1)$. Further, $\{\hat{g}^{(i)}_{b,b'}\}_{b,b' \in \{0,1\}}$ is a function of $\hat{g}^{(i)}_{0,0}, \hat{g}^{(i)}_{1,0}$ and $\alpha_{\mathsf{col}(i-1), r_{\mathsf{col}(i-1)}} \circ \alpha^{-1}_{\mathsf{col}(i-1), r_{\mathsf{col}(i-1)} \oplus 1}$. For brevity, we will denote u_{i-1}, s_{i-1} and x_{i-1} by u, s and x, $r_{\mathsf{col}(i-1)}$ and $r_{\mathsf{col}(i)}$ by r and r'; $g^{(i)}_0$ and $g^{(i)}_1$ by g_0 and g_1; $\alpha_{\mathsf{col}(i),0}$ and $\alpha_{\mathsf{col}(i),0}$ by α_0 and α_1; and $\alpha_{\mathsf{col}(i-1),0}$ and $\alpha_{\mathsf{col}(i-1),1}$ by β_0 and β_1. Recalling the definitions of $\hat{g}^{(i)}_{0,0}, \hat{g}^{(i)}_{1,0}$, the view is determined by

$$\left(x, s, \beta_x(u), r', \alpha_0 \circ g_0 \circ \beta_r^{-1}, \alpha_1 \circ g_1 \circ \beta_r^{-1}, \beta_r \circ \beta^{-1}_{r \oplus 1}\right).$$

Hence, the protocol is private against P_i if the following lemma holds:

Lemma 2. *For all permutations $\hat{\alpha}_0, \hat{\alpha}_1, \hat{\beta}_0, \hat{\beta}_1 \in \mathrm{Sym}(w)$, $b, b' \in \{0,1\}$, and $v \in [w]$, there exists a constant μ such that, for all $x \in \{0,1\}$, and $u \in [w]$,*

$$\mathrm{Prob}\left[\begin{array}{c|c} \alpha_0 \circ g_0 \circ (\beta_r)^{-1} = \hat{\alpha}_0 \circ g_0 \circ \hat{\beta}_0 & \alpha_0, \alpha_1, \beta_0, \beta_1 \leftarrow \mathrm{Sym}(w) \\ \alpha_1 \circ g_1 \circ (\beta_r)^{-1} = \hat{\alpha}_1 \circ g_1 \circ \hat{\beta}_0 & r \leftarrow \{0,1\} \\ \beta_r \circ \beta^{-1}_{r \oplus 1} = \hat{\beta}_0^{-1} \circ \hat{\beta}_1 & s = x \oplus r \\ s = b, \quad r' = b', \quad \beta_x(u) = v & r' \leftarrow \{0,1\} \end{array}\right] = \mu. \tag{5}$$

This is proved in the extended version [ANPP24]. We provide an intuition of the proof. Fix $\hat{\alpha}_0, \hat{\alpha}_1, \hat{\beta}_0, \hat{\beta}_1 \in \mathrm{Sym}(w)$, $b, b' \in \{0,1\}$, and $v \in [w]$. For $x \in \{0,1\}$ and $u \in [w]$, let the LHS of Eq. (5) be defined as $\mu(x, w)$. We observe that, there is a well structured set

$$\Lambda = \{\beta_0 \in \mathrm{Sym}(w) : \exists \alpha, \alpha' \in \mathrm{Sym}(w) \text{ s.t. } (g_0 \circ \beta_0 = \alpha \circ g_0) \wedge (g_1 \circ \beta_0 = \alpha' \circ g_0)\},$$

such that

$$\begin{aligned} \big\{(\beta_r)^{-1} \in \mathrm{Sym}(w) : &\alpha_0 \circ g_0 \circ (\beta_r)^{-1} = \hat{\alpha}_0 \circ g_0 \circ \hat{\beta}_0, \\ &\qquad \alpha_1 \circ g_1 \circ (\beta_r)^{-1} = \hat{\alpha}_1 \circ g_1 \circ \hat{\beta}_0\big\} = \{\beta_0 \circ \hat{\beta}_0 : \beta_0 \in \Lambda\}. \end{aligned}$$

Further, for all $\beta_0 \circ \hat{\beta}_0$ such that $\beta_0 \in \Lambda$, over the randomness of α_0 and α_1 choosen uniformly and independently from $\mathrm{Sym}(w)$, the events $\alpha_0 \circ g_0 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_0 \circ g_0 \circ \hat{\beta}_0$ and $\alpha_1 \circ g_1 \circ \beta_0 \circ \hat{\beta}_0 = \hat{\alpha}_1 \circ g_1 \circ \hat{\beta}_0$ simultaneously occur with the same probability. Using the above two observations, and standard renaming of variables, we simply the expression for $\mu(x, w)$ considerably to get the following: there exists a constant c such that,

$$\mu(x, w)/c = \mathrm{Prob}\left[\begin{array}{c} s = b \\ \Gamma_s(u) = v \end{array}\middle| \begin{array}{cc} \beta_0 \leftarrow \Lambda^0 \cap \Lambda^1, & \beta_1 = \beta_0 \circ \hat{\beta}_1 \\ \multicolumn{2}{c}{s \leftarrow \{0,1\}} \\ \Gamma_0 = (\beta_0 \circ \hat{\beta}_0)^{-1} & \Gamma_1 = \beta_1^{-1} \end{array}\right].$$

At this point, the following suffices to prove the lemma:

$$\begin{aligned} &\mathrm{Prob}\left[(\beta_0 \circ \hat{\beta}_0)^{-1}(u) = v \,\middle|\, \beta_0 \leftarrow \Lambda^0 \cap \Lambda^1\right] \\ &\qquad\qquad = \mathrm{Prob}\left[(\beta_0 \circ \hat{\beta}_1)^{-1}(u) = v \,\middle|\, \beta_0 \leftarrow \Lambda^0 \cap \Lambda^1\right], \quad \forall u \in [w]. \end{aligned}$$

When (g_0, g_1) is strongly regular, we argue that this is indeed the case (see [ANPP24]).

Finally, we prove privacy against P_{n+1}. The view of P_{n+1} is $\{S_{b,y}\}_{(b,y)\in\{0,1\}^2}$, v_n and s_n. We once again, simplify the notation by denoting $x_n, u_n, r_{\mathsf{col}(n)}$, $\alpha_{\mathsf{col}(n),0}$ and $\alpha_{\mathsf{col}(n),1}$ by x, u, r, α_0 and α_1. Since $S_{b,0} = [w] \setminus S_{b,1}$ for $b \in \{0,1\}$, $\{S_{b,y}\}_{b\in\{0,1\},y\in\{0,1\}}$ is a function of $(S_{0,0}, S_{1,0})$ We have $v_n = \alpha_{x_n}(u_n)$, $s_n = x_n \oplus r$, $S_{0,0} = \alpha_r(\phi^{-1}(0))$ and $S_{1,0} = \alpha_{r\oplus 1}(\phi^{-1}(0))$. Here, $\alpha_r(\phi^{-1}(0)) = \{\alpha_r(j) : j \in \phi^{-1}(0)\}$. To prove privacy against P_{n+1}, we will show that, when $x, x' \in \{0,1\}$ and $u, u' \in [w]$ such that $\phi(u) = \phi(u')$,

$$\begin{aligned} &(\alpha_r(\phi^{-1}(0)), \alpha_{r\oplus 1}(\phi^{-1}(0)), \alpha_x(u), x \oplus r) \\ &\qquad\qquad \equiv (\alpha_r(\phi^{-1}(0)), \alpha_{r\oplus 1}(\phi^{-1}(0)), \alpha_{x'}(u'), x' \oplus r). \end{aligned} \tag{6}$$

We show this using a sequence of equivalences:

$$\begin{aligned} &(\alpha_r(\phi^{-1}(0)), \alpha_{r\oplus 1}(\phi^{-1}(0)), \alpha_x(u), x \oplus r) \\ &\quad \equiv (\alpha_0(\phi^{-1}(0)), \alpha_1(\phi^{-1}(0)), \alpha_{x\oplus r}(u), x \oplus r) \\ &\quad \equiv (\alpha_0(\phi^{-1}(0)), \alpha_1(\phi^{-1}(0)), \alpha_{x'\oplus r}(u), x' \oplus r). \end{aligned} \tag{7}$$

The first equivalence is obtained by replacing $(\alpha_r, \alpha_{r\oplus 1})$ with the identically distributed pair (α_0, α_1); and the second equivalence is obtained by replacing r with identically distributed $r \oplus x \oplus x'$. Since $\phi(u) = \phi(u')$, there exists $\hat{\alpha} \in \mathrm{Sym}(w)$ such that, $\hat{\alpha}(u) = u'$ and, for all $u'' \in [w]$, $\phi \circ \hat{\alpha}(u'') = \phi(u'')$. We replace (α_0, α_1) with the identically distributed pair $(\alpha_0 \circ \hat{\alpha}, \alpha_1 \circ \hat{\alpha})$ to obtain the following equivalence:

$$\begin{aligned} &(\alpha_0(\phi^{-1}(0)), \alpha_1(\phi^{-1}(0)), \alpha_{x'\oplus r}(u), x' \oplus r) \\ &\quad \equiv (\alpha_0 \circ \hat{\alpha}(\phi^{-1}(0)), \alpha_1 \circ \hat{\alpha}(\phi^{-1}(0)), \alpha_{x'\oplus r} \circ \hat{\alpha}(u), x' \oplus r) \\ &\quad \equiv (\alpha_0(\phi^{-1}(0)), \alpha_1(\phi^{-1}(0)), \alpha_{x'\oplus r}(u'), x' \oplus r). \end{aligned} \tag{8}$$

The second equivalence used the following facts: $\alpha_b \circ \hat{\alpha}(\phi^{-1}(0))$ is identically distributed as $\alpha_b(\phi^{-1}(0))$ for $b \in \{0,1\}$; and $\hat{\alpha}(u) = u'$. Using the reasoning in Eq. (7) in the reverse direction, we can show that

$$\begin{aligned}(\alpha_0(\phi^{-1}(0)), \alpha_1(\phi^{-1}(0)),&\alpha_{x'\oplus r}(u'), x' \oplus r) \\ &\equiv (\alpha_r(\phi^{-1}(0)), \alpha_{r\oplus 1}(\phi^{-1}(0)), \alpha_{x'}(u'), x' \oplus r). \quad (9)\end{aligned}$$

Equations (7) to (9) prove Eq. (6) concluding the proof of privacy.

Randomness Complexity: Since we need four independent samples from the set of permutations of $[w]$ and two random bits for this protocol, the randomness cost is $\log(2 + 4w!)$ bits, which is $O(w \log w)$. This completes the proof. □

5.2 PSS Protocols From *k*-SRBP

Normal Form SRBP. In our constructions, for a cleaner presentation, we will consider *normal form* (strongly regular) branching programs. This especially makes the presentation of the conflict graph easier. A length ℓ SRBP is said to be in normal form if it satisfies that for all $t \in [\ell - 1]$, $\sigma(t) \neq \sigma(t+1)$. That is, the same party doesn't feed inputs to two consecutive layers of the branching program.

It is easy to modify an k-SRBP for a function (with at least 3 inputs) into a normal form $(2k-1)$-SRBP, with the same width and computing the same function. We show this is in the following lemma which is proved in the extended version [ANPP24].

Lemma 3. *Any function $f : \{0,1\}^n \to \{0,1\}$, where $n \geq 3$, computable using a k-SRBP is also computable using a $(2k-1)$-SRBP in the normal form.*

We now present the protocol for computation of functions having k-SRBP. Our construction follows the blueprint of our construction for 1-SRBP. Since a party feeds their input to the branching program only once in an 1-SRBP, we could get away with using the same permutations for masking the state of the branching program in alternating layers, resulting in a protocol that uses only 4 permutations and two bit masks to realize 1-privacy. In a general SRBP, each party can feed their inputs in several layers of the BP. Hence, the main challenge in the protocol is to come up with a strategy for recycling randomness while ensuring that a reappearing party does not learn any intermediate state of the branching program due to this reuse. We define our strategy for randomness reuse using a conflict graph associated with the branching program we want to compute. We present the protocol in the following proof.

A PSS PROTOCOL COMPUTING f COMPUTABLE USING A k-SRBP

Let $\Pi = \left(\sigma, \{g_b^{(t)}\}_{t\in[\ell],b\in\{0,1\}}, \phi\right)$ be a width-w normal form k-SRBP computing the function $f : \{0,1\}^n \to \{0,1\}$. We define a conflict graph $G_\Pi = ([\ell], E)$ for Π, where $\{t,t'\} \in E$ if $t \neq t' \in [\ell]$ and $\{\sigma(t), \sigma(t+1)\} \cap \{\sigma(t'), \sigma(t'+1)\} \neq \emptyset$. Let $\mathsf{col} : [\ell] \to [\chi]$ be a vertex coloring of G_Π. Define $z_t = x_{\sigma(t)}$ for $t \in [\ell]$. Assign $\varsigma_\pi(t) = \sigma(t)$, $t \in [\ell]$ and $\varsigma_\pi(\ell+1) = \mathsf{P}_{n+1}$.

Preprocessing Phase

1. $(\alpha_{c,0}, \alpha_{c,1}, r_c)_{c\in[\chi]} \leftarrow \mathcal{R}$ is sampled where $\mathcal{R} = (\mathrm{Sym}(w) \times \mathrm{Sym}(w) \times \{0,1\})^{|\chi|}$.
2. For $i \in [n]$ such that $i \neq \sigma(1)$, P_i receives

$$\mathsf{Prep}_\pi(i; (\alpha_{c,0}, \alpha_{c,1}, r_c)_{c\in[\chi]}) := \left(r_{\mathsf{col}(t)}, \hat{g}_{0,0}^{(t)}, \hat{g}_{0,1}^{(t)}, \hat{g}_{1,0}^{(t)}, \hat{g}_{1,1}^{(t)}\right)_{t\in[\ell]:\sigma(t)=i},$$

where

$$\begin{aligned}
\hat{g}_{0,0}^{(t)} &= \alpha_{\mathsf{col}(t),0} \circ g_0^{(t)} \circ \alpha_{\mathsf{col}(t-1),r_{\mathsf{col}(t-1)}}\\
\hat{g}_{0,1}^{(t)} &= \alpha_{\mathsf{col}(t),0} \circ g_0^{(t)} \circ \alpha_{\mathsf{col}(t-1),r_{\mathsf{col}(t-1)}\oplus 1}\\
\hat{g}_{1,0}^{(t)} &= \alpha_{\mathsf{col}(t),1} \circ g_1^{(t)} \circ \alpha_{\mathsf{col}(t-1),r_{\mathsf{col}(t-1)}}\\
\hat{g}_{1,1}^{(t)} &= \alpha_{\mathsf{col}(t),1} \circ g_1^{(t)} \circ \alpha_{\mathsf{col}(t-1),r_{\mathsf{col}(t-1)}\oplus 1}.
\end{aligned}$$

3. $\mathsf{P}_{\sigma(1)}$ receives $\mathsf{Prep}_\pi(\sigma(1); (\alpha_{c,0}, \alpha_{c,1}, r_c)_{c\in[\chi]})$ which is defined as

$$\left((r_{\mathsf{col}(1)}, \alpha_{\mathsf{col}(1),0}, \alpha_{\mathsf{col}(1),1}), \left(r_{\mathsf{col}(t)}, \hat{g}_{0,0}^{(t)}, \hat{g}_{0,1}^{(t)}, \hat{g}_{1,0}^{(t)}, \hat{g}_{1,1}^{(t)}\right)_{t\in[\ell]\setminus\{1\}:\sigma(t)=\sigma(1)}\right),$$

where the $\hat{g}$ functions are as in Step 2 above.

4. P_{n+1} receives $\mathsf{Prep}_\pi(n+1; (\alpha_{c,0}, \alpha_{c,1}, r_c)_{c\in[\chi]}) := (S_{b,y})_{(b,y)\in\{0,1\}^2}$ where

$$S_{b,y} = \left\{j : \alpha^{-1}_{\mathsf{col}(\ell),r_{\mathsf{col}(\ell)}\oplus b}(j) \in \phi^{-1}(y)\right\}, \quad b, y \in \{0,1\}.$$

Computation

1. $\mathsf{P}_{\sigma(1)}$ sends $\mathsf{Next}_\pi(1, m_0, (r_{\mathsf{col}(1)}, \alpha_{\mathsf{col}(1),0}, \alpha_{\mathsf{col}(1),1}), z_1) := (s_1, v_1)$ to $\mathsf{P}_{\sigma(2)}$, where $s_1 = r_{\mathsf{col}(1)} \oplus z_1$ and $v_1 = \alpha_{\mathsf{col}(1),z_1} \circ \hat{g}_{z_1}^{(1)}(1)$. Note that we defined $z_t = x_{\sigma(t)}$ for $t \in [\ell]$.
2. For $2 \leq t \leq \ell - 1$:
   $\mathsf{P}_{\sigma(t)}$ sends $\mathsf{Next}_\pi(t, (s_{t-1}, v_{t-1}), (r_{\mathsf{col}(t)}, \hat{g}_{0,0}^{(t)}, \hat{g}_{0,1}^{(t)}, \hat{g}_{1,0}^{(t)}, \hat{g}_{1,1}^{(t)}), z_t) := (s_t, v_t)$ to $\mathsf{P}_{\sigma(t+1)}$, where $s_t = r_{\mathsf{col}(t)} \oplus z_t$ and $v_t = \hat{g}_{z_t,s_{t-1}}^{(t)}(v_{t-1})$.
3. $\mathsf{P}_{\sigma(\ell)}$ sends $\mathsf{Next}_\pi(\ell, (s_{\ell-1}, v_{\ell-1}), (r_{\mathsf{col}(\ell)}, \hat{g}_{0,0}^{(\ell)}, \hat{g}_{0,1}^{(\ell)}, \hat{g}_{1,0}^{(\ell)}, \hat{g}_{1,1}^{(\ell)}), z_\ell) := (s_\ell, v_\ell)$ to P_{n+1}, where $s_\ell = r_{\mathsf{col}(\ell)} \oplus z_\ell$ and $v_\ell = \hat{g}_{z_\ell,s_{\ell-1}}^{(\ell)}(v_{\ell-1})$.
4. P_{n+1} outputs $\mathsf{Out}_\pi((s_\ell, v_\ell), \{S_{b,y}\}_{b,y\in\{0,1\}}) = y$, where y is s.t. $v_\ell \in S_{s_\ell,y}$.

Fig. 2. A PSS protocol computing f having an k-SRBP.

Proof of Theorem 3. Suppose f is computable using an k-SRBP

$$\Pi = \left(\sigma, \{g_b^{(t)}\}_{i\in[\ell],b\in\{0,1\}}, \phi\right).$$

We will show that the protocol in Fig. 2 computes f with 1-privacy.

Correctness. The computation of s_t for each $t \in [\ell]$ is carried out exactly as in the protocol in Fig. 1, except using a different *coloring* function col. Hence, using the same line of argument used to show Eq. (4) in the proof of Theorem 2,

$$\begin{aligned} v_t &= \alpha_{\mathsf{col}(t),z_t} \circ g_{z_t}^{(t)} \circ g_{z_{t-1}}^{(t-1)} \circ \ldots \circ g_{z_1}^{(1)}(1), \quad t \in [\ell], \\ v_\ell &= \alpha_{\mathsf{col}(\ell),z_\ell} \circ g_{z_\ell}^{(\ell)} \circ g_{z_{\ell-1}}^{(\ell-1)} \circ \ldots \circ g_{z_1}^{(1)}(1), \\ S_{s_\ell,y} &= \{j \in [w] \text{ s.t. } \alpha^{-1}_{\mathsf{col}(\ell),r_{\mathsf{col}(\ell)}\oplus s_\ell}(j) \in \phi^{-1}(y)\}, \end{aligned} \tag{10}$$

where we have used the notation $z_t = x_{\sigma(t)}$ for $t \in [\ell]$ from Fig. 2. Thus, P_{n+1} outputs y such that $y = g_{z_\ell}^{(\ell)} \circ \ldots \circ g_{z_1}^{(1)}(1) = f(x_1, \ldots, x_n)$.

Security. We first show that the protocol is private against P_i for each i such that $\sigma(1) \neq i$, *i.e.*, all the parties except the party who implements the first layer. Fix such a i. Let $(\tilde{x}_1, \ldots, \tilde{x}_n)$ and $(\hat{x}_1, \ldots, \hat{x}_n)$ be any pair of inputs such that $f(\tilde{x}_1, \ldots, \tilde{x}_n) = f(\hat{x}_1, \ldots, \hat{x}_n)$ and $\tilde{x}_i = \hat{x}_i$. We will prove that the view of P_i in an execution of the protocol with $(\tilde{x}_1, \ldots, \tilde{x}_n)$ as inputs is identically distributed as in an execution with $(\hat{x}_1, \ldots, \hat{x}_n)$ as inputs.

Let $\tilde{x}_i = \hat{x}_i = x_i$ and $f(\tilde{x}_1, \ldots, \tilde{x}_n) = f(\hat{x}_1, \ldots, \hat{x}_n) = y$. Let $\tilde{b}_t = \tilde{x}_{\sigma(t)}$ and $\hat{b}_t = \hat{x}_{\sigma(t)}$ for all $t \in [\ell]$. For each $t \in [\ell]$, let $\tilde{u}_t = g_{\tilde{b}_t}^{(t)} \circ \ldots \circ g_{\tilde{b}_1}^{(1)}(1)$ and $\hat{u}_t = g_{\hat{b}_t}^{(t)} \circ \ldots \circ g_{\hat{b}_1}^{(1)}(1)$. The view of P_i in an execution of the protocol with $(\tilde{x}_1, \ldots, \tilde{x}_n)$ as input is

$$\begin{aligned} \widetilde{\text{View}} = \Big(x_i, y, \Big\{&\tilde{s}_{t-1} = r_{\mathsf{col}(t-1)} \oplus \tilde{b}_{t-1}, \tilde{v}_{t-1} = \alpha_{\mathsf{col}(t-1),\tilde{b}_{t-1}}(\tilde{u}_{t-1}), \\ &r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\Big\}_{t:\sigma(t)=i}\Big). \end{aligned}$$

The view of P_i in an execution of the protocol with $(\hat{x}_1, \ldots, \hat{x}_n)$ as input is

$$\begin{aligned} \widehat{\text{View}} = \Big(x_i, y, \Big\{&\hat{s}_{t-1} = r_{\mathsf{col}(t-1)} \oplus \hat{b}_{t-1}, \hat{v}_{t-1} = \alpha_{\mathsf{col}(t-1),\hat{b}_{t-1}}(\hat{u}_{t-1}), \\ &r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\Big\}_{t:\sigma(t)=i}\Big). \end{aligned}$$

We will show $\widetilde{\text{View}} \equiv \widehat{\text{View}}$ using a hybrid argument. Let $t_1, \ldots, t_\zeta$ be an arbitrary ordering of the set $\mathcal{L}_i = \{t : \sigma(t) = i\}$ where $\zeta = |\mathcal{L}_i|$. For each $0 \leq h \leq \zeta$ we define a hybrid view

$$\text{Hyb}_h = \left(\begin{array}{ll} & \Big\{r_{\mathsf{col}(t-1)} \oplus \tilde{b}_{t-1}, \alpha_{\mathsf{col}(t-1),\tilde{b}_{t-1}}(\tilde{u}_{t-1}), \\ & \qquad r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\Big\}_{t\in\{t_1,\ldots,t_h\}} \\ x_i, y, & \\ & \Big\{r_{\mathsf{col}(t-1)} \oplus \hat{b}_{t-1}, \alpha_{\mathsf{col}(t-1),\hat{b}_{t-1}}(\hat{u}_{t-1}), \\ & \qquad r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\Big\}_{t\in\{t_{h+1},\ldots,t_\zeta\}} \end{array}\right).$$

Then, $\mathrm{Hyb}_0 = \widetilde{\mathrm{View}}$ and $\mathrm{Hyb}_\zeta = \widehat{\mathrm{View}}$. Hence, $\widetilde{\mathrm{View}} \equiv \widehat{\mathrm{View}}$ if $\mathrm{Hyb}_{h-1} \equiv \mathrm{Hyb}_h$ for all $h \in [\zeta]$, which we prove below: Since (a) col is a coloring of G_Π, and $\mathcal{L}_i$ forms a clique in G_Π, and (b) $\sigma(t) \neq \sigma(t+1)$ for any $t \in [\ell - 1]$ due to normal form of Π, for any t such that $\sigma(t) = i$,

$$\left(r_{\mathsf{col}(t-1)}, r_{\mathsf{col}(t)}, \{\alpha_{\mathsf{col}(t-1),b}, \alpha_{\mathsf{col}(t),b}\}_{b\in\{0,1\}}\right) \\ \perp\!\!\!\perp \left(r_{\mathsf{col}(t'-1)}, r_{\mathsf{col}(t')}, \{\alpha_{\mathsf{col}(t'-1),c}, \alpha_{\mathsf{col}(t'),c}\}_{c\in\{0,1\}}\right)_{t'\in\mathcal{L}_i\setminus\{t\}}.$$

Hence, for any $h \in [\zeta]$, to prove that $\mathrm{Hyb}_{h-1} \equiv \mathrm{Hyb}_h$, it suffices to show that for $t = t_h$,

$$\left(r_{\mathsf{col}(t-1)} \oplus \hat{b}_{t-1}, \alpha_{\mathsf{col}(t-1),\hat{b}_{t-1}}(\hat{u}_{t-1}), r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\right) \\ \equiv \left(r_{\mathsf{col}(t-1)} \oplus \tilde{b}_{t-1}, \alpha_{\mathsf{col}(t-1),\tilde{b}_{t-1}}(\tilde{u}_{t-1}), r_{\mathsf{col}(t)}, \hat{g}^t_{0,0}, \hat{g}^t_{0,1}, \hat{g}^t_{1,0}, \hat{g}^t_{1,1}\right).$$

But, this follows from Lemma 2; see the proof of privacy against P_i for $2 \leq i \leq n$ in Theorem 2.

Next, we argue privacy against $\mathsf{P}_{\sigma(1)}$. The view of $\mathsf{P}_{\sigma(1)}$ differ from other parties as it receives $\alpha_{\mathsf{col}(1),b} \circ g_b^{(1)}$ for $b \in \{0,1\}$ and $r_{\mathsf{col}(1)}$. Appealing to the properties of graph coloring of G_Π, $\alpha_{\mathsf{col}(1),b} \circ g_b^{(1)}$ for $b \in \{0,1\}$ is independent of the remaining part of the view of P_1. Hence, we can prove privacy against P_1 exactly as we proved the privacy against other parties after excluding these parts of the view.

The view of P_{n+1} is exactly the same as that which is given in the protocol in Theorem 2, *i.e.*, (s_ℓ, v_ℓ) and the set $S_{b,y}$ for $b, y \in \{0,1\}$ and therefore the proof of privacy follows from the proof given for Theorem 2.

Randomness complexity. For every color $c \in [\chi]$, the protocol samples fresh random variables α_c (from a permutation of size w) and coin r_c. Therefore the randomness complexity is bounded by $O(2\chi w \log w)$. From Brooks' theorem [Bro41], the number of colors needed to color a graph greedily is $\Delta_G + 1$ where Δ_G is its maximum degree. In this case, since an input is read at most $2m - 1$ times and from the definition of $G_\Pi = (\ell, E)$, $\{t, t'\} \in E$ if $\sigma(t) = \sigma(t')$, $\sigma(t) = \sigma(t'+1)$, $\sigma(t) = \sigma(t'-1)$ or $\sigma(t-1) = \sigma(t'-1)$, there are at most $8m - 4$ vertices having an edge with t. Therefore $\Delta_G = 4$. This gives that the randomness complexity is $O(mw \log w)$. ☐

6 Private Computation of AND

In this section we focus on private computation of the n-party AND function, which has received significant attention in the literature. We present new results regarding upper and lower bounds on the randomness complexity of AND. Thanks to AND having a 1-SRBP (see Sect. 5), we have a PSS protocol for it, from Fig. 1. However, before we can compare our results fairly to prior results, we need to cast our PSS protocol into a setting without an external source supplying correlated randomness and without a separate output party (with all input-parties getting the output, instead). Towards this, we define an *unassisted* PSS (uPSS) protocol, which is a private protocol (with uncorrelated randomness) in the sense in Sect. 3, but is also sequential and stateless, and further has only one party being randomized.

Unassisted Private Sequential Stateless (uPSS) Protocol. A uPSS protocol for a function $f : \{0,1\}^n \to \{0,1\}$ is specified by a tuple $(\mathsf{Prep}_\pi, \varsigma_\pi, \mathsf{Next}_\pi, \mathsf{Out}_\pi)$, similar to a PSS protocol, but with the following differences:

- There are only n parties, $\mathsf{P}_1, \ldots, \mathsf{P}_n$ (no separate output party).
- In the pre-processing phase P_1 samples $R \leftarrow \mathcal{R}$, computes $r_i = \mathsf{Prep}_\pi(i, R)$ and sends it to P_i for each $i \in [n]$.
- After this each party P_i receives its input x_i, and they all carry out the protocol using ς_π and Next_π exactly as in the PSS model.
- In addition, each of them produces an output over the last $[n]$ rounds. For this we require that in a $T+n$-round protocol, $\varsigma_\pi : [T] \to [n]$, when restricted to the domain $\{T-n+1, \ldots, T\}$, is a bijection with $[n]$; also in round $t > T-n$, party P_i, where $i = \varsigma_\pi(t)$, produces the output $\mathsf{Out}_\pi(t, m_{t-1}, x_i, r_i)$.

We require this protocol to be a 1-private protocol (without correlated randomness) for the n-party functionality $\mathcal{F}_f^*$, which is similar to $\mathcal{F}_f$ but delivers the output to all n parties.

6.1 uPSS Protocol for 1-SRBP

Below, we describe the necessary modifications to be made to the protocol in Fig. 1 to turn it into a uPSS protocol for a function f with a 1-SRBP.

1. **Preprocessing phase.**
   (a) P_1 samples $(\alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}, r_0, r_1) \leftarrow \mathcal{R}$ is sampled where $\mathrm{Sym}(w) \times \mathrm{Sym}(w) \times \mathrm{Sym}(w) \times \mathrm{Sym}(w) \times \{0,1\} \times \{0,1\}$ and sends the appropriate correlated randomness r_i to P_i for each $1 \le i \le n$ (including itself) as in the description of the protocol in Fig. 1.
   (b) Additionally, P_1 samples γ_0 and γ_1 uniformly from $\mathrm{Sym}(w)$ and a random bit r'; it sends r' and (γ_0, γ_1) to P_n, and the sets $S_{c,y} = \{j : \alpha^{-1}_{\mathsf{col}(n), r' \oplus b} \circ \gamma^{-1}_{r' \oplus b}(j) \in \phi^{-1}(y)\}$ for $b \in \{0,1\}$ and $y \in \{0,1\}$ to P_2.
2. **Computation Phase.**
   (a) For $i = 1, \ldots, n-1$, each P_i (including P_1) follows the instruction in the protocol in Fig. 1.
   (b) P_n computes $v_n = \hat{g}^{(n)}_{x_n, s_{n-1}}(v_{n-1})$ as in the previous protocol, but sends (s'_n, v'_n) to P_2, where $s'_n = r' \oplus x_n$ and $v'_n = \gamma_{x_n}(v_n)$.
   (c) P_2 computes y such that $v'_n \in S_{s'_n, y}$.
   (d) Over the next n rounds, each party (starting with P_2) outputs y and sends it to the next party to output.

Theorem 8. *Suppose $f : \{0,1\}^n \to \{0,1\}$ is computable using a 1-SRBP of width w. There exists a uPSS protocol that realizes $\mathcal{F}_f^*$ with 1-privacy using $O(w \log w)$ bits of randomness, all sampled by a single party.*

Proof. The proof of security and correctness follow closely to that of the protocol in Fig. 1 which we established in Theorem 2. Throughout the proof, we refer to this as the 'previous protocol'.

Correctness. The computation of (s_i, v_i) for $1 \leq i \leq n$, proceeds exactly as in the previous protocol. Hence, $v_n = \alpha_{\mathsf{col}(n),x_n} \circ g^{(n)}_{x_n} \circ g^{(n-1)}_{x_{n-1}} \circ \ldots \circ g^{(1)}_{x_1}(1)$, and $v'_n = \gamma_{x_n} \circ \alpha_{\mathsf{col}(n),x_n} \circ g^{(n)}_{x_n} \circ \ldots \circ g^{(1)}_{x_1}(1)$. Since $s'_n = r' \oplus x_n$,

$$S_{s'_n,y} = \{j \in [w] \text{ s.t. } \alpha^{-1}_{\mathsf{col}(n),r'\oplus s'_n} \circ \gamma^{-1}_{r'\oplus s'_n}(j) = \alpha^{-1}_{\mathsf{col}(n),x_n} \circ \gamma^{-1}_{x_n}(j) \in \phi^{-1}(y)\}.$$

Hence, P_2 outputs y such that

$$\begin{aligned}\alpha^{-1}_{\mathsf{col}(n),x_n} \circ \gamma^{-1}_{x_n}(v'_n) &= \alpha^{-1}_{\mathsf{col}(n),x_n} \circ \gamma^{-1}_{x_n} \circ \gamma_{x_n} \circ \alpha_{\mathsf{col}(n),x_n} \circ g^{(n)}_{x_n} \circ g^{(n-1)}_{x_{n-1}} \circ \ldots g^{(1)}_{x_1}(1) \\ &= g^{(n)}_{x_n} \circ g^{(n-1)}_{x_{n-1}} \circ \ldots g^{(1)}_{x_1}(1) \in \phi^{-1}(y).\end{aligned}$$

Thus, $y = f(x_1, \ldots, x_n)$.

Security. The only message received by P_1 throughout the protocl is y from P_n. We have established $y = f(x_1, \ldots, x_n)$, hence the protocol is secure against P_1. For $3 \leq i < n$, the view of P_i is identical to that in the previous protocol (with the exception of y that they receive at the end of the new protocol). Hence, security against them follow from our argument in Theorem 2. The view of P_n additionally consists of r' and functions (γ_0, γ_{r_1}). But, these random variables are sampled independent of all the other messages received by P_n. Hence, the security against P_n in the new protocol follows from that in the old protocol.

Finally, we argue security against P_2. Compared to its view in the previous protocol, the view of P_2 in the new protocol additionally contains $(S_{0,y}, S_{1,y})$ for $y \in \{0,1\}$ that it received in the preprocessing phase, and (s'_n, v'_n) that it receives from P_n. Since (γ_0, γ_1) are sampled independent of $(\alpha_{\mathsf{col}(i),0}, \alpha_{\mathsf{col}(i),1})$ for all $i \in [n]$, the additional values in the view of P_2 are independent of all the other messages it received. Further, the view of P_2 in the previous protocol is established to be secure. Hence, to argue security against P_2, it suffices to show that the additional messages received by P_2 in the new protocol do not break security. But, it can be seen by inspection that $(S_{0,y}, S_{1,y})$ for $y \in \{0,1\}$ and s'_n, v'_n are distributed exactly as the view of P_{n+1} in the previous protocol. But, the previous protocol is secure against P_{n+1} who does not have any input to the protocol. The security against P_2 in the new protocol now follows from the security of the previous protocol against P_2 and P_{n+1}.

Randomness Complexity: Since we need 6 independent samples from the set of permutations of $[w]$ $(\alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}, \gamma_0, \gamma_1)$ and three binary coins (r_0, r_1, r'), the randomness cost of the given protocol is $3 + 6\log(w!)$ bits, which is $O(w \log w)$ bits. $\square$

Since AND has a width-2 branching program, the following corollary follows immediately.

Corollary 1. *There exists a uPSS protocol for $\mathcal{F}^*_{\mathsf{AND}}$ with randomness cost of* 9 *bits.*

However, when the number of inputs n is odd, we obtain an optimization to 6 bits, which matches the state-of-the-art result for (non uPSS) 1-private computation of $\mathcal{F}^*_{\mathsf{AND}}$. [5] The optimized protocol is described in Fig. 3. Instead of presenting the full protocol, we describe how the protocol differs from Fig. 1. We remark that this optimization is possible only because of the properties of AND function. We crucially use the fact that in AND, if a party has 0 as input, it can essentially ignore the information it received so far. Furthermore, we rely on the total number of parties being odd to make the message sent by last party independent of the rest of the view for one of the parties. With this optimization, we bring down the randomness cost from 9 bits in the general uPSS protocol to 6 bits. The following theorem states this fact and it is proved in the extended version [ANPP24].

Theorem 9. *There exists a uPSS protocol for $\mathcal{F}^*_{\mathsf{AND}}$ for an odd number of parties, with randomness cost of 6 bits.*

Preprocessing
- P_1 samples $(\alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}, r_0, r_1) \leftarrow \mathcal{R}$ where $\mathcal{R} = (\mathrm{Sym}(2) \times \mathrm{Sym}(2) \times \mathrm{Sym}(2) \times \mathrm{Sym}(2) \times \{0,1\} \times \{0,1\})$.
- For $3 \leq i \leq n$, P_1 sends the appropriate correlated random variables r_i to P_i as given in the description of the protocol in Figure 1.
- P_1 sends $(\alpha_{0,0}, r_0, \tilde{\alpha}_0, \tilde{\alpha}_1)$ to P_2 where $\tilde{\alpha}_0 = \alpha_{0,r_1}$ and $\tilde{\alpha}_1 = \alpha_{0,r_1 \oplus 1}$.

Computation
- P_1 sends $v_1 = \alpha_{0,1} \circ g^{(1)}_{x_1}(1)$ to P_2.
- P_2 sends (s_2, v_2) to P_3 where $v_2 = \begin{cases} v_1 & x_2 = 1 \\ \alpha_{0,0} & x_2 = 0 \end{cases}$ and $s_2 = x_2 \oplus r_0$.
- For each $i \in \{3, \ldots, n-1\}$, P_i computes and sends (s_i, v_i) to P_{i+1} where (s_i, v_i) is defined as per the description of Figure 1.
- P_n sends (s_n, v_n) to P_2 where (s_n, v_n) is defined as per the description of Figure 1.
- P_2 computes and outputs $y = 1$ if $\tilde{\alpha}^{-1}_{s_n}(v_n) = 1$ $y = 0$ if $\tilde{\alpha}^{-1}_{s_n}(v_n) = 2$ and sends to P_3.
- For $i \in \{3, \ldots, n-1\}$, P_i sends y to P_{i+1}.
- P_n sends y to P_1.

Fig. 3. Optimized uPSS protocol for n-party $\mathcal{F}^*_{\mathsf{AND}}$ for odd n

[5] For even n too, the randomness cost can be improved from 9 to $6 + \log_2 3$ bits. We omit this optimization as it still falls short of the non-uPSS state-of-the-art.

6.2 Lower Bound on Randomness Complexity of AND for 3 Parties

In this section, we prove Theorem 5, namely that the randomness complexity of AND for 3 parties is at least 3 bits.

We show this lower bound through a reduction from a secret sharing problem. Recently, [ARN+23] characterized the randomness complexity of the following problem, referred to as 3-Secret Sharing (3SS): Let $\mathcal{S} \subseteq \{0,1\}^3$. Given $(x_1, x_2, x_3) \in \mathcal{S}$, a dealer produces three different shares $W_{1,2}, W_{2,3}$ and $W_{3,1}$ such that the pair of shares $(W_{1,2}, W_{2,3})$ together reveals x_2 and nothing more about (x_1, x_3) – i.e., nothing other than what can be inferred from learning x_2 and the fact that $(x_1, x_2, x_3) \in \mathcal{S}$. Similarly x_1 (and nothing more about (x_2, x_3)) can be obtained from shares $(W_{1,2}, W_{3,1})$, while $(W_{2,3}, W_{3,1})$ reveals x_3 and nothing more.

We shall argue that a 1-private 3-party MPC protocol for the AND function yields a 3SS scheme for the set $\mathcal{S} = \{0,1\}^3 \setminus \{(1,1,1)\}$ as follows: To share $(x_1, x_2, x_3) \in \mathcal{S}$, let $W_{i,j}$ denote the transcript between parties P_i and P_j in the MPC protocol for AND, in which the input of each party P_i is x_i. Note that $(W_{1,2}, W_{2,3})$ together is part of the view of party P_2 in the MPC protocol, and reveals nothing more about x_1, x_3 beyond what x_2 and $x_1 \wedge x_2 \wedge x_3$ reveals. But the latter only reveals that $(x_1, x_2, x_3) \in \mathcal{S}$. The analogous conditions hold for $(W_{2,3}, W_{3,1})$ (for party P_3) and $(W_{3,1}, W_{1,2})$ (for party P_1). Hence this scheme satisfies the privacy condition of 3SS. It remains to check that this scheme also meets the correctness conditions of 3SS, namely that $(W_{1,2}, W_{2,3})$ do determine x_2, and so on. We verify this in the following lemma, which is proved in the extended version [ANPP24] using elementary arguments based on the properties of private protocols.

Lemma 4. *In any 1-private 3-party MPC protocol for the 3-party AND function where all parties learn the output, the pair of transcripts in the view of each party uniquely determines its input.*

Theorem 5 now follows from the fact that for $\mathcal{S} = \{0,1\}^3 \setminus \{(1,1,1)\}$ 3SS has a lower bound of 3 bits on randomness complexity [ARN+23].

In contrast, the best known upper bound is 6 bits [CR22], and we leave it as an open problem to bridge this gap. Incidentally, our approach cannot yield a higher lower bound than 3, since the randomness complexity of 3SS for $\{0,1\}^3 \setminus \{(1,1,1)\}$ is exactly 3 bits.

References

[ANPP24] Hari Krishnan P. Anilkumar, Varun Narayanan, Manoj Prabhakaran, and Vinod M. Prabhakaran. Randomness in private sequential stateless protocols. Cryptology ePrint Archive, Paper 2024/1448, 2024.

[ARN+23] Hari Krishnan P. Anilkumar, Aayush Rajesh, Varun Narayanan, Manoj M. Prabhakaran, and Vinod M. Prabhakaran. Randomness requirements for three-secret sharing. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 252–257. IEEE, 2023.

[Bar86] David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 1–5, 1986.

[BGP99] Carlo Blundo, Clemente Galdi, and Pino Persiano. Randomness recycling in constant-round private computations (extended abstract). In Prasad Jayanti, editor, *Distributed Computing, 13th International Symposium, Bratislava, Slovak Republic, September 27-29, 1999, Proceedings*, volume 1693 of *Lecture Notes in Computer Science*, pages 138–150. Springer, 1999.

[BGP07] Carlo Blundo, Clemente Galdi, and Giuseppe Persiano. Low-randomness constant-round private XOR computations. *Int. J. Inf. Sec.*, 6(1):15–26, 2007.

[Bro41] Rowland Leonard Brooks. On colouring the nodes of a network. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 37, pages 194–197. Cambridge University Press, 1941.

[BSPV99] Carlo Blundo, Alfredo De Santis, Giuseppe Persiano, and Ugo Vaccaro. Randomness complexity of private computation. *Comput. Complex.*, 8(2):145–168, 1999.

[CKOR00] Ran Canetti, Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Randomness versus fault-tolerance. *Journal of cryptology*, 13:107–142, 2000.

[CR22] Geoffroy Couteau and Adi Rosén. Random sources in private computation. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 443–473. Springer, Heidelberg, December 2022.

[DPP16] Deepesh Data, Vinod M. Prabhakaran, and Manoj M. Prabhakaran. Communication and randomness lower bounds for secure computation. *IEEE Trans. Inf. Theory*, 62(7):3901–3929, 2016.

[FKN94] Uri Feige, Joe Killian, and Moni Naor. A minimal model for secure computation. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 554–563, 1994.

[GIKM98] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 151–160, 1998.

[GIS22] Vipul Goyal, Yuval Ishai, and Yifan Song. Tight bounds on the randomness complexity of secure multiparty computation. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 483–513. Springer, Heidelberg, August 2022.

[GR03] Anna Gál and Adi Rosén. Lower bounds on the amount of randomness in private computation. In *35th ACM STOC*, pages 659–666. ACM Press, June 2003.

[IK97] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, pages 174–183. IEEE, 1997.

[IKO+11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In *Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30*, pages 406–425. Springer, 2011.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 356–364, 1994.

[JLR03] Andreas Jakoby, Maciej Liskiewicz, and Rüdiger Reischuk. Private computations in networks: Topology versus randomness. In Helmut Alt and Michel Habib, editors, *STACS 2003, 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, February 27 - March 1, 2003, Proceedings*, volume 2607 of *Lecture Notes in Computer Science*, pages 121–132. Springer, 2003.

[KM96] Eyal Kushilevitz and Yishay Mansour. Randomness in private computations. In James E. Burns and Yoram Moses, editors, *15th ACM PODC*, pages 181–190. ACM, August 1996.

[KOP+19] Eyal Kushilevitz, Rafail Ostrovsky, Emmanuel Prouff, Adi Rosén, Adrian Thillard, and Damien Vergnaud. Lower and upper bounds on the randomness complexity of private computations of AND. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 386–406. Springer, Heidelberg, December 2019.

[KOR96] Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Characterizing linear size circuits in terms of privacy. In *28th ACM STOC*, pages 541–550. ACM Press, May 1996.

[KOR98] Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Amortizing randomness in private multiparty computations. In Brian A. Coan and Yehuda Afek, editors, *17th ACM PODC*, pages 81–90. ACM, June / July 1998.

[KR94] Eyal Kushilevitz and Adi Rosén. A randomnesss-rounds tradeoff in private computation. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 397–410. Springer, Heidelberg, August 1994.

[LPV23] Chin Ho Lee, Edward Pyne, and Salil Vadhan. On the power of regular and permutation branching programs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023).* Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

[RU19] Adi Rosén and Florent Urrutia. A new approach to multi-party peer-to-peer communication complexity. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 64:1–64:19. LIPIcs, January 2019.

Secret Sharing

Evolving Secret Sharing Made Short

Danilo Francati[1](✉) and Daniele Venturi[2]

[1] Royal Holloway, University of London, Egham, UK
danilo.francati@rhul.ac.uk
[2] Sapienza University of Rome, Rome, Italy

Abstract. Evolving secret sharing (Komargodski, Naor, and Yogev, TCC'16) generalizes the notion of secret sharing to the setting of *evolving access structures*, in which the share holders are added to the system in an online manner, and where the dealer does not know neither the access structure nor the maximum number of parties in advance. Here, the main difficulty is to distribute shares to the new players without updating the shares of old players; moreover, one would like to minimize the share size as a function of the number of players.

In this paper, we initiate a systematic study of evolving secret sharing in the *computational setting*, where the maximum number of parties is polynomial in the security parameter, but the dealer still does not know this value, neither it knows the access structure in advance. Moreover, the privacy guarantee only holds against computationally bounded adversaries corrupting an unauthorized subset of the players.

Our main result is that for many interesting, and practically relevant, evolving access structures (including graphs access structures, DNF and CNF formulas access structures, monotone circuits access structures, and threshold access structures), under standard hardness assumptions, there exist efficient secret sharing schemes with computational privacy and in which the shares are *succinct* (i.e., much smaller compared to the size of a natural computational representation of the evolving access structure).

Keywords: secret sharing · evolving access structures · computational security

1 Introduction

A (threshold) secret sharing scheme, as introduced independently by Blakley [11] and Shamir [35], allows a dealer to share a secret message μ between n parties, obtaining shares $\sigma_1, \dots, \sigma_n$, in such a way that the following two properties are satisfied for a fixed threshold parameter $t \le n$:

D. Francati—Work done in part while at Aarhus University (Aarhus, Denmark) and George Mason University (Fairfax, Virginia, USA).

D. Venturi—The author is member of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15490, pp. 69–99, 2025.
https://doi.org/10.1007/978-981-96-0941-3_3

– **Correctness:** Any subset of at least t parties can reconstruct th e message (by pulling their shares together).
– **(Perfect) Privacy:** Any subset of at most $t-1$ parties obtains no information (statistically) on the message.

Ito, Saito, and Nishizeki [22] extended the concept of secret sharing to general *access structures* $\mathcal{A}$, in which certain subsets of the n parties are allowed to reconstruct the message (i.e., the so-called *authorized* subsets $\mathcal{I} \in \mathcal{A}$), and all other subsets of the n parties (i.e., the so-called *unauthorized* subsets $\mathcal{U} \notin \mathcal{A}$) obtain no information on the message. It is natural to require that access structures should be *monotone*, meaning that if a subset $\mathcal{I}$ is authorized, then any subset of the parties that includes $\mathcal{I}$ is also authorized.

An important efficiency measure in the context of secret sharing schemes is the *share size*, defined as the bit-length of the largest share given to the n parties. Indeed, a large body of work has focused on minimizing the share size for different access structures $\mathcal{A}$. For instance, Shamir's scheme (based on polynomial interpolation) for threshold access structures achieves share size $\max\{\ell, \log n\}$, where ℓ is the length of the message, which is known to be optimal [12,33]. More generally, Csirmaz [14,15] proved that there is an (explicit) access structure that requires a *total* share size of $\Omega(n^2/\log n)$. In contrast, the best scheme [3] for general access structures achieves share size $1.5^{n+o(n)}$, which is pretty far from the lower bound. We refer to the excellent survey by Beimel [6] for an overview of the main results in classical secret sharing.

Evolving Secret Sharing. In a beautiful work, Komargodski, Naor, and Yogev [24], generalized secret sharing to the setting of *evolving* access structures, in which the number of parties n is not known in advance and can potentially go to infinity. Intuitively, an evolving access structures consists of a monotone sequence of subsets $\mathcal{A} = \{\mathcal{A}_n\}_{n\geq 1}$, where $\mathcal{A}_n$ denotes the access structure when there are n parties. Importantly, the dealer does not know n, neither it knows the access structure $\mathcal{A}_n$ before the n-th party enters the system. The main result in [24] is a secret sharing scheme for the evolving threshold access structure (i.e., the number of parties n grows, but the threshold $t = \mathsf{poly}(\lambda)$ is fixed and known to the dealer), in which the share size of party n, for messages of size $\ell = 1$, is $(t-1)\cdot \log n + O(\log\log n)$. They also give a secret sharing scheme for any evolving access structure, in which the share size of party n is 2^{n-1} (i.e., with exponential share size).

In a follow-up work, Komargodski and Paskin-Cherniavsky [26] give a secret sharing scheme for the more general evolving *dynamic* threshold access structure, which is represented by a sequence of thresholds $\{t_n\}_{n\geq 1}$ of increasing[1] size, so that the authorized subsets when there are n parties are all the subsets of at least t_n parties. The share size of this construction, when the message length is $\ell = 1$, is $O(n^4 \cdot \log n)$. Note that, in this case, each of the thresholds t_n can depend on n; this captures, e.g., the so-called *evolving majority* access structure,

[1] The fact that the thresholds should be increasing is required to ensure monotonicity of the access structure.

in which qualified subsets are those which form a majority of the current number of parties at *some* point in time. Xin and Yuan [38] show that the share size can be improved to $O(n^4)$ by relying on techniques from algebraic geometry.

More recently, at Eurocrypt'20, Beimel and Othman [9] gave constructions of secret sharing schemes for the dynamic threshold *ramp* access structure, which is represented by two functions $\tau, \rho : \mathbb{N} \to \mathbb{N}$ such that, when there are n parties in the system, the threshold for reconstruction is $\rho(n)$ (i.e., any subset of at least $\rho(n) \leq n$ parties can reconstruct the message) while the threshold for privacy is $\tau(n)$ (i.e., every subset of at most $\tau(n) < \rho(n)$ parties has no information about the message). The share size in their construction depends on the gap between the reconstruction and privacy thresholds; in particular, when the message length is $\ell = 1$, the share size is $\mathsf{polylog}(n)$ when $\rho(n) - \tau(n) = n/\mathsf{polylog}(n)$, and $O(n \cdot \log n)$ when $\rho(n) - \tau(n) = \sqrt{n}$; furthermore, they also give a direct construction for the special case in which $\rho(n) = t$ and $\tau(n) = t/2$ for any constant $t = O(1)$, with share size $O(\log t \cdot \log n)$. This improves over a previous construction by the same authors [8], which achieves share size $O(1)$, always when the message length is $\ell = 1$, but when $\rho(n) = b \cdot n$ and $\tau(n) = a \cdot n$ for any constants $0 < a < b < 1$ (i.e., when the gap between the reconstruction and the privacy thresholds is a constant fraction of the number of parties).

Lastly, Alon et al. [1] gave a construction for infinite layered branching programs. In particular, their scheme can handle any (evolving) access structure which can be computed by a (evolving) branching program, where the latter can evolve over time increasing in width $\omega(n)$ (this represents the addition of users). The share size of their scheme is $2^{O\left(\min\{\epsilon(n)\cdot\log(n), \sqrt{\epsilon(n)}\}\right)\cdot n}$ when the width of the corresponding branching program is bounded by $\omega(n) \leq 2^{\epsilon(t)\cdot t}$ for $\epsilon(n) < 0.04$.

All of the above constructions achieve *perfect* privacy (i.e., unauthorized subsets have no information about the message, in an information-theoretic sense), and can potentially accommodate an infinite number of users. Moreover, they are all tailored to variations of the evolving threshold access structure, and have share size that is at least linear in the number of parties n or in the threshold t (the only exception being ramp secret sharing schemes). Given this state of affairs, the following question arises naturally:

> *Can we get secret sharing schemes with* succinct *shares (i.e., with size smaller than the size of a natural computational representation of the evolving access structure) for richer* evolving *access structures, possibly under computational assumptions?*

1.1 Our Contributions

We provide a positive answer to the above question by initiating a systematic study of secret sharing schemes for evolving access structures in the *computational setting*. Our contributions are summarized in Table 1, where we compare our results with the state of the art in terms of access structure, share size, and computational assumptions.

Table 1. Comparing our results with state-of-the-art constructions for evolving secret sharing, in terms of access structure, share size, and computational assumptions. See Sect. 1.2 for the definition of the various parameters. The share size refers to the size of the share received by the n-th party, where n is the current number of parties in the system. In information-theoretic constructions, n is unbounded; in the computational setting $n = \mathsf{poly}(\lambda)$ (but the dealer does not know an upper bound on n). For simplicity, we only consider message length $\ell(\lambda) = \lambda$; when a scheme is for $\ell(\lambda) = 1$, we consider the share size obtained by repeating the sharing procedure λ times in parallel.

Reference	Access Structure	Parameters	Share Size	Assumptions
[24]	Any	–	$\lambda \cdot 2^{n-1}$	–
	Static Threshold	$t = \mathsf{poly}(\lambda)$	$(t-1) \cdot \log n + \mathsf{poly}(\lambda, t) \cdot o(\log n)$	–
[26]	Dynamic Threshold	$t_1 \le \cdots \le t_n \le n$	$\lambda \cdot O(n^4 \cdot \log n)$	–
[8]	Ramp Dynamic Threshold	$\rho(n) - \tau(n) = c \cdot n$	$O(\lambda)$	–
[9]	Ramp Dynamic Threshold	$\rho(n) - \tau(n) = n/\mathsf{polylog}(n)$	$\lambda \cdot \mathsf{polylog}(n)$	–
	Ramp Dynamic Threshold	$\rho(n) - \tau(n) = \sqrt{n}$	$\lambda \cdot O(n \cdot \log n)$	–
	Ramp Static Threshold	$\rho(n) = t; \tau(n) = t/2$	$\lambda \cdot O(\log t \cdot \log n)$	–
[38]	Static Threshold	$0 < t \le n; \epsilon > 0$	$\lambda \cdot O(t^{1+\epsilon} \cdot \log n)$	–
	Dynamic Threshold	$t_1 \le \cdots \le t_n \le n$	$\lambda \cdot O(n^4)$	–
[1]	Layered Infinite Branching Programs	$\epsilon(n) < 0.04; \omega(n) \le 2^{\epsilon(n) \cdot n}$	$\lambda \cdot 2^{O\left(\min\{\epsilon(n) \cdot \log(n), \sqrt{\epsilon(n)}\}\right) \cdot n}$	–
Sect. 4.2	Any	$d_n = \mathsf{poly}(\lambda, n)$	$\lambda \cdot (d_n + 1)$	OWFs
Sect. 5.1	Static Threshold	$t = \mathsf{poly}(\lambda)$	λ	–
	Dynamic Threshold	$t_1 \le \cdots \le t_n \le n$	$\lambda \cdot (n+1)$	OWFs
Sect. 5.2	Graphs	–	$\mathsf{poly}(\lambda)$	RSA/iO + SSB
Sect. 5.3	Monotone Circuits	$m_g = m_g^{\wedge} + m_g^{\vee}; g \ge 1$	$m_g \cdot \mathsf{poly}(\lambda)$	RSA/iO + SSB
			$(m_g^{\vee})^2 + m_g^{\wedge} \cdot O(\lambda)$	DDH/BDDH
			$m_g^{\vee} + m_g^{\wedge} \cdot O(\lambda)$	LWE
Full version [20]	CNF Formulas	$m \ge 1$	$\mathsf{poly}(\lambda)$	RSA/iO + SSB
			$m^2 \cdot \mathsf{poly}(\lambda)$	DDH/BDDH
			$m \cdot \mathsf{poly}(\lambda)$	LWE
Full version [20]	DNF Formulas	$t_n \ge 1$	$\lambda \cdot (t_n + 1)$	OWFs

In a nutshell, we provide constructions of *computationally private* secret sharing schemes for a plethora of evolving access structures, under standard hardness assumptions. In all of our constructions, the number of parties n is upper bounded by an arbitrary polynomial in the security parameter, but the dealer does not know this polynomial, neither it knows the overall access structure (it only knows the new authorized subsets when a new party joins the system). For some access structures, the share size is *succinct* (i.e., much smaller compared to the size of a natural computational representation of the evolving access structure). More in details, we given constructions of secret sharing schemes:

– For any evolving access structure in which the n-th participant appears in at most $d_n = \mathsf{poly}(\lambda, n)$ authorized subsets. This construction requires one-way functions (OWFs), and yields share size $\lambda \cdot (d_n + 1)$.
– For the dynamic threshold access structure. This construction requires OWFs, and yields share size $\lambda \cdot (n + 1)$
– For graphs access structures, in which the parties are added to the nodes of an evolving (undirected) graph, and the authorized subsets consist of all the pair of nodes for which there is an edge in the graph. This construction

requires either the RSA assumption, or indistinguishability obfuscation (iO) and somewhere statistically binding (SSB) hash functions, and yields share size $\mathsf{poly}(\lambda)$.
- For monotone circuits access structures, in which the parties correspond to the inputs wires of an evolving boolean circuit made of AND and OR gates with unbounded fan-in. This construction requires either of the following assumptions: (i) RSA, (ii) iO plus SSB hash functions, (iii) DDH/BDDH, (iv) LWE, and yields share size that is roughly linear[2] in the number of gates that are added to the circuit.
- For CNF and DNF formulas access structures, which are a special case of monotone circuits. In fact, in these cases, we give direct constructions that are slightly better in terms of assumptions and/or share size.

All of the above constructions allow to share secret messages of size $\ell(\lambda) = \lambda$. In the final part of the paper, we deal with the problem of domain extension for evolving secret sharing schemes and show that, under mild assumptions, all of our schemes can be generically upgraded to support messages of length $\ell(\lambda) = \mathsf{poly}(\lambda)$, by paying only an additive (in fact, linear in ℓ) overhead in terms of share size. This transformation only requires OWFs.

1.2 Technical Overview

We now give a detailed overview of the main techniques we use in order to obtain our results, starting with the notion of computational privacy for evolving secret sharing, and then explaining the ideas behind each of our constructions.

Computational Evolving Secret Sharing. The definition of computationally private secret sharing for evolving access structures is the natural adaptation of the corresponding information-theoretic definition. In particular, in the computational setting, the number of parties is $n = \mathsf{poly}(\lambda)$. An evolving access structure is a monotone sequence $\{\mathcal{A}_n\}_{n\geq 1}$, where $\mathcal{A}_n$ denotes the access structure when there are n parties in the system. We note that the dealer does not know the actual polynomial that upper bounds n, neither it knows the access structure $\mathcal{A}_n$ before the n-th party enters the system. This limitation is rather important, as if the dealer knows that $n < \tilde{n}$ for some polynomial $\tilde{n}$, along with the access structure $\mathcal{A}_{\tilde{n}}$, it can simply use a standard secret sharing scheme for $\mathcal{A}_{\tilde{n}}$ and distribute the shares to the parties as they arrive.

Computational privacy simply requires that for every (polynomial) $n \geq 1$, for all unauthorized subsets $\mathcal{U} \notin \mathcal{A}_n$, and for every pair of messages (μ_0, μ_1), no computationally bounded adversary given the shares $(\sigma_i)_{i\in\mathcal{U}}$ can distinguish, with better than negligible probability, whether the shares are generated using message μ_0 or message μ_1.

[2] This is a very rough approximation. See Table 1 and Sect. 1.2 for more precise parameters.

Rigidity. In principle, an evolving access structure $\mathcal{A} = \{\mathcal{A}_n\}_{n \geq 1}$ only needs to be monotone, in the sense that, for every $\mathcal{I}$ such that $\mathcal{I} \in \mathcal{A}_n$, every superset $\mathcal{I}' \supset \mathcal{I}$ also satisfies $\mathcal{I}' \in \mathcal{A}_n$. Now, say that $n = 10$ and that parties 1 and 3 are not authorized (i.e., $\mathcal{U} = \{1, 3\} \notin \mathcal{A}_{10}$); when party 11 arrives, the set $\mathcal{U}$ might become authorized without violating monotonicity. An access structure is called *rigid* if the above never happens, namely the subset $\mathcal{U}$ never becomes authorized.

It turns out that rigidity plays a rather important role when defining how more complex access structures (e.g., monotone circuits) can evolve. Indeed, in Sect. 3.2, we show that in any secret sharing for a *non-rigid* evolving access structure $\mathcal{A}$, the dealer must update the shares of old players at some point. This observation becomes immediately apparent when we look again at the above example: Since $\mathcal{U} = \{1, 3\}$ is unauthorized when $n = 10$, by the privacy property, the shares σ_1 and σ_3 reveal no information about the message; hence, unless we update these shares, σ_1 and σ_3 are not enough to reconstruct the message when $n = 11$, which contradicts the correctness property.

General Access Structures. In Sect. 4, we give a simple construction of a secret sharing scheme for general rigid evolving access structures $\hat{\mathcal{A}}$. However, the scheme is provably secure (and efficient) only when we make the restriction that each party n is added to $d_n = \mathsf{poly}(\lambda, n)$ authorized subsets. The share size of party n is going to be $\lambda \cdot (d_n + 1) = \mathsf{poly}(\lambda, n)$.

This construction roughly works as follows. Let G be a pseudorandom generator (PRG) with unbounded polynomial stretch (this exists assuming OWFs); we think of the PRG output as a sequence of λ-bit blocks, which we denote by $\mathsf{G}(\kappa)[j]$. The share σ_n given to the n-th party consists of a λ-bit seed κ_n for the PRG, as well as a ciphertext $\gamma_{\mathcal{I}}$ for every new subset $\mathcal{I}$ containing n that is added to the access structure $\hat{\mathcal{A}}_n$. This ciphertext is obtained by masking the message μ with a pad $\rho_{\mathcal{I}} = \bigoplus_{i \in \mathcal{I}} \mathsf{G}(\kappa_i)[j(\mathcal{I})]$, where κ_i is the seed of party i, and $j(\mathcal{I})$ is the index corresponding to the subset $\mathcal{I}$ in some lexicographic order. Note that the number of ciphertexts given to party n is exactly d_n; furthermore, since both n and d_n are polynomial, the number of PRG blocks that are needed is still polynomial, and thus the construction is efficient.

Correctness follows by observing that the parties corresponding to an authorized subset $\mathcal{I} \in \hat{\mathcal{A}}_n$ can recover the pad $\rho_{\mathcal{I}}$ and reveal the message. Privacy follows by an hybrid argument, in which we replace the ciphertext $\gamma_{\mathcal{I}}$ for every $\mathcal{I}$ that contains an index corresponding to an honest party with a random string. Since the attacker does not know the seed of honest players, these hybrids are all computationally indistinguishable by security of the PRG. Hence, one observes that in the final hybrid, the distribution of the shares corresponding to an unauthorized subset of the players is independent of the message. Interestingly, our construction shares similarities with an old scheme proposed by Cachin in the context of *online* secret sharing [13,16]. The main difference is that Cachin's scheme directly uses OWFs (or hard-core bits) instead of PRGs, and additionally requires a public bulletin board that must be updated by the dealer.[3] However, in retrospect, the only reason why the bulletin board is needed is because Cachin's definition of evolving access structure does not require rigidity. This is consistent with our impossibility result showing that secret sharing schemes for non-rigid evolving access structures require the dealer to update old shares.

In the information-theoretic setting, Mazor [30] demonstrated a lower bound for evolving secret sharing, showing that there exists an evolving access structure $\hat{\mathcal{A}}$ for which every information-theoretic evolving secret sharing scheme for $\hat{\mathcal{A}}$ (where the dealer does not know the access structure in advance) is such that the share size of the first n parties is at least $2^{n-o(n)}$. We highlight that our (computationally secure) scheme for general access structure circumvents such lower bound. Intuitively, we are able to accomplish this thanks to PRGs which allow generating an arbitrary polynomial number of random blocks from a short seed. We provide more details in Sect. 4 and Remark 2. This is a clear example of the main benefit of studying evolving secret sharing in the computational setting.

Dynamic Threshold Access Structures. Next, we move to concrete access structures that do not obey the restrictions of the above generic construction, starting with the evolving threshold access structure. The static case, where there is a single threshold $t = \mathsf{poly}(\lambda)$ known to the dealer, is rather simple to deal with: although the dealer does not know the maximum number of users, it knows that $n = \mathsf{poly}(\lambda)$ and thus can define the share of the n-th party as the evaluation $\sigma_n = f(n)$ of a random polynomial f of degree $t-1$ with coefficients over a field of size $\lambda^{\omega(1)}$, subject to the constraint that $f(0) = \mu$; this yields share size $\omega(1) \cdot \log(\lambda)$, ensures both correctness and privacy as per Shamir's secret sharing, and moreover allows to accommodate an arbitrary polynomial number of users.[4]

Hence, we move to the more challenging setting of the evolving dynamic threshold access structure. As explained above, this access structure is specified by a sequence of thresholds $\{t_n\}_{n \geq 1}$ such that $t_n \geq t_{n-1}$, and the authorized subsets when there are n parties are all the subsets of size at least $0 < t_n \leq n$. Here, the threshold t_n may depend on n, and the dealer does not know t_n before party n arrives. Our construction, which can be found in Sect. 5.1, works as follows. Let G be a PRG with unbounded polynomial stretch. When party n arrives, the dealer samples a random seed κ_n, along with a random polynomial f_n of degree $t_n - 1$ over a field of exponential size 2^λ, subject to the constraint that $f_n(0) = \mu$. The share σ_n of party n consists of the seed κ_n, of the

[3] Online secret sharing is the ancestor of evolving secret sharing. Csirmaz and Tardos [16] show how to remove the public bulletin board, obtaining information-theoretic privacy for infinitely many parties; however, they require the dealer knows an upper bound on the maximum number of authorized subsets a party can join.

[4] Note that, in case $t = O(1)$, the evolving threshold access structure actually satisfies the condition required by our generic construction, as the number of new authorized subsets in which party n participates is exactly $d_n = \binom{n}{t} = \mathsf{poly}(\lambda)$. However, the above direct construction based on Shamir secret sharing works even for $t = \mathsf{poly}(\lambda)$, and does not require computational assumptions.

evaluation $f_n(n)$, along with $n-1$ ciphertexts $(\gamma_i)_{i<n}$, where each ciphertext γ_i is an encryption of $f_n(i)$ under the pad $\rho_i = \mathsf{G}(\kappa_i)[n-i]$ (i.e., the next unused block output by the PRG $\mathsf{G}(\kappa_i)$ associated to party i).

Correctness follows by observing that the parties corresponding to an authorized subset can recover at least t_n pads used to encrypt the evaluations of the polynomial f_n, and thus reconstruct the message via polynomial interpolation. Privacy follows by an hybrid argument, in which we replace all of the ciphertexts γ_j corresponding to honest parties with a uniformly random ciphertext. Since the attacker does not know the seed of honest players, these hybrids are all computationally indistinguishable by security of the PRG. Hence, one observes that for every $k \in [n]$, the adversary now knows at most $t_k - 1$ evaluations of the polynomial f_k, and thus the message is information-theoretically hidden.

An interesting feature of our construction is that it works unmodified even assuming the sequence of thresholds $\{t_n\}_{n\geq 1}$ is not increasing. Namely, it is allowed that $t_n < t_{n-1}$ for some number of parties n, so long as the newly added authorized subsets (i.e., those that are in $\hat{\mathcal{A}}_n$, but not in $\hat{\mathcal{A}}_{n-1}$) always include party n. The latter ensures monotonicity (and rigidity is also preserved). We find this to be a natural extension of the evolving dynamic threshold access structure.

The aforementioned constructions are presented in Sect. 5.1.

Graphs Access Structures. Consider now the case of graphs access structures, in which the parties correspond to nodes $v_1, \ldots, v_n$ in an undirected graph $G = (\mathcal{V}, \mathcal{E})$, and parties (i, j) are authorized if and only if $(v_i, v_j) \in \mathcal{E}$ (i.e., (v_i, v_j) is an edge in the graph). Applebaum *et al.* [2] recently gave a *succinct* secret sharing scheme for this access structure, based on a new primitive called *projective* PRG. Intuitively, a projective PRG, as any standard PRG, allows to stretch a short seed msk (a.k.a. the master secret key) into a pseudorandom string of *bounded* length $m \in \mathbb{N}$. Additionally, one can use the master secret key msk to generate projective keys $\alpha_{\mathcal{T}}$ associated to sets $\mathcal{T} \subset [m]$, in such a way that it is possible to use $\alpha_{\mathcal{T}}$, along with the master public key mpk associated to msk, to recover the PRG output corresponding to the seed msk in the positions indexed by $\mathcal{T}$. On the other hand, all the remaining positions still look pseudorandom (this property is known as *robustness*). For this notion to be non-trivial, the projective keys must be succinct (i.e., with length sub-linear in $|\mathcal{T}|$).

Applebaum *et al.* [2] first give a construction for *bipartite* graphs access structures, in which the nodes belong to two sets $\mathcal{V} = (\mathcal{V}^{(0)}, \mathcal{V}^{(1)})$, and the edges are only between nodes pertaining to different sets. It is known that secret sharing schemes for bipartite graphs imply ones for arbitrary graphs. To secret share a message $\mu \in \{0, 1\}$, the dealer picks a random seed msk for the projective PRG, and lets $y_1, \ldots, y_m$ be the full PRG output. Hence, the share of a left node $v_i \in \mathcal{V}^{(0)}$ is set to $\mu \oplus y_i$, whereas the share of a right node $v_j \in \mathcal{V}^{(1)}$ is set to the projective key for the set $\mathcal{T}_j = \{i : (v_i, v_j) \in \mathcal{E}\}$ of v_j's neighbors.

In Sect. 5.2, we extend the above construction to the evolving setting. Here, the underlying graph evolves as nodes and edges are added to it, whenever new players enter the system. However, new edges can be added only if those involve a new player, otherwise we would violate rigidity. This yields the rigid evolving bipartite graphs access structure. Our construction is based on the following observation: when the n-th party corresponds to a right node $v_n \in \mathcal{V}^{(1)}$, the dealer can define its share as the projective key corresponding to the set of left neighbors $\mathcal{T}_n$ of node v_n. On the other hand, when the n-th party corresponds to a left node $v_n \in \mathcal{V}^{(0)}$, we would need to encrypt the message with the next available output bit from the projective PRG. This generates two technical problems:

– The output length of the projective PRG is fixed to some value m, which needs to be known in advance. We solve this problem by requiring that the projective PRG has *unbounded* polynomial stretch, so that the setup does not depend on m. Note that increasing the stretch of a projective PRG (in a black-box way) is a non-trivial task: For instance, the standard trick to increase the stretch of a PRG by running it sequentially poly-many times simply does not work, as it destroys succinctness. Fortunately, in Sect. 2.1 and the full version of this work [20], we show that some of the constructions in Applebaum *et al.* [2] can be adapted to our purpose. In particular, there exist projective PRGs with unbounded polynomial stretch and, with master public keys and projective keys of size $\mathsf{poly}(\lambda)$, assuming either the RSA assumption, or indistinguishability obfuscation (iO) and somewhere statistically binding (SSB) hash functions.
– Setting the share σ_n to $\gamma_n = \mu \oplus y_n$, where y_n is the next output bit produced by the projective PRG, requires to update the shares of v_n's right neighbors, as the projective keys given to these nodes needs to allow them to recover y_n. We solve this problem by repeating the construction in parallel two times, namely the share of party n is obtained by considering two independent executions of the construction by Applebaum *et al.* [2]: one in which v_n is a left node, and one in which v_n is a right node. This way, assuming rigidity, when a new party enters the system, we never need to update old shares.

The above yields a secret sharing scheme for the rigid evolving *bipartite* graphs access structure, with shares of size $\mathsf{poly}(\lambda)$, under either the RSA assumption or assuming iO and SSB hash functions. Additionally, the latter directly implies an evolving secret sharing scheme, with the same share size, for (non-bipartite) graphs access structures.

Monotone Circuits Access Structures. Finally, consider the case of monotone circuits access structures, in which the parties correspond to an input $x = (x_1, \ldots, x_n)$ for a boolean circuit C consisting of AND and OR gates with unbounded fan-in, and a subset of the parties $\mathcal{I}$ is authorized if $C(x_{\mathcal{I}}) = 1$, where $x_{\mathcal{I}}$ is the input associated to $\mathcal{I}$ (i.e., $x_i = 1$ if $i \in \mathcal{I}$ and $x_i = 0$ otherwise). For simplicity, let us assume[5] that the circuit alternates layers made only by either OR or AND gates, starting always with OR gates; hereafter, we refer to such circuits as AND-OR circuits. Applebaum *et al.* [2] give a construction of

[5] This assumption is essentially without loss of generality, at least if one is willing to pay an additive factor of n in the number of OR gates, which does not impact the final share size by too much. See Sect. 5.3 for more on this point.

a *succinct* secret sharing scheme for this access structure, based on projective PRGs. In particular, the share size in their construction grows with the number of gates in the circuit, which improves over the classical construction of Yao [37], in which the share size grows linearly with the number of wires in the circuit.

At a high level, the construction by Applebaum *et al.* [2] allocates to the i-th gate a secret key κ_i, and makes sure that a set of authorized parties corresponding to input $x \in \{0,1\}^n$ will be able to learn the keys of the gates that are satisfied by x, while all other keys remain secret. The keys associated to the OR gates are pseudorandom blocks from the output of the projective PRG, whereas the keys associated to the AND gates are the projective keys for the set $\mathcal{T}_i = \{j : i \to j \text{ in } C\}$ corresponding to the out-neighbors of the i-th gate. The share of the parties include the master public key mpk of the projective PRG, as well as a ciphertext for each AND gate, which essentially allows one to move from an OR gate to an AND gate during reconstruction; the only exception are the input OR gates, for which the secret key is a random label that is given in the clear to the associated players.

In Sect. 5.3, we extend the above construction to the evolving setting. Here, the underlying circuit evolves as wires and gates are added to it, whenever new players enter the system. Some care is needed when specifying how the circuit evolves: say that we have a circuit over $n-1$ inputs; when the n-th player arrives, monotonicity may forbid to add the corresponding input wire as input to an already existing AND gate (e.g., if the AND gate is an output gate). Still, we could add such wires as part of other AND and OR gates, or add new gates to the circuit. However, the latter cannot be done arbitrarily if we want to also consider rigidity, which, as explained above, is a necessary condition in order to have an evolving secret sharing scheme where old shares do not need to be updated. To allow more flexibility, we will consider a strict generalization in which the parties arrive in generations instead of one by one. We denote by $n_g \geq 1$ the number of parties in generation $g \geq 1$, so that $n = \sum_g n_g$. Hence, when the first generation arrives, the access structure is specified by a circuit $\hat{C}_1(x_1, \ldots, x_{n_1})$ for some AND-OR circuit $\hat{C}_1$; when the second generation arrives the access structure is specified by the circuit $\hat{C}_1(x_1, \ldots, x_{n_1}) \vee \hat{C}_2(x_1, \ldots, x_{n_1+n_2})$, where $\hat{C}_2$ is any AND-OR circuit such that $\hat{C}_2(x_1, \ldots, x_{n_1}, 0, \ldots, 0) = 0$. The above preserves monotonicity (as the output of the two circuits are input to an OR gate), and ensures rigidity (as any assignment $x_1, \ldots, x_{n_1}$ that does not satisfy $\hat{C}_1$ won't satisfy $\hat{C}_2$ unless some of the parties in $[n] \setminus [n_1]$ is present).[6]

Given the above characterization, our construction proceeds as follows. When the n-th generation begins, the dealer knows the circuit $\hat{C}_g$; say such a circuit is made of $m_g = m_g^{\vee} + m_g^{\wedge}$ gates, where $m_g^{\vee}$ (resp., $m_g^{\wedge}$) denotes the number of OR (resp., AND) gates. The dealer now distributes the shares to the parties of the g-th generation as in the construction by Applebaum *et al.* [2], with the only difference that the secret key associated to the input OR gates is defined using a standard PRG with unbounded polynomial stretch, evaluated using a seed

[6] An equivalent way to preserve monotonicity and rigidity is to assume that $\hat{C}_1(x_1, \ldots, x_{n_1}) = \hat{C}_2(x_1, \ldots, x_{n_1}, 0, \ldots, 0)$.

that is only known by the corresponding player. This way, the wires associated to old players can be used as inputs in the new circuits, without the need for updating old shares. Moreover, the dealer knows the sub-circuit representing each generation, and thus can define the secret keys associated to the OR gates using a fresh pair of keys $(\mathsf{mpk}_g, \mathsf{msk}_g)$ for a projective PRG with *bounded* output length $m_g^{\vee}$. By using the constructions of projective PRGs from Applebaum *et al.* [2], we can instantiate our scheme from either (i) the RSA assumption, or (ii) iO and SSB hash functions, or (iii) the DDH/BDDH assumption, or (iv) the LWE assumption, with different trade-offs in terms of share size (see Table 1).

We remark that the rigid evolving monotone circuits access structure captures several interesting evolving access structures as a special case, such as:

- The rigid evolving monotone conjunctive normal form (CNF) formulas access structure, in which the authorized subsets $\mathcal{I}$ are those corresponding to inputs $x_{\mathcal{I}}$ such that $\bigwedge_{i\in[m]} C_i$, where each clause C_i is a disjunction over a subset of the n players. Note that, while n increases, the clauses change over time. However, by monotonicity, no clause can be added to the access structure, and, by rigidity, no clause can be removed from the access structure. Thus, the number of clauses m is a fixed parameter of the access structure.
- The rigid evolving monotone disjunctive normal form (DNF) formulas access structure, in which the authorized subsets $\mathcal{I}$ are those corresponding to inputs $x_{\mathcal{I}}$ such that $\bigvee_{i\in[m]} C_i$, where each clause C_i is a conjunction over a subset of the n players. Note that, while n increases, the clauses change over time. However, by monotonicity, we can neither remove old clauses nor add new variables to old clauses, and, by rigidity, we can add new clauses, so long as each new clause including old inputs must also include at least one new input. Thus, the number of clauses m is not fixed, and grows over time together with the number of players.

While the above construction directly implies an evolving secret sharing scheme for rigid evolving CNF/DNF formulas access structures, in the full version [20], we give direct constructions which are slightly better in terms of share size and/or assumptions. More in details, for the case of CNF formulas, we again rely on projective PRGs with *bounded* polynomial stretch m, and obtain instantiations from assumptions (i)–(iv) listed above with different trade-offs in terms of share size (see Table 1). For the case of DNF formulas, we only rely on OWFs and get a scheme with shares size $t_n \cdot (\lambda + 1)$, where t_n is the number of clauses in which the input associated to player n appears.

Domain Extension. As our final contribution, we study the question of domain extension for evolving secret sharing schemes in the computational setting. Here, one starts with a secret sharing scheme supporting messages of size λ, for a rigid evolving access structure $\hat{\mathcal{A}}$, and the goal is to obtain a secret sharing scheme for the same access structure, but supporting messages of length $\ell \gg \lambda$.

In the non-evolving setting, this question was first studied by Krawczyk [27] for the case of threshold access structures. The main idea is to use a so-called information dispersal, which allows to divide the message into n fragments, in such a way that the message can be recovered from any $t \leq n$ fragments, while the size of each fragment is only ℓ/t (which is optimal). In other words, an information dispersal for the t-threshold access structure offers the same functionality of a secret sharing scheme for the same access structure, but without any privacy guarantee (which is the reason why the fragments can be shorter than the message). A simple example of information dispersal comes from Reed-Solomon codes: Parse the message μ into t blocks $\mu = (\mu_0, \ldots, \mu_{t-1})$, and interpret each block as an element of $\mathbb{GF}(q)$; if needed, the original message can be padded so that the message length ℓ is a multiple of the threshold t. Hence, let $f(X) = \mu_0 + \mu_1 \cdot X + \cdots + \mu_{t-1} \cdot X^{t-1}$ be the polynomial over $\mathbb{GF}(q)$, whose coefficients are the fragments of the message. The fragment γ_i assigned to party $i \in [n]$ is $f(i)$. Now, any subset of t parties can successfully reconstruct the polynomial, and thus recover the message. Moreover, the size of each fragment is $\log q = \ell/t$, which is optimal.

Given a secret sharing scheme (with domain $\{0,1\}^\lambda$) and an information dispersal for the t-threshold access structure, Krawczyk's compiler works as follows: First sample a random secret key $\kappa \in \{0,1\}^\lambda$ for a symmetric encryption scheme; then, encrypt the message $\mu \in \{0,1\}^\ell$. The share of party $i \in [n]$ is defined to be $\sigma'_i = (\sigma_i, \gamma_i)$, where σ_i is the i-th share of a secret sharing of the key κ, and γ_i is the i-th fragment of an information dispersal of the ciphertext γ. This results in shares of size $\ell/t + \lambda$, which is asymptotically optimal (as $\ell \to \infty$). In a follow-up work, Bèguin and Cresti [5] showed that the above construction still works assuming the underlying secret sharing scheme and information dispersal support an arbitrary access structure. Moreover, they observed that an information dispersal for access structure $\mathcal{A}$ can be obtained by dispersing the message using an information dispersal for the t-threshold access structure, where t is the minimum[7] size of an authorized set in $\mathcal{A}$.

In the full version [20], we show that Krawczyk's compiler works unmodified even in the evolving setting. Naturally, this requires to assume an information dispersal for any rigid evolving access structure $\mathcal{A}$. To this end, we first show how to obtain an information dispersal for the evolving t-threshold access structure. Basically, this is an erasure code where one can disperse the message into a growing number of fragments (potentially infinite), with the guarantee that the message can be recovered from any fraction of t fragments. When $n = \mathsf{poly}(\lambda)$, an easy solution comes again using Reed-Solomon codes, i.e. we simply disperse the message using the above defined polynomial $f(X)$ over an exponentially large field $\mathbb{GF}(2^\lambda)$. This allows to accommodate an arbitrary polynomial number of users. A drawback of this solution is that it requires an exponentially large field, and thus it is not very efficient in practice, as reconstruction takes quadratic (in t) time. This can be improved using more sophisticated techniques from coding theory. In particular, using so-called *digital fountains* [31] (e.g., Tornado codes [29], LT codes [28], or Raptor codes [36]), one can support a potentially

[7] This solution is optimal in terms of minimizing the maximum share size. Bèguin and Cresti also propose simple variants that minimize the total size of the shares, but for simplicity we do not consider these variants in our paper.

infinite number of players with reconstruction time that is only linear (in t). To the best of our knowledge, this is the first application of rateless codes in cryptography, and thus we believe our work establishes an interesting connection with information theory.

Given an information dispersal for the evolving t-threshold access structure, one can obtain an information dispersal for any rigid evolving access structure $\hat{\mathcal{A}}$ by setting the threshold to the size of a minimum authorized subset in $\hat{\mathcal{A}}$, as proposed by Bèguin and Cresti [5]. A small caveat is that the latter requires the dealer to know the size of a minimal authorized subset in $\hat{\mathcal{A}}$; while this assumption is for free for some evolving access structures (e.g., dynamic threshold access structures, graphs access structures, and CNF formulas access structures), it is not always true in general (e.g., in the case of DNF formulas and monotone circuits access structures). We prove that this limitation is somewhat inherent for general access structures: whenever a rigid evolving access structure $\hat{\mathcal{A}}$ is such that the minimal size of an authorized subset goes from t_1 (when there are n_1 parties) to $t_2 < t_1$ (when there are $n_2 > n_1$ parties), no information dispersal for $\hat{\mathcal{A}}$ can be optimal in terms of share size (i.e., have maximum share size ℓ/t, where t is the minimum size of an authorized subset) without updating old shares.

1.3 Additional Related Work

Paskin-Cherniavsky [32] gives a secret sharing scheme for arbitrary evolving access structures with slightly better (but still exponential) share size (compared to [24]. Dutta *et al.* [18] consider a simple generalization of the evolving threshold access structure in which parties belong to different compartments, and each compartment has associated a threshold specifying the minimum size of an authorized subset in that compartment. Both of these constructions require that the dealer knows the access structure in advance.

Desmedt, Dutta, and Morozov [17] give an interpretation of evolving secret sharing from the lens of so-called evolving perfect hash families. This is useful in order to obtain schemes where the message space is a non-abelian group.

Komargodski and Paskin-Cherniavsky further show how to generically transform any secret sharing scheme for the evolving t-threshold access structure into a scheme which is *robust* [34], where the latter means that the message can be recovered even if some parties hand-in incorrect shares.

2 Projective Pseudorandom Generators

We review the recent notion of *projective* pseudorandom generators (pPRGs) from Applebaum *et al.* [2], which will be used as a tool in some of our constructions. For space reasons, we refer the reader to the full version [20] for other auxiliary standard definitions.

Intuitively, a projective PRG is a PRG $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^m$ with the additional property that, given a master key msk and a subset $\mathcal{T} \subseteq [m]$, one can produce a *succinct*[8] projective key $\alpha_{\mathcal{T}}$ which can be used to recover the output bits $\mathsf{G}(\alpha)|_{\mathcal{T}}$, but reveals nothing about the other bits.

More formally, a projective PRG is a tuple of polynomial-time algorithms $\mathsf{pPRG} = (\mathsf{pPRG.Setup}, \mathsf{pPRG.KeyGen}, \mathsf{pPRG.Eval})$ specified as follows:

- The randomized setup algorithm $\mathsf{pPRG.Setup}(1^\lambda, 1^m)$ takes as input the security parameter $\lambda \in \mathbb{N}$ and an output length parameter $m \in \mathbb{N}$, and outputs public parameters mpk along with a master secret key msk.
- The deterministic key generation algorithm $\mathsf{pPRG.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathcal{T})$ takes as input the public parameters mpk, the master secret key msk, and a target set $\mathcal{T} \subseteq [m]$, and outputs a projective key $\alpha_{\mathcal{T}}$.
- The deterministic evaluation algorithm $\mathsf{pPRG.Eval}(\mathsf{mpk}, \alpha_{\mathcal{T}}, \mathcal{T})$ takes as input the public parameters mpk, a projective key $\alpha_{\mathcal{T}}$, and a target set $\mathcal{T} \subseteq [m]$, and outputs a string $y \in \{0,1\}^{|\mathcal{T}|}$.

Abusing notation, we write $\mathsf{pPRG.Eval}(\mathsf{mpk}, \mathsf{msk})$ to denote the output of the PRG corresponding to $\mathsf{pPRG.Eval}(\mathsf{mpk}, \alpha_{[m]}, [m])$ (i.e., when the target set $\mathcal{T}$ corresponds to the entire output length). A projective PRG is required to satisfy three properties. The first property is a correctness requirement saying that a projective key for target set $\mathcal{T}$ allows to learn the PRG output in the positions indexed by $\mathcal{T}$. The second property is a succinctness requirement saying that the size of a projective key for a target set $\mathcal{T}$ is significantly shorter than $|\mathcal{T}|$. The third property is a security requirement saying that an adversary obtaining a projective key for the union of different subsets $\mathcal{T}^*$ learn no information about the output of the PRG in the positions outside $\mathcal{T}^*$.

Definition 1 (Correctness of pPRGs). *We say that $\mathsf{pPRG} = (\mathsf{pPRG.Setup}, \mathsf{pPRG.KeyGen}, \mathsf{pPRG.Eval})$ is correct if for all $\lambda, m \in \mathbb{N}$, all $(\mathsf{mpk}, \mathsf{msk}) \in \mathsf{pPRG.Setup}(1^\lambda, 1^m)$, all subsets $\mathcal{T} \subseteq [m]$, and all projective keys $\alpha_{\mathcal{T}} = \mathsf{pPRG.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathcal{T})$, it holds that $y_{\mathcal{T}} = \mathsf{pPRG.Eval}(\mathsf{mpk}, \alpha_{\mathcal{T}}, \mathcal{T})$ equals all of the bits of $y = \mathsf{pPRG.Eval}(\mathsf{mpk}, \mathsf{msk})$ indexed by the positions in $\mathcal{T}$.*

Definition 2 (Succinctness of pPRGs). *We say that $\mathsf{pPRG} = (\mathsf{pPRG.Setup}, \mathsf{pPRG.KeyGen}, \mathsf{pPRG.Eval})$ is (fully) succinct if for all $\lambda, m \in \mathbb{N}$, all $(\mathsf{mpk}, \mathsf{msk}) \in \mathsf{pPRG.Setup}(1^\lambda, 1^m)$, and all subsets $\mathcal{T} \subseteq [m]$, it holds that the projective key $\alpha_{\mathcal{T}} = \mathsf{pPRG.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathcal{T})$ has size $|\alpha_{\mathcal{T}}| = \mathsf{poly}(\lambda, \log m)$.*

Definition 3 (Robustness of pPRGs). *We say that $\mathsf{pPRG} = (\mathsf{pPRG.Setup}, \mathsf{pPRG.KeyGen}, \mathsf{pPRG.Eval})$ is robust if for all PPT adversaries A it holds that $\{\mathbf{G}^{\mathsf{rob}}_{\mathsf{pPRG},\mathsf{A}}(\lambda, 0)\}_{\lambda \in \mathbb{N}} \approx_c \{\mathbf{G}^{\mathsf{rob}}_{\mathsf{pPRG},\mathsf{A}}(\lambda, 1)\}_{\lambda \in \mathbb{N}}$, where the game $\mathbf{G}^{\mathsf{rob}}_{\mathsf{pPRG},\mathsf{A}}(\lambda, b)$ is defined as follows:*

- *Given 1^λ, the adversary sends 1^m and $\mathcal{T}_1, \ldots, \mathcal{T}_q \subset [m]$ to the challenger.*
- *The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \leftarrow_\$ \mathsf{pPRG.Setup}(1^\lambda, 1^m)$ and lets $y_0 = y_{\overline{\mathcal{T}}}$ and $y_1 \leftarrow_\$ \{0,1\}^{|\overline{\mathcal{T}}|}$, where $\overline{\mathcal{T}} = [m] \setminus \mathcal{T}$, $\mathcal{T} = \mathcal{T}_1 \cup \cdots \cup \mathcal{T}_q$, and $y_{\overline{\mathcal{T}}} = \mathsf{pPRG.Eval}(\mathsf{mpk}, \mathsf{pPRG.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \overline{\mathcal{T}}), \overline{\mathcal{T}})$. Then, the challenger forwards $(\mathsf{mpk}, (\alpha_{\mathcal{T}_i})_{i \in [q]}, y_b)$ to A, where $\alpha_{\mathcal{T}_i} = \mathsf{pPRG.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathcal{T}_i)$.*

[8] Succinctness is a crucial requirement, as otherwise a projective key could just be $\mathsf{G}(\alpha)|_{\mathcal{T}}$.

2.1 Unbounded Polynomial Stretch

For some of our applications, we will require a stronger variant of projective PRGs in which the output length is an unbounded polynomial $m = \mathsf{poly}(\lambda)$. For concreteness, we define this variant below.

Definition 4 (pPRGs with unbounded stretch). *We say that* $\mathsf{pPRG} = (\mathsf{pPRG.Setup}, \mathsf{pPRG.KeyGen}, \mathsf{pPRG.Eval})$ *has* unbounded polynomial stretch *if algorithm* Setup *takes only the security parameter* $\lambda \in \mathbb{N}$ *as input, whereas algorithms* KeyGen *and* Eval *take as input target sets* $\mathcal{T}$ *of arbitrary polynomial size* $|\mathcal{T}| = \mathsf{poly}(\lambda)$.

The definitions of correctness, succinctness and robustness can be easily adapted to the case of projective PRGs with unbounded polynomial stretch. In particular, correctness is immediate, while succinctness still requires that the size of the projective keys $\alpha_{\mathcal{T}}$ are $\mathsf{poly}(\lambda)$, whereas the running time of algorithms KeyGen and Eval are $\mathsf{poly}(\lambda, |\mathcal{T}|)$. We remark that, when the stretch is unbounded, the master public key mpk is always succinct (i.e., of size $\mathsf{poly}(\lambda)$), as the setup algorithm does not depend on m.

As for robustness, the security game remains unchanged as the adversary can already specify the output length 1^m of the challenge. Note the adversary has to commit to 1^m and to the subsets $\mathcal{T}_1, \cdots, \mathcal{T}_q$ before receiving the master public key mpk. This flavor of "selective" security is sufficient for our applications.

Outputting Blocks. Sometimes, it is convenient to think of the pPRG output as a sequence of t blocks of size λ. In such a case, the key generation takes as input subsets $\mathcal{T} \subseteq [t]$ and generates projective keys corresponding to the blocks indexed by the positions in $\mathcal{T}$; the evaluation algorithm is modified analogously. The latter can be obtained by mapping $\mathcal{T}$ into $\mathcal{T}' \subseteq [t \cdot \lambda]$, where $\mathcal{T}'$ consists of the set of all location that fall inside the blocks whose indexes are in $\mathcal{T}$.

Instantiations. The following theorem summarizes known constructions of projective PRGs under a variety of assumptions.

Theorem 1 ([2]). *There exist constructions of projective PRGs from the following assumptions and with the following parameters:*

- *Under the RSA assumption, with* unbounded *polynomial stretch, and with master public keys and projective keys of size* $\mathsf{poly}(\lambda)$.
- *Assuming indistinguishability obfuscation and somewhere statistically binding hash functions, with* unbounded *polynomial stretch, and with projective keys of size* $\mathsf{poly}(\lambda)$ *(and empty master public keys).*
- *Under the DDH and the BDDH assumption, with* bounded *polynomial stretch, and with master public keys of size* $m^2 \cdot \mathsf{poly}(\lambda)$ *and projective keys of size* $O(\lambda)$. *In the second construction the master public key is reusable (i.e., it is independent of the master secret key).*
- *Under the LWE assumption, with* bounded *polynomial stretch, and with master public keys of size* $m \cdot \mathsf{poly}(\lambda)$ *and projective keys of size* $O(\lambda)$.

We remark that Applebaum *et al.* [2] actually prove a slightly different statement about the constructions based on RSA and on obfuscation. In particular, they prove that the first construction achieves sub-exponential stretch under the sub-exponential RSA assumption, and that the second construction achieves *bounded* polynomial stretch. Nevertheless, it is easy to adapt these constructions and show that they indeed achieve *unbounded* polynomial stretch under polynomial hardness. We refer the reader to [20] for more details.

3 Computational Evolving Secret Sharing

Let $n \in \mathbb{N}$, and denote by $[n] = \{1, \ldots, n\}$. A collection of subsets $\mathcal{A} \subseteq 2^{[n]}$ is *monotone* if for every $\mathcal{I} \in \mathcal{A}$, with $\mathcal{I} \subseteq \mathcal{I}'$, it holds that $\mathcal{I}' \in \mathcal{A}$.

Definition 5 (Access structure). *An access structure $\mathcal{A} \subseteq 2^{[n]}$ is a monotone collection of non-empty subsets. Subsets in $\mathcal{A}$ are called* qualified*, whereas subsets not in $\mathcal{A}$ are called* unqualified*.*

If $\mathcal{A}$ is an access structure, then a subset $\mathcal{B} \in \mathcal{A}$ is *minimal* if $\mathcal{B}' \notin \mathcal{A}$ whenever $\mathcal{B}' \subset \mathcal{B}$. We will typically assume $\mathcal{A}$ only consists of minimal authorized subsets, which we sometimes refer to as the minimal representation form.

Standard secret sharing schemes are typically defined in a setting where the number of parties n is fixed. In evolving secret sharing [24], instead, parties arrive one by one and thus the value n represents the number of parties currently in the system. In the information-theoretic setting, the number n can grow up to infinity; in the computational setting, instead, $n = \mathsf{poly}(\lambda)$. We denote by $\mathcal{A}_n$ the access structure when there are n parties in the system. We stress, that the dealer does not know the value n, neither it knows the access structure $\mathcal{A}_n$ before the n-th party enters the system.

Definition 6 (Evolving access structure). *Let $n \in \mathbb{N}$, with $n(\lambda) = \mathsf{poly}(\lambda)$. An evolving access structure $\mathcal{A} = \{\mathcal{A}_n\}$ is a monotone collection of subsets $\mathcal{A}_n \subseteq 2^{[n]}$ such that, for every $n \in \mathbb{N}$, it holds that $\mathcal{A}_n \subseteq \mathcal{A}_{n+1}$.*

We are now ready to define evolving secret sharing schemes. A secret sharing scheme $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Recon})$, with message space $\mathcal{M}$ and share space $\mathcal{S}$, for an evolving access structure $\mathcal{A}$ consists of two polynomial-time algorithms specified as follows (we implicitly assume all algorithms take as input the security parameter 1^λ):

- The randomized sharing algorithm $\mathsf{SS.Share}(\mu, (\sigma_i)_{i \in [n-1]}, \mathcal{A}_n)$ takes as input a message $\mu \in \mathcal{M}$, a collection of $n-1$ shares $\sigma_1, \ldots, \sigma_{n-1} \in \mathcal{S}$, and an access structure $\mathcal{A}_n$, and outputs the share σ_n for the n-th party.
- The deterministic reconstruction algorithm $\mathsf{SS.Recon}((\sigma_i)_{i \in \mathcal{I}}, \mathcal{I}, \mathcal{A}_n)$ takes as input a collection of shares $(\sigma_i)_{i \in \mathcal{I}}$, along with a subset $\mathcal{I} \subseteq 2^{[n]}$ of the parties, and outputs a message $\mu \in \mathcal{M}$.

The share size $s(n, \ell, \lambda)$ of party n in a secret sharing scheme for evolving access structure $\mathcal{A}$ is defined as the maximum of $|\sigma_n|$ over all messages of length $\ell = \ell(n)$ and over all possible previous assignments $\sigma_1, \ldots, \sigma_{n-1}$. As considered in [2], we are interested in building *succinct* computational secret sharing schemes, i.e., schemes with share size smaller than the representation of representation of the access structure structure.

Remark 1 (On representing the access structure). When dealing with efficiency of computational secret sharing schemes, it is important to define how an evolving access structure is represented. In particular, we can associate to each access structure $\mathcal{A}_n$ over n parties a boolean function $f_n : \{0,1\}^n \to \{0,1\}$ such that $f(x) = 1$ if and only if the set $\mathcal{I}_x$ consisting of all the parties in $[n]$ for which $x_i = 1$ is such that $\mathcal{I}_x \in \mathcal{A}_n$. Hence, we assume there is a universal (polynomial-time computable) representation model $\mathsf{U} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ such that $\mathsf{U}(\Pi_n, x) = f_n(x)$ for all $n \geq 1$ and for all $x \in \{0,1\}^n$, where Π_n is a program representing the function f_n. For simplicity, in the rest of the paper, we implicitly assume that the sharing and reconstruction algorithm take as input the program Π_n representing the function f_n corresponding to the access structure $\mathcal{A}_n$; sometimes, we abuse notation and write $\mathcal{A}_n$ instead of the program Π_n.

3.1 Defining Computational Privacy

Evolving secret sharing schemes are required to satisfy the following two properties. The first property is a correctness requirement saying that, at any point in time, qualified subsets of parties can recover the message. The second property is a security requirement saying that, at any point in time, a computationally bounded adversary knowing the shares corresponding to an unqualified subset of parties obtains no information about the message.

Definition 7 (Correctness of evolving secret sharing). *We say that a secret sharing scheme SS over message space $\mathcal{M}$ for the evolving access structure $\mathcal{A}$ is correct if for every message $\mu \in \mathcal{M}$, for every number of parties $n \geq 1$, and for every qualified subset $\mathcal{I} \in \mathcal{A}_n$ it holds that*

$$\Pr\left[\mu = \mu' : \begin{array}{c} \forall i \leq n : \sigma_i \leftarrow\!\!\$\ \mathsf{SS.Share}(\mu, (\sigma_j)_{j<i}, \mathcal{A}_i) \\ \mu' = \mathsf{SS.Recon}((\sigma_i)_{i \in \mathcal{I}}, \mathcal{I}) \end{array}\right] = 1.$$

Definition 8 (Privacy of evolving secret sharing). *We say that a secret sharing scheme SS over message space $\mathcal{M}$ for the evolving access structure $\mathcal{A}$ is computationally private if $\{\mathbf{G}^{\mathsf{priv}}_{\mathsf{SS},\mathsf{A}}(\lambda, 0)\}_{\lambda \in \mathbb{N}} \approx_c \{\mathbf{G}^{\mathsf{priv}}_{\mathsf{SS},\mathsf{A}}(\lambda, 1)\}_{\lambda \in \mathbb{N}}$, where the game $\mathbf{G}^{\mathsf{priv}}_{\mathsf{SS},\mathsf{A}}(\lambda, b)$ is defined as follows:*

- *The adversary chooses a pair of messages $\mu_0, \mu_1 \in \mathcal{M}$, an integer $n \geq 1$, and an unqualified subset $\mathcal{U} \notin \mathcal{A}_n$, and forwards them to the challenger.*
- *The challenger computes $\sigma_i \leftarrow\!\!\$\ \mathsf{SS.Share}(\mu_b, (\sigma_j)_{j<i}, \mathcal{A}_i)$ for all $i \leq n$, and forwards $(\sigma_i)_{i \in \mathcal{U}}$ to A.*
- *Finally, A outputs a bit $b' \in \{0,1\}$ which is also the output of the experiment.*

3.2 Rigid Access Structures

We remark that our definition of evolving access structures is less stringent than previous definitions considered in the literature (see, e.g., [24]). In particular, previous definitions require that if at some point there is an unqualified subset $\mathcal{U} \subseteq 2^{[n]}$, where n is the current number of parties, then the set $\mathcal{U}$ will always remain unqualified when more parties are added to the evolving access structure. In contrast, Definition 6 potentially allows sets that previously were unqualified to become qualified at a later point in time (without necessarily involving new players). The theorem below states that Definition 6 is impossible to achieve unless the dealer is allowed to update the shares.

Theorem 2. *Let $\mathcal{A}$ be an evolving access structure such that there exist indexes n_1, n_2, with $n_1 < n_2$, along with a subset $\mathcal{U} \in 2^{[n_1]}$ which satisfy the following conditions: (i) $\mathcal{U} \notin \mathcal{A}_{n_1}$; (ii) $\mathcal{U} \in \mathcal{A}_{n_2}$. Then, any computationally private secret sharing scheme for $\mathcal{A}$ requires to update the shares of party n_1 when party n_2 enters the system.*

Proof. Let $\mathcal{A}$, n_1, and n_2 be as in the statement of the theorem. Fix an arbitrary message $\mu \in \mathcal{M}$, and denote by $(\sigma_i)_{i\in[n_2]}$ the shares obtained by sharing μ among n_2 parties. Condition (ii) on the access structure $\mathcal{A}$, along with the correctness property of SS, imply that $\mathsf{SS.Recon}((\sigma_i)_{i\in\mathcal{U}}, \mathcal{U})$ outputs μ with probability one over the randomness of the sharing algorithm $\mathsf{SS.Share}$.

Now, consider the adversary A that plays the computational privacy game by choosing messages $\mu_0 = \mu$ and $\mu_1 = \mu' \neq \mu$, number of parties n_1, and subset $\mathcal{U}$. The adversary obtains $(\sigma_i)_{i\in\mathcal{U}}$ from the challenger and outputs $b' = 1$ if and only if $\mathsf{SS.Recon}((\sigma_i)_{i\in\mathcal{U}}, \mathcal{U}) = \mu$. By the above argument, A wins with probability one. Moreover, condition (i) implies that $\mathcal{U}$ is unqualified, and thus A is valid. This finishes the proof.

An evolving access structure that does not meet the properties (i) and (ii) in Theorem 2 (i.e., if $\mathcal{U} \notin \mathcal{A}_i$ for some i, then $\mathcal{U} \notin \mathcal{A}_j$ for all $j > i$) is called *rigid*. The definition below, which also appears in [24], formalizes this property.

Definition 9 (Rigid evolving access structure). *A* rigid *evolving access structure $\mathcal{A} = \{\mathcal{A}_n\}$ is a monotone collection of subsets $\mathcal{A}_n \subseteq 2^{[n]}$ such that, for any $n \in \mathbb{N}$, it holds that $\mathcal{A}_n = \mathcal{A} \cap [n]$.*

Since in this paper we are not interested in evolving secret sharing schemes in which the dealer can update the shares, in what follows we only focus on rigid evolving access structures. Throughout the paper, we use the hat symbol $\hat{\mathcal{A}}$ to denote access structures that evolve while preserving rigidity.

4 Construction for General Access Structures

4.1 Exponential-Time Construction

We present a construction of a secret sharing scheme for any rigid evolving access structure. The construction is based on any PRG with unbounded polynomial stretch (and thus only requires one-way functions). Unfortunately, for certain access structures, the running time of the sharing algorithm in our construction may be exponential, and thus we cannot prove computational privacy for all rigid evolving access structures. Nevertheless, we show that the construction runs in polynomial time for a fairly natural family of rigid evolving access structures, and in this case we can also prove computational privacy.

Construction 1

Let $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^*$ be a PRG. For a seed $\kappa \in \{0,1\}^\lambda$, we parse the output of $\mathsf{G}(\kappa)$ into blocks of size λ, and we denote with $\mathsf{G}(\kappa)[i]$ the i-th such block. Consider the following secret sharing scheme $SS = (SS.\mathsf{Share}, SS.\mathsf{Recon})$ over message space $\mathcal{M} = \{0,1\}^\lambda$ for an arbitrary rigid evolving access structure $\hat{\mathcal{A}}$. We assume a lexicographic order $\xi : 2^{[n]} \to \mathbb{N}$ over the subsets in $\hat{\mathcal{A}}$, and that $\hat{\mathcal{A}}$ is in its minimal representation form.

Sharing: When the n-th party arrives, the dealer proceeds as follows:

- Sample $\kappa_n \leftarrow\!\!\$\ \{0,1\}^\lambda$.
- For each $\mathcal{I} \in \hat{\mathcal{A}}_n \setminus \hat{\mathcal{A}}_{n-1}$, let $\mathcal{I} = (i_1, \ldots, i_{t-1}, n)$ for some $t \geq 1$ and compute $\gamma_\mathcal{I} = \mu \bigoplus_{j=1}^{t-1} \mathsf{G}(\kappa_{i_j})[\xi(\mathcal{I})]$.
- Return $\sigma_n = (\kappa_n, (\gamma_\mathcal{I})_{\mathcal{I} \in \hat{\mathcal{A}}_n \setminus \hat{\mathcal{A}}_{n-1}})$ to the party.

Reconstruction: The reconstruction algorithm $SS.\mathsf{Recon}((\sigma_i)_{i\in\mathcal{I}}, \mathcal{I})$, given the shares $(\sigma_i)_{i\in\mathcal{I}}$ such that $\sigma_i = (\kappa_i, (\gamma_\mathcal{I})_{\mathcal{I}\in\hat{\mathcal{A}}_i\setminus\hat{\mathcal{A}}_{i-1}})$, and a minimal authorized subset $\mathcal{I} = \{i_1, \ldots, i_{t-1}, n\} \in \hat{\mathcal{A}}_n$, returns the same as $\gamma_\mathcal{I} \bigoplus_{j=1}^{t-1} \mathsf{G}(\kappa_{i_j})[\xi(\mathcal{I})]$, where $\gamma_\mathcal{I}$ is taken from the share σ_n of party n.

Recall that, by rigidity, every new authorized subset $\mathcal{I} \in \hat{\mathcal{A}}_n \setminus \hat{\mathcal{A}}_{n-1}$ must include the new party n. Correctness then follows by the fact that each of the party i_j in $\mathcal{I}$ knows the seed κ_{i_j}, and moreover can determine the correct index $\xi(\mathcal{I})$ given the lexicographic order ξ of the access structure $\hat{\mathcal{A}}$. However, the the sharing algorithm requires to generate a ciphertext $\gamma_\mathcal{I}$ for each minimal authorized subset $\mathcal{I} \in \hat{\mathcal{A}}_n \setminus \hat{\mathcal{A}}_{n-1}$ that party n completes, which can be exponential in n, and thus one cannot prove computational security of the above construction for arbitrary rigid evolving access structures.

4.2 Polynomial-Time Instantiation

We note that if the number of authorized subsets that each new arriving party completes is $\mathsf{poly}(\lambda, n)$, then the sharing and reconstruction algorithms in Construction 1 always run in polynomial time. The size of each share is $\mathsf{poly}(\lambda, n)$, and becomes $\mathsf{poly}(\lambda)$ in case the number of added authorized subsets is independent of the number of parties currently in the system n. Moreover, in this case, the above secret sharing scheme is also computationally private. Below, we report the formal result whose proof appears in the full version of this work [20].

Theorem 3. *Assuming G is secure, then the scheme SS described in Construction 1 is a computationally private secret sharing scheme over $\mathcal{M} = \{0,1\}^\lambda$ for every rigid evolving access structure $\hat{\mathcal{A}}$ such that, for each $n \geq 1$, it holds that $|\hat{\mathcal{A}}_n| - |\hat{\mathcal{A}}_{n-1}| = \mathsf{poly}(\lambda, n)$.*

While the above construction is general, it fails to capture many important evolving access structures in which the number of added authorized subsets is super-polynomial in the number of parties (e.g., the dynamic threshold access structure). We will provide more efficient solutions for many of these access structures in the following section.

Remark 2 (On Mazor's lower bound [30]). Mazor [30] demonstrated that there exists an evolving access structure $\hat{\mathcal{A}} = \{\mathcal{A}_n\}$ for which every information-theoretic evolving secret sharing scheme for $\mathcal{A}$ (where the dealer does not know the access structure in advance) is such that the share size of the first n parties is at least $2^{n-o(n)}$. In particular, this is achieved by the access structure $\hat{\mathcal{A}}$ such that $\mathcal{A}_n$ is empty for every n. Note that the number of authorized subsets that the n-th party completes in the above access structure $\hat{\mathcal{A}}$ is 0, and thus our Construction 1 gives a computational evolving secret sharing scheme for $\hat{\mathcal{A}}$ with share size λ, thus circumventing Mazor's lower bound. This is because, in our scheme applied to $\hat{\mathcal{A}}$, the dealer gives a short PRG seed to every party (independently whether it is authorized or not) that can be stretched in the future when needed, i.e., when a party will later be part of an authorized set. The same approach cannot be used in the information-theoretic setting), as it would force the dealer to include an exponential number of random pads in the share of each party (yielding the bound $2^{n-o(n)}$) to make the latter ready to become eventually authorized in the future.

5 Constructions for Specific Access Structures

5.1 Dynamic Threshold Access Structure

In this section, we focus on the so-called dynamic threshold evolving access structure [26]. This access structure is a generalization of the t-threshold evolving access structure, in which the authorized parties consist of all subsets of at least t parties, where $t = O(1)$ is fixed and independent of n. In the more general case of the dynamic threshold evolving access structure $\hat{\mathcal{A}}^{\mathsf{dthr}}$, instead, we have a sequence of thresholds $t_1 \leq t_2 \leq \cdots$, such that when there are n parties the qualified sets are those of size at least t_n; note that now, at least in general, the thresholds can depend on n. The condition that $t_n \geq t_{n-1}$ is necessary to ensure monotonicity, namely for the sequence of access structures to be a valid evolving access structure. Moreover, by definition, $\hat{\mathcal{A}}^{\mathsf{dthr}}$ is automatically rigid.

Below, we give a construction of secret sharing for $\hat{\mathcal{A}}^{\mathsf{dthr}}$ with message space $\{0,1\}^\lambda$ and share size $\lambda \cdot (n+1)$. The scheme is based on Shamir's secret sharing and on any standard PRG with unbounded polynomial stretch.

Construction 2

Let $\mathbb{GF}(2^\lambda)$ be a field of size 2^λ, and $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^*$ be a standard PRG with unbounded polynomial stretch. For a seed $\kappa \in \{0,1\}^\lambda$, we parse the output of $\mathsf{G}(\kappa)$ into blocks of size λ, and we denote with $\mathsf{G}(\kappa)[i]$ the i-th such block. Consider the following secret sharing scheme $SS = (SS.\mathsf{Share}, SS.\mathsf{Recon})$ over message space $\mathcal{M} = \{0,1\}^\lambda$ for the dynamic threshold access structure $\hat{\mathcal{A}}^{\mathsf{dthr}}$ with sequence of thresholds $t_1 \leq t_2 \leq \cdots$.

Sharing: When the n-th party arrives, the dealer proceeds as follows:

- Sample a random polynomial f_n of degree $t_n - 1$ over $\mathbb{GF}(2^\lambda)[X]$, subject to $f_n(0) = \mu$.
- For every $i \in [n-1]$, compute $\gamma_i = f_n(i) \oplus \mathsf{G}(s_i)[n-i]$.
  Finally, set the share of the n-th player to:

$$\sigma_n = (s_n, (\gamma_i)_{i<n}, f_n(n))$$

  where $s_n \leftarrow\$ \{0,1\}^\lambda$.

Reconstruction: The reconstruction algorithm $SS.\mathsf{Recon}((\sigma_i)_{i\in\mathcal{I}}, \mathcal{I})$, given the shares $(\sigma_i)_{i\in\mathcal{I}} = ((s_i, (\gamma_j)_{j<i}, y_i))_{i\in\mathcal{Q}}$, and a minimal authorized subset $\mathcal{I} = \{i_1, \ldots, i_{t_{n-1}}, n\} \in \hat{\mathcal{A}}^{\mathsf{dthr}}_n$, proceeds as follows:

- For every $j \in [t_n - 1]$, compute $y'_j = \gamma_{i_j} \oplus \mathsf{G}(s_{i_j})[n - i_j]$ where $(\gamma_k)_{k<n}$ are the ciphertexts contained in σ_n.
- Use Lagrange interpolation over the points $((i_1, y'_1), \ldots, (i_{t_n-1}, y'_{t_n-1}), (n, y_n))$, yielding a polynomial $f' \in \mathbb{GF}(2^\lambda)[X]$, where $y_n = f_n(n)$ is the (plaintext) evaluation contained in σ_n.

Finally, output $f'(0) = \mu$.

Let $\mathcal{I} = \{i_1, \ldots, i_{t_n-1}, n\} \in \hat{\mathcal{A}}^{\mathsf{dthr}}_n$ be a minimal authorized subset, i.e., $\mathcal{I}$ is of size t_n. Correctness follows by observing that, for $j \in [n-1]$, each ciphertext γ_{i_j} contained in σ_n can be correctly decrypted by means of the seed s_{i_j} contained in σ_{i_j}, thus yielding $y'_j = \gamma_{i_j} \oplus \mathsf{G}(s_{i_j})[n - i_j] = f_n(i_j)$. Hence, the authorized parties can retrieve t_n evaluations (as $f_n(n) = y_n$ is already contained in σ_n) of the polynomial f_n of degree $t_n - 1$. Thus, by correctness of Lagrange interpolation, the parties in $\mathcal{I}$ can correctly reconstruct $f' = f_n$ and, in turn, recover the correct message $f'(n) = f_n(0) = \mu$. As for security, we prove the following result. We refer the reader to [20] for the corresponding proof.

Theorem 4. *Assuming G is secure, the scheme SS described in Construction 2 is a computationally private secret sharing scheme for the dynamic threshold evolving access structures $\hat{\mathcal{A}}^{\mathsf{dthr}}$.*

Remark 3 (Static threshold). A special case of the dynamic threshold access structure is the (static) t-threshold access structure $\hat{\mathcal{A}}_t^{\mathsf{thr}}$, i.e., the threshold t is fixed and independent of n (i.e., $t_1 = t_2 = \cdots = t = O(1)$). Of course, Construction 2 yields a secret sharing scheme for $\hat{\mathcal{A}}_t^{\mathsf{thr}}$, with share size $\lambda \cdot (n+1)$.[9] However, the share size can be improved with a direct construction based on Shamir's secret sharing scheme: The dealer samples a single random polynomial f over $\mathbb{GF}(2^\lambda)$, of degree $t-1$, and subject to $f(0) = \mu$. Then, for each $n \geq 1$, the share of party n is simply $\sigma_n = f(n)$. The share size is λ. Note that this works because, in the computational setting, the number of parties is upper bounded by an unknown polynomial, but the field is of exponential size.

Remark 4 (Flexible dynamic threshold). Consider the evolving *flexible* dynamic threshold access structure $\hat{\mathcal{A}}^{\mathsf{flex\text{-}dthr}} = \{t_n\}$, in which the *new* authorized subsets when the n-th party arrives (i.e., the subsets in $\hat{\mathcal{A}}_n^{\mathsf{flex\text{-}dthr}} \setminus \hat{\mathcal{A}}_{n-1}^{\mathsf{flex\text{-}dthr}}$) are all the subsets of at least t_n players *that always include* n. We note that this access structure is still monotone, even if we remove the condition that $t_1 \leq \cdots \leq t_n$; in fact, when the latter condition is added, the access structure $\hat{\mathcal{A}}^{\mathsf{flex\text{-}dthr}}$ collapses to $\hat{\mathcal{A}}^{\mathsf{dthr}}$. One can show that Construction 2 yields a computationally private secret sharing scheme for $\hat{\mathcal{A}}^{\mathsf{flex\text{-}dthr}}$, with exactly the same parameters. See the proof of Theorem 4 for more on this point.

Corollary 1. *Assuming OWFs, there exists a computationally private secret sharing scheme over $\mathcal{M} = \{0,1\}^\lambda$ for the evolving flexible dynamic threshold access structure $\hat{\mathcal{A}}^{\mathsf{flex\text{-}dthr}}$, where the share size of party $n \geq 1$ is $\lambda \cdot (n+1)$.*

5.2 Graphs

We start by recalling the concept of secret sharing schemes for graph access structures. Given an undirected graph $G = (\mathcal{V}, \mathcal{E})$, we view the parties as the vertexes in $\mathcal{V}$ and authorized sets are those sets that contain vertexes such that there is at least a pair of vertexes corresponding to an edge in the graph. Namely, the access structure is specified by a function $f_G : \{0,1\}^n \to \{0,1\}$, where $n = |\mathcal{V}|$, and $f_G(x) = 1$ if and only if there exist indexes $i, j \in [n]$ such that $x_i = x_j = 1$ and $(i,j) \in \mathcal{E}$.

We can generalize the above access structure to the evolving setting as follows. W.l.o.g., parties are added to the graph in an online manner; every new

[9] Note that a secret sharing scheme for $\hat{\mathcal{A}}_t^{\mathsf{thr}}$ can also be obtained as a special case of our construction for rigid evolving DNF formulas access structures (see [20]), by taking $n_1 = t$, $n_g = 1$ for every $g \geq 2$, and $m_g = \binom{n}{t}$ for every $g \geq 1$. However, the latter yields an even worse share size of $\mathsf{poly}(\lambda, n)$.

player is added to the vertex set $\mathcal{V}$ and can be connected via one or more edges to the previous nodes in the graph. However, no new edges can be added between old nodes in the set $\mathcal{V}$, i.e. we only consider the rigid evolving graphs access structure, which we denote by $\hat{\mathcal{A}}^{\mathsf{graph}} = \{\mathcal{E}_n\}$, where $G_n = (\mathcal{V}_n, \mathcal{E}_n)$ is the graph when there are n parties. By Theorem 2, the above limitation is inherent as the evolving secret sharing scheme we describe below does not require to update the shares of old players.

Following [7], we focus on the simpler case where the graph $G_n = (\mathcal{V}_n, \mathcal{E}_n)$ is bipartite, namely the vertex set $\mathcal{V}_n$ consists of two sets $(\mathcal{V}_n^{(0)}, \mathcal{V}_n^{(1)})$, and the edge set $\mathcal{E}_n$ is of the form $\mathcal{E}_n \subseteq \mathcal{V}_n^{(0)} \times \mathcal{V}_n^{(1)}$. This naturally extends to the evolving setting as well (keeping the rigidity condition), and we write $\hat{\mathcal{A}}^{\mathsf{2graph}} = \{\mathcal{E}_n\}$ to denote the rigid evolving bipartite graphs access structure. We will later show that secret sharing for $\hat{\mathcal{A}}^{\mathsf{2graph}}$ implies secret sharing for $\hat{\mathcal{A}}^{\mathsf{graph}}$.

Construction 3

Let $\mathsf{pPRG} = (\mathsf{pPRG.Setup}, \mathsf{pPRG.KeyGen}, \mathsf{pPRG.Eval})$ be a pPRG with unbounded polynomial stretch. Consider the following secret sharing scheme $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Recon})$ over message space $\mathcal{M} = \{0, 1\}$ for the rigid evolving bipartite graphs access structure $\hat{\mathcal{A}}^{\mathsf{2graph}}$.

Sharing: At the onset, the dealer computes $(\mathsf{mpk}_b, \mathsf{msk}_b) \leftarrow\!\!\$\, \mathsf{pPRG.Setup}(1^\lambda)$ for every $b \in \{0, 1\}$. When the n-th party arrives, the dealer proceeds as follows:

- Let $v_n \in \mathcal{V}_n^{(b)}$ (for some $b \in \{0, 1\}$) be the node in the graph corresponding to the n-th party, where $G_n = ((\mathcal{V}_n^{(0)}, \mathcal{V}_n^{(1)}), \mathcal{E}_n)$ is the graph specified in $\hat{\mathcal{A}}_n^{\mathsf{2graph}}$.
- Run $\alpha_{\{n\}} = \mathsf{pPRG.KeyGen}(\mathsf{mpk}_b, \mathsf{msk}_b, \{n\})$ and $y_n = \mathsf{pPRG.Eval}(\mathsf{mpk}_b, \alpha_{\{n\}}, \{n\})$, and let $\gamma_n = \mu \oplus y_n$.
- Let $\mathcal{T}_n = \{j : (v_n, v_j) \in \mathcal{E}_n\}$ be the set of indexes corresponding to the neighbors of the node $v_n \in \mathcal{V}_n^{(b)}$ in the graph. Let $\alpha_{\mathcal{T}_n} = \mathsf{pPRG.KeyGen}(\mathsf{mpk}_{1-b}, \mathsf{msk}_{1-b}, \mathcal{T}_n)$.
- Return $\sigma_n = (\mathsf{mpk}_{1-b}, \gamma_n, \alpha_{\mathcal{T}_n})$.

Reconstruction: The reconstruction algorithm $\mathsf{SS.Recon}((\sigma_i)_{i\in\mathcal{I}}, \mathcal{I})$, given the shares $(\sigma_i)_{i\in\mathcal{I}} = ((\mathsf{mpk}, \gamma_i, \alpha_{\mathcal{T}_i}))_{i\in\mathcal{I}}$ (for some $b \in \{0, 1\}$), and a minimal authorized subset $\mathcal{I} = \{v_{i_0}, v_{i_1}\}$ such that $(v_{i_0}, v_{i_1}) \in \mathcal{E}_n = \hat{\mathcal{A}}_n^{\mathsf{2graph}}$, proceeds as follows:

- W.l.o.g., assume that $v_{i_0} \in \mathcal{V}_n^{(b)}$ and $v_{i_1} \in \mathcal{V}_n^{(1-b)}$ (otherwise, simply swap v_{i_0} and v_{i_1}). Hence, $\sigma_0 = (\mathsf{mpk}_{1-b}, \gamma_{i_0}, \alpha_{\mathcal{T}_{i_0}})$ and $\sigma_1 = (\mathsf{mpk}_b, \gamma_{i_1}, \alpha_{\mathcal{T}_{i_1}})$

- If $i_0 < i_1$, compute $y' = \mathsf{pPRG.Eval}(\mathsf{mpk}_b, \alpha_{\mathcal{T}_{i_1}}, \mathcal{T}_{i_1})$ and $\mu = \gamma_{i_0} \oplus y'_k$, where y'_k is the k-th bit of y' such that the k-th index $i_k \in \mathcal{T}_{i_1}$ is equal to i_0.
- Otherwise, if $i_1 < i_0$, compute $y' = \mathsf{pPRG.Eval}(\mathsf{mpk}_{1-b}, \alpha_{\mathcal{T}_{i_0}}, \mathcal{T}_{i_0})$ and $\mu = \gamma_{i_1} \oplus y'_k$, where y'_k is the k-th bit of y' such that the k-th index $i_k \in \mathcal{T}_{i_0}$ is equal to i_1.
- Output μ.

Correctness follows readily from the correctness property of the underlying projective pPRG. More in details, fix two vertexes v_{i_1} and v_{i_0} such that $(v_{i_0}, v_{i_1}) \in \mathcal{E}_n$ and $v_{i_j} \in \mathcal{V}_n^{(j)}$ for $j \in \{0,1\}$. Assume $i_0 < i_1$ (the case $i_1 < i_0$ follows by using a symmetrical argument). Then, we have that $\gamma_{i_0} = \mu \oplus y_{i_0}$ (which is part of σ_{i_0}), where $y_{i_0} = \mathsf{pPRG.Eval}(\mathsf{mpk}_0, \alpha_{i_0}, \{i_0\})$. Moreover, by definition, we have that $\alpha_{\mathcal{T}_{i_1}} = \mathsf{pPRG.KeyGen}(\mathsf{mpk}_0, \mathsf{msk}_0, \mathcal{T}_{i_1})$ (which is part of σ_{i_1}), where $i_0 \in \mathcal{T}_{i_1}$. By correctness of the projective PRG, we conclude that the output $\mu = \gamma_{i_0} \oplus y'_k$ is correct, since y'_k is the k-th bit of $y' = \mathsf{pPRG.Eval}(\mathsf{mpk}_0, \alpha_{\mathcal{T}_{i_1}}, \mathcal{T}_{i_1})$ such that $i_0 = i_k \in \mathcal{T}_{i_1}$ (i.e., $y'_k = y_{i_0}$). As for security, we prove the following theorem. The formal proof is deferred to the full version of this work [20].

Theorem 5. *Assuming* pPRG *is robust (Definition 3), then the scheme* SS *described in Construction 3 is a computationally private secret sharing scheme over* $\mathcal{M} = \{0,1\}$ *for the the rigid evolving bipartite graphs access structure* $\hat{\mathcal{A}}^{\mathsf{2graph}}$.

For each $n \geq 1$, the share σ_n in Construction 3 consists of a one-bit ciphertext γ_n, a projective key $\alpha_{\mathcal{T}_n}$, and the master public key mpk of the projective PRG. Moreover, while the construction only deals with message space $\mathcal{M} = \{0,1\}$, it is immediate to obtain a scheme for $\mathcal{M} = \{0,1\}^\lambda$ by repeating the construction λ times in parallel. Hence, by invoking Theorem 1, we obtain:

Corollary 2. *Under the RSA assumption, or assuming iO and SSB hash functions, there exists a computationally private secret sharing scheme over* $\mathcal{M} = \{0,1\}^\lambda$ *for the rigid evolving bipartite graphs access structure* $\hat{\mathcal{A}}^{\mathsf{2graph}}$*, where the share size of party* $n \geq 1$ *is* $\mathsf{poly}(\lambda)$.

Arbitrary Graphs. As observed in [7], in the non-evolving setting, secret sharing schemes for bipartite graph access structures imply ones for arbitrary graph access structures. In more details, given a graph $G = (\mathcal{V}, \mathcal{E})$ one can construct a bipartite graph H by taking two copies of each vertex, and for every $(u, v) \in \mathcal{E}$ connect the first copy of u to the second copy of v; the share of each party in $\mathcal{V}$ consists of the shares of the corresponding two copies of this vertex in H. The above transformation clearly preserves rigidity, and thus readily adapts to the setting of rigid evolving graphs access structures. Thus, by leveraging Corollary 2, we obtain:

Corollary 3. *Under the RSA assumption, or assuming iO and SSB hash functions, there exists a computationally private secret sharing scheme over $\mathcal{M} = \{0,1\}^\lambda$ for the rigid evolving graphs access structure $\hat{\mathcal{A}}^{\mathsf{graph}}$, where the share size of party $n \geq 1$ is $\mathsf{poly}(\lambda)$.*

5.3 Monotone Circuits

Next, we focus on access structures represented as circuits $C : \{0,1\}^n \to \{0,1\}$ with AND and OR gates of unbounded fan-in, which we refer to as an AND-OR circuit. We denote by $x = (x_1, \ldots, x_n)$ the input to the circuit, where $x_i = 1$ means that the i-th player is part of the reconstruction. Following [2], we make some conventions on the structure of the circuit.

- We assume w.l.o.g. that all the outgoing wires of an OR gate are connected as incoming wires to AND gates, and viceversa; if this is not the case and, say, an OR gate has an outgoing wire that enters another OR gate, we can duplicate all the input wires of the first OR gate and connect them directly to the second OR gate. (The same can be done with AND gates.)
- We assume, for simplicity, that each input wire is only[10] connected to an OR gate with fan-in 1; this can be achieved by adding an OR gate with fan-in 1 for every input x_i that goes into an AND gate with fan-in ≥ 2, and by adding both an OR gate with fan-in 1 followed by an AND gate with fan-in 1 for every input x_i that goes into an OR gate with fan-in ≥ 2 (at the cost of increasing the number of gates by at most $2n$).
- We assume that gates are numbered from 1 to m according to some topological order and that the first n gates correspond to the inputs $x_1, \ldots, x_n$. We write $i \to j$ when an output of the i-th gate is being fed to the j-th gate as an input.

As explained in the introduction, we can generalize the above to the evolving setting as follows. The rigid evolving monotone circuits access structure $\hat{\mathcal{A}}^{\mathsf{ckts}} = \{\hat{\varphi}_g\}_{g \geq 1}$ consists of a sequence of monotone formulas $\hat{\varphi}_g$ (with $g = \mathsf{poly}(\lambda)$) defined as follows:

$$\hat{\varphi}_g(x) = \bigvee_g \hat{C}_g(x_1, \ldots, x_n), \tag{1}$$

where $\hat{C}_g : \{0,1\}^n \to \{0,1\}$ is an arbitrary AND-OR circuit such that $\hat{C}_g(x_1, \ldots, x_{n-n_g}, 0, \ldots, 0) = 0$, and $n = \sum_g n_g$. Furthermore, without loss of generality, we will assume the output gates in $\hat{C}_g$ are all AND gates, as if an OR gate is an output gate we can remove it and connect its output to the final OR gate in the above formula (i.e., the operator $\bigvee_g$ in Eq. (1)).

Based on the above formalization, we propose an evolving secret sharing scheme for $\hat{\mathcal{A}}^{\mathsf{ctks}}$ based on projective PRGs with *bounded* polynomial stretch (and standard PRGs with unbounded polynomial stretch).

[10] This assumption is slightly different from [2], where one adds an OR gate with fan-in 1 only for the input wires that go into AND gates; the modification is needed in order to obtain a construction in the evolving setting.

Construction 4

Let $\mathsf{pPRG} = (\mathsf{pPRG.Setup}, \mathsf{pPRG.KeyGen}, \mathsf{pPRG.Eval})$ be a block projective PRG with bounded polynomial stretch, and $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^*$ be a standard PRG with unbounded polynomial stretch. For a seed $\kappa \in \{0,1\}^\lambda$, we parse the output of $\mathsf{G}(\kappa)$ into blocks of size equal to the size of a projective key produced by pPRG, and denote with $\mathsf{G}(\kappa)[i]$ the i-th such block. Consider the following secret sharing scheme $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Recon})$ over message space $\mathcal{M} = \{0,1\}^\lambda$ for the rigid evolving monotone circuits access structure $\hat{\mathcal{A}}^{\mathsf{ckts}}$.

Sharing: When the generation $g \geq 1$ begins, the dealer proceeds as follows:

– Let $\hat{C}_g$ be the AND-OR circuit corresponding to the arrival of the g-th generation (see Eq. (1)). Let $m_g = m_g^\wedge + m_g^\vee$ be the number of gates in $\hat{C}_g$, where $m_g^\wedge$ and $m_g^\vee$ are, respectively, the number of AND and OR gates (including the input and output gates).
– Compute $(\mathsf{mpk}_g, \mathsf{msk}_g) \leftarrow\!\!\$\ \mathsf{pPRG.Setup}(1^\lambda, 1^{m_g^\vee})$.
– For each $i \in [m_g^\vee]$, associate to the i-th OR gate in $\hat{C}_g$ a key $\kappa_i^{(g)}$ determined as follows:
  - If the gate is an input OR gate, set $\kappa_i^{(g)} = \mathsf{G}(\kappa_i^*)[1]$ where $\kappa_i^* \leftarrow\!\!\$\ \{0,1\}^\lambda$ for each $i \in [n] \setminus [n - n_g]$, and $\kappa_i^{(g)} = \mathsf{G}(\kappa_i^*)[g - g_i + 1]$ for each $i \in [n - n_g]$, where g_i is the generation corresponding to the arrival of party $i \in [n - n_g]$. (Observe that, for every $i \in [n - n_g]$, the key κ_i^* corresponds to the PRG seed contained in the shares of parties of previous generations.)
  - If the gate is a non-input OR gate, set $\kappa_i^{(g)} = y_i^{(g)} = \mathsf{pPRG.Eval}(\mathsf{mpk}_g, \alpha_{\{i\}}^{(g)}, \{i\})$, where $\alpha_{\{i\}}^{(g)} = \mathsf{pPRG.KeyGen}(\mathsf{mpk}_g, \mathsf{msk}_g, \{i\})$.
– For each $i \in [m_g^\wedge]$, associate to the i-th AND gate in $\hat{C}_g$ a key $\kappa_i^{(g)}$:
  - If the i-th gate is a non-output AND gate, set $\kappa_i^{(g)} = \alpha_{\mathcal{T}_i^{(g)}} = \mathsf{pPRG.KeyGen}(\mathsf{mpk}_g, \mathsf{msk}_g, \mathcal{T}_i^{(g)})$, where $\mathcal{T}_i^{(g)} = \{j : i \to j \text{ in } \hat{C}_g\}$ consists of all out-neighbor of the i-th AND gate in $\hat{C}_g$.
  - If the i-th gate is an output AND gate, set $\kappa_i^{(g)} = \mu$.
– For each $i \in [m_g^\wedge]$, associate to the i-th AND gate in $\hat{C}_g$ (including the output gates) the ciphertext $\gamma_i^{(g)} = \kappa_i^{(g)} \oplus \rho_i^{(g)}$, which is viewed as an encryption of $\kappa_i^{(g)}$ under the mask $\rho_i^{(g)} = \bigoplus_{j:j \to i} \mathsf{G}(\kappa_j^{(g)})[i]$. (For each of the output AND gates, we only take the first λ bits of the corresponding PRG blocks.)
– Finally, the share of party $i \in [n] \setminus [n - n_g]$ (i.e, the share of the new parties belonging to the g-th generation) is defined to be $\sigma_i = (\mathsf{mpk}_g, \kappa_i^*, (\gamma_j^{(g)})_{j \in [m_g^\wedge]})$.

Reconstruction: Let $x \in \{0,1\}^n$ be the input that corresponds to the parties that want to reconstruct the message. For each generation, we traverse the circuit $\hat{C}_g$ from the inputs to the outputs, and recover the key $\kappa_i^{(g)}$ associated to each gate i that is satisfied by x (i.e., the gate is evaluated to 1 under the assignment x) in $\hat{C}_g$. In case the latter allows to obtain the key associated to any of the output AND gates in $\hat{C}_g$, output that value as the reconstructed message.

Correctness can be seen as follows. Clearly, for any qualified subset $\mathcal{I}$ corresponding to a satisfying assignment $x_\mathcal{I}$ for $C(x)$, there exists some $g \geq 1$ such that at least one output AND gate in $\hat{C}_g$ is satisfied by $x_\mathcal{I}$. Furthermore, the parties in $\mathcal{I}$ can recover the key $\kappa_i^{(g)}$ associated to each gate i that is satisfied by $x_\mathcal{I}$ in $\hat{C}_g$, and thus can correctly recover the message. Indeed:

- For an input OR gate, the key $\kappa_i^{(g)}$ is given as part of the shares of the parties corresponding to the assignment x.
- For a non-input OR gate, the key $\kappa_i^{(g)} = y_i^{(g)}$ can be recovered based on the key $\kappa_j^{(g)}$ of the first gate j that is satisfied, and whose outgoing wire enters i, i.e. $j \to i$ in $\hat{C}_g$. Indeed, j is an AND gate, and its key $\kappa_j^{(g)}$, which was already recovered, consists of a projective key $\alpha_{\mathcal{T}_j^{(g)}}$ for a set $\mathcal{T}_j^{(g)}$ that contains i.
- For an AND gate, the key $\kappa_i^{(g)}$ can be recovered by XOR-ing the ciphertext $\gamma_i^{(g)}$ with the mask $\rho_i^{(g)}$. This mask can be computed based on all the keys $\{\kappa_j^{(g)} : j \to i \text{ in } \hat{C}_g\}$ that were already recovered (since $j < i$ and since all the gates $j : j \to i$ must be satisfied by $x_\mathcal{I}$).

Turning to security, we establish the following result whose proof is included in the full version of this work [20].

Theorem 6. *Assuming* pPRG *is robust (Definition 3) and* G *is secure, the scheme* SS *described in Construction 4 is a computationally private secret sharing scheme over* $\mathcal{M} = \{0,1\}^\lambda$ *for the rigid evolving monotone circuits access structure* $\hat{\mathcal{A}}^{\mathsf{ckts}}$.

We notice that the share of each *new* party within the generation $g \geq 1$ (i.e., any party that is not part of previous generations) consists of a master public key for a projective PRG outputting $m_g^\vee$ blocks of dimension λ, of a λ-bit seed for the standard PRG G, and of an encryption of the projective key[11] associated to each AND gate in the circuit $\hat{C}_g$. Recall that we increased the number of OR gates in the circuit $\hat{C}_g$ by at most n_g (in order to make sure that input wires only enter OR gates), and thus we can simply upper bound $m_g^\vee$ with m_g. Hence, by invoking Theorem 1, we obtain:

[11] The key associated to the output AND gate actually equals the message, but this difference is immaterial when evaluating the share size.

Corollary 4. *Let $\hat{\mathcal{A}}^{\mathsf{ckts}} = \{\hat{C}_g\}$ be the monotone circuits access structure, and denote by $m_g^{\wedge}$ (resp. $m_g^{\vee}$) the number of AND (resp. OR) gates in $\hat{C}_g$ for any $g \geq 1$. There exists a computationally private secret sharing scheme over $\mathcal{M} = \{0,1\}^{\lambda}$ for $\hat{\mathcal{A}}^{\mathsf{ckts}}$, from the following assumptions and with the following parameters:*

- *Under the RSA assumption (or iO and SSB hash functions), where the share size of all parties belonging to the generation $g \geq 1$ is $m_g^{\wedge} \cdot \mathsf{poly}(\lambda)$.*
- *Under either the DDH or the BDDH assumption, where the share size of all parties belonging to the generation $g \geq 1$ is $(m_g^{\vee})^2 \cdot \mathsf{poly}(\lambda) + m_g^{\wedge} \cdot O(\lambda)$.*
- *Under the LWE assumption, where the share size of all parties belonging to the generation $g \geq 1$ is $m_g^{\vee} \cdot \mathsf{poly}(\lambda) + m_g^{\wedge} \cdot O(\lambda)$.*

Remark 5 (On the number of gates). Recall that we assumed for simplicity that each input wire in the circuits $\hat{C}_g$ is only connected to OR gates with fan-in 1. For this to hold, one needs to add an AND gate with fan-in 1 for every input wire that goes into an OR gate with fan-in ≥ 2 (in order to maintain the invariant that the circuit alternates OR and AND layers). Hence, the number of AND gates $m_g^{\wedge}$ in the above corollary must be increased by an additive factor of at most n_g for every $g \geq 1$.

6 Conclusions

We have initiated a systematic study of *evolving* secret sharing schemes in the *computational* setting. Our main finding is that switching to computational security allows to obtain secret sharing schemes for a plethora of evolving access structures, including dynamic threshold, graphs, CNF and DNF formulas, and monotone circuits access structures. Furthermore, for many of there access structures, our secret sharing schemes are *succinct*, i.e., much smaller compared to the size of a natural computational representation of the evolving access structure.

A first natural direction for future research would be to obtain secret sharing schemes for more evolving access structures (e.g., monotone NP [25], branching programs [1], weighted threshold access structures [10]), or to improve our constructions in terms of hardness assumptions and/or share size. Another interesting open problem is to study evolving secret sharing in the context of *adaptive security* [23], or with additional properties such as verifiability [4], and non-malleability [19,21].

Acknowledgements. The first author was partially supported by the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the second author was supported by project SERICS (PE00000014) and by project PARTHENON (B53D23013000006), under the MUR National Recovery and Resilience Plan funded by the European Union—NextGenerationEU.

References

1. Alon, B., Beimel, A., David, T.B., Omri, E., Paskin-Cherniavsky, A.: New upper bounds for evolving secret sharing via infinite branching programs. Cryptology ePrint Archive (2024), https://eprint.iacr.org/2024/419
2. Applebaum, B., Beimel, A., Ishai, Y., Kushilevitz, E., Liu, T., Vaikuntanathan, V.: Succinct computational secret sharing. In: Proceedings of the 55th Annual ACM Symposium on Theory of Computing. pp. 1553–1566 (2023). https://doi.org/10.1145/3564246.3585127
3. Applebaum, B., Nir, O.: Upslices, downslices, and secret-sharing with complexity of 1.5^n. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 627–655. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84252-9_21
4. Backes, M., Kate, A., Patra, A.: Computational verifiable secret sharing revisited. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 590–609. Springer, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_32
5. Béguin, P., Cresti, A.: General short computational secret sharing schemes. In: Guillou, L.C., Quisquater, J.J. (eds.) EUROCRYPT'95. LNCS, vol. 921, pp. 194–208. Springer, Heidelberg (May 1995). https://doi.org/10.1007/3-540-49264-X_16
6. Beimel, A.: Secret-sharing schemes: A survey. In: Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings. vol. 6639, pp. 11–46. Springer (2011)
7. Beimel, A., Farràs, O., Mintz, Y.: Secret-sharing schemes for very dense graphs. Journal of Cryptology **29**(2), 336–362 (Apr 2016). https://doi.org/10.1007/s00145-014-9195-8
8. Beimel, A., Othman, H.: Evolving ramp secret-sharing schemes. In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 313–332. Springer, Heidelberg (Sep 2018). https://doi.org/10.1007/978-3-319-98113-0_17
9. Beimel, A., Othman, H.: Evolving ramp secret sharing with a small gap. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 529–555. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45721-1_19
10. Beimel, A., Tassa, T., Weinreb, E.: Characterizing ideal weighted threshold secret sharing. In: Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005. Proceedings 2. pp. 600–619. Springer (2005). https://doi.org/10.1007/978-3-540-30576-7_32
11. Blakley, G.R.: Safeguarding cryptographic keys. Proceedings of AFIPS 1979 National Computer Conference **48**, 313–317 (1979)
12. Bogdanov, A., Guo, S., Komargodski, I.: Threshold secret sharing requires a linear-size alphabet. Theory of Computing **16**(1), 1–18 (2020)
13. Cachin, C.: On-line secret sharing. In: Boyd, C. (ed.) 5th IMA International Conference on Cryptography and Coding. LNCS, vol. 1025, pp. 190–198. Springer, Heidelberg (Dec 1995)
14. Csirmaz, L.: The size of a share must be large. In: Santis, A.D. (ed.) EUROCRYPT'94. LNCS, vol. 950, pp. 13–22. Springer, Heidelberg (May 1995). https://doi.org/10.1007/BFb0053420
15. Csirmaz, L.: The size of a share must be large. In: Santis, A.D. (ed.) EUROCRYPT'94. LNCS, vol. 950, pp. 13–22. Springer, Heidelberg (May 1995). https://doi.org/10.1007/BFb0053420

16. Csirmaz, L., Tardos, G.: On-line secret sharing. Des. Codes Cryptogr. **63**(1), 127–147 (2012)
17. Desmedt, Y., Dutta, S., Morozov, K.: Evolving perfect hash families: A combinatorial viewpoint of evolving secret sharing. In: Mu, Y., Deng, R.H., Huang, X. (eds.) CANS 19. LNCS, vol. 11829, pp. 291–307. Springer, Heidelberg (Oct 2019). https://doi.org/10.1007/978-3-030-31578-8_16
18. Dutta, S., Roy, P.S., Fukushima, K., Kiyomoto, S., Sakurai, K.: Secret sharing on evolving multi-level access structure. In: You, I. (ed.) WISA 19. LNCS, vol. 11897, pp. 180–191. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-39303-8_14
19. Faonio, A., Venturi, D.: Non-malleable secret sharing in the computational setting: Adaptive tampering, noisy-leakage resilience, and improved rate. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 448–479. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_16
20. Francati, D., Venturi, D.: Evolving secret sharing made short. Cryptology ePrint Archive (2023), https://eprint.iacr.org/2023/1534
21. Goyal, V., Kumar, A.: Non-malleable secret sharing. In: Diakonikolas, I., Kempe, D., Henzinger, M. (eds.) 50th ACM STOC. pp. 685–698. ACM Press (Jun 2018). https://doi.org/10.1145/3188745.3188872
22. Ito, M., Saito, A., Nishizeki, T.: Secret sharing schemes realizing general access structure. In: Proc. IEEE Global Telecommunication Conf. (Globecom'87). pp. 99–102 (1987)
23. Jafargholi, Z., Kamath, C., Klein, K., Komargodski, I., Pietrzak, K., Wichs, D.: Be adaptive, avoid overcommitting. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 133–163. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_5
24. Komargodski, I., Naor, M., Yogev, E.: How to share a secret, infinitely. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 485–514. Springer, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_19
25. Komargodski, I., Naor, M., Yogev, E.: Secret-sharing for NP. Journal of Cryptology **30**(2), 444–469 (Apr 2017). https://doi.org/10.1007/s00145-015-9226-0
26. Komargodski, I., Paskin-Cherniavsky, A.: Evolving secret sharing: Dynamic thresholds and robustness. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 379–393. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70503-3_12
27. Krawczyk, H.: Secret sharing made short. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 136–146. Springer, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48329-2_12
28. Luby, M.: Lt codes. In: 43rd FOCS. pp. 271–282. IEEE Computer Society Press (Nov 2002). https://doi.org/10.1109/SFCS.2002.1181950
29. Luby, M., Mitzenmacher, M., Shokrollahi, M.A., Spielman, D.A.: Efficient erasure correcting codes. IEEE Trans. Inf. Theory **47**(2), 569–584 (2001)
30. Mazor, N.: A lower bound on the share size in evolving secret sharing. In: 4th Conference on Information-Theoretic Cryptography (ITC 2023). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2023). https://doi.org/10.4230/LIPIcs.ITC.2023.2
31. Mitzenmacher, M.: Digital fountains: a survey and look forward. In: 2004 IEEE Information Theory Workshop, San Antonio, TX, USA, 24-29 October, 2004. pp. 271–276. IEEE (2004)
32. Paskin-Cherniavsky, A.: How to infinitely share a secret more efficiently. Cryptology ePrint Archive, Report 2016/1088 (2016), https://eprint.iacr.org/2016/1088

33. Pueyo, I.C., Cramer, R., Xing, C.: Bounds on the threshold gap in secret sharing and its applications. IEEE Trans. Inf. Theory **59**(9), 5600–5612 (2013)
34. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: 21st ACM STOC. pp. 73–85. ACM Press (May 1989). https://doi.org/10.1145/73007.73014
35. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery **22**(11), 612–613 (Nov 1979)
36. Shokrollahi, M.A., Luby, M.: Raptor codes. Found. Trends Commun. Inf. Theory **6**(3-4), 213–322 (2009)
37. Vinod, V., Narayanan, A., Srinathan, K., Rangan, C.P., Kim, K.: On the power of computational secret sharing. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 162–176. Springer, Heidelberg (Dec 2003)
38. Xing, C., Yuan, C.: Evolving secret sharing schemes based on polynomial evaluations and algebraic geometry codes. Cryptology ePrint Archive, Report 2021/1115 (2021), https://eprint.iacr.org/2021/1115

Verifiable Secret Sharing from Symmetric Key Cryptography with Improved Optimistic Complexity

Ignacio Cascudo[1(✉)], Daniele Cozzo[1], and Emanuele Giunta[1,2]

[1] IMDEA Software Institute, Madrid, Spain
{ignacio.cascudo,daniele.cozzo,emanuele.giunta}@imdea.org
[2] Universidad Politecnica de Madrid, Madrid, Spain

Abstract. In this paper we propose verifiable secret sharing (VSS) schemes secure for any honest majority in the synchronous model, and that only use *symmetric-key* cryptographic tools, therefore having plausibly post-quantum security. Compared to the state-of-the-art scheme with these features (Atapoor et al., Asiacrypt '23), our main improvement lies on the complexity of the *"optimistic"* scenario where the dealer and all but a small number of receivers behave honestly in the sharing phase: in this case, the running time and download complexity (amount of information read) of each honest verifier is *polylogarithmic* and the total amount of broadcast information by the dealer is *logarithmic*; all these complexities were linear in the aforementioned work by Atapoor et al. At the same time, we preserve these complexities with respect to the previous work for the "pessimistic" case where the dealer or $O(n)$ receivers cheat actively. The new VSS protocol is of interest in multi-party computations where each party runs one VSS as a dealer, such as distributed key generation protocols.

Our main technical handle is a distributed zero-knowledge proof of low degreeness of a polynomial, in the model of Boneh et al. (Crypto '19) where the statement (in this case the evaluations of the witness polynomial) is distributed among several verifiers, each knowing one evaluation. Using folding techniques similar to FRI (Ben-Sasson et al., ICALP '18) we construct such a proof where each verifier receives polylogarithmic information and runs in polylogarithmic time.

1 Introduction

A (t, n)-threshold secret sharing scheme allows a dealer D to share a secret s among n parties $P_1, \ldots, P_n$ in such a way that any subset of t parties or less has no information about s while any set of $t + 1$ or more parties can recover s if they collaborate. The most famous example of threshold secret sharing is the scheme proposed by Shamir [35]. The dealer samples a polynomial $f(X)$ of degree at most t with coefficients in a finite field $\mathbb{F}$ such that its evaluation in a distinguished point $\alpha_0 \in \mathbb{F}$ is the secret s, i.e. $f(\alpha_0) = s$, and then sends an evaluation $s_i = f(\alpha_i)$ to each party individually, where α_i are pairwise distinct

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15490, pp. 100–128, 2025.
https://doi.org/10.1007/978-981-96-0941-3_4

points in $\mathbb{F}$ (and different from α_0). From the basic properties of polynomials it follows that t or less parties do not have any information about s, while $t+1$ parties can collaboratively reconstruct $f(X)$ and hence recover s by Lagrange interpolation. The above scheme however only provides security against passive adversaries and does not prevent a dishonest dealer from sampling a polynomial of the wrong degree or dishonest parties from sending incorrect values at the moment of reconstruction.

A verifiable secret sharing (VSS) scheme solves these problems by ensuring that the shares parties receive are part of a correct sharing of a secret according to the specified underlying secret sharing scheme, and later that the honest parties reconstruct the secret correctly. VSS is typically used for realizing distributed schemes, for example realizing distributed key generation (DKG), which in turn is a fundamental building block for threshold cryptography [25], and general purpose multiparty computation (MPC) protocols [8].

VSS schemes have been realized from different assumptions, in both the synchronous and asynchronous model. In this work we focus on the former. On one side, starting with [8] there are VSS that offer perfect security, see [21] for an exhaustive overview. These schemes do not rely on computational hardness assumptions and are typically computationally efficient, but require high communication and assume more than two thirds of parties being honest for reconstructing the secret. If one is willing to admit an exponentially small probability of error in the reconstruction (statistical security), there are schemes that offer unconditional security assuming only a simple honest majority starting with [33], but they still require a large amount of communication between parties. On the other hand, computationally secure VSS schemes based on public-key cryptography have been proposed [7,16–19,24,27,32,34]. Thanks to public-key tools some of these schemes are able to achieve the property of public verifiability. A publicly verifiable secret sharing (PVSS) is a VSS scheme where all communication is done through a public channel and a public verifier can ensure the correctness of the sharing and reconstruction. This typically employs a linearly homomorphic encryption scheme by which the dealer encrypts the shares and then publishes the ciphertexts, so that parties can use their own secret keys to get the shares. The correctness of the shares is ensured by a proof that the plaintexts of those ciphertexts are indeed shares of a polynomial of degree t, which can be checked by everyone, not only the shareholders. This is particularly useful in some scenarios such as the construction of randomness beacons [16,17] and multiparty computation in some restricted settings, e.g. in the so-called YOSO model [18,19].

Many of these schemes, in particular all cited above, only require simple honest majority for reconstructing the secret. The downside of this approach is that it suffers from large bandwidth and high cost from the prover due to public-key operations.

A third class of VSS lies in the middle in terms of required security assumptions and attain computational security but only use symmetric-key cryptography. Giving up on perfect secrecy puts these schemes in the regime of honest majority and at the same time relying on symmetric-key tools gives an advantage in terms of efficiency over those relying on public-key primitives. While these schemes are not publicly verifiable, this is enough for many applications, for example multiparty computation and threshold cryptography.

Gennaro, Rabin and Rabin [26] proposed the first computational VSS that relies solely on symmetric-key tools. However their scheme only achieves a weaker notion of security and communication is linear in the number of parties. Backes, Kate and Patra [6] proposed a VSS whose security relies solely on the binding property of commitment schemes, that can be instantiated with hash functions. However, sharing a secret involves using a bivariate polynomial that yields $O(n^2)$ in communication.

The state-of-the-art is a recent work by Atapoor et al. [3], which uses a novel approach for constructing VSS starting from distributed zero-knowledge (dZK) proofs. The notion of distributed dZK proof was formally introduced in [13]. In this setting a prover wants to convince n verifiers $V_1, \dots, V_n$ that a statement x lies in some language $\mathcal{L}$ (in the case of NP-languages that there exists a witness for which $x \in \mathcal{L}$). The main difference with the standard notion of interactive (zero-knowledge) proof is that now x is distributed among the verifiers, but no single verifier knows x in full. A distributed zero-knowledge proof system is required to satisfy correctness, meaning that if all verifiers accept then $x \in \mathcal{L}$; soundness, meaning that if there is no w for which $x \in \mathcal{L}$ then the proof rejects even if the prover colludes with $t-1$ verifiers; and zero-knowledge, namely that the proof does not reveal any information about w to up to t colluding verifiers.

As shown in [3], one can realize a VSS from a distributed dZK proof. The dealer runs a dZK proof for membership to the following language

$$\mathcal{L} = \{x = (x_1, \dots, x_n) \in R^n \ : \ \exists f(X) \in R[X] \ : \ x_i = f(\alpha_i),\ \deg(f) \leq t\}. \quad (1)$$

where R is a (commutative, with 1) ring, and $(\alpha_1, \dots, \alpha_n)$ is a (fixed, public) exceptional set in R.[1] The share-receivers play the role of verifiers for this dZK proof. Then a consensus protocol allows the parties to resolve conflicts in case of rejections, and eventually disqualify the dealer.

The starting point of [3] is a simple distributed Σ-protocol for (1). At a high level their construction works as follows: the dealer with input a secret polynomial $f(X)$ of degree $\leq t$ with $x_0 = f(\alpha_0)$ and $x_i = f(\alpha_i)$, samples a uniformly random polynomial $b(X)$ also of degree $\leq t$, broadcasts commitments to its evaluations, and then computes a random linear combination $r(X) = b(X)+\mu f(X)$ where $\mu \in \mathbb{F}$ is sampled by the verifiers. The prover then broadcasts the polynomial $r(X)$ while it sends, privately to each party i, the share x_i and the opening to the commitments to $b(\alpha_i)$. Each verifier V_i can individually check the proof by checking the openings of the commitments and that $r(\alpha_i) = b(\alpha_i)+\mu x_i$.

The resulting VSS has a $O(n \log n)$ cost for the dealer, which is inherent due to the need of evaluating a polynomial of degree $O(n)$ on n points. Their VSS works for more general structures than fields, for example rings with a large

[1] An exceptional set is a set where the pairwise differences of distinct elements in the set are all invertible in the ring.

enough exceptional set. However their protocol requires at least $O(n)$ computational cost for the shareholders already in the sharing phase, as verification requires to evaluate the polynomial $r(X)$. In terms of communication, the dealer needs to broadcast the whole polynomial $r(X)$ (which amounts to a $O(n)$ amount of broadcast communication) and consequently each verifier has linear download complexity, i.e. each verifier receives $O(n)$ amount of communication.

An important observation for this work is that the above complexities hold *even in the "optimistic" case* where no party eventually acts dishonestly. In particular, the verifiers complexity is $O(n)$ in that case. If there are $\Theta(n)$ corrupt verifiers or if the dealer is corrupt, then the complexity becomes $O(n \log n)$.

This leads us to the question of whether we can design VSS protocols that have a *better optimistic complexity*, i.e., that allow for sublinear $o(n)$ or even polylogarithmic $O(\text{polylog } n)$ verifier work, in the case where the dealer is honest and up to a "small" number (say $O(1)$) of verifiers are corrupted, while still not worsening the verifier complexity of the pessimistic cases where either the dealer and/or a large number of verifiers up to certain bound $t = O(n)$ are corrupted.

Note that even though there is one party (namely, the dealer) who may on their own force the pessimistic case to happen, the scenario above can be of interest in many uses of VSS in multiparty computation protocols where, at a certain round, n instances of VSS are run, one for each of the parties acts as a dealer. One of the most well known examples of this is the case of distributed key generation protocols for discrete-log based threshold schemes, where each party chooses and VSSs a random field element, and the secret key is computed as the sum of the correctly VSSed elements. In cases as the above, if the adversary actively corrupts, say, $O(1)$ parties during the execution, these parties will be able to force the worst case complexity in the $O(1)$ instances where they are acting as dealers, but the remaining instances will enjoy the optimistic complexity. The total work of each honest party across the n VSS instances will be $O(n \text{ polylog } n)$, which would be an improvement over the $O(n^2)$ complexity that would arise from using [3]. At the same time, the VSS would still remain secure against a more powerful adversary corrupting $O(n)$ parties.

While verifier complexity is our main concern in this paper, we are also interested in improving the *download complexity* per party and *broadcast communication* in these optimistic cases. The latter is interesting in blockchain ecosystems, where one wants to limit as much as possible the amount of information stored on-chain. We do need to remark that this will come at the cost of increasing the amount of private communication sent by the dealer to each party, as we describe in the next paragraphs.

Our Contributions. In this work we present a VSS construction that only employs secret key cryptography, tolerates $t < n/2$ corrupt parties, and improves on the state of the art with regards to *optimistic verifier complexity* as well as optimistic broadcast and download complexity, while still matching the complexities of previous works in the pessimistic case. In particular, in the optimistic case where the dealer is honest and there are $O(1)$ corrupt verifiers, the verifier complexity is $O(\log(n)^2)$, the dealer broadcasts $O(\log n)$ information and each party needs to receive $O(\log(n)^2)$ information in total, via the broadcast and private channels[2]. All these complexities were $O(n)$ in [3]. In the pessimistic case (where the dealer and/or $O(n)$ verifiers are corrupt) the asymptotic complexities match those of [3]: the broadcast and download complexities are $O(n)$ and the verifier complexity is $O(n \log n)$. In all cases the prover complexity is $O(n \log n)$, same as in [3]. This all comes at the cost of increasing the private communication: while in [3] the dealer needs to send a constant amount of information privately to each party, in our work this will be $O(\log(n)^2)$. While this increases the total amount of information communicated by the dealer, the decreased use of the broadcast channel may be beneficial in some applications as we have argued above.

Our main technical handle is a new distributed ZK proof for language (1) with polylogarithmic proof size in the degree of the polynomial that only leverages on symmetric-key primitives and supports rings with a large enough exceptional set, and which we believe to be of independent interest.

In more detail, in our VSS protocol the dealer initially broadcasts $O(\log n)$ information (the "public" part of the aforementioned distributed ZK proof), as well as sending $O(\log(n)^2)$ information to each party privately, which contains the shares and private part of the distributed proof. The verifiers are required to perform $O(\log(n)^2)$ computation[3]. Only if there are complaints, the protocol incurs in more communication and computation as the prover then needs to broadcast the private communication previously sent to the complaining parties. In the worst case where $O(n)$ share receivers complain, the dealer needs to broadcast $O(n)$ information (and no additional private communication) and the share receivers need to perform $O(n \log n)$ computation. As mentioned above, these costs are the same as in the state-of-the-art [3], with the one aforementioned caveat that we require more private communication. See Table 1 for comparisons.

Our VSS tolerates up to $t = n/2 - 1$ corruptions and is proven secure in the random oracle model and, as in [3], we are able to support rings with a large enough exceptional set.

Technical Overview of Our Approach. To construct our VSS, we follow the paradigm of [3] starting from a distributed ZKP of the existence of a low degree polynomial interpolating the shares. At the heart of our work is an efficient distributed proof (without zero-knowledge) for the same task.

More precisely, the n verifiers each have a piece $x_i \in R$ of the statement, and the prover wishes to convince them that there is a polynomial $f(X)$ of degree $< d$ such that $x_i = f(\alpha_i)$ for all $i \in \{1, \ldots, n\}$ (or more precisely, for all the honest verifiers V_i).

Our interactive proof is recursive and based on a folding technique, similar to proofs in the literature such as FRI [9] and DARK compilers [14]. At every step k of the recursion the prover claims that a certain polynomial $f^{(k)}(X)$ has degree $< d/2^k$ (where in addition $f^{(0)} = f$); the recursion reduces this task to proving that some related *randomized* polynomial $f^{(k+1)}(X)$ has degree $< d/2^{k+1}$. At the last

[2] Precise asymptotic costs according to active corruptions are given in Sect. 4.3.
[3] Specifically $O(\log n)$ ring operations and $O(\log(n)^2)$ hashes.

step $k = \tau$ of the recursion, the prover simply broadcasts $f^{(\tau)}$ which is of small enough degree $d/2^\tau$. For the folding of $f^{(k)}$ into $f^{(k+1)}$ we offer two alternatives: in our first alternative, which is inspired by [14], the prover splits $f^{(k)}$ in high and low degree terms (i.e. $f^{(k)}(X) = g_0^{(k+1)}(X) + X^{d/2^{k+1}} g_1^{(k+1)}(X)$), receives a random challenge $\mu^{(k)}$ and constructs $f^{(k+1)}$ as $f^{(k+1)}(X) = g_0^{(k+1)}(X) + \mu^{(k+1)} g_1^{(k+1)}(X)$. The second alternative is the one in FRI: the prover splits $f^{(k)}$ in odd and even degree terms, i.e. as $f^{(k)}(X) = g_0^{(k)}(X^2) + X g_1^{(k)}(X^2)$, and again sets $f^{(k+1)}(X) = g_0^{(k+1)}(X) + \mu^{(k+1)} g_1^{(k+1)}(X)$.

The reason why we have two alternatives is that, for general rings R, the first alternative requires fewer assumptions on the set of evaluation points $\{\alpha_1, \ldots, \alpha_n\}$; namely, we require that this is an exceptional set, i.e. that the pairwise differences of all the elements in the set are invertible in the ring. On the other hand, using the second alternative requires in addition that *all* sets $\{\alpha_1^{2^k}, \ldots, \alpha_n^{2^k}\}$ for $k = 0, \ldots, \tau$ are exceptional. However, if we do have this guarantee, for example if R is a finite field, the second alternative may also lead to more efficient protocols.

A technical difference with FRI is in how the prover shows that this splitting has been done correctly. Instead of committing to the evaluation of $f^{(k)}(X)$, $g_0^{(k+1)}$, $g_1^{(k+1)}$ in a large set of points and then opening a random subset chosen by the (single) verifier, in our case the prover commits to the evaluations of these polynomials in $\{\alpha_1, \ldots, \alpha_n\}$ (or $\{\alpha_1^{2^k}, \ldots, \alpha_n^{2^k}\}$ in the second alternative) and then opens the evaluations in the i-th point privately to the i-th verifier. Note that in our case, the prover cannot cheat, even with small probability, in this step as essentially the honest verifiers are checking that the splitting is correct in all honest points. The only source of soundness error in our case is the fact that the degree of $g_0^{(k+1)}(X) + \mu_1 g_1^{(k+1)}(X)$ may be smaller than the degrees of both $g_0^{(k+1)}(X)$ and $g_1^{(k+1)}(X)$, which happens with small probability if μ is sampled from a large exceptional set of the ring.

The recursive nature of our protocol allows us to achieve a proof where the size of the communication received by each verifier (both broadcast and privately) is polylogarithmic in the degree of the polynomial. Instantiating the commitments with Merkle trees allows for efficient openings and give the protocol computational security based only on the security of hash functions.

Our technique allows us to work over rings with a large enough exceptional set, and we believe this to be of independent interest as it would allow constructing ring-friendly polynomial commitments that are plausibly post-quantum secure.

We then show how to easily add zero-knowledge to the protocol above by adding two additional rounds. In the first round the prover samples a uniformly random polynomial $b(X)$ of the same degree of $f(X)$ and sends a commitment to it to the verifiers, that will respond with a random challenge μ_0. The prover then computes the random linear combination $r(X) = b(X) + \mu_0 f(X)$ and applies to it the above folding protocol. The zero knowledge property comes from the fact that $b(X)$ information-theoretically hides $f(X)$.

By interpreting these two additional rounds as the first two rounds of the distributed Σ-protocol of [3], we can see our construction as a distributed version of the technique used to compress standard Σ-protocols [4] where the prover replaces the third message of a Σ-protocol by a (non zero-knowledge) proof that this last message satisfies a certain property (in this case that $\deg r(X) < d$). We then turn the above $(2\tau + 3)$-rounds dZK proof into a non-interactive dZK proof using the Fiat-Shamir transform.

Following the construction of [3], we obtain a VSS by adding a consensus protocol on the execution of the non-interactive dZK proof. The resulting VSS inherits the logarithmic communication complexity and computational costs from the dZKP.

Comparison with Previous Work. We compare our VSS with other honest majority schemes in Table 1. For [3] and our work we include the costs for both the optimistic (no complaints) and pessimistic ($O(n)$ complaints) case. Such a distinction does not exist in the case of PVSS thanks to the public verifiability feature. Such a feature, however, comes at the price of having $O(n)$ broadcast communication and $O(n \log n)$ computational complexity for both dealer and parties in terms of (expensive) group operations. Compared to the state-of-the-art [3], in the optimistic case we decrease the amount of information to be broadcasted from $O(n)$ to $O(\log n)$, as well as the total download per party, from $O(n)$ to $O(\log(n)^2)$, and parties computational cost from $O(n)$ to $O(\log n)$. Instead we add a polylogarithmic factor $\log(n)^2$ to the amount of private communication. The computational cost for the dealer is the same, $O(n \log n)$ ring operations, which is an inherent cost from the evaluation of the secret polynomial defining the shares. In the pessimistic case, the cost are the same, except we still pay for the polylogarithmic factor in private communication.

Other Related Work. In this work we only focus on protocols that operate in the so-called synchronous model. Here the parties are synchronized by a global clock and there are strict (publicly-known) upper bounds on the message delays. On the other hand, there is a line of works that explores protocols in the asynchronous model [1,2,22,36], where the parties are not synchronized and where the we assume that the adversary can take control of the network and arbitrarily delay the messages sent by the parties. Designing VSS protocols in this setting is more challenging and inherently support at most $n/3 - 1$ corruptions. Our techniques, in particular, do not apply to this asynchronous case, where parties would need to broadcast and read decision bits (even in the case everyone accepts) so communication and verification time would be linear even in the optimistic case.

Outline. In Sect. 2 we recall and revise known definitions. Our distributed proof of low degree is detailed in Sect. 3, starting from two (non ZK) protocols (Sects. 3.1–3.2) and later adding zero knowledge (Sect. 3.3) and removing interaction (Sect. 3.4). Finally, Sect. 4 is devoted to building VSS from distributed proofs of low degree.

Table 1. Comparison of our VSS with previous computationally secure VSS For comparison, we specialize our scheme and that of [3] to $R = \mathbb{F}$, a finite field. PV: public verifiability, BC: broadcast, PC: private communication to each party, DW: download per party, $O^{\mathbb{F}}(\bullet)$: complexity in terms of field operations, $O^{\mathbb{G}}(\bullet)$: complexity in terms of group operations. Note that publicly verifiable secret sharing schemes assume an initial PKI setup, or otherwise need an additional round to establish this PKI.

Scheme	Assumption	Rounds	Prover complexity	Verifier complexity	Communication	PV
[34]	DDH, RO	1	$O^{\mathbb{G}}(n \log n)$	$O^{\mathbb{G}}(n^2 \log n)$	BC: $O(n)$ PC: – DW: $O(n)$	yes
[16,17], [19]	DDH, RO	1	$O^{\mathbb{G}}(n \log n)$	$O^{\mathbb{G}}(n \log n)$	BC: $O(n)$ PC: – DW: $O(n)$	yes
[18,29]	Class group assump., RO	1	$O^{\mathbb{G}}(n \log n)$	$O^{\mathbb{G}}(n \log n)$	BC: $O(n)$ PC: – DW: $O(n)$	yes
[3] optimistic case	SK, RO	2	$O^{\mathbb{F}}(n \log n)$	$O^{\mathbb{F}}(n)$	BC: $O(n)$ PC: $O(1)$ DW: $O(n)$	no
[3] pessimistic case	SK, RO	3	$O^{\mathbb{F}}(n \log n)$	$O^{\mathbb{F}}(n \log n)$	BC: $O(n)$ PC: $O(1)$ DW: $O(n)$	no
This work optimistic case	SK, RO	2	$O^{\mathbb{F}}(n \log n)$	$O^{\mathbb{F}}(\log^2 n)$	BC: $O(\log n)$ PC: $O(\log^2 n)$ DW: $O(\log^2 n)$	no
This work pessimistic case	SK, RO	3	$O^{\mathbb{F}}(n \log n)$	$O^{\mathbb{F}}(n \log n)$	BC: $O(n)$ PC: $O(\log^2 n)$ DW: $O(n)$	no

2 Preliminaries

2.1 Notation

We denote by $[n]$ the set $\{1, \ldots, n\}$. In what follows R denotes a ring and $R[X]$ (resp. $R[X]_t$) the ring of polynomials (resp. degree $\leq t$ polynomials) with coefficients in R. Vectors are denoted in bold. For a vector $\mathbf{x}$ we denote by x_i its i-th entry, while $\mathbf{x}_H$ denotes the subvector $(x_i)_{i \in H}$ for a given set of indices H. All logarithms are assumed in base two.

2.2 Adversarial and Communication Model

In our protocols we will consider a network of n parties $P_1, \ldots, P_n$. We work in the synchronous communication model and assume that parties are conncted to each other by private, authenticated and bidirectional channels.

We further assume that all parties have access to a broadcast channel [31]. This means that parties can send a message reliably to each other. In addition, if a party receives a message via a broadcast, then it knows that all other honest parties received the same value.

A synchronous network allows protocols to operate in a sequence of rounds. In each round, parties perform some local computation, send messages (if any) through the private and authenticated link, and broadcast some information over the broadcast channel. At the end of each round, they receive all messages sent or broadcast by the other parties in the same round.

For the adversarial model, we assume a static, malicious adversary that corrupts up to $t < \frac{n}{2}$ parties in the protocol. While honest parties send messages following the protocol, corrupt parties may send arbitrary messages. Also, the adversary may be rushing, meaning that at each round of the protocol it waits to see what the other parties have broadcasted before broadcasting its own messages.

2.3 Vector Commitments

Definition 1. *A vector commitment with message space* $\mathsf{VC}.M$ *and commitment space* $\mathsf{VC}.C$ *is a tuple of algorithms* $\mathsf{VC} = (\mathsf{VC.Setup}, \mathsf{VC.Com}, \mathsf{VC.Open}, \mathsf{VC.Vfy})$ *defined as follows:*

- $\underline{\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda, n)}$: *given the security parameter and size of the vectors to be committed returns public parameters.*
- $\underline{(\mathsf{cm}, \mathsf{aux}) \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x})}$: *given public parameters and a vector* $\mathbf{x} \in \mathsf{VC}.M^n$ *outputs a commitment* cm *and auxiliary information* aux *used for opening.*
- $\underline{\mathsf{op} \leftarrow \mathsf{VC.Open}(\mathsf{pp}, \mathsf{cm}, i, \mathsf{aux})}$: *given public parameters, an index* $i \in [n]$ *and a commitment* cm *with auxiliary information* aux*, returns an opening proof* op *for position* i*.*
- $\underline{b \leftarrow \mathsf{VC.Vfy}(\mathsf{pp}, \mathsf{cm}, x, i, \mathsf{op})}$: *given public parameters, an element* x*, position* i*, proof* op *and a commitment* cm*, it checks the validity of the opening proof.*

Properties that a secure VC is required to satisfy are correctness, position binding and succinctness. In this work we further consider *hiding* VC.

Correctness. A vector commitment is correct if for any $\mathbf{x} \in \mathsf{VC}.M^n$

$$\Pr\left[\mathsf{VC.Vfy}(\mathsf{pp}, \mathsf{cm}, x_i, i, \mathsf{op}) = 1 \ : \ \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda) \\ \mathsf{cm}, \mathsf{aux} \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x}) \\ \mathsf{op} \leftarrow \mathsf{VC.Open}(\mathsf{pp}, \mathsf{cm}, i, \mathsf{aux}) \end{array}\right] = 1.$$

Position Binding. Define the advantage:

$$\mathrm{Adv}^{\mathrm{PBinding}}_{\mathsf{VC}}(\mathcal{A}) =$$
$$\Pr\left[\begin{array}{l} \mathsf{VC.Vfy}(\mathsf{pp}, \mathsf{cm}, x_0, i, \mathsf{op}_0) = 1 \\ \mathsf{VC.Vfy}(\mathsf{pp}, \mathsf{cm}, x_1, i, \mathsf{op}_1) = 1 \end{array} \ : \ (\mathsf{cm}, i, x_0, \mathsf{op}_0, x_1, \mathsf{op}_1) \leftarrow \mathcal{A}(\mathsf{pp}) \right].$$

Then a vector commitment VC satisfies position binding if, for all honestly generated public parameters pp, for all PPT adversaries $\mathcal{A}$ one has

$$\mathrm{Adv}^{\mathrm{PBinding}}_{\mathsf{VC}}(\mathcal{A}) = \mathsf{negl}(\lambda).$$

Hiding. The hiding property [28] informally states that opening a number of positions in a vector commitment should keep the unopened ones hidden. Formally, for any non-empty subset $T \subset [n]$ define the advantage

$$\mathrm{Adv}^{\mathrm{Hiding}}_{\mathsf{VC}}(\mathcal{A}, T) \;=\; \Pr\left[b = b' \;:\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda),\; b \leftarrow \{0,1\} \\ \mathbf{x}_0, \mathbf{x}_1 \leftarrow \mathcal{A}(\mathsf{pp}),\; \mathbf{x}_0[j] = \mathbf{x}_1[j]\; \forall j \in T \\ (\mathsf{cm}, \mathsf{aux}) \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x}_b) \\ \mathsf{op}_j \leftarrow \mathsf{VC.Open}(\mathsf{pp}, \mathsf{cm}, j, \mathsf{aux})\; \forall j \in T \\ b' \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{cm}, \{\mathsf{op}_j\}_{j \in T}) \end{array} \right].$$

Then we say that VC is hiding if, for all non-empty $T \subset [n]$ and all PPT adversaries $\mathcal{A}$ one has that

$$\mathrm{Adv}^{\mathrm{Hiding}}_{\mathsf{VC}}(\mathcal{A}, T) = \frac{1}{2} + \mathsf{negl}(\lambda).$$

Succintness. A vector commitment is succint if both the commitment cm and opening proof op for a position i are independent of the size of the committed vector. In this work, we will slightly relax this notion, by allowing the opening proof for a position to be *logarithmic* in the size n of the vector. For example, this is the case of Merkle trees, which are discussed next.

Merkle Trees. In this work we will be using Merkle trees to instantiate (hiding) vector commitments.[4] For Merkle trees we will be using the specific notation $\mathsf{MT} = (\mathsf{MT.Setup}, \mathsf{MT.Com}, \mathsf{MT.Open}, \mathsf{MT.Vfy})$. Let n be a power-of-two[5] and let H be a collision resistant hash function. A Merkle tree commitment to a vector $\mathbf{x}$ of n elements consists in a binary tree, where the leaves are the hash of the elements x_i and each node is the hash of its children. It is a tree of height $\log n$. Formally, setting $d = \log n$ if we label the entries of $\mathbf{x}$ with the binary representation of their indexes then

$$\begin{array}{lr} \text{leaves:} & h_{b_0 b_1 \ldots b_d} = H(x_{b_0 b_1 \ldots b_d}) \\ \text{level } 1: & h_{b_0 b_1 \ldots b_{d-1}} = H(h_{b_0 b_1 \ldots b_{d-1} 0} \| h_{b_0 b_1 \ldots b_{d-1} 1}) \\ \text{level } k: & h_{b_0 b_1 \ldots b_{d-k}} = H(h_{b_0 b_1 \ldots b_{d-k} 0} \| h_{b_0 b_1 \ldots b_{d-k} 1}) \end{array}$$

[4] The reason being that Merkle trees can be realized from hash functions only which is fundamental to our purpose of realizing a VSS from symmetric-key cryptography only. Furthermore they do not require setup.

[5] If not we can pad the vector with zeros.

A public commitment to $\mathbf{x}$ then consists in the root of such a tree, with the convention that the root is indexed by the empty string ϵ, namely $\mathsf{cm} = h_\epsilon = H(h_0 \| h_1)$. Opening the commitment at position i can be done by revealing the corresponding leaf x_i and all the sibling values of all the nodes in the path from x_i till the root, which is logarithmic in the size of $\mathbf{x}$.

Formally, for a bit b, let $\overline{b}$ denote $1 - b$. Then an opening for position i with binary representation $b_0 \dots b_d$ is given by the list

$$\left(x_{b_0 \dots b_{d-1} b_d}, h_{b_0 \dots b_{d-1} \overline{b_d}}, h_{b_0 \dots b_{d-2} \overline{b_{d-1}}}, \dots, h_{b_0 \dots b_{d-k} \overline{b_{d-k+1}}}, \dots, h_{\overline{b_0}}\right).$$

Verifying the opening requires hashing the siblings to recompute the nodes in the path, and checking that the last node is indeed the same as the initial commitment root. The position binding of the Merkle tree can be reduced to the collision resistance of H.

It is possible to turn such a construction into an hiding vector commitment by computing the leaves using a uniformly random string from $\{0,1\}^\lambda$, one for each leaf. In other words $h_{b_0 b_1 \dots b_d} = H(x_{b_0 b_1 \dots b_d} \| r_{b_0 b_1 \dots b_d})$. Then an opening for position i must include r_i. For simplicity, in the rest of the paper we will assume that all Merkle trees are hiding.

2.4 Distributed Zero-Knowledge Proofs

Here we recall the definition of zero-knowledge proof for distributed relations, introduced in [13], suitably adapted to rings.

Definition 2 (Distributed Inputs, Languages, and Relations [13][). *Let n be a number of parties, R be a ring, and $l, l_1, l_2, \cdots, l_n \in \mathbf{N}$ be length parameters, where $l = l_1 + l_2 + \cdots + l_n$. An n-distributed input over R^l (or just distributed input) is a vector $x = x^{(1)} \| x^{(2)} \| \cdots \| x^{(n)} \in R^l$ where each $x^{(i)} \in R^{l_i}$ is called a piece (or share) of x. An n-distributed language $\mathcal{L} \subset R^l$ is a set of n-distributed inputs. A distributed NP relation with witness length h is a binary relation $\mathcal{R} \subset R^l \times R^h$ of pairs (x, w) with n-distributed input $x \in R^l$ and witness $w \in R^h$, such that $(x, w) \in \mathcal{R}$ can be checked in polynomial time. Finally, we let $\mathcal{L}_\mathcal{R} = \{x \in R^l : \exists w \in R^h,\ (x, w) \in \mathcal{R}\}$.*

Definition 3 (n-Verifier Interactive Proofs [13]). *An n-*Verifier Interactive Proof *protocol over R is an interactive protocol $\Pi = (P, V_1, V_2, \cdots, V_n)$ involving a prover P and n verifiers $V_1, V_2, \cdots, V_n$. The protocol proceeds as follows.*

- *In the beginning of the protocol the prover holds an n-distributed input $x = x^{(1)} \| x^{(2)} \| \cdots \| x^{(n)} \in R^l$, a witness $w \in R^h$, and each verifier V_j holds an input piece (or share) $x^{(j)}$.*
- *The protocol allows the parties to communicate in synchronous rounds over secure point-to-point channels and a broadcast channel.*
- *At the end of the protocol each verifier outputs either* 1 *(accept) or* 0 *(reject) based on its view, where the view of V_j consists of its input piece $x^{(j)}$, its random input $r^{(j)}$, and messages it received during the protocol execution.*

The protocol accepts if all verifiers accept, and the protocol rejects if at least one verifier rejects.[6]

In the following, $\Pi(x, w)$ denotes running Π on shared input x and witness w, and says that $\Pi(x, w)$ accepts (respectively, rejects) if at the end all verifiers (resp. at least one honest verifier) output 1 (resp., 0). $View_{\Pi,T}(x, w)$ denotes the (joint distribution of) views of verifiers $\{V_j\}_{j \in T}$ in the execution of Π on distributed input x and witness w.

Let $\mathcal{R}(x, w)$ be an n-distributed relation over a ring R. We say that an n-verifier interactive proof protocol $\Pi = (P, V_1, \cdots, V_n)$ is a distributed strong ZK proof protocol for R with t-security against malicious prover *and* malicious verifiers, and with soundness error ε, if Π satisfies the following properties:

Definition 4 (Correctness). *For every n-distributed input $x = x^{(1)} \| x^{(2)} \| \cdots \| x^{(n)} \in R^l$ and $w \in R^h$ such that $(x, w) \in \mathcal{R}$, the execution of $\Pi(x, w)$ accepts with probability* 1*. Note this definition assumes the prover and all verifiers behave honestly.*

Definition 5 (ε-Soundness Against Prover and t Verifiers). *For every $T \subseteq [n]$ of size $|T| \le t$, an adversary $\mathcal{A}$ controlling the prover P and verifiers $\{V_j\}_{j \in T}$, n-distributed input $x = x^{(1)} \| x^{(2)} \| \cdots \| x^{(n)} \in R^l$, and every $w \in R^h$, the following holds. If there is no n-distributed input $x' \in \mathcal{L}_{\mathcal{R}}$ such that $x'_H = x_H$, where $H = [n] \backslash T$, the execution of $\Pi^\star$ rejects except with probability at most ε, where $\Pi^\star$ denotes the interaction of $\mathcal{A}$ with the honest verifiers.*

In analogy to ordinary interactive proofs, we say soundness holds *adaptively* if the input is chosen by the malicious prover (see for instance [5]), potentially after observing the public parameters, or interacting with the random oracle.

Definition 6 (Strong Zero-Knowledge against t Verifiers). *For every $T \subseteq [n]$ of size $|T| \le t$ and a malicious adversary $\mathcal{A}$ controlling the verifiers $\{V_j\}_{j \in T}$, there exists a simulator* Sim *such that for every n-distributed input $x = x^{(1)} \| x^{(2)} \| \cdots \| x^{(n)} \in R^l$, and witness $w \in R^h$ such that $(x, w) \in \mathcal{R}$, we have $\mathsf{Sim}((x^{(j)})_{j \in T}) \equiv View_{\Pi^\star,T}(x, w)$. Here, $\Pi^\star$ denotes the interaction of adversary $\mathcal{A}$ with the honest prover P and the honest verifiers $\{V_j\}_{j \in [n] \backslash T}$.*

In order to later provide a compiler analogous to the Fiat-Shamir transform, we define a distributed proof to be *public coin* if in the interactive phase verifiers only executes a coin-tossing protocol $\mathcal{F}_{coin}$. Finally we also consider the stronger notion of *round-by-round soundness* [15], adapted to the distributed proof setting and restricted for simplicity to the public coin case.

[6] We stress the fact that in our definition, the protocol rejects if at least one verifier rejects, unlike [13], where the protocol rejects if all verifiers reject. Although the latter is a stronger notion of soundness, it is always possible to achieve by adding some rounds of consensus among the verifiers. Later we'll construct a VSS from a ZK proof on distributed inputs. This is done by adding such a consensus protocol outside the proof system. We made this choice so as to mark clearly the step from proof system to VSS.

Definition 7. *A distributed public coin proof has ε-round-by-round soundness against a prover and t verifiers if there exists a set D of doomed transcripts such that, for any $T \subseteq [n]$ of size $|T| \leq t$*

1. *Given $(x_i)_{i \notin T}$ such that $\nexists x' \in \mathcal{L}_{\mathcal{R}} : x'_i = x_i$ for $i \notin T$, then $(x_i, \varnothing)_{i \notin T} \in D$.*
2. *Given $(x_i, v_i)_{i \notin T} \in D$, where v_i denotes the view of verifier V_i till the current state, for any reply (M, m_i) where M, m_i denote the messages broadcast and privately sent to V_i respectively at the end of the current round, and random coins μ tossed by the verifiers*

$$\Pr\left[(x_i, (v_i \| M \| m_i \| \mu))_{i \notin T} \notin D\right] \leq \varepsilon(\lambda).$$

3. *For any list of full transcripts*[7], *$(x_i, v_i)_{i \notin T} \in D \Rightarrow \exists j \notin T : V_j$ rejects.*

2.5 Interpolation and Shamir Secret Sharing over Rings

A (t, n)-Shamir secret sharing scheme allows n parties to individually hold a share x_i of a common secret x_0, such that any subset of t parties or less are not able to learn any information about the secret x_0, while any subset of at least $t+1$ parties are able to efficiently reconstruct the common secret x_0. While Shamir's scheme is originally defined over a finite field $\mathbb{F}$, meaning the secret and all shares are values in $\mathbb{F}$, it can be extended to rings through exceptional sets, as described in [23]. We recall the details here.

Definition 8. *Let R be a ring. An* exceptional set *is a set $S \subset R$, where for every pair $x, x' \in S$ with $x \neq x'$, the difference $x - x'$ is invertible in R.*

Lemma 1. *Let $m > t \geq 0$ be integers, R a ring, $S = \{\alpha_1, \ldots, \alpha_m\} \subseteq R$ an exceptional set. Then for every $Q = \{i_1, \ldots, i_{t+1}\} \subseteq [m]$ the map $R[X]_t \to R^{t+1}$ given by $f(X) \mapsto (f(\alpha_{i_1}), \ldots, f(\alpha_{i_{t+1}}))$ is an R-module isomorphism.*

In details, the inverse of the isomorphism above is given by mapping $(x_1, \ldots, x_{t+1})$ to $f(X) = \sum_{i \in Q} x_i \cdot L_i^Q(X)$ where

$$L_i^Q(X) := \prod_{j \in Q \setminus \{i\}} \frac{\alpha_j - X}{\alpha_j - \alpha_i}$$

are the Lagrange basis polynomials, which are all of degree t, and well defined thanks to Q being exceptional.

Now (t, n)-Shamir secret sharing can be defined over R, as long as it contains an exceptional set of "evaluation points" $\mathcal{E} = \{\alpha_0, \alpha_1, \ldots, \alpha_n\} \subset R$ of size $n+1$. Each party P_i, $i \in [n]$, is associated to the element α_i while α_0 is associated to the secret. To share secret x_0, a polynomial $f(x)$ is chosen uniformly at random in the set of polynomials in $R[X]_t$ with $f(\alpha_0) = x_0$. Each party P_i is assigned the secret share $x_i = f(\alpha_i)$. Then any subset $Q \subseteq \{1, \ldots, n\}$ of at least $t+1$ parties can reconstruct the secret x_0 via Lagrange interpolation by computing $x_0 = f(\alpha_0) = \sum_{i \in Q} x_i \cdot L_i^Q(\alpha_0)$, where L_i^Q is as above. Moreover, also based on

[7] The transcript until the point when the verifier halts.

Lemma 1, a subset of t or fewer parties are not able to find $x_0 = f(\alpha_0)$, as this is information theoretically hidden from the other shares. See [23] for details.

Finally we will need another technical lemma involving exceptional sets, which can be derived easily from Lemma 1.

Lemma 2. *Let $S \subset R$ be an exceptional set. Let $N \geq 1$ be an integer and $h^{(0)}(X)$, $h^{(1)}(X), \ldots, h^{(N)}(X)$ be $N+1$ arbitrary polynomials in $R[X]$. Let $d := \max\left\{\deg h^{(i)}(X) : i \in \{0, \ldots, N\}\right\}$.*

Then if $\nu_1, \ldots, \nu_N$ are sampled independently and uniformly at random in S,

$$\Pr\left[\deg\left(h^{(0)}(X) + \sum\nolimits_{i=1}^{N} \nu_i h^{(i)}(X)\right) < d\right] \leq \frac{1}{|S|}$$

Proof. If $d > \deg h^{(i)}(X)$ for all $i \in \{1, \ldots, N\}$ (hence also $\deg h^{(0)} = d$), then the claim is trivial.

Otherwise there exists a (possibly non-unique) $\ell \in \{1, \ldots, N\}$ such that $\deg h^{(\ell)} = d$. We show that for *every* fixed choice of $(\nu_i)_{i \neq \ell} \in S^{N-1}$, there exists at most one $\nu_\ell \in S$ such that $\deg(h_0(X) + \sum_{i=1}^{N} \nu_i h^{(i)}(X)) < d$. This is clearly enough to show the claim.

Let $f(X) = h^{(0)}(X) + \sum_{i=1, i\neq\ell}^{N} \nu_i h^{(i)}(X)$. Note $f(X) + \nu_\ell h^{(\ell)}(X)$ is the polynomial whose degree we want to bound. Let f_d, $h_d^{(\ell)}$ respectively denote the coefficients of X^d in $f(X)$ and $h^{(\ell)}(X)$ (if $\deg f < d$ then $f_d = 0$). Note that $h_d^{(\ell)} \neq 0$. Let $m(X) = f_d + h_d^{(\ell)} X \in R[X]_1$. There is at most one value $s \in S$ such that $m(s) = 0$, otherwise if $m(s') = 0$ for some other $s' \in S$, then $m(X)$ and 0 would be two different polynomials in $R[X]_1$ with the same evaluations in a set $\{s, s'\} \subset S$ of size 2, which is impossible by Lemma 1. Therefore there is at most one value $\nu_\ell = s$ in S for which the coefficient of X^d in $f(X) + \nu_\ell h^{(\ell)}(X)$ is 0, and hence for which this polynomial can have degree less than d.

2.6 Verifiable Secret Sharing Scheme

Definition 9 (From [6,21]). *A (t, n)-VSS protocol is an interactive protocol between n parties $P_1, \ldots, P_n$ and a distinguished party, the dealer, denoted by D and consists of two phases, a sharing phase and a reconstruction phase, defined as follows:*

1. Share*: initially D holds an input x_0, referred to as the secret. The sharing phase may consist of several rounds of interaction between the parties. At the end of the sharing phase, each honest party P_i holds a view v_i that may be used later to reconstruct the dealer's secret.*
2. Reconstruction*: in this phase each party P_i publishes its entire view v_i from the sharing phase, and a reconstruction algorithm* Reconstruction$(v_1, \ldots, v_n)$ *is run and the output is taken as the protocol output.*

Definition 10. *A (t, n)-VSS is secure if for every adversary that controls parties $\{P_i\}_{i \in T}$ belonging to a subset $T \subseteq [n]$ of size $|T| \leq t$, possibly including the dealer, it satisfies the following properties up to negligible probability.*

1. ***Correctness.*** *If the dealer is honest (i.e., not controlled by the adversary), then all honest parties output x_0 at the end of* Reconstruction*.*
2. *t-**Privacy.** If the dealer is honest, then the adversary's view at the end of* Share *reveals no information about the secret x_0. In other words, the adversary's view is identically distributed for all possible secrets x_0.*
3. ***Commitment.*** *If the dealer is dishonest (i.e. controlled by the adversary), then at the end of* Share *there exists a unique value $x_0^* \in R \cup \{\perp\}$ such that at the end of* Reconstruction *all parties return x_0^*.*

A stronger notion of VSS requires correctness, t-privacy and strong commitment defined as follows:

3. ***Strong commitment.*** *The scheme has the commitment property above and in addition, if the dealer is dishonest, at the end of the sharing phase each (honest) party locally outputs a share of the secret chosen only in R, such that the joint shares output by honest parties are consistent with a specified secret sharing scheme.*

In this work we will focus on Shamir secret sharing schemes. Then the definition of strong commitment means that at the end of the sharing phase, the shares x_i held by the honest parties implicitly define a polynomial $f(X) \in R[X]_t$ of degree t such that $f(\alpha_i) = x_i$.

Our constructions will actually only achieve a computational flavor of privacy. In this case, we argue that for any two different secrets x_0, x_0^*, the distributions of the views of any adversary corrupting at most t share-receivers in respectively a sharing of x_0 and x_0^* are computationally indistinguishable. We capture this by stating that for any such an adversary there is a simulator that can produce a view that is computationally indistinguishable of the sharing of any secret x_0.

Definition 11. ***Computational t-privacy.*** *For any adversary $\mathcal{A}$ corrupting a set of parties $\{P_i\}_{i \in T}$, with $|T| \leq t$, there exists a simulator $\mathcal{S}$ that interacts with $\mathcal{A}$, playing the role of the honest dealer and honest parties $\{P_i\}_{i \in [n] \setminus T}$, such that for any $x_0 \in R$, $View_{\mathcal{A},\mathcal{S}} \equiv_c View_{\mathcal{A},Share}(x_0)$ where $View_{\mathcal{A},\mathcal{S}}$ and $View_{\mathcal{A},\mathsf{Share}}(x_0)$ are the random variables describing the view of $\mathcal{A}$ when interacting with $\mathcal{S}$ and when interacting with the dealer and $\{P_i\}_{i \in [n] \setminus T}$ in the sharing phase* Share *of the VSS, where the dealer has input x_0.*

3 Distributed Low-Degree Proofs

3.1 Interactive (non-ZK) Distributed Low-Degree Proof

We describe now a distributed (interactive) proof for the existence of a low-degree polynomial interpolating a distributed input, under the assumption that no more than t out of n verifiers collude. More precisely, let R be a ring, $\mathcal{E} = \{\alpha_1, \ldots, \alpha_n\} \subseteq R$ an exceptional set and $d \in \mathbb{N}$. Consider the relation

$$\mathcal{R}^{d,\mathcal{E}}_{\mathsf{lowdeg}} = \{(x, f) : x = (x_1, \ldots, x_n) \in R^n,\ f \in R[X]_{d-1},\ f(\alpha_i) = x_i\ \forall i \in [n]\}.$$

Provided each verifier V_i holds x_i, with $x = (x_1, \ldots, x_n)$, our protocol proves x is in the language induced by the relation above. Note we are not concerned with zero-knowledge here, which will be addressed later in Sect. 3.3.

The proof is based on folding, resembling FRI [9] and [14]: the problem of showing low-degree is self-reduced at each round to an instance with half the initial degree. Specifically, let $f^{(k-1)}$ be the polynomial claimed to have degree d_{k-1} at the start of round k.

The prover *splits* $f^{(k-1)}$ deterministically into $g_0^{(k)}, g_1^{(k)}$, both polynomials of degree at most $d_k = d_{k-1}/2$. Each verifier V_i receives an evaluation of the two polynomials above, at a certain point which depends on α_i and k. Finally, a random challenge is sampled, and the prover proceeds recursively, showing $f^{(k)} = g_0^{(k)} + \mu g_1^{(k)}$ has degree at most d_k. After τ rounds, $f^{(\tau)}$ is eventually sent in clear. Verifiers can then recursively check the split was correctly computed, relative to their assigned point α_i, using $f^{(\tau)}$ and the evaluations provided by the prover. The proof is eventually accepted if *all* honest verifiers accept it.

As for how to split the polynomials, we consider two possibilities: the first one used in $\Pi^{d,\mathcal{E}}_{\mathsf{lowdeg}}$ (Fig. 1) consists on splitting f into high-degree and low degree terms as in [14]. If we can however further assume each of the sets $\mathcal{E}^{(k)} = \{\alpha_i^{2^k} : i \in [n]\}$ for $k \in \{0, \ldots, \tau\}$ to be exceptional (which is the case for fields)[8], we could improve on efficiency setting $f^{(k-1)}(X) = g_0^{(k)}(X^2) + X \cdot g_1^{(k)}(X^2)$ as done in FRI. This is especially beneficial when $\mathcal{E}$ is the set of 2^m-th roots of unity over a field, for some m. This second variant $\Pi^{d,\mathcal{E}}_{\mathsf{lowdeg-alt}}$, is detailed in Fig. 2.

For simplicity, we assume that d is a power of 2, but this can easily be extended to general d, see Sect. 3.6. For our proof we need a large exceptional set S in the ring R, which may overlap with $\mathcal{E}$. The soundness error will be inversely proportional to the size of S. Soundness amplification techniques are discussed in Sect. 3.7. The parameter $\tau \le \log d$ is set for flexibility, so that we stop the recursion when the current polynomial $f^{(\tau)}$ has degree $d/2^\tau$. Nonetheless for asymptotics we will always consider the choice $\tau = \Theta(\log d)$.

Theorem 1. *Protocol $\Pi^{d,\mathcal{E}}_{\mathsf{lowdeg}}$ in Fig. 1 is a correct distributed proof for $\mathcal{R}^{d,\mathcal{E}}_{\mathsf{lowdeg}}$ with $1/|S|$-round by round soundness against a malicious prover and up to n corrupted verifiers.*

Proof. Correctness is trivial to verify. Regarding soundness we explicitly describe a *doomed* set D. Let T be the set of corrupted parties, $H = [n]\backslash T$ and $\mathcal{E}_H = \{\alpha_i\}_{i \in H}$. Given a state $(x_i, v_i)_{i \in H}$ until the h-th round, we call $g_0^{(k)}$, $g_1^{(k)}$ the polynomials obtained interpolating the values honest verifiers receive at round k, $f = f^{(0)}$ the interpolation of $(x_i)_{i \in H}$ and $f^{(k-1)} = g_0^{(k)} + X^{d_k} g_1^{(k)}$ for $1 < k \le \tau$, with μ_k being the k-th round challenge from $\mathcal{F}_{coin}$. Then such tuple of views lies if D if and only if at least one of the following conditions is satisfied:

1. $\deg\left(g_0^{(h)} + \mu_h g_1^{(h)}\right) \ge d_h$.

[8] But it may be a stronger condition on rings; as an example consider $R = \mathbb{Z}_{15}$, where $\mathcal{E} = \{2, 3\}$ is exceptional, while $\mathcal{E}^{(1)} = \{4, 9\}$ is not, since $9 - 4$ divides 0.

Protocol $\Pi^{d,\mathcal{E}}_{\mathsf{lowdeg}}(x, f)$

n-verifier interactive proof for the relation

$$\mathcal{R}^{d,\mathcal{E}}_{\mathsf{lowdeg}} = \{(x, f) \,:\, x \in R^n,\ f \in R[X]_{d-1},\ f(\alpha_i) = x_i\ \forall i \in [n]\}$$

where x is distributed among n verifiers (verifier V_i having as input x_i), and $\mathcal{E} = \{\alpha_1, \ldots, \alpha_n\} \subseteq R$ is an exceptional set. We assume d to be a power of 2 The proof is parametrized by $\tau \in \mathbb{N}$ with $\tau \leq \log_2 d$. For $k \in [\tau]$ we set $d_k = d/2^k$. The proof requires $S \subseteq R$ be a (large) exceptional set, which may overlap with $\mathcal{E}$. $\mathcal{F}^S_{coin}$ is a coin-tossing functionality sampling and broadcasting a random element in S on each invocation.

Interactive Protocol: The prover P sets $f^{(0)} \leftarrow f$. Then for $k \in [\tau]$:

1. P computes $g_0^{(k)}$, $g_1^{(k)}$ of degree $< d_k$ such that $f^{(k-1)} = g_0^{(k)} + X^{d_k} \cdot g_1^{(k)}$
2. P **privately sends** $g_0^{(k)}(\alpha_i)$, $g_1^{(k)}(\alpha_i)$ to each V_i
3. The verifiers call the coin-tossing functionality $\mu_k \xleftarrow{\$} \mathcal{F}^S_{coin}$ with $\mu_k \in S$
4. P computes $f^{(k)} \leftarrow g_0^{(k)} + \mu_k \cdot g_1^{(k)}$

Finally P **broadcasts** the polynomial $f^{(\tau)}$ **Verification:** Each V_i sets $f^{(k)}(\alpha_i) \leftarrow g_0^{(k+1)}(\alpha_i) + \alpha_i^{d_{k+1}} g_1^{(k+1)}(\alpha_i)$ for $k \in [\tau - 1]$ and accepts if and only if the following checks hold:

5. $\deg f^{(\tau)} < d/2^\tau$
6. $x_i = g_0^{(1)}(\alpha_i) + \alpha_i^{d/2} \cdot g_1^{(1)}(\alpha_i)$
7. $f^{(k)}(\alpha_i) = g_0^{(k)}(\alpha_i) + \mu_k \cdot g_1^{(k)}(\alpha_i)$ for all $k \in [\tau]$

Fig. 1. Distributed low-degree proof $\Pi^{d,\mathcal{E}}_{\mathsf{lowdeg}}$.

2. $f^{(0)}(X) \neq g_0^{(1)}(X) + X^{d/2} \cdot g_1^{(1)}(X)$.
3. $f^{(k)}(X) \neq g_0^{(k)}(X) + \mu_k \cdot g_1^{(k)}(X)$ for some $k \leq h$, $k < \tau$.

If the given input does not lie in the projection of the associated language over indices H, i.e. if there is no polynomial of degree $d-1$ interpolating $(x_i)_{i \in H}$, then we have $(x_i)_{i \in H} \in D$ from the first condition. Next, if a complete transcript lies in D, then conditions 2, 3 implies one verifier rejects, as they are all explicitly checked. Conversely if the complete transcript satisfies condition 1, then either all verifiers reject as $\deg f^{(\tau)} \geq d_\tau$ or $\deg f^{(\tau)} < d_\tau$. The second case however implies $f^{(\tau)}(\alpha_i) \neq g_0^{(\tau)}(\alpha_i) + \mu_\tau g_1^{(\tau)}(\alpha_i)$ for some α_i. This is true as $g_0^{(\tau)} + \mu_\tau g_1^{(\tau)}$ has degree at least d_τ and it is the polynomial of minimum degree taking its values in $\mathcal{E}_H$ since both $g_0^{(\tau)}$ and $g_1^{(\tau)}$ must have degree smaller than $|\mathcal{E}_H| - 1$.

Finally, we show escaping D is hard (condition 2 in Definition 7) Regarding the first message, if $(x_i, \varnothing)_{i \in H} \in D$ then $\deg f^{(0)} \geq d$. Thus for any messages resulting in $g_0^{(1)}, g_1^{(1)}$, if condition 2 is not satisfied, the extended view lies in D. If not instead, at least one of the two polynomials must have degree $\geq d/2$. By Lemma 2 we then have that for a uniformly sampled μ_1 the first condition is not satisfied with probability $\leq 1/|S|$. For a transcript until the h-th round lying in D instead, if conditions 2 and 3 are satisfied no reply can end outside of D. Conversely if they are both not satisfied, the first one must be. The next message then is either such that

$$g_0^{(h+1)}(X) + X^{d_h/2} g_1^{(h+1)} \neq g_0^{(h)} + \mu_h g_1^{(h)}$$

which implies that the third condition is false, or not, which implies that at least one of $g_0^{(h+1)}, g_1^{(h+1)}$ has degree at least $d_h/2 = d_{h+1}$. Using again Lemma 2, we have that their random linear combination also has degree at least d_{h+1} up to probability $1/|S|$. This conclude our argument for round-by-round soundness.

3.2 Improvements for Specific Rings

Assume now that, for a given exceptional set $\mathcal{E} = \{\alpha_1, \ldots, \alpha_n\}$, all the sets $\mathcal{E}^{(k)} = \{\alpha_i^{2^k} : \alpha_i \in \mathcal{E}\}$ for $k \in [\tau]$ are exceptional and $|\mathcal{E}^{(k)}| = 2 \cdot |\mathcal{E}^{(k+1)}|$. In particular, these assumptions hold if n is a power of 2, R is a finite field $\mathbb{F}$ with a primitive n-th root of unity (i.e. the multiplicative order $|\mathbb{F}| - 1$ is a multiple of n) and $\mathcal{E}$ is the set of all n-th roots of the unity.[9] In these conditions, we present an alternative protocol $\Pi^{d,\mathcal{E}}_{\mathsf{lowdeg\text{-}alt}}$ in Fig. 2. The only difference with $\Pi^{d,\mathcal{E}}_{\mathsf{lowdeg}}$ lies in the splitting procedure for f, now divided into even and odd powers terms.

The main advantage of this approach is that, as the domain decreases in size, evaluating the intermediates polynomials becomes faster for the prover. More specifically, P performs $O(n/2^k \cdot \log n)$ ring operations in round k for two FFTs, which in total amounts to $O(n \log n)$. Looking forward, this also improves proof size for the compiled non-interactive proof by a factor 2, see Sect. 3.4.

3.3 Zero-Knowledge Compiler

We now show how to turn the protocol in Fig. 1 into a distributed zero-knowledge proof. The (standard) idea is to make the prover mask $f(X)$ with a random low-degree polynomial $b(X)$. Specifically, P first shares evaluations of $b(X)$ among the verifiers, and later shows $b(X) + \mu_0 \cdot f(X)$, for a randomly sampled μ_0, to have low degree. The protocol is detailed in Fig. 3.

Theorem 2. *Let Π be a correct distributed proof for $\mathcal{R}^{d,\mathcal{E}}_{\mathsf{lowdeg}}$ with ε-round by round soundness against any number of corruptions. Then $\Pi^{d,\mathcal{E}}_{\mathsf{dZKlowdeg}}$ (Fig. 3) is a correct distributed proof for the same relation with* $\max(\varepsilon, 1/|S|)$*-round by round soundness and perfect zero-knowledge against any number of corruptions.*

A full proof appears in the full version

Remark 1. Applying either of our distributed proof for low-degree to the protocol in Fig. 3 yields a perfect zero-knowledge proof with round-by-round soundness error $1/|S|$.

[9] For instance, the scalar fields of BLS12-377 and BLS12-381 admit 2^m-th roots of unity respectively for all $m \leq 44$ and $m \leq 32$.

Protocol $\Pi^{d,\mathcal{E}}_{\mathsf{lowdeg\text{-}alt}}(x, f)$

Notation as in Figure 1. $\mathcal{E} = \{\alpha_1, \ldots, \alpha_n\}$ and $\mathcal{E}^{(k)} = \{\alpha_1^{2^k}, \ldots, \alpha_n^{2^k}\}$ are exceptional for $k \leq \tau$. Note $\mathcal{E} = \mathcal{E}^{(0)}$. We denote $\alpha_i^{(k)} = \alpha_i^{2^k}$.

Interactive Protocol: As in Figure 1 up to replacing line 1 and 2 in the loop by:

1*. P computes $g_0^{(k)}$, $g_1^{(k)}$ of degree $< d_k$ so that $f^{(k-1)}(X) = g_0^{(k)}(X^2) + X \cdot g_1^{(k)}(X^2)$

2*. P **privately sends** $g_0^{(k)}\left(\alpha_i^{(k)}\right)$, $g_1^{(k)}\left(\alpha_i^{(k)}\right)$ to each V_i

Verification: Each V_i sets $f^{(k)}(\alpha_i^{(k)}) \leftarrow g_0^{(k+1)}(\alpha_i^{(k+1)}) + \alpha_i^{(k)} g_1^{(k+1)}(\alpha_i^{(k+1)})$ for $k \in [\tau - 1]$ and accepts iff the checks in Figure 1 hold, up to replacing line 6 with:

6*. $x_i = g_0^{(1)}(\alpha_i^2) + \alpha_i \cdot g_1^{(1)}(\alpha_i^2)$

Fig. 2. Distributed low-degree proof with alternative splitting $\Pi^{d,\mathcal{E}}_{\mathsf{lowdeg\text{-}alt}}$.

Protocol $\Pi^{d,\mathcal{E}}_{\mathsf{dZKlowdeg}}(x, f)$

n-verifier interactive proof for the relation $\mathcal{R}^{d,\mathcal{E}}_{\mathsf{lowdeg}}$. Notation is as in Figure 1. Π is a (non-zk) distributed proof for $\mathcal{R}^{d,\mathcal{E}}_{\mathsf{lowdeg}}$.

Interactive Protocol:

1. P samples $b \xleftarrow{\$} R[X]_{d-1}$ and *privately sends* $b_i \leftarrow b(\alpha_i)$ to V_i
2. The verifiers call the random coin functionality $\mu_0 \xleftarrow{\$} \mathcal{F}^S_{coin}$
3. All parties execute Π: P with input $b(X) + \mu_0 \cdot f(X)$, V_i with input $b(\alpha_i) + \mu_0 \cdot x_i$

Verification: Each verifier V_i accepts if and only if its execution of Π is accepting.

Fig. 3. Zero-knowledge distributed proof of low-degree.

3.4 Removing Interaction in the ROM

In this Section we describe a generic Fiat-Shamir like compiler for distributed proofs in the Random Oracle Model (ROM). This is essentially an adaptation of the one presented in [11] to the distributed setting.

At a high level, let Π be a *public-coin* τ-rounds distributed protocol. That is, one where verifiers can only invoke $\mathcal{F}_{coin}$ at the end of each round, and decide whether to accept or not based on their view after the protocol ends. Given such Π and random oracle $\mathcal{H}$ the non-interactive protocol works as follows. The prover internally simulates Π. At the end of round k, it collects the message M_k it wishes to broadcast and $m_{k,i}$ the message it would have privately sent to V_i. Then it computes c_k as a commitment to the vector of all $m_{k,i}$, $i \in [n]$, and obtain its next challenge μ_k (i.e. the expected output of $\mathcal{F}_{coin}$) as the hash of all the commitments c_j and virtually broadcast message M_j computed so far. When Π halts, P broadcasts all c_k, M_k and privately sends $m_{k,i}$ to V_i along with opening information. Finally, verifiers accept if all openings are correct and if the transcript is accepting. A detailed description is provided in Fig. 4.

```
FS-like Compiler $\mathsf{FS}[\Pi](x, w)$
    $\mathcal{H}$ is a random oracle. $\mathsf{MT}$ is a Merkle tree vector commitment realized with $\mathcal{H}$.
    $\Pi$ is a $\tau$ rounds distributed proof for relation $\mathcal{R}$.

Interactive Protocol: The prover $\mathsf{P}$ performs the following:
 1. Computes $(c_0, \mathsf{aux}_0) \leftarrow \mathsf{MT.Com}(\mathsf{pp}, x)$ commitment to the distributed input
 2. $\mathsf{op}_{0,i} \leftarrow \mathsf{MT.Open}(\mathsf{pp}, i, \mathsf{aux}_0)$ openings of $c_0$
 3. Internally runs $\Pi$ on input $(x, w)$
 4. For $k \in [\tau]$, at the end of the $k$-th round:
 5.     Let $m_{k,i}$ be the messages it would have privately sent to $V_i$
 6.     Let $M_k$ be the message it would have broadcast
 7.     $c_k, \mathsf{aux}_k \leftarrow \mathsf{MT.Com}(\mathsf{pp}, m_{k,1}, \ldots, m_{k,n})$, commitment to private messages
 8.     $\mathsf{op}_{k,i} \leftarrow \mathsf{MT.Open}(\mathsf{pp}, i, \mathsf{aux}_k)$, opening of $c_k$ in position $i$
 9.     $\mu_k \leftarrow \mathcal{H}(c_0, c_1, M_1, \ldots, c_k, M_k)$ and set it as the output of $\mathcal{F}_{coin}$ in $\Pi$
10. When the execution of $\Pi$ terminates sets:
11.     $\pi \leftarrow (c_0, c_1, M_1, \ldots, c_\tau, M_\tau)$ and $\pi_i \leftarrow (\mathsf{op}_{0,i}, m_{1,i}, \mathsf{op}_{1,i}, \ldots, m_{\tau,i}, \mathsf{op}_{\tau,i})$
12. Broadcast $\pi$ and privately send $\pi_i$ to $V_i$

Verification: Each verifier $V_i$ retrieves $c_0, \mathsf{op}_{0,i}$ and $M_k, m_{i,k}, c_k, \mathsf{op}_{k,i}$ for $k \in [\tau]$
from $(\pi, \pi_i)$. Next it computes $\mu_k$ as done in Line 9, and accepts if and only if
13. $1 = \mathsf{MT.Vfy}(\mathsf{pp}, c_0, x_i, i, \mathsf{op}_{0,i})$
14. $1 = \mathsf{MT.Vfy}(\mathsf{pp}, c_k, m_{k,i}, i, \mathsf{op}_{k,i})$ for all $k \in [\tau]$
15. Its local view $(x_i, m_{1,i}, M_1, \mu_1, \ldots, m_{\tau,i}, M_\tau, \mu_\tau)$ is accepting according to $\Pi$
```

Fig. 4. Generic Fiat-Shamir like compiler for distributed proofs.

Theorem 3. *Let Π be a correct τ-rounds distributed proof for $\mathcal{R}$ with ε-round by round soundness against up to t verifiers. Then $\mathsf{FS}[\Pi]$ is correct and ε'-sound for adaptively chosen inputs in the ROM against a q-query adversary $\mathcal{A}$ corrupting up to t verifiers, where*

$$\varepsilon' = (q + \tau)\varepsilon + (2q^2)/2^\lambda .$$

In order to preserve zero-knowledge a necessary condition is for the used MT commitment to be hiding (see Sect. 2.3). Let then $\mathsf{FS}_{\mathsf{zk}}$ be the compiler FS up to replacing MT with hiding MT. This can be shown to preserve zero-knowledge.

Theorem 4. *Let Π be a public-coin zero-knowledge distributed proof for $\mathcal{R}$ against t malicious verifiers. Then $\mathsf{FS}_{\mathsf{zk}}[\Pi]$ is computationally zero-knowledge in the ROM against t malicious verifiers.*

We include the proofs of Theorems 3 and 4 in the full version.

3.5 Efficiency

Following are the asymptotic costs of protocols in Fig. 1 and 2 made ZK (Fig. 3) and then non-interactive via $\mathsf{FS}_{\mathsf{zk}}$ (Fig. 4). We set $\tau = \Theta(\log d)$ and assume that the points α_i were pre-assigned to the respective verifier.

Communication. In the general ring setting, the prover broadcasts $\Theta(\log d)$ hash values (the MT roots) and $d\,2^{-\tau} = \Theta(1)$ ring elements. It further privately communicates $\Theta(\log n \cdot \log d)$ hash values and $\Theta(\log d)$ ring elements. For rings supporting the alternative protocol instead, private communication involves only $\leq (\tau+1)\log n - 1/2 \cdot \tau^2$ hashes. For $d \approx n$ those are roughly half as before.

Verifier Computation. Each verifier performs $O(\log d)$ operations[10] in R for checks 6–7 (Fig. 1) and $O(\log d{\cdot}\log n)$ hash evaluations to check the MT openings. As before, the second protocol requires about half hash evaluations for $d \approx n$.

Prover Computation. Here the computation is dominated by the evaluations of intermediate polynomials in over $\mathcal{E}$. Without additional assumptions, Hörner's method allows evaluating in $O(nd)$ ring operations[11]. If we can use FFTs then computation is reduced to $O(n \log n \log d)$. Finally, if the alternative protocol is used, this is further reduced to $O(n \log d)$ ring operations as we perform 2 FFTs evaluation on a domain of size $n/2^k$ for a degree $d/2^k$ polynomial, for $0 \leq k \leq \tau$.

In addition to the above, the basic protocol requires $O(n \log n)$ hash evaluations, whereas the alternative one only requires $O(n)$.

3.6 Dealing with Any Degree d

The protocols in Fig. 1 and Fig. 2 can be extended to deal with the case where d is not a power of 2, as follows. Let d_1 be the largest power of two strictly smaller than d, and $d_k = d_1/2^{k-1}$, for $k \geq 1$. Modify then the first loop iteration for $k = 1$ computing $g_0^{(1)}$, $g_1^{(1)}$ such that $f^{(0)}(X) = g_0^{(1)}(X) + X^{d-d_1} g_1^{(1)}(X)$, with the degree conditions $\deg g_0^{(1)}(X) < d - d_1$, $\deg g_1^{(1)}(X) < d_1$ (we use this splitting for the first step even in the case of Fig. 2)

From there on all successive steps hold in the same way (with the small change that check 1 by the verifier should be that $\deg f^{(\tau)} < d_1/2^{\tau-1}$). It is easy to see that Theorem 1 would still hold.

3.7 Soundness Amplification

All our interactive constructions have round-by-round soundness error $1/|S|$, where S is the largest exceptional set in R, thus requiring S to be exponentially large in λ. Note however that there is in principle no guarantee such S exists:

[10] Assuming $\alpha_i^{(k)} = \alpha_i^{2^k}$ was precomputed.

[11] Here we are using the fact that each evaluation of a polynomial of degree d_k would take $O(d_k)$ operations in R, and $\sum_{k=0}^{\tau} d_k = \sum_{k=0}^{\tau} d/2^k < 2d$.

indeed, while the relation $\mathcal{R}^{d,\mathcal{E}}_{\mathsf{lowdeg}}$ requires the existence of an exceptional set $\mathcal{E}$, this only needs to be of size n, which is not enough by itself.

Unfortunately, parallel repetitions of (FS-compiled) multi-round proofs may not amplify soundness efficiently [5]. For this reason we show a simple way to improve soundness assuming a ring extension $R \subseteq R'$ with an exponentially large exceptional set $S' \subseteq R'$ is known. The idea is to execute the protocol in Fig. 1 replacing R by R' and with S' playing the role of S. However, while this guarantees soundness, it only shows, in principle, the existence of a witness $f \in R'[X]$ but not in $R[X]$. Nevertheless, provided each party also checks $f(\alpha_i) \in R$, this is sufficient as the following lemma shows.

Lemma 3. *Let d, n be positive integers, R a ring with an exceptional set $\mathcal{E} = \{\alpha_1, \ldots, \alpha_n\}$ and extension $R \subseteq R'$ and call $\mathcal{E}_H = \{\alpha_i : i \in H\} \subset \mathcal{E}$ for a given subset $H \subseteq [n]$ of size m. Finally let $x_i \in R$ for all $i \in H$. If there exists a polynomial $f' \in R'[X]_{d-1}$ with $f'(\alpha_i) = x_i$ for all $i \in H$, then there exists a polynomial $f \in R[X]_{d-1}$ with $f(\alpha_i) = x_i$ for $i \in H$. Moreover $d < m \Rightarrow f = f'$.*

Proof. First, by Lemma 1, since $\mathcal{E}_H$ is exceptional and contained in R, we know there is a polynomial $f \in R[X]$ of degree $\leq m - 1$ such that $f(\alpha_i) = x_i$ for $i \in H$. If $d \geq m$ we are done. Assume then that $d < m$. Note that f is also in $R'[X]$, hence f and f' are two polynomials in $R'[X]$ with $f(\alpha_i) = f'(\alpha_i)$ for i in H. Since they are both polynomials of degree $\leq m - 1$ (because we are in the case $d < m$) then Lemma 1 guarantees $f = f'$ and therefore $f \in R[X]_{d-1}$.

For the simple and commonly encountered case of $R = \mathbb{Z}_{p^k}$ such extension consists of the Galois ring $R' = GR(p^k, r)$ for extension degree $r = \lambda / \log p$. Indeed, it is well known that $GR(p^k, r)$ contains an exceptional set of p^r elements [12].

4 Verifiable Secret Sharing

4.1 Verifiable Secret Sharing Scheme

In this section we construct a VSS (see Definition 10) starting from the non-interactive distributed proof from the previous section. Our VSS supports Shamir secret sharing of elements in a ring R among n users, as long as there exists an exceptional set $\mathcal{E}^* = \{\alpha_0, \alpha_1, \ldots, \alpha_n\} \subseteq R$ of size at least $n+1$. We also assume a large $S \subseteq R$ exceptional set to instantiate the non-interactive distributed proof in Sect. 3 with high soundness. Otherwise the techniques presented in Sect. 3.7 could be used.

At a high level, following [3], our VSS is obtained combining a (non-interactive) distributed proof of low degree with a complaining phase. The sharing phase consists of three rounds. In the first round, the dealer computes shares x_i of a secret x_0 along with a distributed proof $(\pi, \pi_1, \ldots, \pi_n)$ of low degreeness. It then broadcasts π and privately sends each proof piece to the corresponding verifier. Next, in the second round, every user checks the received proof and, if

Verifiable Secret Sharing scheme

(t, n)-VSS for sharing a secret $x_0 \in R$. We assume $\mathcal{E}^* = \{\alpha_0, \alpha_1, \dots, \alpha_n\} \subseteq R$ is an exceptional set, call $\mathcal{E} = \{\alpha_1, \dots, \alpha_n\}$ and let $S \subseteq R$ be the largest exceptional subset. We call $\Pi = \mathsf{FS}_{\mathsf{zk}}\left[\Pi^{t+1,\mathcal{E}}_{\mathsf{dZKlowdeg}}\right]$. See Sections 3.3-3.4.

Share: Let $x_0 \in R$ be the secret to be shared.

- **First Round**. The dealer proceeds as follows:
  1. Sample a uniformly random $f(X) \xleftarrow{\$} R[X]_t$ such that $f(\alpha_0) = x_0$
  2. Set $x_i := f(\alpha_i)$ the i-th user's share, for $i \in [n]$
  3. Run Π on input (x, f) where $x = (x_i)_{i \in [n]}$ and get $\pi, (\pi_i)_{i\in[n]}$.
  4. **Broadcast** π and **privately send** (x_i, π_i) to V_i
- **Second Round**. Each user P_i, upon receiving x_i and π, π_i, verifies its proof running Π as the i-th verifier. If it rejects, P_i **broadcasts** a complaint bit.
- **Third Round**. For each complaining P_j, the dealer **broadcasts** (x_j, π_j).
- **Finalization**. If all broadcast π_j are accepting, the protocol continues as normal (P_i with share x_i). Otherwise, parties disqualify the dealer and set their share to the default value 0.

Reconstruction: To reconstruct the secret x_0, parties proceed in one round as follows:

- Each party P_i **broadcasts** its local view (x_i, π_i) from the sharing phase
- Each party P_i performs Lagrange interpolation on values x_j with an accepting proof (π, π_j), and returns the evaluation of such polynomial in α_0.

Fig. 5. Verifiable Secret Sharing from non-interactive distributed ZKP.

incorrect or missing, broadcasts a complain bit. Finally, in case of complaints, there is a third round where the dealer publicly broadcasts the share and proof of each complaining users, which all parties locally verify.

In order to reconstruct, users simply broadcast their (local) views. If the dealer was previously disqualified[12], then parties agree on 0 being the final secret. Otherwise, each party collects the shares x_j sent with an accepting proof, computes $f(X) \in R[X]_t$ interpolating them, and obtains the secret $x_0 = f(\alpha_0)$. The full protocol is detailed in Fig. 5.

Theorem 5. *Suppose $2t + 1 \le n$ and Π is a non-interactive ZK distributed proof with ε-adaptive soundness, with negligible ε. Then the protocol in Fig. 5 is a VSS satisfying correctness, t-privacy and strong commitment in the ROM.*

Proof. We prove each property separately.

Correctness. Assuming the dealer is honest, let $\mathcal{A}$ be an adversary corrupting T verifiers. Let $\mathcal{B}$ be an adversary playing against the soundness game of the

[12] This may happen if the dealer addressed a complaint by broadcasting an incorrect proof.

distributed ZKP. Specifically, $\mathcal{B}$ will start the protocol and play the role of the honest parties, until the reconstruction phase, where it picks an index i uniformly at random from the parties in $[n]\backslash H$ that output an accepting view. Then it outputs the partial proof π_i along with the public part of the proof π_{pub}.

Note that the dealer cannot be disqualified at the end of the sharing phase and in case of complaints from corrupt parties it will output views that are going to be accepted by the correctness of the distributed ZKP. Therefore the only step where correctness might fail is in the reconstruction, in the event that a corrupt party opens an incorrect view that passes verification. More specifically, call E the event that at least one party P_i controlled by the adversary $\mathcal{A}$ outputs an incorrect view that verifies in the reconstruction phase. If E happens then $\mathcal{B}$ will pick the corresponding view with probability at least $\frac{1}{t}$. Thus the advantage of $\mathcal{B}$ in winning the soundness game is bounded by $\frac{1}{t}\Pr[\mathsf{E}]$. If the event E does not happen, then all parties output correct views that by the property of correctness of the distributed ZKP are accepted and thus honest parties are able to reconstruct the correct share with probability 1, hence $\mathcal{A}$ does not win the game against the correctness of the VSS. This means that $\text{Adv}^{\text{corr}}_{\text{VSS}}(\mathcal{A}) \leq \Pr[E] \leq t\text{Adv}^{\text{snd}}_{\text{dZKP}}(\mathcal{B})$.

t-Privacy. Given an adversary $\mathcal{A}$ corrupting parties $\{P_i\}_{i\in T}$ for some $T \subseteq [n]$ with $|T| \leq t$, we describe a simulator $\mathcal{S}$ playing the role of the honest parties. First, let $\mathcal{S}_\Pi$ be a simulator for Π against an adversary corrupting T parties[13] Then, $\mathcal{S}$ initially samples uniformly randomly $x_i \xleftarrow{\$} R[X]_t$ for $i \in T$. Next, it computes (π, π_i) executing $\mathcal{S}_\Pi$ on input $(x_i)_{i\in T}$, and proceed sending (π, x_i, π_i) to corrupted parties. In case of complaints by a corrupted P_i, $\mathcal{S}$ broadcasts its local view (x_i, π_i). For all x_0, we show the resulting transcript to be indistinguishable form one obtained through Share with secret x_0 using a sequence of hybrid distributions.

- D_1: The distribution of corrupted parties' transcripts when $\mathcal{A}$ interacts with $\mathcal{S}$ as above.
- D_2: As D_1 but $\mathcal{S}$ computes $x_i = f(\alpha_i)$ for $f \xleftarrow{\$} R[X]_t$ such that $f(\alpha_0) = x_0$.
- D_3: The distribution of corrupted parties' transcripts when $\mathcal{A}$ interacts with a dealer using Share on input x_0.

D_1 and D_2 follow the same distribution because (by Lemma 1) $|T| \leq t$ implies that $(f(\alpha_i))_{i\in T}$ is a uniformly random vector in $R^{|T|}$ when f is a random polynomial such that $f(\alpha_0) = x_0$. On the other hand, the difference between D_2 and D_3 is that in the former the simulator $\mathcal{S}_\Pi$ for Π is used while the latter uses the actual proof Π, but we know the transcripts are indistinguishable by the zero-knowledge property of Π.

Strong Commitment. If the malicious dealer gets disqualified for broadcasting an incorrect proof in the third round, all honest players set their own share to 0. Moreover, in the reconstruction phase they always return 0. Thus in this case strong commitment holds perfectly.

[13] In a non-interactive distributed proof, an adversary's behavior is fully determined by the set of parties it corrupts.

Conversely, if the dealer is not disqualified, each honest user ends Share with share x_i and accepting proof π_i. As we assumed $n \geq 2t+1$, then $H = [n] \backslash T$ has size $|H| \geq t+1$. From ε-soundness we have that, up to probability ε, the shares x_i are the evaluation of a (unique) polynomial $f(X)$ of degree $\leq t$, which identifies a unique secret $x_0 = f(\alpha_0)$.

Next, during the reconstruction phase, let Bad_j for $j \in T$ be the event in which P_j broadcast (x_j, π_j) where π_j is accepting but $f(\alpha_j) \neq x_j$. We then argue $\Pr[\mathsf{Bad}_j] \leq \varepsilon$. Indeed, consider $\mathcal{A}_i$ an adversary for the adaptive-input soundness of Π which corrupts users in $T \backslash \{j\}$. Initially, in runs the t-privacy adversary which return $\pi, (x_i, \pi_i)_{i \in H}$. Next, it correctly simulates an execution of $\mathsf{Reconstruct}$ with the same adversary from which it retrieves (x_j, π_j). If all partial proofs verifier correctly, it returns $(x_i)_{i \in H \cup \{j\}}$ and proof $\pi, (\pi_i)_{i \in H \cup \{j\}}$. If all proofs are accepting but Bad_j occurs, then $\mathcal{A}$ violates adaptive soundness. Thus $\Pr[\mathsf{Bad}_j] \leq \varepsilon$.

Finally, setting $\mathsf{Bad} = \vee_{j \in T} \mathsf{Bad}_j$ we have that $\Pr[\mathsf{Bad}] \leq t\varepsilon$ by a union bound. Assuming $\neg\mathsf{Bad}$ instead, we have that all x_j with an accepting proof are such that $f(\alpha_j) = x_j$. Thus, regardless of the chosen interpolation set, all honest parties recover f during $\mathsf{Reconstruct}$ and in particular they all return $x_0 = f(\alpha_0)$.

4.2 Optimizations

We now detail two optimizations to the VSS scheme presented in the previous section.

Path-Pruning. In the third round of the sharing protocol, assume parties P_i for $i \in T$ with $|T| = \vartheta$ issued a complain. Naïvely broadcasting each complaining user's proof would require $O_\eta(\vartheta \log(n)^2)$ communication, with η being the hash output length. This however includes several repeated hash values, as we authenticate every path for every index in T individually. Removing redundant values allows opening a MT of n leaves in $O(\vartheta \log(n/\vartheta))$ hashes ([30], see the full version for a proof). This implies that over arbitrary rings, the complain phase involves the communication of $O(\vartheta \log(n/\vartheta) \log(n))$ hashes (which is in particular $O(n \log n)$). Conversely, over rings where the protocol in Fig. 2 can be used, the k-th MT only involves $n \cdot 2^{-k}$ leaves. This implies that with path-pruning communication is bounded by $O(\vartheta \log(n/\vartheta)^2)$ which is in particular $O(n)$.

Two-Rounds Reconstruction. Since each proof has size $O(\log(n)^2)$, in order to avoid a broadcast complexity of $O(n \log(n)^2)$ we could split the reconstruction phase in two. Initially all parties broadcast their share. Next, they check whether there exists a polynomial of degree t which interpolates those values. This can be checked probabilistically in linear time [16]. Finally, all parties broadcast their local proofs as in the previous protocol.

In spite of the possible fall back to an $O(n \log(n)^2)$ sized broadcast, this only occurs when at least one *actively* malicious user is identified. Such case could then be mitigated through incentives/stake, although analyzing such option in detail is outside of our scope.

4.3 Efficiency

In light of the optimizations previously discussed, we now detail the overall sharing costs of our VSS. We call η the hash output length, $\rho = \log_2 |R|$. In the optimistic case in which no complaint is raised we have:

- *Communication.* The dealer needs to broadcast $O_\eta(\log(n)) + O_\rho(1)$ bits and privately send $O_\eta(\log(n)^2) + O_\rho(\log(n))$ bits to each user.
- *Dealer computation.* This is dominated by the proof cost, which amounts to $O(n \log(n)^2)$ and $O(n \log n)$ ring operations, when using respectively the protocol in Fig. 1 or Fig. 2.
- *Parties computation.* Each party needs to do $O(\log(n))$ ring operations and $O(\log(n)^2)$ hash evaluations to check the openings of the commitments they received.

Next, we study the overhead induced by ϑ complains in the sharing phase. We remark verification time remains sub-linear as long as $\vartheta = o(n/\log(n)^2)$.

- *Communication.* Using *path-pruning*, Protocol 1 requires $O_\mu(\vartheta \log(n/\vartheta) \log n) + O_\rho(\vartheta)$ extra broadcast communication, there the first term is $O_\mu(n \log n)$ in the worst case. Conversely, Protocol 2 requires $O_\mu(\vartheta \log(n/\vartheta)^2) + O_\rho(\vartheta)$ additional bits, where the first term is $O_\mu(n)$ in the worst case.
- *Dealer computation.* The dealer costs remains the same asymptotically.
- *Parties computation.* Each party needs to do extra $O(\vartheta \log(n/\vartheta) \log(n))$ hash evaluations (respectively $O(\vartheta \log(n/\vartheta)^2)$ using Protocol 2) and $O(\vartheta)$ ring operations.

Acknowledgements. The authors would like to thank Dario Fiore and Thomas Attema for providing helpful comments.

This work is supported by Juan de la Cierva grant JDC2022-049711-I, the PRODIGY Project (TED2021-132464B-I00) and the ESPADA project (PID2022-142290OB-I00), all funded by MCIN/AEI/10.13039/501100011033. The Juan de la Cierva grant and the PRODIGY project are cofunded by the European Union «NextGenerationEU/PRTR»; ESPADA is cofunded by FEDER, UE. This work has also been supported by the PICOCRYPT project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101001283); and by the CONFIDENTIAL6G funded by the European Union (GA 101096435).

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

References

1. Abraham, I., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G.: Bingo: Adaptivity and asynchrony in verifiable secret sharing and distributed key generation. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 39–70. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38557-5_2

2. Applebaum, B., Kachlon, E., Patra, A.: The round complexity of perfect MPC with active security and optimal resiliency. In: 61st FOCS. pp. 1277–1284. IEEE Computer Society Press (Nov 2020).https://doi.org/10.1109/FOCS46700.2020.00121
3. Atapoor, S., Baghery, K., Cozzo, D., Pedersen, R.: VSS from distributed ZK proofs and applications. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part I. LNCS, vol. 14438, pp. 405–440. Springer, Heidelberg (Dec 2023).https://doi.org/10.1007/978-981-99-8721-4_13
4. Attema, T., Cramer, R.: Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 513–543. Springer, Heidelberg (Aug 2020).https://doi.org/10.1007/978-3-030-56877-1_18
5. Attema, T., Fehr, S., Klooß, M.: Fiat-shamir transformation of multi-round interactive proofs. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part I. LNCS, vol. 13747, pp. 113–142. Springer, Heidelberg (Nov 2022).https://doi.org/10.1007/978-3-031-22318-1_5
6. Backes, M., Kate, A., Patra, A.: Computational verifiable secret sharing revisited. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 590–609. Springer, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_32
7. Baghery, K.: Π: A unified framework for verifiable secret sharing. IACR Cryptol. ePrint Arch. p. 1669 (2023), https://eprint.iacr.org/2023/1669
8. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 1–10. ACM (1988).https://doi.org/10.1145/62212.62213, https://doi.org/10.1145/62212.62213
9. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) ICALP 2018. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl (Jul 2018).https://doi.org/10.4230/LIPIcs.ICALP.2018.14
10. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 103–128. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_4
11. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_2
12. Bois, A., Cascudo, I., Fiore, D., Kim, D.: Flexible and efficient verifiable computation on encrypted data. In: Garay, J. (ed.) PKC 2021, Part II. LNCS, vol. 12711, pp. 528–558. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75248-4_19
13. Boneh, D., Boyle, E., Corrigan-Gibbs, H., Gilboa, N., Ishai, Y.: Zero-knowledge proofs on secret-shared data via fully linear PCPs. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 67–97. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8_3
14. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 677–706. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45721-1_24
15. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D.: Fiat-Shamir from simpler assumptions. Cryptology ePrint Archive, Report 2018/1004 (2018), https://eprint.iacr.org/2018/1004

16. Cascudo, I., David, B.: SCRAPE: Scalable randomness attested by public entities. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 17. LNCS, vol. 10355, pp. 537–556. Springer, Heidelberg (Jul 2017).https://doi.org/10.1007/978-3-319-61204-1_27
17. Cascudo, I., David, B.: ALBATROSS: Publicly AttestabLe BATched Randomness based On Secret Sharing. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 311–341. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64840-4_11
18. Cascudo, I., David, B.: Publicly verifiable secret sharing over class groups and applications to DKG and YOSO. To appear at Eurocrypt 24. IACR Cryptol. ePrint Arch. p. 1651 (2023), https://eprint.iacr.org/2023/1651
19. Cascudo, I., David, B., Garms, L., Konring, A.: YOLO YOSO: Fast and simple encryption and secret sharing in the YOSO model. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part I. LNCS, vol. 13791, pp. 651–680. Springer, Heidelberg (Dec 2022).https://doi.org/10.1007/978-3-031-22963-3_22
20. Cascudo, I., Giunta, E.: On interactive oracle proofs for boolean R1CS statements. In: Eyal, I., Garay, J.A. (eds.) FC 2022. LNCS, vol. 13411, pp. 230–247. Springer, Heidelberg (May 2022).https://doi.org/10.1007/978-3-031-18283-9_11
21. Chandramouli, A., Choudhury, A., Patra, A.: A survey on perfectly secure verifiable secret-sharing. ACM Comput. Surv. **54**(11s), 232:1–232:36 (2022).https://doi.org/10.1145/3512344, https://doi.org/10.1145/3512344
22. Choudhury, A., Patra, A.: On the communication efficiency of statistically secure asynchronous MPC with optimal resilience. J. Cryptol. **36**(2), 13 (2023). https://doi.org/10.1007/S00145-023-09451-9, https://doi.org/10.1007/s00145-023-09451-9
23. Cramer, R., Damgård, I., Nielsen, J.B.: Secure Multiparty Computation and Secret Sharing. Cambridge University Press (2015), http://www.cambridge.org/de/academic/subjects/computer-science/cryptography-cryptology-and-coding/secure-multiparty-computation-and-secret-sharing?format=HB&isbn=9781107043053
24. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987. pp. 427–437. IEEE Computer Society (1987).https://doi.org/10.1109/SFCS.1987.4, https://doi.org/10.1109/SFCS.1987.4
25. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. Journal of Cryptology **20**(1), 51–83 (Jan 2007). https://doi.org/10.1007/s00145-006-0347-3
26. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In: Coan, B.A., Afek, Y. (eds.) Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, PODC '98, Puerto Vallarta, Mexico, June 28 - July 2, 1998. pp. 101–111. ACM (1998).https://doi.org/10.1145/277697.277716, https://doi.org/10.1145/277697.277716
27. Gentry, C., Halevi, S., Lyubashevsky, V.: Practical non-interactive publicly verifiable secret sharing with thousands of parties. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 458–487. Springer, Heidelberg (May / Jun 2022).https://doi.org/10.1007/978-3-031-06944-4_16

28. Giunta, E.: On the impossibility of algebraic NIZK in pairing-free groups. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 702–730. Springer, Heidelberg (Aug 2023)https://doi.org/10.1007/978-3-031-38551-3_22
29. Kate, A., Mangipudi, E.V., Mukherjee, P., Saleem, H., Thyagarajan, S.A.K.: Non-interactive VSS using class groups and application to DKG. IACR Cryptol. ePrint Arch. p. 451 (2023), https://eprint.iacr.org/2023/451
30. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_3
31. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. Journal of the ACM (JACM) **27**(2), 228–234 (1980)
32. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (Aug 1992).https://doi.org/10.1007/3-540-46766-1_9
33. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: 21st ACM STOC. pp. 73–85. ACM Press (May 1989).https://doi.org/10.1145/73007.73014
34. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 148–164. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_10
35. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery **22**(11), 612–613 (Nov 1979). https://doi.org/10.1145/359168.359176
36. Shoup, V., Smart, N.P.: Lightweight asynchronous verifiable secret sharing with optimal resilience. IACR Cryptol. ePrint Arch. p. 536 (2023), https://eprint.iacr.org/2023/536

Timed Secret Sharing

Alireza Kavousi[1](✉), Aydin Abadi[2], and Philipp Jovanovic[3]

[1] University College London, London, UK
a.kavousi@cs.ucl.ac.uk
[2] Newcastle University, Newcastle, UK
aydin.abadi@newcastle.ac.uk
[3] University College London, London, UK
p.jovanovic@ucl.ac.uk

Abstract. This paper introduces the notion of *timed secret sharing* (TSS), which establishes *lower* and *upper* time bounds for secret reconstruction in a threshold secret sharing scheme. Such time bounds are particularly useful in scenarios where an early or late reconstruction of a secret matters. We propose several new constructions that offer different security properties and show how they can be instantiated efficiently using novel techniques. We highlight how our ideas can be used to break the *public goods game*, which is an issue inherent to threshold secret sharing-based systems, without relying on incentive mechanism. We achieve this through an upper time bound that can be implemented either via *short-lived proofs*, or the *gradual release of additional shares*, establishing a trade-off between time and fault tolerance. The latter independently provides *robustness* in the event of dropout by some portion of shareholders.

1 Introduction

Threshold secret sharing [54] is a widely used primitive in cryptography and distributed computing. A (t, n)-threshold secret sharing scheme lets a dealer distribute a secret s among n shareholders such that any subset of at least $t + 1$ shares can recover s, whereas no subset of at most t shares reveal any information about s. This primitive is useful in a wide range of applications from password-protection [8,37] and federated learning [42], to verifiable management of on-chain secrets [39] and many more. Protocols using secret sharing usually specify conditions under which shareholders release their shares to reconstruct the secret [20,30]. In many cases, these conditions depend on the notion of time in one way or another. In practice, however, shareholders may violate these time-dependent conditions intentionally or unintentionally by releasing their shares too early or late. These issues may arise due to the use of unsynchronized clocks by the shareholders [7,12,34] or due to a (temporary) dishonest majority [22,23]. The latter could occur particularly when incentives are misaligned so that shareholders collude and reconstruct secrets earlier than what specified [36,44].

The practical applications of threshold secret sharing motivate this work, where elaborate on two concrete scenarios as follows.

Maximal Extractable Value. In cryptocurrency platforms, consensus nodes such as proof-of-stake validators may engage in *maximal extractable value* (MEV) processes [27] to gain some benefit from users by learning their transactions and affect their ordering in the block. A principal MEV countermeasure deploys threshold secret sharing to protect the privacy of transactions up to a time where their inclusion/ordering in a block is ensured. This is done by encrypting the transaction using a random key and then sharing the key towards validators with a threshold secret sharing scheme [44,61].

However, it largely overlooks the fact that consensus nodes have significant incentives to prematurely reconstruct the secrets to capitalize on MEV rewards. Observe that this type of collusion (*i.e.,* dishonest majority) does not violate the protocol's liveness (*i.e.,* reconstruction) as the success of MEV depends on the completion of the secret reconstruction, and thus colluding parties are incentivized to make progress. In many cases, such behavior is particularly problematic since corrupt shareholders can carry out the process without leaving any public traces and thus collusion is *unobservable* [51].[1]

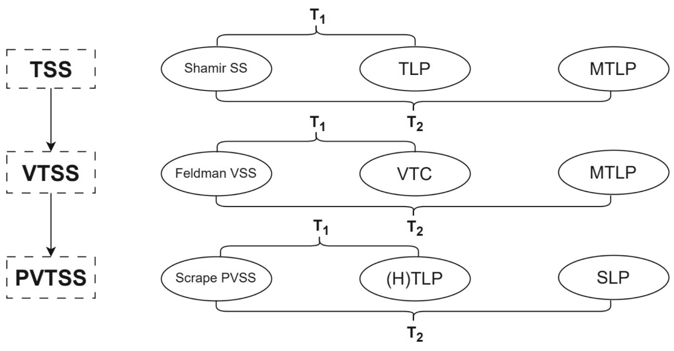
Public Goods Game. An independent issue with threshold secret sharing-based schemes is that they could constitute a *public goods game* [4,11]. This is essentially because only a subset of the shareholders needs to release their shares to reconstruct the secret. Consequently, the shareholders may choose to remain inactive, hoping that others will step forward and contribute. As a mitigation mechanism an incentive system is usually assumed [6,39] which may, however, not be available or feasible to implement under all circumstances.

Our schemes with lower and upper time bounds T_1 and T_2, respectively, address the aforementioned issues: T_1 prevents shareholders from reconstructing the secret early, and T_2 prevents public goods game dilemma without having to rely on financial incentives, providing an alternative solution. We stress that the motivations for lower and upper time bounds are different and independent. In the case of the former, we must *ensure* that the reconstruction does not occur before T_1. In the case of the latter, the goal is to *encourage* (rational) shareholders to appear early and initiate the reconstruction. For the sake of better consistency, we present the schemes with both time bounds rather than treating them separately.

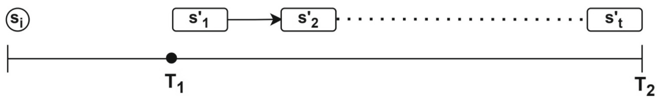
1.1 Technical Overview

Our constructions enjoy novel techniques and build upon time-based primitives with efficient instantiation in a modular way. In particular, we use *time-lock puzzles* (TLPs) [1,43,50], *verifiable timed commitments* (VTCs) [57], and *verifiable delay functions* (VDFs) [48,60]. In the remainder of this section, we give an overview of our proposed constructions.

[1] Using time-lock puzzles (TLPs) [50] are not sufficient to address the issue as protected transactions may actually not make it into the block and then lose confidentiality after the TLP has been opened, demanding *pending transaction privacy* [24].



(a) A visual representation of our constructions. It depicts the underlying tools and techniques to establish the lower and upper time bounds for different variants of secret sharing protocols.



(b) A visual representation of secret sharing with gradual release of additional shares that could break public goods game and provide robustness.

Fig. 1. Timed Secret Sharing variants

Timed Secret Sharing (TSS). This is our basic construction, where the dealer encapsulates the shares into TLPs [43,50] to realize a lower time bound T_1. Consequently, no computationally bounded adversary can learn the secret before T_1, even if it corrupts *all the shareholders* [23]. Moreover, TLPs provide a consistent relative measure of time (*i.e.,* computational timing), eliminating the need for a shared global clock. For the upper time bound, we rely on the underlying timing assumption of the secret sharing scheme and later show how to relax it.

Verifiable Timed Secret Sharing (VTSS). We enhance TSS with verification mechanisms, to deal with *malicious* dealers and shareholders. First, we ensure that a malicious dealer cannot distribute malformed puzzles, *i.e.,* puzzles that either are not extractable or contain invalid shares. Second, we ensure that malicious shareholders cannot send invalid shares during the reconstruction phase. Here we need to tackle technical challenges in realizing lower and upper time bounds. We overcome the former with a novel trick in using verifiable secret sharing (VSS) [31] and verifiable timed commitment (VTC) [57] that allows checking the validity of embedded share before the shareholder invests computational effort to retrieve it. For the latter, we introduce the novel idea of secret sharing with *gradual release of additional shares* that relaxes the assumption made in the previous scheme and also that could be of independent interest.

Publicly Verifiable Timed Secret Sharing (PVTSS). We further extend our schemes to support *public verifiability*. To do so, we take a different route in realizing the lower and upper time bounds. First, we deploy an efficient non-interactive zero-knowledge (NIZK) protocol, and the cut-and-choose technique [40] to let *anyone* (not just shareholders) ensure the validity of the embedded encrypted shares and the extractability of puzzles. Second, we bind the attestation of the distributed shares to time and impose an upper bound T_2 by utilizing short-lived proofs (SLPs) [5] that come with time-sensitive soundness and public verifiability. We crucially rely on the observation that the secret (and shares) are uniformly distributed, allowing us to securely use SLPs that require indistinguishability property. This essentially puts an upper time bound by making the system usable up to some time T_2, *i.e.,* the correct reconstruction is only guaranteed before T_2.

It is worth mentioning that our idea of secret sharing with additional shares could be useful in scenarios where a sufficient number of (honest) shareholders is not available for reconstruction and thus the additional shares allow the remaining parties to nevertheless reconstruct the secret, providing *robustness* to the system. As an application, this could help with *dropout resilience* in secure aggregation protocols for federated learning [42]. We provide a visual representation of our protocol variants at Fig. 1.

1.2 Our Contributions

– We formally define and construct (t, n)*-timed secret sharing* (TSS) which enables a timely reconstruction of a secret shared by a dealer to a set of n shareholders within the time interval $[T_1, T_2]$.
– We enhance TSS with verifiability by formally defining and constructing *verifiable timed secret sharing* (VTSS), which protects against a malicious dealer during share distribution and against malicious shareholders during secret reconstruction.
– We further extend VTSS with public verifiability by formally defining and constructing *publicly verifiable timed secret sharing* (PVTSS).
– We introduce two novel ideas to break the public goods game in threshold secret sharing systems. One is based on using *short-lived proofs* and the other is based on *gradual release of additional shares*. As a side contribution, we formally define and propose a construction for the latter which is also useful to provide *robustness* against shareholder's dropouts.

2 Related Work

There is a large body of literature on the combination of computational timing and cryptographic primitives such as commitment [3,14,29,45,58], encryption [17,25,41], signature [9,28,33,57], and more. The essence of almost all of these works is to enable the receiver(s) to forcefully open the locked object after a predefined period by working through some computational operation.

The work of [57] proposed efficient constructions for encapsulating a signature into a TLP, ensuring the receiver can extract the valid signature after carrying out sequential computation. Roughly speaking, the sender secret shares the signature and embeds each share in a linearly homomorphic TLP [43]. Then, the sender and receiver run a cut-and-choose protocol for verifying the correctness of the puzzles. Moreover, to enable the receiver to compact all the pieces of time-locked signatures and solve one single puzzle, a range proof is used to guarantee that no overflow occurs. Manevich and Akavia in [45] augment the timed commitment of Boneh and Naor [14] with zero-knowledge proofs, enabling the sender to prove *any* arbitrary attribute regarding the committed value.

With a focus on reducing the interaction in MPC protocols with limited-time secrecy, the authors in [3] developed a gage time capsule (GaTC), allowing a sender to commit to a value that others can obtain after putting a total computational cost which is parallelizable to let solvers claim a monetary reward in exchange for their work. The security guarantee of GaTC resemble ours when using secret sharing with additional shares in the sense that over time it gradually decays, as the adversary can invest more and more computational resources. Doweck and Eyal [29] constructed a multi-party timed commitment that enables a group of parties to jointly commit to a secret to be opened by an aggregator later on via brute-force computation.

The authors in [10] explore multi-party computation with output-independent abort, having each participant in an MPC protocol lock their output until some time in the future. This is to force the adversary to decide whether to cause an abort before learning the output. As performing sequential computations might be beyond the capacity of some users, Thyagarajan et al. [59] developed a system to allow users to outsource their tasks to some servers in a privacy-preserving manner. Srinivasan et al. [56] constructed a TLP that supports unbounded batch-solving while enjoying a transparent setup and a puzzle size independent of the batch size. Although their construction is of theoretical interest and does not have practical efficiency due to the reliance on indistinguishability obfuscation, it enables a party to solve many puzzle instances simultaneously at the cost of solving one puzzle. It is worth noting that such a setting is not applicable to our PVTSS as each shareholder just needs to know their own share and solving other parties' puzzles gives her no information as they are already encrypted under the parties' public keys. One of the motivating reasons for batch-solving is to enable a party to solve the puzzles of others in case a large number of parties abort. We refer the reader to [47] for a more detailed overview of relevant works.

3 Preliminaries

3.1 Threat Model and Assumptions

We consider a standard synchronous network where each pair of parties in a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ is connected via an authenticated channel, and each message is delivered at most by a known delay. There is also a dealer D that takes the role of distributing the secret among participating parties.

As common in the literature for verifiable secret sharing, we assume the existence of broadcast channels. For a publicly verifiable scheme, we assume the existence of an authenticated public bulletin board. In this work, we consider a static adversary that may corrupt up to t out of n parties before the start of protocol execution. D may also be corrupted. We consider both semi-honest and malicious types of adversaries. In the former, the corrupted parties are assumed to follow the protocol but may try to learn some information by observing the protocol execution. In the latter, however, the corrupted parties are allowed to do any adversarial action of their choice. The adversary's computational power is bounded with respect to a security parameter λ that gives it a negligible advantage in breaking the security of underlying primitives. Such algorithms are often known as probabilistic polynomial time (PPT). Finally, we denote by $[n]$ the set $\{1, \ldots, n\}$ an by $\mathbf{v}$ a vector of elements $\{v_i\}_{i\in[n]}$.

3.2 Secret Sharing

A (threshold) secret sharing scheme is a cryptographic protocol that enables a dealer D to distribute a secret s among n parties. The scheme typically consists of two main phases; *distribution* and *reconstruction*. In the former, D sends each party their corresponding share, and in the latter, any proper subset of parties reconstruct the secret by pooling their shares.

A (t, n)-threshold secret sharing offers two main properties: (1) correctness: the secret is reconstructed by any subset of at least $t+1$ shares, and (2) t-security: no information is revealed about the secret by gathering t or fewer shares. In this work, we develop our protocols based on the popular Shamir secret sharing [54]. We note that our proposed definitions can capture any (linear) secret sharing.

Verifiable Secret Sharing (VSS). The basic (t, n)-threshold secret sharing scheme (e.g., [54]) only provides security against a *semi-honest* adversary. When dealing with malicious adversaries, it is essential for (1) the dealer to prove the validity of the shares it produces in the distribution phase, and (2) the shareholders to prove the validity of the shares they provide in the reconstruction phase. To satisfy these properties, various VSS schemes have been proposed, following the celebrated work by Feldman [31].

Publicly Verifiable Secret Sharing (PVSS). To extend the scope of verifiability to the public and not only participating parties, PVSS schemes [18,19,53] deploy cryptographic primitives such as encryption and NIZK proofs. PVSS enables anyone to verify the distribution and reconstruction phases. Cascudo and David [18] proposed an efficient scheme called Scrape PVSS, which is an improvement over [53] and has been deployed extensively in many recent cryptographic protocols. The Scrape protocol works as follows. The dealer D chooses a random value $s \overset{\$}{\leftarrow} \mathbb{Z}_q$, sets the secret as a group element of form $S = h^s$, splits s into shares $\{s_i\}_{i\in[n]}$, and computes the encrypted shares $\{\hat{s}_i\}_{i\in[n]}$ using corresponding parties' public keys $\{pk_i\}_{i\in[n]}$.

Then, D publishes a set of commitments to shares $\{v_i\}_{i\in[n]}$ together with a proof π_D, enabling anyone to check the consistency of the shares (*i.e.,* shares are evaluations of the same polynomial of proper degree) and validity of the ciphertexts (*i.e.,* encrypted shares correspond to the committed shares). Upon receiving a threshold number of valid shares (*i.e.,* shares with correct decryptions), anyone can use Lagrange interpolation [2] in the exponent to reconstruct the secret S. The authors proposed two versions, one in the random oracle model under the Decisional Diffie-Hellman (DDH) assumption and the other in the plain model under the Decisional Bilinear Squaring (DBS) assumption. We use the non-pairing variant which offers *knowledge soundness*. This is vital to ensure the secret chosen by the adversary is independent of those of honest parties. Also, we require the knowledge soundness property for deploying short-live proofs [5].

3.3 Time-Lock Puzzles (TLPs)

The idea of TLPs was introduced by Rivest et al. [50]. TLP locks a secret such that it can only be retrieved after a predefined amount of sequential computation. It consists of two algorithms: TLP.Gen, which takes as input a time parameter T and a secret s, and returns a puzzle Z, and TLP.Solve, that takes as input a puzzle Z and returns a secret s. A TLP must satisfy *correctness* and *security*. The correctness ensures that the solution is indeed obtained if the protocol gets executed as specified. The security ensures that no PPT adversary running in parallel obtains the solution within the time bound T, except with negligible probability. We provide the formal definitions in Appendix A.

Homomorphic Time-Lock Puzzles (HTLP). Malavolta and Thyagarajan [43] proposed homomorphic TLP, enabling one to homomorphically combine many instances of TLPs into a single TLP. An HTLP consists of a tuple of algorithms (HTLP.Setup, HTLP.Gen, HTLP.Solve, HTLP.Eval). In particular, HTLP.Setup generates public parameters pp on input a security parameter, and HTLP.Eval performs a homomorphic operation on input a set of puzzles to output a single puzzle.

Multi-instance Time-Lock Puzzle (MTLP). Abadi and Kiayias [1] proposed a primitive called multi-instance TLP. This variant of TLP is suitable for the case where the solver is given multiple puzzles at the same time but must discover each solution at different points in time. It allows solving the instances sequentially one after the other without needing to run parallel computations on them. An MTLP consists of a tuple of algorithms (MTLP.Setup, MTLP.Gen, MTLP.Solve, Prove, Verify), where the last two algorithms are used to check the correctness of a solver's claimed solution.

3.4 Timed Commitment

An inherent limitation of the well-known time-lock puzzles such as [43,50] is the lack of verifiability, meaning that the receiver cannot check the validity of the received puzzle unless after putting time and effort into solving it. To fill this gap, a timed commitment scheme [14] enables the receiver to make sure about the well-formedness (*i.e.,* extractability) of the puzzle before performing a sequential computation. In an attempt to make the timed commitment of [14] efficiently verifiable, the recent work of Thyagarajan et al. [57] proposed verifiable timed commitment (VTC), enabling the sender to verifiably[2] commit to signing keys of form $pk = g^{sk}$, $sk \in \{0,1\}^{\lambda}$. The VTC primitive consists of a tuple of algorithms ($\mathsf{VTC.Setup}$, $\mathsf{VTC.Commit}$, $\mathsf{VTC.Verify}$, $\mathsf{VTC.Solve}$). Note that we deploy VTC to design construction for our verifiable time secret sharing (VTSS) scheme.

3.5 Sigma Protocols

A zero-knowledge protocol enables proving the validity of a claimed statement by the prover P to the verifier V without revealing any information further. While zero-knowledge protocols involve various settings and notions, we particularly consider the well-known Sigma protocols which are useful building blocks in many cryptographic constructions. Let v denote an instance that is known to both parties and w denote a witness that is only known to the P. Let $R = \{(v; w)\} \in \mathcal{V} \times \mathcal{W}$ denote a relation containing the pairs of instances and corresponding witnesses. A Sigma protocol Σ on $(v; w) \in R$ is an interactive protocol with three movements between P and V. Using Fiat-Shamir heuristic [32] in the random oracle model, one can make the protocol non-interactive with public verifiability. A Sigma protocol satisfies two security properties: (1) *soundness*, ensuring the verifier about the validity of the statement v, and (2) *zero-knowledge*, ensuring the prover about the secrecy of the witness w.

Zero Knowledge Proof of Equality of Discrete Logarithm. One of the well-used Sigma protocols is discrete logarithm equality (DLEQ) proof. It considers a tuple of publicly known values (g_1, x, g_2, y), where g_1, g_2 are random generators and x, y are two elements of the cyclic group $\mathbb{G}$ of order q. DLEQ proof enables a prover P to prove to the verifier V that it knows a witness α such that $x = g_1^{\alpha}$ and $y = g_2^{\alpha}$. A DLEQ proof is an AND-composition of two Sigma protocols for relation $R = \{(v_i; w) : v_i = g_i^{w}\}$ with the *same* witness and challenge. The following protocol is a Sigma protocol for generating a DLEQ proof due to Chaum-Pedersen [21].

1. P chooses a random element $u \xleftarrow{\$} \mathbb{Z}_q$, computes $a_1 = g_1^{u}$ and $a_2 = g_2^{u}$, and sends them to the V.
2. V sends back a randomly chosen challenge $c \xleftarrow{\$} \mathbb{Z}_q$.
3. P computes $r = u + c\alpha$ and sends it to V.
4. V checks if both $g_1^{r} = a_1 x^{c}$ and $g_2^{r} = a_2 y^{c}$ hold.

Throughout the paper we use the non-interactive version of this protocol which produces a single message $\mathsf{DLEQ.P}(\alpha, g_1, x, g_2, y)$ as proof π verified via $\mathsf{DLEQ.V}(\pi, g_1, x, g_2, y)$. The challenge is computed by the prover as

[2] Ensuring the extractability together with validity of the committed message that is the discrete logarithm of a public key.

$c = H(x, y, a_1, a_2)$, where H is a cryptographic hash function modeled as a random oracle.

3.6 Short-Lived Proofs

Arun et al. [5] recently introduced the notion of *short-lived proofs* (SLPs) which can be roughly defined as types of proofs with expiration, such that their soundness will disappear after certain time. They are only sound if being observed before a determined time, afterwards, they may be forgery indistinguishable from the valid proofs. At a high level, an SLP is proof of an OR-composition $R \vee R_{VDF}$, where R is an arbitrary relation and R_{VDF} is a VDF evaluation relation. Interestingly, this proof is only convincing to the verifier for a determined time T as forging the proof is possible for anybody after evaluating the VDF. Due to the nature of VDF, short-lived proofs offer efficient public variability. One notable point is that the primitive makes use of a *randomness beacon* [26] which outputs unpredictable values b periodically.

An SLP scheme consists of four algorithms ($\mathsf{SLP.Setup}$, $\mathsf{SLP.Gen}$, $\mathsf{SLP.Forge}$, $\mathsf{SLP.Verify}$) with the following descriptions. $\mathsf{SLP.Setup}$ generates public parameters pp on input the security parameter and time parameter T. $\mathsf{SLP.Eval}$ takes pp, an input x, a random beacon value b, and generates a proof π. $\mathsf{SLP.Forge}$ takes pp, x, b, and produces a proof π. Lastly, $\mathsf{SLP.Verify}$ validates the proof π on input pp, x, π, and b. A short-lived proof must satisfy four security properties including *forgeability*, enabling anyone running in time $(1 + \epsilon)T$ to generate a valid proof, *soundness*, preventing a malicious prover P^* running with parallel processors to generate a convincing proof in time less than T, *zero knowledge*, preserving the privacy of the witness w, and *indistinguishability*, making the real and forged proofs indistinguishable from the actual proof.

4 Timed Secret Sharing (TSS)

With timed secret sharing (TSS), we make a secret sharing scheme dependent on time, having the reconstruction phase occur within a determined time interval, $[T_1, T_2]$, where T_1 is the lower time bound and T_2 is the upper time bound. These time bounds might be required by the dealer or as part of the system requirements, or even a combination of these two. An important consideration, however, is that the dealer's *availability* should not be affected by making the scheme time-based, meaning that the dealer's role should finish after the distribution phase similar to the original setting.

4.1 TSS Definition

In this section, we present a formal definition of TSS. This definition builds upon the original definition of threshold secret sharing.

Definition 1 (Timed Secret Sharing). *A timed secret sharing (TSS) scheme involves the following algorithms.*

1. **Initialization:**
   - Setup: $\mathsf{TSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp$, *on input security parameter* λ, *lower time bound* T_1, *and upper time bound* T_2, *outputs public parameters* pp.
2. **Distribution:**
   - Sharing: $\mathsf{TSS.Sharing}(pp, s) \rightarrow \{C_i\}_{i\in[n]}$, *on input* pp *and secret* $s \in S_\lambda$, *outputs a locked share* C_i *with time parameter* T_1 *for each party* P_i *in the set* $\mathcal{P}$.
3. **Reconstruction:**
   - Recovering: $\mathsf{TSS.Recover}(pp, C_i) \rightarrow s_i$, *on input* pp *and* C_i, *recovers the share* s_i. *The algorithm is run by each party* P_i *in* $\mathcal{P}$.
   - Pooing: $\mathsf{TSS.Pool}(pp, \mathcal{S}, T_2) \rightarrow s$, *on input* pp *and a set* $\mathcal{S}$ *of shares (where* $|\mathcal{S}| > t$ *and* $t \in pp$*), outputs the secret* s *if* T_2 *has not elapsed. Otherwise, it outputs* $\perp$.

A correct TSS scheme must satisfy *privacy*, ensuring no share is obtained before T_1 and *security*, ensuring any set of shares less than a threshold $t+1$ reveals no information about the secret before T_2.

Definition 1.1 (Correctness). *A TSS satisfies correctness if for all secret* $s \in S_\lambda$ *and a set of shares* $|\mathcal{S}| > t$ *it holds*

$$\Pr\left[\mathsf{TSS.Pool}(pp, \mathcal{S}, T_2) \rightarrow s \ : \begin{array}{c} \mathsf{TSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \mathsf{TSS.Sharing}(pp, s) \rightarrow \{C_i\}_{i\in[n]}, \\ \mathsf{TSS.Recover}(pp, C_i) \rightarrow s_i \end{array}\right] = 1$$

Definition 1.2 (Privacy). *TSS satisfies privacy if for all parallel algorithms* $\mathcal{A}$ *whose running time is at most less than* T_1 *there exists a simulator* Sim *and a negligible function* μ *such that for all secret* $s \in S_\lambda$, *all* $\lambda \in \mathbb{N}$, *and all* $i \in [n]$ *it holds*

$$\left|\Pr\left[\mathcal{A}(pp, s, C_i) = 1 \ : \begin{array}{l} \mathsf{TSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \mathcal{A}(pp, 1^\lambda) \rightarrow s, \\ \mathsf{TSS.Sharing}(pp, s) \rightarrow \{C_i\}_{i\in[n]} \end{array}\right] - \right.$$

$$\left.\Pr\left[\mathcal{A}(pp, s', C_j) = 1 \ : \begin{array}{l} \mathsf{TSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \mathcal{A}(pp, 1^\lambda) \rightarrow s', \\ \mathsf{Sim}(pp) \rightarrow \{C_j\}_{j\in[n]} \end{array}\right]\right| \le \mu(\lambda)$$

Definition 1.3 (Security). *TSS satisfies security if an adversary* $\mathcal{A}$ *controlling a set* $\mathcal{S}'$ *of parties, where* $|\mathcal{S}'| \le t$ *and* $s \in S_\lambda$, *learns no information about* s. *Thus, it must hold*

$$\Pr\left[\mathcal{A}(pp, \mathcal{S}', T_2) \rightarrow s \ : \begin{array}{c} \mathsf{TSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \mathsf{TSS.Sharing}(pp, s) \rightarrow \{C_i\}_{i\in[n]}, \\ \mathsf{TSS.Recover}(pp, C_i) \rightarrow s_i \end{array}\right] \le \mu(\lambda) + \frac{1}{|S_\lambda|}$$

Π_{TSS}

1. **Initialization:**
   – Setup: $\mathsf{TSS.Setup}(1^\lambda, T_1, T_2) \to pp$, the protocol works over $\mathbb{Z}_q$, where $q > n$. The dealer D runs $\mathsf{TLP.Setup}(1^\lambda, T_1)$ and publishes public parameters pp.
2. **Distribution:**
   – Sharing: $\mathsf{TSS.Sharing}(pp, s) \to \{Z_i\}_{i\in[n]}$, the dealer D picks a secret $s \in Z_p$ to be shared among n parties. It samples a degree-t Shamir polynomial $f(\cdot)$ such that $f(0) = s$ and $f(i) = s_i$ for $i \in [n]$. It runs $\mathsf{TLP.Gen}(1^\lambda, T_1, s_i)$ to create puzzle Z_i with time parameter T_1, locking the share s_i for all $i \in [n]$. Finally, D privately sends each party P_i their corresponding puzzle Z_i.
3. **Reconstruction:**
   – Recovering: $\mathsf{TSS.Recover}(pp, Z_i) \to s_i$, upon receiving the puzzle Z_i, party P_i starts solving it by running $\mathsf{TLP.Solve}(T_1, Z_i)$ to recover the share s_i.
   – Pooling: $\mathsf{TSS.Pool}(pp, \mathcal{S}, T_2) \to s$, upon having sufficient number of shares $(\geq t + 1)$ received before T_2, the reconstructor (a party in $\mathcal{P}$) reconstructs the secret s using Lagrange interpolation at $f(0)$; otherwise, it returns $\perp$.

Fig. 2. Timed Secret Sharing (TSS) protocol

4.2 TSS Construction

We present an instantiation of TSS in Fig. 2. To enforce a lower time bound T_1, the dealer uses TLPs [43,50] to lock the shares into puzzles, enforcing a computational delay for each party to recover their corresponding share. Note that we treat T_2 mostly as a matter of formalization and rely on the underlying assumption of having common knowledge of time for participating parties. We later in Sect. 5 show how to relax this assumption using computational timing.

Theorem 1. *If the time-lock puzzle TLP and Shamir secret sharing are secure, then timed secret sharing protocol Π_{TSS} presented in Fig. 2 satisfies privacy and security, w.r.t. Definitions 1.2 and 1.3 respectively.*

Proof. Correctness is straightforward. The privacy property follows directly from that of the underlying TLP which implies the indistinguishability of a puzzle produced by algorithm $\mathsf{TSS.Sharing}$ and the one produced by Sim. Since all the puzzles are communicated through private channels, no party can learn the other party's share after T_1. Finally, the security stems from the underlying threshold secret sharing, where a subset of shares $\mathcal{S}'$ whose size is less than t reveals no information about the secret s.

5 Secret Sharing with Additional Shares

A threshold secret sharing scheme guarantees *t-security*. There is also $t + 1$-*robustness* assumption, ensuring the availability of a sufficient number of valid shares during the reconstruction phase. However, it is natural to challenge such a liveness assumption and consider a scenario in which a *large* fraction of honest parties goes offline, particularly when having a determined period for reconstruction, putting the system under threat of failure (*i.e.,* lack of liveness). To be concrete, a possible scenario that may lead to having less than a threshold of (honest) parties available is explored in [57] known as *denial of spending* (DoSp) attack where the set of available parties cannot reach the threshold and their investment will remain locked. In a federated learning setting [42], real-world factors such as hardware failure or poor network coverage can also cause this issue, leading to shareholders' dropouts.

Our goal is to provide *robustness* using the capabilities of time-based cryptography. We observe this is feasible by having the dealer provide parties with *additional time-locked shares.* By additional, we mean some shares other than the individual one each party already receives during the distribution phase of the protocol. Thus, even if there is less than a threshold of parties (even a single one) available at the reconstruction period (*i.e.,* $[T_1, T_2]$), they will be able to open the additional time-locked shares after carrying out some computation and retrieve the secret. We remark that a large body of literature on threshold secret sharing assumes all the parties, not only those interacting in the reconstruction phase, learn the secret [18,38]. Given this, we argue that the availability of a (threshold) number of additional time-locked shares at the proper time (*i.e.,* T_2) does not violate the security of the system since it enables all the parties to eventually learn the secret at the same time if they have not already learned it.

5.1 Decrementing-Threshold Timed Secret Sharing (DTSS)

It is possible to derive an interesting *trade-off* between time and fault tolerance by having some additional time-locked shares to be realized periodically at *different* points in time. The consequence of this *gradual* release is twofold. Firstly, it enables (honest) parties requiring some more shares (not necessarily t) to reconstruct the secret without going through the sequential computation for the whole period, *i.e.,* $[T_1, T_2]$. They can stop working up to a point where a sufficient number of additional shares is gained, as T_2 might be considerably later than T_1. Secondly, as time goes by and the reconstruction is not initiated, the adversary may get more additional shares by investing computational effort, causing security decay over time [3]. Looking ahead, this feature happens to be useful to impose an upper time bound and thus *break the public goods game* as it ties the security of the system to time; the later parties initiate the reconstruction, the more chances the adversary learns the secret.[3]

[3] It is clear that since all parties can recover the secret by T_2, this essentially puts an upper time bound for the system. We use this technique to relax the assumption made to realize an upper time bound for TSS.

5.2 DTSS Definition

Now, we present a formal definition for our scheme called decrementing-threshold timed secret sharing (DTSS).

Definition 2 (Decrementing-threshold Timed Secret Sharing). *A (t, n) DTSS scheme consists of a tuple of algorithms ($\mathsf{DTSS.Setup}$, $\mathsf{DTSS.Sharing}$, $\mathsf{DTSS.ShaRecover}$, $\mathsf{DTSS.Verify}$, $\mathsf{DTSS.AddRecover}$, $\mathsf{DTSS.Pool}$) as follows.*

1. **Initialization:**
   - Setup: $\mathsf{DTSS.Setup}(1^\lambda, T_1, T_2, t) \rightarrow \{pp, pk, sk\}$, *on input security parameter λ, lower time bound T_1, and a value t, outputs public parameters pp and key pair (pk, sk) to be used for generating additional locked shares by the dealer D.*
2. **Distribution:**
   - Sharing: $\mathsf{DTSS.Sharing}(pp, s, pk, sk) \rightarrow \{\{C_i\}_{i\in[n]}, \{O_j\}_{j\in[t]}\}$, *on input pp, a secret s, and a key pair (pk, sk), outputs locked share C_i with time parameter T_1. Moreover, it outputs t additional locked shares $\{O_j\}_{j\in[t]}$, with O_j being locked with time parameter $(j+1)T_1$.*
3. **Reconstruction:**
   - Share recovery: $\mathsf{DTSS.ShaRecover}(pp, C_i) \rightarrow s_i$, *on input pp and C_i, outputs a share s_i. The algorithm is run by each party P_i.*
   - Additional share recovery: $\mathsf{DTSS.AddRecover}(pp, pk, \{O_j\}_{j\in[t]}) \rightarrow \{s'_j\}$, *on input pp, pk, and $\{O_j\}_{j\in[t]}$, forcibly outputs the additional share s'_j at time $(j+1)T_1$. The algorithm is run by anyone in $\mathcal{P}$ wishing to obtain additional shares.*
   - Pooling: $\mathsf{DTSS.Pool}(pp, \mathcal{S}, T_2) \rightarrow s$, *on input pp and a set $\mathcal{S}$ of shares (where $|\mathcal{S}| > t$ and $t \in pp$), outputs the secret s if T_2 has not elapsed.*

A correct DTSS scheme must satisfy *privacy*, *security*, and *robustness* with the following definitions.

Definition 2.1 (Privacy). *A DTSS satisfies privacy if for all algorithms $\mathcal{A}$ running in time $T < jT_1$, where $1 \leq j \leq t$, with at most T_1 parallel processors, there exists a simulator Sim and a negligible function μ such that for all secret $s \in S_\lambda$ and $\lambda \in \mathbb{N}$ it holds that*

$$\left|\Pr\left[\begin{array}{l}\mathcal{A}(pp, pk, s, \\ C_i, \{O_j\}_{j\in[t]}) = 1\end{array} : \begin{array}{l}\mathsf{DTSS.Setup}(1^\lambda, T_1) \rightarrow \{pp, pk, sk\}, \\ \mathcal{A}(1^\lambda, pp) \rightarrow s \\ \mathsf{DTSS.Sharing}(pp, s) \\ \rightarrow \{\{C_i\}_{i\in[n]}, \{O_j\}_{j\in[t]}\}\end{array}\right] - \right.$$

$$\left.\Pr\left[\begin{array}{l}\mathcal{A}(pp, pk, s', \\ C_i, \{O_j\}_{j\in[t]}) = 1\end{array} : \begin{array}{l}\mathsf{DTSS.Setup}(1^\lambda, T_1) \rightarrow \{pp, pk, sk\}, \\ \mathcal{A}(1^\lambda, pp) \rightarrow s' \\ \mathsf{Sim}(pp) \rightarrow \{\{C_i\}_{i\in[n]}, \{O_j\}_{j\in[t]}\}\end{array}\right]\right| \leq \mu(\lambda)$$

Definition 2.2 (t-Security). *Let $2T_1, \dots, (t+1)T_1$ be times at which each additional time-locked share is forcibly obtained. A DTSS is t-secure if prior to $(j+1)T_1$, where $1 \le j \le t$, the adversary controlling a set $|\mathcal{S}'| \le t-(j-1)$ of parties learns no information about $s \in S_\lambda$ in a computational sense. Thus, it holds*

$$\Pr\left[\mathcal{A}(pp, pk, \mathcal{S}', T_2) \rightarrow s \;:\; \begin{array}{l} \mathsf{DTSS.Setup}(1^\lambda, T_1, t) \rightarrow \{pp, pk, sk\} \\ \mathsf{DTSS.Sharing}(pp, s) \\ \rightarrow \{\{C_i\}_{i\in[n]}, \{O_j\}_{j\in[t]}\}, \\ \mathsf{DTSS.ShaRecover}(pp, C_i) \rightarrow s_i, \\ \mathsf{DTSS.AddRecover}(pp, pk, \{O_j\}_{j\in[t]}) \\ \rightarrow \{s'_j\}, 1 \le j \le t. \end{array}\right] \le \mu(\lambda) + \frac{1}{|S_\lambda|}$$

Definition 2.3 (Robustness). *A DTSS is robust if each party in $\mathcal{P}$ can* eventually *reconstruct the secret s, after receiving a sufficient number of other parties' shares and/or obtaining the additional time-locked shares.*

$$\Pr\left[\mathsf{DTSS.Pool}(pp, \mathcal{S}, T_2) \rightarrow s \;:\; \begin{array}{l} \mathsf{DTSS.Setup}(1^\lambda, T_1, t) \rightarrow \{pp, pk, sk\} \\ \mathsf{DTSS.Sharing}(pp, s) \\ \rightarrow \{\{C_i\}_{i\in[n]}, \boldsymbol{v}, \{O_j\}_{j\in[t]}\}, \\ \mathsf{DTSS.ShaRecover}(pp, C_i) \rightarrow s_i, \\ \mathsf{DTSS.AddRecover}(pp, pk, \{O_j\}_{j\in[t]}) \\ \rightarrow \{s'_j\}_{j\in[t]} \end{array}\right] = 1$$

5.3 DTSS Construction

We present a construction for DTSS in Fig. 3. We would like a protocol in which anyone can obtain each additional share s'_j at time $(j+1)T_1$ given that the dealer's role must end with the distribution phase.[4] In a naive way, the dealer should create t puzzles each embedding one additional share to be opened at t different points in time. However, this inefficient solution comes with a high computation cost as anyone wishing to access the shares needs to solve each puzzle separately in parallel, demanding up to $T_1 \sum_{j=1}^{t} j$ operations. To get away with this issue, we use multi-instance time-lock puzzle (MTLP) [1], a primitive allowing sequential (chained) release of solutions where the overall computation cost of solving t puzzles is equal to that of solving only the last one.

Theorem 2. If the multi-instance time-lock puzzle MTLP and timed secret sharing TSS are secure, then our DTSS protocol Π_{DTSS} presented in Fig. 3 satisfies the properties described in Sect. 5.2.

Proof. Privacy follows from that of Π_{TSS} together with the underlying Π_{MTLP} protocol for additional time-locked shares. The t-security is satisfied concerning

[4] Without loss of generality we assume $T_2 = (t+1)T_1$, accommodating the periodic release of additional shares.

Π_{DTSS}

1. **Initialization:**
   - Setup: $\mathsf{DTSS.Setup}(1^\lambda, T_1, t) \rightarrow \{pp, pk, sk\}$, the dealer D invokes two algorithms of $\mathsf{TSS.Setup}(1^\lambda, T_1, T_2)$ and $\mathsf{MTLP.Setup}(1^\lambda, T_1, t+1)$, and publishes the set of public parameters pp, pk.
2. **Distribution:**
   - Sharing: $\mathsf{DTSS.Sharing}(pp, s, pk, sk) \rightarrow \{\{C_i\}_{i\in[n]}, \{O_j\}_{j\in[t]}\}$, the dealer D first picks a secret $s \leftarrow \mathbb{Z}_q$ and invokes $\mathsf{TSS.Sharing}(pp, s)$ to generate n locked shares $\{C_i\}_{i\in[n]}$. Moreover, it computes t additional shares $f(a_j) = s'_j$ for $j \in [t]$, where $f(0) = s$ and $\{a_1, \dots, a_t\}$ are some known distinct points. Finally, it invokes $\mathsf{MTLP.Gen}(\mathbf{m}, pk, sk)$, where $\mathbf{m} = \{\perp, s'_1, \dots, s'_t\}$ to generate an MTLP containing $\{s'_j\}_{j\in[t]}$.
3. **Reconstruction:**
   - Share recovery: $\mathsf{DTSS.ShaRecover}(pp, C_i) \rightarrow s_i$, each party P_i runs $\mathsf{TSS.Recover}(pp, C_i)$ to recover their share s_i.
   - Additional share recovery: $\mathsf{DTSS.AddRecover}(pp, pk, \{O_j\}_{j\in[t]}) \rightarrow \{s'_j\}_{j\in[t]}$, anyone wishing to obtain additional time-locked shares $\{s'_j\}_{j\in[t]}$ runs $\mathsf{MTLP.Solve}(pp, \{O_j\}_{j\in[t]})$.
   - Pooling: $\mathsf{DTSS.Pool}(pp, \mathcal{S}, T_2) \rightarrow s$, upon having sufficient number of valid shares (*i.e.*, $\geq t+1$), the reconstrctor $V \in \mathcal{P}$ reconstructs the secret s using Lagrange interpolation at $f(0)$.

Fig. 3. Decrementing-threshold Timed Secret Sharing (DTSS) protocol

the gradual release of additional time-locked shares s'_j over time. That is, the adversary can forcibly learn s'_j by $(j+1)T_1$, reducing fault tolerance to $t-j$. The protocol is robust as each party P_i can eventually learn the secret by the time T_2 due to the t additional time-locked shares.

6 Verifiable Timed Secret Sharing (VTSS)

In this section, we present verifiable timed secret sharing (VTSS), an enhanced TSS which considers malicious adversaries. It protects against a malicious dealer who may send incorrect shares (or even no shares) during the distribution phase and against a malicious shareholder who may send an incorrect share during the reconstruction phase.

6.1 VTSS Definition

We present a formal definition of VTSS. Our definition extends the original verifiable secret sharing (VSS) of Feldman [31], incorporating the notion of time.

Definition 3 (Verifiable Timed Secret Sharing). *A verifiable timed secret sharing (VTSS) scheme involves the following algorithms.*

1. **Initialization:**
   – Setup: $\mathsf{VTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp$, *on input security parameter* λ, *lower time bound* T_1 *and upper time bound* T_2, *outputs public parameters pp.*
2. **Distribution:**
   – Sharing: $\mathsf{VTSS.Sharing}(pp, s) \rightarrow \{C_i, \pi_i\}_{i\in[n]}$, *on input pp and a secret s, outputs locked share* C_i *with time parameter* T_1 *and a proof of validity* π_i *for each party* $P_i \in \mathcal{P}$.
   – Share verification: $\mathsf{VTSS.Verify_1}(pp, C_i, \pi_i) \rightarrow 1/0$, *on input pp,* C_i, *and* π_i, *checks the validity of share to ensure the locked share* C_i *is well-formed and contains a valid share of secret s. The algorithm returns* 1 *if both checks pass. Otherwise, it returns* 0.
3. **Reconstruction:**
   – Recovering: $\mathsf{VTSS.Recover}(pp, C_i) \rightarrow s_i$, *on input pp and* C_i, *forcibly outputs a share* s_i. *The algorithm is run by each party* P_i.
   – Recovery verification: $\mathsf{VTSS.Verify_2}(pp, s_i, \pi_i) \rightarrow 1/0$, *on input pp,* s_i, *and* π_i, *checks the validity of submitted share. The algorithm is run by a verifier* $V \in \mathcal{P}$.
4. Pooling: $\mathsf{VTSS.Pool}(pp, \mathcal{S}, T_2) \rightarrow s$, *on input pp and a set* $\mathcal{S}$ *of shares (where* $|\mathcal{S}| > t$ *and* $t \in pp$*), outputs the secret s if* T_2 *has not elapsed and* $\perp$ *otherwise.*

A correct VTSS scheme must satisfy *soundness*, ensuring extractability and verifiability of the shares, *privacy*, and *security.*

Definition 3.1 (Correctness). *A VTSS satisfies correctness if for all secret* $s \in S_\lambda$ *and a set of shares* $|\mathcal{S}| > t$ *it holds*

$$\Pr\left[\begin{array}{l|l} \mathsf{VTSS.Verify_1}(pp, C_i, \pi_i) = 1 & \mathsf{VTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \mathsf{VTSS.Verify_2}(pp, s_i, \pi_i) = 1 & : \mathsf{VTSS.Sharing}(pp, s) \rightarrow \{C_i, \pi_i\}_{i\in[n]} \\ \mathsf{VTSS.Pool}(pp, \mathcal{S}, T_2) \rightarrow s & \mathsf{VTSS.Recover}(pp, C_i) \rightarrow s_i \end{array}\right] = 1$$

Definition 3.2 (Soundness). *A VTSS scheme is sound if there exists a negligible function* μ *such that for all PPT adversaries* $\mathcal{A}$ *and all* $\lambda \in \mathbb{N}$ *it holds*

$$\Pr\left[b_1 = 1 \vee b_2 = 1 : \begin{array}{l} \mathsf{VTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \mathcal{A}(pp) \rightarrow (\{C_i, \pi_i\}_{i\in[n]}, \{s_i, \pi'_i\}), \\ b_1 := \mathsf{VTSS.Verify}_{v1}(pp, C_i, \pi_i) \wedge \nexists s \ s.t. \\ \mathsf{VTSS.Sharing}(pp, s) \rightarrow (\{C_i\}_{i\in[n]}, \cdot), \\ b_2 := \mathsf{VTSS.Verify_2}(pp, s_i, \pi'_i) \wedge \nexists C_i \ s.t. \\ \mathsf{VTSS.Recover}(pp, C_i) \rightarrow s_i \end{array} \right] \leq \mu(\lambda)$$

Π_{VTSS}

1. **Initialization:**
   - Setup: $\mathsf{VTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp$, let g be a generator of a group $\mathbb{G}$ of order q. The dealer D runs $\mathsf{VTC.Setup}(1^\lambda, T_1)$ and publishes a set of public parameters pp.
2. **Distribution:**
   - Sharing: $\mathsf{VTSS.Sharing}(pp, s) \rightarrow \{C_i, \pi_i\}_{i\in[n]}$, D picks a secret $s \xleftarrow{\$} \mathbb{Z}_q$ to be shared among n parties. It samples a degree-t random polynomial $f(\cdot)$ such that $f(0) = s$ and $f(i) = s_i$ for $i \in [n]$. It then commits to f by computing $v_i = g^{s_i}$ and broadcasting $\mathbf{v} = \{v_i\}_{i\in[n]}$. Then, D runs $\mathsf{VTC.Commit}(pp, s_i)$ to create a locked share C_i and a corresponding proof of validity π'_i with respect to v_i, locking the share s_i to be opened forcibly at T_1, $\forall i \in [n]$. Let $\pi_i = \{\pi'_i, \mathbf{v}\}$. D privately sends each party P_i their sharing $\{C_i, \pi'_i\}$.
   - Share verification: $\mathsf{VTSS.Verify}_1(pp, C_i, \pi_i) \rightarrow 1/0$, party P_i runs $\mathsf{VTC.Verify}(pp, v_i, C_i, \pi'_i)$ to check the locked share C_i is well-formed and embeds the share s_i corresponding to v_i. They then validate the consistency of the shares by sampling a code word $\mathbf{y}^\perp \in \mathcal{C}^\perp$, where $\mathbf{y}^\perp = \{y_1^\perp, \dots, y_n^\perp\}$, and checking if $\prod_{j=1}^n v_j{}^{y_j^\perp} = 1$.
   - Complaint round: If a set of parties of size $\geq t+1$ complain about sharing, then D is disqualified. Otherwise, D reveals the corresponding locked shares with proofs by broadcasting $\{C_i, \pi'_i\}$. If the verification fails (or D does not broadcast), the dealer is disqualified.
3. **Reconstruction:**
   - Recovering: $\mathsf{VTSS.Recover}(pp, C_i) \rightarrow s_i$, each P_i wishing to participate in reconstruction runs $\mathsf{VTC.Solve}(pp, C_i)$ to obtain a share s_i.
   - Recovery verification: $\mathsf{VTSS.Verify}_2(pp, s_i, \pi_i) \rightarrow 1/0$, for each received share s_i from P_i, the reconstructor checks its validity by computing g^{s_i} and comparing it with v_i.
   - Pooling: $\mathsf{VTSS.Pool}(pp, \mathcal{S}, T_2) \rightarrow s$, upon having sufficient number of valid shares (*i.e.,* $\geq t+1$) received before T_2, the reconstrrctor (a party in $\mathcal{P}$) reconstructs the secret s using Lagrange interpolation at $f(0)$ or aborts otherwise.

Fig. 4. Verifiable Timed Secret Sharing (VTSS) protocol

Definition 3.3 (Privacy). *A VTSS satisfies privacy if for all parallel algorithms $\mathcal{A}$ whose running time is at most T_1 there exists a simulator Sim and a negligible function μ such that for all secret $s \in S_\lambda$ and all $\lambda \in \mathbb{N}$, it holds*

$$\left| \Pr\left[\mathcal{A}(pp, s, \{C_i, \pi_i\}) = 1 \; : \; \begin{array}{l} \mathsf{VTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \mathcal{A}(1^\lambda, pp) \rightarrow s \\ \mathsf{VTSS.Sharing}(pp, s) \;\; \rightarrow \{C_i, \pi_i\}_{i\in[n]} \end{array} \right] - \right.$$

$$\left|\Pr\left[\mathcal{A}(pp, s', \{C_j, \pi_j\}) = 1 \;:\; \begin{array}{r}\mathsf{VTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp,\\ \mathcal{A}(1^\lambda, pp) \rightarrow s'\\ \mathsf{Sim}(pp) \rightarrow \{C_j, \pi_j\}_{j\in[n]}\end{array}\right]\right| \leq \mu(\lambda)$$

Definition 3.4 (Security). *A VTSS satisfies security if there exists a negligible function μ such that for an adversary controlling a subset $\mathcal{S}'$ of parties, where $|\mathcal{S}'| \leq t$ and $s \in S_\lambda$ it holds*

$$\Pr\left[\mathcal{A}(pp, \mathcal{S}', T_2) \rightarrow s \;:\; \begin{array}{c}\mathsf{VTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp,\\ \mathsf{VTSS.Sharing}(pp, s) \rightarrow \{C_i, \pi_i\}_{i\in[n]}\\ \mathsf{VTSS.Recover}(pp, C_i) \rightarrow s_i\end{array}\right] \leq \mu(\lambda) + \frac{1}{|S_\lambda|}$$

6.2 VTSS Construction

We present a protocol for VTSS in Fig. 4. Following Feldman VSS [31], we make a crucial change in the protocol to adapt it for VTSS so that the dealer coudl convince each individual shareholder about the validity of their shares. Notably, in VTSS we have the dealer commit to the *shares* rather than the *coefficients* of the Shamir polynomial. This modification has two consequences.

First, it allows shareholders to check the consistency of the shares (*i.e.,* all lie on a polynomial of degree t) using properties of error-correcting code, particularly the Reed-Solomon code [49]. This is due to the equivalency of the Shamir secret sharing with Reed-Solomon encoding observed by [46].[5] We restate the basic fact of linear error correcting code in Lemma 1. We remark that in Feldman VSS the checking of each share is done against the commitment to the whole polynomial, but here it is done with respect to an individual commitment to each share, requiring the this step to ensure the sharing phase has been performed correctly.

Lemma 1. *Let $\mathcal{C}^\perp$ be the dual code of $\mathcal{C}$ that is a linear error correcting code over $\mathbb{Z}_q$ of length n. If $\boldsymbol{x} \in \mathbb{Z}_q^n \backslash \mathcal{C}$, and $\boldsymbol{y}^\perp$ is chosen uniformly at random from $\mathcal{C}^\perp$, the probability that the inner product of the vectors $\langle \boldsymbol{x}, \boldsymbol{y}^\perp \rangle = 0$ is exactly $1/q$.*

Second, it enables us to make use of VTC primitive [57] to *non-interactively* ensure each party P_i that they indeed obtains its correct share s_i at T_1. As mentioned, VTC allows committing to a signing key sk where its corresponding public key $pk = g^{sk}$ is publicly known. Our main insight is that we can think of $v_i = g^{s_i}$ published by the dealer as a public key for each share s_i committed by VTC. So, each party P_i can check the verifiability of its locked share C_i while ensuring the consistency of the shares $\{s_i\}_{i\in[n]}$.

Remark 1. We can realize the upper time bound in VTSS similarly to TSS by using the idea of secret sharing with additional shares (Sect. 5.1). We implicitly assume the additional time-locked shares are honestly generated due to our

[5] We refer the reader to [18] for a detailed description of the verification procedure.

motivation which is realizing an upper time bound (and thus breaking public goods game).[6]

Theorem 3. If the verifiable timed commitments VTC and Feldman verifiable secret sharing [31] are secure, then verifiable timed secret sharing protocol Π_{VTSS} presented in Fig. 4 satisfies soundness, privacy, and security, w.r.t. Definitions 3.2, 3.3, and 3.4 respectively.

Proof. Correctness is straightforward. The soundness property of the protocol follows directly from that of the underlying Π_{VTC} primitive for every single share s_i committed with respect to the v_i in $\mathbf{v}$. A maliciously generated $\mathbf{v}$ can pass the verification check $\mathsf{VTSS.Verify}_1$ only with probability $1/q$. A maliciously submitted s_i by P_i cannot pass the verification check $\mathsf{VTSS.Verify}_2$, except with negligible probability. The privacy property also follows directly from that of the underlying Π_{VTC} which implies the indistinguishability of a puzzle produced by $\mathsf{VTC.Sharing}$ and the one produced by Sim. Note that the commitment to shares $\mathbf{v}$ does not reveal any information about the secret s under the discrete logarithm assumption. It is important to note that for the assumption to hold the secret s should have a random distribution. Observe that before T_1 the privacy property essentially implies the security; afterward, the security follows directly from that of Feldman VSS due to the security of the commitment $\mathbf{v}$.

7 Publicly Verifiable Timed Secret Sharing (PVTSS)

In this section, we make our timed secret sharing scheme publicly verifiable, meaning that anyone, not only a participating party, is able to verify different phases of the scheme. To achieve this, we use a publicly verifiable secret sharing (PVSS) scheme as the main building block that compels parties to behave correctly by non-interactively proving the validity of the messages sent during the distribution and reconstruction phases.

7.1 PVTSS Definition

In this section, we present a formal definition of PVTSS according to the existing ones in the literature such as [18,19,53].

Definition 4 (Publicly Verifiable Timed Secret Sharing). *A PVTSS scheme involves the following algorithms.*

1. **Initialization:**
   - Setup: $\mathsf{PVTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp$, *on input security parameter* λ, *lower time bound* T_1, *and upper time bound* T_2, *outputs public parameters* pp. *Each party* P_i *announces a registered public key* pk_i *which the corresponding secret key* sk_i *is only known to them.*

[6] Should a malicious dealer attempt to misbehave, this assumption could be lifted by using less efficient cryptograhpic protocols.

2. **Distribution:**
   - Sharing: $\mathsf{PVTSS.Sharing}(pp, S, \{pk_i\}_{i\in[n]}) \rightarrow \{\{C_i\}_{i\in[n]}, \pi_D\}$, *on input pp, $\{pk_i\}_{i\in[n]}$, and a secret S, generates locked encrypted share C_i with time parameter T_1 for each party $P_i \in \mathcal{P}$. It also generates a proof π_D for the validity of shares.*
   - Share verification: $\mathsf{PVTSS.Verify_1}(pp, \{pk_i, C_i\}_{i\in[n]}, \pi_D) \rightarrow 1/0$, *on input pp, $\{pk_i, C_i\}_{i\in[n]}$, and π_D, checks the validity of the shares. This includes verifying the published locked encrypted shares are well-formed and contain correct shares of secret S. The algorithm is run by any verifier V.*
3. **Reconstruction:**
   - Recovering: $\mathsf{PVTSS.Recover}(pp, C_i, pk_i, sk_i) \rightarrow \{\tilde{s}_i, \pi_i\}$, *on input pp, C_i, pk_i, and sk_i, outputs a decrypted share $\tilde{s}_i$ together with proof π_i of valid decryption. The algorithm is run by each party $P_i \in \mathcal{P}$.*
   - Recovery verification: $\mathsf{PVTSS.Verify_2}(pp, C_i, \tilde{s}_i, \pi_i) \rightarrow \{0, 1\}$, *on input pp, C_i, $\tilde{s}_i$, and π_i, checks the validity of the decryption. The algorithm is run by any verifier V.*
   - Pooling: $\mathsf{PVTSS.Pool}(pp, \mathcal{S}, T_2) \rightarrow S$, *on input pp and a set $\mathcal{S}$ of decrypted shares $\tilde{s}_i$ (where $|\mathcal{S}| > t$ and $t \in pp$), outputs the secret S if T_2 has not elapsed.*

A PVTSS scheme must satisfy the following properties.

Definition 4.1 (Correctness). *A PVTSS satisfies correctness if for all secret $s \in S_\lambda$ and a set of shares $|\mathcal{S}| > t$ it holds that*

$$Pr\left[\begin{array}{l|l} \mathsf{PVTSS.Verify_1}(pp, \{C_i\}_{i\in[n]}, & \mathsf{PVTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \pi_D, \{pk_i\}_{i\in[n]}) = 1 & \mathsf{PVTSS.Sharing}(pp, S, \{pk_i\}_{i\in[n]}) \\ \mathsf{PVTSS.Verify_2}(pp, C_i, \tilde{s}_i, \pi_i) = 1 & \rightarrow \{\{C_i\}_{i\in[n]}, \pi_D\}, \\ \mathsf{PVTSS.Pool}(pp, \mathcal{S}, T_2) \rightarrow S & \mathsf{PVTSS.Recover}(pp, C_i, pk_i, sk_i) \\ & \rightarrow \{\tilde{s}_i, \pi_i\} \end{array}\right] = 1$$

Definition 4.2 (Soundness). *A PVTSS scheme is sound if there exists a negligible function μ such that for all PPT adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$ it holds that*

$$\Pr\left[b_1 = 1 \vee b_2 = 1 \,:\, \begin{array}{l} \mathsf{PVTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \mathcal{A}(pp) \rightarrow (\{pk_i, C_i\}_{i\in[n]}, \pi_D, \tilde{s}, \pi), \\ b_1 := \mathsf{PVTSS.Verify_1}(pp, \{pk_i, C_i\}_{i\in[n]}, \pi_D) \\ \wedge \nexists s \ s.t. \\ \mathsf{PVTSS.Sharing}(pp, S, \{pk_i\}_{i\in[n]}) \\ \rightarrow \{\{C_i\}_{i\in[n]}, \cdot\}, \\ b_2 := \mathsf{PVTSS.Verify_2}(pp, C, \tilde{s}, \pi) \wedge \nexists sk \ s.t. \\ \mathsf{PVTSS.Recover}(pp, C, pk, sk) \rightarrow \{\tilde{s}, \cdot\}, \end{array} \right] \le \mu(\lambda)$$

Definition 4.3 (t-Privacy). *A PVTSS satisfies t-privacy if for all parallel algorithms $\mathcal{A}$ whose running time is at most T_1, and set $I \subset [n]$ with $|I| = t+1$,* *there exists a simulator* Sim *and a negligible function* μ *such that for all secret* $s \in S_\lambda$ *and* $\lambda \in \mathbb{N}$ *it holds that*

$$\left| \Pr\left[\mathcal{A}(pp, S, \{C_i\}_{i\in[I]}, \pi_D) = 1 \;:\; \begin{array}{l} \mathsf{PVTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \mathcal{A}(1^\lambda, pp) \rightarrow S, \\ \mathsf{PVTSS.Sharing}(pp, S, \{pk_i\}_{i\in[I]}) \\ \rightarrow \{\{C_i\}_{i\in[I]}, \pi_D\} \end{array} \right] - \right.$$

$$\left. \Pr\left[\mathcal{A}(pp, S', \{C_j\}_{j\in[I]}, \pi_D) = 1 \colon \begin{array}{l} \mathsf{PVTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \mathcal{A}(1^\lambda, pp) \rightarrow S', \\ \mathsf{Sim}(pp) \rightarrow (\{C_j\}_{j\in[I]}, \pi_D) \end{array} \right] \right| \leq \mu(\lambda)$$

Definition 4.4 (Security). *A PVTSS satisfies security if there exists a negligible function* μ *such that for an adversary controlling a set* $\mathcal{S}'$ *of parties, where* $|\mathcal{S}'| \leq t$ *and* $s \in S_\lambda$*, together with the public information denoted by* PI*, it holds that*[7]

$$\Pr\left[\mathcal{A}(pp, \mathcal{S}', \mathsf{PI}, T_2) \rightarrow S : \begin{array}{l} \mathsf{PVTSS.Setup}(1^\lambda, T_1, T_2) \rightarrow pp, \\ \mathsf{PVTSS.Sharing}(pp, s, \{pk_i\}_{i\in[n]}) \\ \rightarrow \{\{C_i\}_{i\in[n]}, \pi_D\}, \\ \mathsf{PVTSS.Recover}(pp, C_i, pk_i, sk_i) \\ \rightarrow \{\tilde{s}_i, \pi_i\} \end{array} \right] \leq \mu(\lambda) + \frac{1}{|S_\lambda|}$$

An indistinguishability game given in [35,52] and adopted by [18] formalizes the security definition.

7.2 PVTSS Construction

We present a detailed description of the PVTSS protocol in Fig. 5. In what follows, we elaborate on several techniques used in our construction. In particular, it turns out that the public verifiability requirement of the scheme demands taking different approaches toward realizing the lower and upper time bounds.

Dealing with a Malicious Dealer. What makes the protection mechanism challenging for PVTSS is that *anyone*, before performing sequential computation, should be able to check the correctness of shares including consistency, validity, and extractability of the shares having a set of *encrypted* shares locked by the dealer. That is to say, a solution should *simultaneously* ensure (1) all shares lie on the same polynomial of degree t, (2) locked encrypted shares contain the committed shares, and (3) shares are obtainable in time T_1, all concerning some public information. We first discuss how to guarantee consistency and verifiability followed by our approach regarding extractability.

[7] This property is presented as IND1-Secrecy in [35,52].

Blinded DLEQ. Our solution to meet the first two aforementioned requirements is based on having the dealer blind each encrypted shares $\tilde{s}_i$ using some randomness β_i, put the randomness into a puzzle Z_i, and publish all the puzzles together with locked encrypted shares and commitments for $i \in [n]$. The dealer needs to show that the locked encrypted shares contain the same shares as the commitments, while the consistency of the shares can be checked using the commitments (as discussed in Sect. 6.2). To do so, we slightly modify the DLEQ proof (Sect. 3.5) and make it blinded. It allows proving simultaneous knowledge of two witnesses, one of which is common in two statements. The following is a protocol Π_{BDLEQ} for the language

$$L_{\mathsf{BDLEQ}} = \{(g_1, x, g_2, g_3, y) \mid \exists(\alpha, \beta) : x = g_1^{\alpha} \wedge y = g_2^{\alpha} g_3^{\beta}\}$$

1. P chooses two random elements $u_1, u_2 \xleftarrow{\$} \mathbb{Z}_q$, computes $a_1 = g_1^{u_1}$ and $a_2 = g_2^{u_1} g_3^{u_2}$, and sends them to V.
2. V sends back a randomly chosen challenge $c \xleftarrow{\$} \mathbb{Z}_q$.
3. P computes $r_1 = u_1 + c\alpha$ and $r_2 = u_2 + c\beta$ and sends them to V.
4. V checks if both $g_1^{r_1} = a_1 x^c$ and $g_2^{r_1} g_3^{r_2} = a_2 y^c$ hold.

Theorem 4. Protocol Π_{BDLEQ} is a public-coin honest-verifier zero-knowledge argument of knowledge corresponding to the language L_{BDLEQ}.

Proof. We show that the Π_{BDLEQ} satisfies the properties of a Sigma protocol. Completeness holds, as

$$g_1^{r_1} = g_1^{u_1 + c\alpha} = g_1^{u_1} g_1^{c\alpha} = a_1 x^c$$
$$g_2^{r_1} g_3^{r_2} = g_2^{u_1 + c\alpha} g_3^{u_2 + c\beta} = g_2^{u_1} g_3^{u_2} (g_2^{u_1} g_3^{u_2})^c = a_2 y^c$$

For knowledge soundness, given two accepting transcripts $(a_1, a_2; c; r_1, r_2)$ and $(a_1, a_2; c'; r'_1, r'_2)$ the witness (α, β) can be found as follows

$$g_1^{r_1} = a_1 x^c,\ g_2^{r_1} g_3^{r_2} = a_2 y^c \ ; \ g_1^{r'_1} = a_1 x^{c'},\ g_2^{r'_1} g_3^{r'_2} = a_2 y^{c'}$$
$$g_1^{r_1 - r'_1} = x^{c - c'} \Leftrightarrow x = g_1^{\frac{r_1 - r'_1}{c - c'}}$$
$$g_2^{r_1 - r'1} g_3^{r_2 - r'_2} = y^{c - c'} \Leftrightarrow y = g_2^{\alpha} g_3^{\frac{r_2 - r'_2}{c - c'}}$$

Hence, the witness β can be found as $\beta = (r_2 - r'_2)/(c - c')$ given the witness $\alpha = (r_1 - r'_1)/(c - c')$.
Let c be a given challenge. Zero-knowledge property is implied by the fact that the following two distributions, namely real protocol distribution and simulated distribution, are identically distributed.

$$\texttt{Real} : \{(a_1, a_2; c; r_1, r_2) : u_1, u_2 \xleftarrow{\$} \mathbb{Z}_q, a_1 = g_1^{u_1}, a_2 = g_2^{u_1} g_3^{u_2}; r_1 = u_1 + c\alpha, r_2 = u_2 + c\beta\}$$
$$\texttt{Sim} : \{(a_1, a_2; c; r_1, r_2) : r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q; a_1 = g_1^{r_1} x^{-c}, a_2 = g_2^{r_1} g_3^{r_2} y^{-c}\}$$

Note that the probability of occurring for each distribution is the same and equals $1/q^2$.

Cut-and-Choose. The dealer needs to convince the parties they can obtain their shares at time T_1. This is equivalent to saying that Z_i has indeed the value β_i embedded. A natural way to show the correctness of puzzle generation is by utilizing the cut-and-choose technique as in previous works [9,56]. This technique forces a sender to behave correctly by randomly opening a (fixed) set of puzzles it has already sent to the receiver based on the receiver's choice.

We remark that it is possible to deploy the cut-and-choose technique in our construction without sacrificing security. Given that opening just reveals a (random) set of size t of encrypted shares, we are still guaranteed that the secret remains hidden up to time T_1 as $t+1$ shares are needed for reconstruction. Each party is supposed to open their corresponding locked encrypted share, which is not among the opened ones by the dealer. Given public verification, we can stick to an honest majority assumption (*i.e.,* $\mathrm{t} < \mathrm{n}/2$) while ensuring soundness. We can borrow concrete numbers from related work in the same setting: For example, setting $n = 40$ would give a soundness error of 10^{-12} (Table 3, [57]).

Realizing an Upper Time Bound. Due to the public verifiability, PVTSS protocol is executed over a public bulletin board. As a result, the secret may be reconstructed/used by any external party after T_2. This demands taking a different approach towards realizing the upper time bound to make it more strict. Our solution is based on deploying *short-lived proofs* (SLPs) [5]. We Observe that the use of SLPs allows tying the *correctness* of the system to time, meaning that the secret is only guaranteed to be correct if it is reconstructed before the upper time bound. Correctness intuitively states if the distribution phase succeeds, then the reconstruction phase will output the *same* secret initially shared by the dealer. Let us now briefly explain how we make use of SLPs in our construction.

Upper Time Bound with SLPs. Our approach is to take advantage of the *forgeability* property of SLPs in the PVTSS construction. We piggyback on the *proof of decryptions* π_i generated by each party P_i as part of the reconstruction phase, turning them into SLPs where their expiration time matches the upper time bound T_2. Therefore, given the properties of short-lived proofs and also relying on that the secret has *uniformly random* distribution in Scrape PVSS,[8] the correctness of a share submitted by a party P_i is only guaranteed if being observed before T_2, otherwise it could be an invalid share accompanied with a valid proof. A short-lived proof for any arbitrary relation R for which there exists a Sigma protocol can be efficiently constructed [5]. For completeness, we present the short-lived proof for a relation R using pre-computed VDFs in Fig. 6.

In our protocol, we make a black box use of short-lived DLEQ proof generation denoted by $\mathsf{DLEQ.SLP}$ and verification denoted by $\mathsf{DLEQ.SLV}$. It is required that the beacon value b used to compute π_i is not known until the time T_1, with $T = T_2 - T_1$ being the time parameter for the underlying VDF. Therefore,

[8] This essentially implies any set of shares is indistinguishable from a set of random strings. Note that in normal Shamir secret sharing this is limited to a set of size at most t shares as the secret is not uniformly distributed [13].

anyone verifying the proof before T_2 knows that it could have not been computed through forgery. We highlight that, to deploy short-lived proofs we need to use the DDH-based version of Scrape PVSS which its DLEQ proof comes with *knowledge soundness* property.

Remark 2. Several recent works focus on the notion of forgeability over time, particularly for developing short-lived signature or forward-forgeable signature [5,55]. To the best of our knowledge, Arun et al. [5] is the only one exploring the time-based forgeability in proof systems. This in turn enables us to deploy their primitive to provide the upper time bound for PVTSS, binding the correctness of the secret reconstruction to time.

Remark 3. We do not assume the availability of an *online* verifier who observes the protocol over time. In fact, due to the characteristic of SLPs, their use is meaningful when the verifier does not necessarily remain online during the reconstruction period $[T_1, T_2]$; otherwise, it can always reject the proofs sent afterward, negating the forgeability property. Moreover, as pointed out in [5], convincingly timestamping the messages published on the bulletin board is opposed to the usability of SLPs.

In our PVTSS construction, we explicitly feed the upper time bound T_2 and a beacon value b in two algorithms, $\mathsf{PVTSS.Recover}$ and $\mathsf{PVTSS.Verify}_2$. This is essentially due to the necessity of the knowledge of time parameters $T = T_2 - T_1$ and b for short-lived proof generation and verification. Moreover, as discussed in [5], T does not need to be hardcoded when $\mathsf{PVTSS.Setup}$ is run. This allows the use of VDFs with any time parameter $T' > T$, while still generating short-lived proofs with respect to time T. That is, even if different parties use different time parameters with $T' > T$ for their VDF evaluations, only those proofs observed before time T are convincing.

Theorem 5. If the time-lock puzzle TLP, short-lived proofs SLP, and Scrape PVSS are secure, then publicly verifiable timed secret sharing protocol Π_{PVTSS} (presented in Fig. 5) satisfies soundness, t-privacy, and security, w.r.t. Definitions 4.2, 4.3, and 4.4 respectively.

Proof. Before T_2, the correctness is straightforward. Afterward, the correctness may fail with overwhelming probability due to the forgeability and indistinguishability properties of the underlying SLPs together with the uniform distribution of the secret s (and thus shares s_i). Anyone observing the public bulletin board after T_2 cannot distinguish an erroneous decryption share $\tilde{s_i}$ from a valid one as both pass the verification check $\mathsf{PVTSS.Verify}_2$. The soundness of the protocol follows from the underlying cut-and-choose argument and BDLEQ's soundness property. Note that by choosing parameters properly the soundness error for the cut-and-choose technique can be negligible in n. The property of t-privacy stems from the fact that given a random set of t opened locked encrypted shares produced by $\mathsf{VTC.Sharing}$, the simulator Sim can produce a locked encrypted share indistinguishable from any locked encrypted share that remained unopened due to the privacy properties of the underlying TLP. Security of the protocol follows directly from the underlying PVSS protocol. Note that blinded encrypted shares c_i distributed by the dealer provide semantic security due to the independent randomness β_i, while the original encryption method used in [18] to generate $\hat{s_i}$ is not IND-CPA-secure.

8 Discussion

In the following, we explore and discuss several aspects of our constructions.

On the Setup Phase. In all of our schemes, Setup algorithm is responsible for generating a set of public parameters pp, encapsulating the parameters for the underlying secret sharing and time-based cryptographic primitive. In particular, our VTSS construction in Fig. 4 requires a trusted setup to generate the parameters for the underlying VTC primitive. This is due to the linearly homomorphic TLP of [43] deployed in VTC construction. The functionality of the primitive depends on such an assumption; otherwise, either the puzzle is not solvable or one can efficiently solve it upon receipt. Using class groups of imaginary quadratic fields [16] as a family of groups of unknown order instead of the well-known RSA group is an option to reduce the trust, but comes with higher (offline) computational investment for the puzzle generator to compute the parameters through sequential computation [43]. Deploying the class groups solely does not eliminate the need for a trusted setup as it is still feasible that a malicious sender fools a receiver into accepting locked shares that will never be opened. Moreover, the VDF used in SLPs can be instantiated efficiently via class groups [60] without making any trusted setup assumption.

On the Use of SLPs. As previously mentioned, the use of SLPs necessitates the availability of a reconstructor prior to the upper time bound for a correct reconstruction. Moreover, we deploy short-lived proofs using precomputed VDFs [5] which do not offer reusable forgeability, *i.e.,* forging a proof for any statement v without computing a new VDF. However, this essentially fits a secret sharing setting (in particular, PVSS) which is inherently one-time use, *i.e.,* after reconstruction the secret is known and the system is not reusable.

Failure Probability. Although just some chances of reconstruction failure after T_2 should be enough to break the public goods game, here We briefly analyze the probability of a reconstruction failure after T_2 when deploying SLPs with an honest majority assumption. Let t be the number of adversarial shares and n be the total number of shares publicly available. Given that the incorporation of even *one* invalid share results in an invalid reconstruction and the fact that shares are uniformly distributed, the success probability can be computed as $p = \frac{p_1}{p_2}$, where $p_1 = \binom{n-t}{t+1}$ and $p_2 = \binom{n}{t+1}$. We can easily show that by a proper choice of the parameters n, t the reconstruction fails with overwhelming probability. Setting $t = \lceil \frac{n}{2} \rceil - 1$, we have $p \leq n2^{-(\lceil \frac{n}{2} \rceil + 1)}$ which is a negligible value in λ for a choice of $n = \lambda$.

Π_{PVTSS}

1. **Initialization:**
   – Setup: $\mathsf{PVTSS.Setup}(1^\lambda, T_1) \rightarrow pp$, the public parameters pp include independently chosen generators g_1, g_2, g_3 in a DDH-hard group $\mathbb{G}$, a field $\mathbb{Z}_q$, a hash function $H : \{0,1\}^* \rightarrow I \subset [n]$ with $|I| = t$, and a public bulletin board. Each party P_i announces a registered public key $pk_i = g_1^{sk_i}$ which its secret key sk_i is only known to them.
2. **Distribution:**
   – Sharing: $\mathsf{PVTSS.Sharing}(pp, S, \{pk_i\}_{i\in[n]}) \rightarrow \{\{C_i\}_{i\in[n]}, \pi_D\}$, the dealer D randomly chooses $s \xleftarrow{\$} \mathbb{Z}_q$ and defines the secret $S = g_1^s$ to be shared among n parties with public keys $\{pk_i\}_{i\in[n]}$. D computes Shamir shares $f(i) = s_i$, commitments $v_i = g_2^{s_i}$, and encrypted shares $\hat{s}_i = pk_i^{s_i}$ for all $i \in [n]$ using a degree-t Shamir polynomial $f(\cdot)$, where $f(0) = s$. It blinds the encrypted shares $\{\hat{s}_i\}_{i\in[n]}$ using some independent randomness β_i, resulting in $\{c_i\}_{i\in[n]}$, where $c_i = \hat{s}_i g_3^{\beta_i}$. The dealer then locks every randomness β_i in a TLP by running $\mathsf{TLP.Gen}(1^\lambda, T_1, \beta_i)$. Let denote $C_i = \{c_i, Z_i\}$. To show the consistency and validity of the locked encrypted shares, D runs Π_{BDLEQ}, resulting in proof $\pi =: (v_i, e, r_{1,i}, r_{2,i})$ for $i \in [n]$. Finally, D publishes the locked encrypted shares $\{C_i\}_{i\in[n]}$ and proof π_D on a public bulletin board. Moreover, D computes $H(\{C_i\}_{i\in[n]}, \pi) \rightarrow I$ as a random challenge (for cut and choose) and outputs $\pi_D = \{I, \pi, \beta_i, \hat{s}_i\}_{i\in[I]}$.
   – Share verification: $\mathsf{PVTSS.Verify_1}(pp, \{C_i\}_{i\in[n]}, \pi_D, \{pk_i\}_{i\in[n]}) \rightarrow 1/0$, the verifier V first validates the consistency of the shares by sampling a code word $\mathbf{y}^\perp \in C^\perp$, where $\mathbf{y}^\perp = \{y_1^\perp, \ldots, y_n^\perp\}$, and checking if $\prod_{j=1}^n {v_j}^{y_j^\perp} = 1$. V then checks the proof π_D is valid. After re-computing I, the verifier checks the puzzles are correctly constructed by invoking $\mathsf{TLP.Gen}$ algorithm and comparing the encrypted share sent by the dealer with the one being unlocked using β_i.
3. **Reconstruction:**
   – Recovering: $\mathsf{PVTSS.Recover}(pp, C_i, pk_i, sk_i, b, T_2) \rightarrow \{\tilde{s}_i, \pi_i\}$, after checking the validity of sharing phase, any party P_i wishing to obtain their share at T_1, unlocks the blinding factor β_i by running $\mathsf{TLP.Solve}(pp, Z_i)$, and obtains their share $\tilde{s}_i$ after decrypting $\hat{s}_i$ as $\tilde{s}_i = {\hat{s}_j}^{1/sk_i}$. Then, the party P_i reveals the share $\tilde{s}_i$ together with a short-lived proof $\pi_i =: \{\mathsf{DLEQ.SLP}(sk_i, g_1, pk_i, \tilde{s}_i, \hat{s}_i), \beta_i\}$ of valid decryption. Note that $\mathsf{DLEQ.SLP}$ involves calling $\mathsf{SLP.Gen}$ for the relation $R_{DLEQ} = \{(g_1, pk_i, \tilde{s}_i, \hat{s}_i; sk_i)\}$ given a beacon value b publicly known no sooner than T_1.
   – Recovery verification: $\mathsf{PVTSS.Verify_2}(pp, C_i, \tilde{s}_i, \pi_i, b, T_2) \rightarrow 1/0$, any (external) verifier V can check the validity of published share $\tilde{s}_i$ via $\mathsf{DLEQ.SLV}(\pi_i, g_1, pk_i, \tilde{s}_i, \hat{s}_i)$. Note that having C_i, the verifier first obtains $\hat{s}_i$ with β_i.
   – Pooling: $\mathsf{PVTSS.Pool}(pp, \mathcal{S}, T_2) \rightarrow S$, upon having sufficient number of shares ($\geq t+1$) received before time T_2, denoted by $\mathcal{S}$, anyone can reconstruct the secret $S = g_1^s$ using Lagrange interpolation in the exponent.

Fig. 5. Publicly Verifiable Timed Secret Sharing (PVTSS) protocol

Breaking Public Goods Game. A common method to break the public goods game is to reward those parties who publish their shares sooner via harnessing the financial capabilities of the blockchain systems [6,11,39]. That is, the shareholder receives some *reward* if their submitted share is among the first $t + 1$ shares published on the chain. This in turn creates a race and motivates the shareholder to show up sooner. Our two solutions, namely gradual release of additional shares and using short-lived proofs, can be considered as *orthogonal* methods that are *off-chain*. More precisely, the former approach essentially binds the security of the protocol to time by causing security reduction over time. The latter approach binds the correctness of the protocol to time, meaning that if the reconstruction does not occur sometime before T_2, then the correctness is not guarantee.[9]. As a result, in both approaches the shareholders are pushed to act as soon as possible to avoid any pitfalls.

Acknowledgements. The authors would like to thank Dan Ristea for the helpful discussions and Asiacrypt anonymous reviewers for their useful comments. Aydin Abadi was supported in part by REPHRAIN: The National Research Centre on Privacy, Harm Reduction and Adversarial Influence Online, under UKRI grant: EP/V011189/1.

A Cryptographic Primitives and Definitions

A.1 Time-Lock Puzzles (TLP)

Definition 5 (Time-lock Puzzle). *A time-lock puzzle (TLP) consists of the following two algorithms:*

1. $\mathsf{TLP.Gen}(1^\lambda, T, s) \rightarrow Z$, *a probabilistic algorithm that takes time parameter* T *and a secret* s*, and generates a puzzle* Z*.*
2. $\mathsf{TLP.Solve}(T, Z) \rightarrow s$, *a deterministic algorithm that solves the puzzle* Z *and retrieves the secret* s*.*

We recall the correctness and security definition of standard time-lock puzzles:

Correctness [43]**.** A TLP scheme is correct if for all $\lambda \in \mathbb{N}$, all polynomials $T(\cdot)$ in λ, and all $s \in S_\lambda$, it holds that

$$\Pr\left[\mathsf{TLP.Solve}(T(\lambda), Z) \rightarrow s : \mathsf{TLP.Gen}(1^\lambda, T(\lambda), s) \rightarrow Z\right] = 1$$

Security [43]**.** A TLP scheme is secure with gap $\epsilon < 1$ if there exists a polynomial $\tilde{T}(\cdot)$ such that for all polynomials $T(\cdot) \geq \tilde{T}(\cdot)$ and every polynomial-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in N}$ of depth $\leq T^\epsilon(\lambda)$, there exists a negligible function $\mu(\cdot)$, such that for all $\lambda \in \mathbb{N}$ and $s_0, s_1 \in \{0,1\}^\lambda$ it holds that $\Pr\left[\mathcal{A}(Z) \rightarrow b : \mathsf{TLP.Gen}(1^\lambda, T(\lambda), s_b) \rightarrow Z,\, b \overset{\$}{\leftarrow} \{0,1\}\right] \leq \frac{1}{2} + \mu(\lambda)$.

[9] This is a generic argument, independent of the adversarial behavior.

In particular, the seminal work of [50] introduced the notion of *encrypting to the future* using an RSA-based TLP. Loosely speaking, the sender encrypts a message m under a key k derived from the solution s to a puzzle Z. So, anyone can obtain m after running $\mathsf{TLP.Solve}(T, Z)$, and learning the key.

A.2 Homomorphic Time-Lock Puzzles (HTLP)

Definition 6 (Homomorphic Time-Lock Puzzles [43]). *Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a class of circuits and S_λ be a finite domain. A homomorphic time-lock puzzle (HTLP) with respect to $\mathcal{C}$ and with solution space S_λ is a tuple of algorithms (HTLP.Setup, HTLP.Gen, HTLP.Solve, HTLP.Eval) as follows.*

1. $\mathsf{HTLP.Setup}(1^\lambda, T) \rightarrow pp$, *a probabilistic algorithm that takes a security parameter 1^λ and time parameter T, and generates public parameters pp.*
2. $\mathsf{HTLP.Gen}(pp, s) \rightarrow Z$, *a probabilistic algorithm that takes public parameters pp and a solution $s \in S_\lambda$, and generates a puzzle Z.*
3. $\mathsf{HTLP.Solve}(pp, Z) \rightarrow s$, *a deterministic algorithm that takes public parameters pp and puzzle Z, and retrieves a secret s.*
4. $\mathsf{HTLP.Eval}(C, pp, Z_1, \ldots, Z_n) \rightarrow Z'$, *a probabilistic algorithm that takes a circuit $C \in \mathcal{C}_\lambda$ and a set of n puzzles $(Z_1, \ldots, Z_n)$, and outputs a puzzle Z'.*

Security [43]. An HTLP scheme ($\mathsf{HTLP.Setup}$, $\mathsf{HTLP.Gen}$, $\mathsf{HTLP.Solve}$, $\mathsf{HTLP.Eval}$) is secure with gap $\epsilon < 1$ if there exists a polynomial $\tilde{T}(\cdot)$ such that for all polynomials $T(\cdot) \geq \tilde{T}(\cdot)$ and every polynomial-size adversary $(\mathcal{A}_1, \mathcal{A}_2) = \{(\mathcal{A}_1, \mathcal{A}_2)_\lambda\}_{\lambda \in N}$ where the depth of $\mathcal{A}_2$ is bounded from above by $T^\epsilon(\lambda)$, there exists a negligible function $\mu(\cdot)$, such that for all $\lambda \in \mathbb{N}$ it holds that

$$\Pr\left[\mathcal{A}_2(pp, Z, \tau) \rightarrow b : \begin{array}{l} \mathcal{A}_1(1^\lambda) \rightarrow (\tau, s_0, s_1) \\ \mathsf{HTLP.Setup}(1^\lambda, T(\lambda)) \rightarrow pp \\ b \xleftarrow{\$} \{0,1\} \\ \mathsf{HTLP.Gen}(pp, s_b) \rightarrow Z \end{array}\right] \leq \frac{1}{2} + \mu(\lambda)$$

The puzzle is defined over a group of unknown order and is of the form $Z = (u, v)$, where $u = g^r$ and $v = h^{r \cdot N}(1+N)^s$. One notable point regarding the construction is that a trusted setup assumption is needed to generate the public parameters $pp = (T, N, g, h)$, where N is a safe modulus[10] and $h = g^{2^T}$. Such a setup phase is responsible for generating the parameters as specified and keeping the random coins secret; otherwise, either the puzzle is not solvable or one can efficiently solve it in time $t \ll T$. Having said that, the authors in [43] point out that this assumption can be removed if construction gets instantiated over class groups instead of an RSA group of unknown order. However, this comes at the cost of a higher computational overhead by the puzzle generator.

[10] A safe modulus is a product of two safe primes $P = 2p' + 1, Q = 2q' + 1$, where p' and q' are prime numbers.

A.3 Multi-instance Time-Lock Puzzle (MTLP)

Definition 7 (Multi-instance Time-lock Puzzle [1]**).** *A Multi-instance Time-lock Puzzle (MTLP) consists of the following five algorithms.*

1. $\mathsf{MTLP.Setup}(1^\lambda, T, z) \rightarrow \{pk, sk, \mathbf{d}\}$, *a probabilistic algorithm that takes a security parameter λ, a time parameter T, and the number of puzzle instances z, and outputs a key pair (pk, sk) and a secret witness vector* $\boldsymbol{d}$.
2. $\mathsf{MTLP.Gen}(\mathbf{m}, pk, sk, \mathbf{d}) \rightarrow \{\mathbf{o}, \mathbf{h}\}$, *a probabilistic algorithm that takes a message vector $\boldsymbol{m}$, the public-private key (pk, sk), secret witness vector $\boldsymbol{d}$, and outputs a puzzle vector $\boldsymbol{o}$ and a commitment vector $\boldsymbol{h}$.*
3. $\mathsf{MTLP.Solve}(pk, \mathbf{o}) \rightarrow \boldsymbol{s}$, *a deterministic algorithm that takes the public key pk and the puzzle vector $\boldsymbol{o}$, and outputs a solution vector $\boldsymbol{s}$, where s_j is of form $m_j \mid\mid d_j$.*
4. $\mathsf{Prove}(pk, s_j) \rightarrow \pi_j$, *a deterministic algorithm that takes the public key pk and a solution s_j, and outputs a proof π_j.*
5. $\mathsf{Verify}(pk, \pi_j, h_j) \rightarrow \{0, 1\}$, *a deterministic algorithm that takes the public key pk, proof π_j, and commitment h_j. If verification succeeds, it outputs 1, otherwise 0.*

Security [1]**.** A multi-instance time-lock puzzle is secure if for all λ and T, any number of puzzle: $z \geq 1$, any j (where $1 \leq j \leq z$), any pair of randomised algorithm $\mathcal{A} : (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ runs in time $O(poly(jT, \lambda))$ and $\mathcal{A}_2$ runs in time $\delta(jT) < jT$ using at most $\pi(T)$ parallel processors, there exists a negligible function $\mu(.)$ such that

$$\Pr\left[\begin{array}{l} \mathcal{A}_2(pk, \ddot{o}, \tau) \rightarrow \ddot{a} \\ \text{s.t.} \\ \ddot{a} : (b_i, i) \\ m_{b_i, i} = m_{b_j, j} \end{array} : \begin{array}{l} \mathsf{MTLP.Setup}(1^\lambda, \Delta, z) \rightarrow (pk, sk, \mathbf{d}) \\ \mathcal{A}_1(1^\lambda, pk, z) \rightarrow (\tau, \mathbf{m}) \\ \forall j', 1 \leq j' \leq z : b_{j'} \xleftarrow{\$} \{0, 1\} \\ \mathsf{MTLP.Gen}(\mathbf{m}', pk, sk, \mathbf{d}) \rightarrow \ddot{o} \end{array}\right] \leq \frac{1}{2} + \mu(\lambda)$$

A.4 Verifiable Delay Function

Definition 8 (Verifiable Delay Function). *A verifiable delay function (VDF) consists of the following three algorithms:*

1. $\mathsf{VDF.Setup}(1^\lambda, T) \rightarrow pp$, *a probabilistic algorithm that takes security parameter λ and time parameter T, and generates system parameters pp.*
2. $\mathsf{VDF.Eval}(pp, x) \rightarrow \{y, \pi\}$, *a deterministic algorithm that given system parameters pp and a randomly chosen input x, computes a unique output y and a proof π.*
3. $\mathsf{VDF.Verify}(pp, x, y, \pi) \rightarrow \{0, 1\}$, *a deterministic algorithm that verifies y indeed is a correct evaluation of the x. If verification succeeds, the algorithm outputs 1, and otherwise 0.*

Intuitively, there are three security properties that a valid VDF should satisfy. There must be a run time constraint of $(1+\epsilon)T$ for a positive constant ϵ to limit the evaluation algorithm, called *ϵ-evaluation*. The VDF should have *sequentially*, meaning no adversary using parallel processors can successfully compute the output without executing proper sequential computation. Lastly, the VDF evaluation should be a function with *uniqueness* property. That is, the verification algorithm must accept only one output per input.

VDF Constructions. Among a variety of constructions, VDFs based on repeated squaring have gained more attention as they offer a simple evaluation function that is more compatible with the hardware and provides better accuracy in terms of the time needed to perform the computation. The two concurrent works of [48,60] suggest evaluating the function $y = x^{2^T}$ over a hidden-order group. Despite similarities in construction, they present two independent ways of proof generation. Particularly, the one proposed by Wesolowski [60] enjoys the luxury of having a constant size proof and verification cost. In addition, Wesolowski's construction can be instantiated over class groups of imaginary quadratic fields [16] which do not require a trusted setup assumption.

A.5 Verifiable Timed Commitment

Definition 9 (Verifiable Timed Commitment [57]**).** *A verifiable timed commitment consists of the following algorithms:*

1. $\mathsf{VTC.Setup}(1^\lambda, T) \rightarrow pp$, *a probabilistic algorithm that takes a security parameter* 1^λ *and time parameter* T*, and generates public parameters* pp*.*
2. $\mathsf{VTC.Commit}(pp, s) \rightarrow \{C, \pi\}$, *a probabilistic algorithm that takes public parameters* pp *and a secret* s*, and generates a commitment* C *and proof* π*.*
3. $\mathsf{VTC.Verify}(pp, pk, C, \pi) \rightarrow \{0, 1\}$, *a deterministic algorithm that takes public parameters* pp*, a public key* pk*, the commitment* C*, and proof* π*, and checks if the commitment contains a valid* s *with respect to* pk*.*
4. $\mathsf{VTC.Solve}(pp, C) \rightarrow s$, *a deterministic algorithm that takes commitment* C*, and outputs a secret* s*.*

Intuitively, a correct VTC should satisfy *soundness*, ensuring the commitment C indeed embeds a valid secret s with respect to pk, and *privacy*, ensuring that no parallel adversary with a running time of less than T succeeds in extracting s, except with negligible probability.

A.6 Sigma Protocols

Let $R = \{(v; w)\} \in \mathcal{V} \times \mathcal{W}$ denote a relation containing the pairs of instances and corresponding witnesses. A Sigma protocol Σ on the $(v; w) \in R$ is an interactive protocol with three movements between P and V as follows.

1. $\Sigma.\mathsf{Ann}(v, w) \rightarrow a$, runs by P and outputs a message a to V.

2. $\Sigma.\mathsf{Cha}(v) \rightarrow c$, runs by V and outputs a message c to P.
3. $\Sigma.\mathsf{Res}(v, w, c) \rightarrow r$, runs by P and outputs a message r to V.
4. $\Sigma.\mathsf{Ver}(v, a, c, r) \rightarrow \{0, 1\}$, runs by V and outputs 1 if statement holds.

A Sigma protocol has three main properties including *completeness*, *knowledge soundness*, and *zero-knowledge*. Completeness guarantees the verifier gets convinced if parties follow the protocol. Special soundness states that a malicious prover P^* cannot convince the verifier of a statement without knowing its corresponding witness except with a negligible probability. This is formalized by considering an efficient algorithm called *extractor* to extract the witness given a pair of valid protocol transcripts with different challenges showing the computational infeasibility of having such pairs and therefore guaranteeing the knowledge of the witness by P. The notion of zero-knowledge ensures that no information is leaked to the verifier regarding the witness. This is formalized by considering an efficient algorithm called *simulator* which given the instance v, and also the challenge c, outputs a simulated transcript that is indistinguishable from the transcript of the actual protocol execution. Note that this property only needs to hold against an *honest verifier* which seems to be a limitation of the description, but allows for having much more efficient constructions compared to generic models. The interactive protocol described above can be easily turned into a non-interactive variant using the Fiat-Shamir heuristic [32] in the random oracle model, making it publicly verifiable with no honest verifier assumption.

A.7 Short-Lived Proofs

Definition 10 (Short-lived Proofs [5]). *A short-lived proof scheme includes a tuple of the following algorithms:*

1. $\mathsf{SLP.Setup}(1^\lambda, T) \rightarrow pp$, *a probabilistic algorithm that takes security parameter λ and time parameter T, and generates public parameters pp.*
2. $\mathsf{SLP.Gen}(pp, v, w, b) \rightarrow \pi$, *a probabilistic algorithm that takes a $(v; w) \in R$ and a random value b, and generates a proof π.*
3. $\mathsf{SLP.Forge}(pp, v, b) \rightarrow \pi$, *a probabilistic algorithm that takes any instance v and a random value b, and generates a proof π.*
4. $\mathsf{SLP.Verify}(pp, v, \pi, b) \rightarrow 1/0$, *a probabilistic algorithm verifying that π indeed is a valid short-lived proof of the instance v. If verification succeeds, the algorithm outputs 1, and otherwise 0.*

Note that the definition assumes there exists a *randomness beacon* which outputs an unpredictable value b periodically at certain times. There are various ways to implement such beacons including using a public blockchain [15], financial market [26], and more. Such an assumption is necessary to eliminate the need for having a shared global clock (*i.e.,* timestamping). As parties agree on the initial point in time (implied by b), the proof π tied to b must have been observed before time T to be convincing, otherwise might be a forgery.

Π_{SLP}

1. Initialization: On input a random value b^*, compute $\mathsf{VDF.Eval}(pp, b^*) \rightarrow \{y^*, \pi^*_{VDF}\}$
2. Proof generation: $\mathsf{SLP.Gen}(pp, v, w, b) \rightarrow \pi$,
   - Compute $\Sigma.\mathsf{Announce}(v, w) \rightarrow a$
   - Compute $c = H(v \parallel b \parallel a)$
   - Set sub-challenge $c_2 = b^* \oplus b$
   - Compute sub-challenge $c_1 = c \oplus c_2$
   - Compute $\Sigma.\mathsf{Response}(v, w, a, c_1) \rightarrow r$
   - Output $\pi =: \{a, c_1, r, c_2, y^*, \pi^*_{VDF}\}$
3. Forgery: $\mathsf{SLP.Forge}(pp, v, b) \rightarrow \tilde{\pi}$,
   - Compute $\Sigma.\mathsf{Simulator}(v) \rightarrow (\tilde{a}, \tilde{c_1}, \tilde{r})$
   - Compute $c = H(v \parallel b \parallel \tilde{a})$
   - Set sub-challenge $c_2 = c \oplus \tilde{c_1}$
   - Compute $\mathsf{VDF.Eval}(pp, b \oplus c_2) \rightarrow \{y, \pi_{VDF}\}$
   - Output $\tilde{\pi} =: \{\tilde{a}, \tilde{c_1}, \tilde{r}, c_2, y, \pi_{VDF}\}$
4. Proof verification: $\mathsf{SLP.Verify}(pp, v, \pi/\tilde{\pi}, b) \rightarrow \{0, 1\}$
   - Compute $c = H(v \parallel b \parallel a)$
   - Accept if:
     - $c = c_1 \oplus c_2$
     - $\Sigma.\mathsf{Verify}(v, a, c_1, r) = 1$
     - $\mathsf{VDF.Verify}(pp, b \oplus c_2, y, \pi_{VDF}) = 1$

Fig. 6. Short-lived proof for a relation $R = \{(v; w)\}$ using pre-computed VDFs [5]

SLP Using Sigma protocols. Short-lived proofs can be instantiated both using generic (non-interactive) zero-knowledge proofs and efficient Sigma protocols. However, as shown in [5], making a Sigma protocol short-lived is rather tricky as it needs some modification in the protocol for OR-composition to be secure according to SLP properties. The modification is done in such a way to let the honest prover create an SLP in a short time without needing to wait for time T to compute the VDF but forces the malicious prover to do the sequential computation, preventing her from computing a forgery before time T. More accurately, in an Or-composition the prover can convince the verifier even if it only knows the witness to one of the relations. To do so, the verifier lets the prover somehow cheat by using the simulator for the relation that it does not know the witness for. Thus, having one degree of freedom the prover chooses two sub-challenges c_1 and c_2 under the constraint that $c_1 + c_2 = c$. Note that the prover is free to fix one of them and compute the other one under the constraints. The observation made in [5] to let the honest prover quickly generate the short-lived proof is to involve the beacon b in the generation of the challenge. Therefore, an honest prover just needs to pre-compute the VDF on a random value b^* allowing her to use it when computing the forgery by freely setting one of the sub-challenges, say c_2, to $b^* \oplus b$ and letting $c_1 = c \oplus c_2$. A malicious prover, however, should compute the VDF on demand as it does not know a witness w for the relation R and c_1 gets fixed by the simulator, taking away the possibility of setting c_2 as specified.

As an optimization, some alternative ways for generating a VDF solution by the honest prover instead of pre-computing a VDF from scratch have been proposed that we refer the reader to [5] for more details.

References

1. A. Abadi and A. Kiayias. Multi-instance publicly verifiable time-lock puzzle and its applications. In *International Conference on Financial Cryptography and Data Security*, pages 541–559. Springer, 2021.
2. A. V. Aho and J. E. Hopcroft. *The design and analysis of computer algorithms.* Pearson Education India, 1974.
3. G. Almashaqbeh, F. Benhamouda, S. Han, D. Jaroslawicz, T. Malkin, A. Nicita, T. Rabin, A. Shah, and E. Tromer. Gage mpc: Bypassing residual function leakage for non-interactive mpc. *Proceedings on Privacy Enhancing Technologies*, 2021.
4. M. Archetti and I. Scheuring. Game theory of public goods in one-shot social dilemmas without assortment. *Journal of theoretical biology*, 299:9–20, 2012.
5. A. Arun, J. Bonneau, and J. Clark. Short-lived zero-knowledge proofs and signatures. In *Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part III*, pages 487–516. Springer, 2023.
6. Z. Avarikioti, E. Kokoris-Kogias, R. Wattenhofer, and D. Zindros. B rick: Asynchronous incentive-compatible payment channels. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*, pages 209–230. Springer, 2021.
7. C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas. Dynamic ad hoc clock synchronization. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 399–428. Springer, 2021.
8. A. Bagherzandi, S. Jarecki, N. Saxena, and Y. Lu. Password-protected secret sharing. In *Proceedings of the 18th ACM conference on Computer and Communications Security*, pages 433–444, 2011.
9. W. Banasik, S. Dziembowski, and D. Malinowski. Efficient zero-knowledge contingent payments in cryptocurrencies without scripts. In *Computer Security–ESORICS 2016: 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part II 21*, pages 261–280. Springer, 2016.
10. C. Baum, B. David, R. Dowsley, R. Kishore, J. B. Nielsen, and S. Oechsner. Craft: C omposable r andomness beacons and output-independent a bort mpc f rom t ime. In *IACR International Conference on Public-Key Cryptography*, pages 439–470. Springer, 2023.
11. D. Beaver, K. Chalkias, M. Kelkar, L. K. Kogias, K. Lewi, L. de Naurois, V. Nicolaenko, A. Roy, and A. Sonnino. Strobe: Stake-based threshold random beacons. *Cryptology ePrint Archive*, 2021.
12. A. Beimel, Y. Ishai, and E. Kushilevitz. Ad hoc psm protocols: Secure computation without coordination. In *Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part III 36*, pages 580–608. Springer, 2017.

13. M. Bellare, W. Dai, and P. Rogaway. Reimagining secret sharing: Creating a safer and more versatile primitive by adding authenticity, correcting errors, and reducing randomness requirements. *Proceedings on Privacy Enhancing Technologies*, 2020(4), 2020.
14. D. Boneh and M. Naor. Timed commitments. In *Annual international cryptology conference*, pages 236–254. Springer, 2000.
15. J. Bonneau, J. Clark, and S. Goldfeder. On bitcoin as a public randomness source. *Cryptology ePrint Archive*, 2015.
16. J. Buchmann and H. C. Williams. A key-exchange system based on imaginary quadratic fields. *Journal of Cryptology*, 1(2):107–118, 1988.
17. J. Burdges and L. D. Feo. Delay encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 302–326. Springer, 2021.
18. I. Cascudo and B. David. Scrape: Scalable randomness attested by public entities. In *International Conference on Applied Cryptography and Network Security*, pages 537–556. Springer, 2017.
19. I. Cascudo, B. David, L. Garms, and A. Konring. Yolo yoso: fast and simple encryption and secret sharing in the yoso model. In *Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part I*, pages 651–680. Springer, 2023.
20. M. Chase, H. Davis, E. Ghosh, and K. Laine. Acsesor: A new framework for auditable custodial secret storage and recovery. *Cryptology ePrint Archive*, 2022.
21. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Annual international cryptology conference*, pages 89–105. Springer, 1992.
22. M. Chen, C. Hazay, Y. Ishai, Y. Kashnikov, D. Micciancio, T. Riviere, A. Shelat, M. Venkitasubramaniam, and R. Wang. Diogenes: lightweight scalable rsa modulus generation with a dishonest majority. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 590–607. IEEE, 2021.
23. Y.-H. Chen and Y. Lindell. Feldman's verifiable secret sharing for a dishonest majority. *IACR Communications in Cryptology*, 1(1), 2024.
24. A. R. Choudhuri, S. Garg, J. Piet, and G.-V. Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3513–3529. USENIX Association, 2024.
25. P. Chvojka, T. Jager, D. Slamanig, and C. Striecks. Versatile and sustainable timed-release encryption and sequential time-lock puzzles. In *European Symposium on Research in Computer Security*, pages 64–85. Springer, 2021.
26. J. Clark and U. Hengartner. On the use of financial data as a random beacon. *Evt/wote*, 89, 2010.
27. P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927. IEEE, 2020.
28. Y. Dodis and D. H. Yum. Time capsule signature. In *International Conference on Financial Cryptography and Data Security*, pages 57–71. Springer, 2005.
29. Y. Doweck and I. Eyal. Multi-party timed commitments. *arXiv preprint* arXiv:2005.04883, 2020.
30. S. D. Dwilson. What happened to julian assange's dead man's switch for the wikileaks insurance files? https://heavy.com/news/2019/04/julian-assange-dead-mans-switch-wikileaks-insurance-files/, Apr. 2019. Section: News.

31. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 427–438. IEEE, 1987.
32. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.
33. J. A. Garay and M. Jakobsson. Timed release of standard digital signatures. In *International Conference on Financial Cryptography*, pages 168–182. Springer, 2002.
34. J. Y. Halpern, B. Simons, R. Strong, and D. Dolev. Fault-tolerant clock synchronization. In *Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 89–102, 1984.
35. S. Heidarvand and J. L. Villar. Public verifiability from pairings in secret sharing schemes. In *International Workshop on Selected Areas in Cryptography*, pages 294–308. Springer, 2008.
36. L. Heimbach and R. Wattenhofer. Sok: Preventing transaction reordering manipulations in decentralized finance. *arXiv preprint* arXiv:2203.11520, 2022.
37. S. Jarecki, A. Kiayias, and H. Krawczyk. Round-optimal password-protected secret sharing and t-pake in the password-only model. In *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20*, pages 233–253. Springer, 2014.
38. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 177–194. Springer, 2010.
39. E. Kokoris-Kogias, E. C. Alp, L. Gasser, P. Jovanovic, E. Syta, and B. Ford. Calypso: private data management for decentralized ledgers. *Proceedings of the VLDB Endowment*, 14(4):586–599, 2020.
40. Y. Lindell. Fast cut-and-choose-based protocols for malicious and covert adversaries. *Journal of Cryptology*, 29(2):456–490, 2016.
41. A. F. Loe, L. Medley, C. O'Connell, and E. A. Quaglia. Tide: A novel approach to constructing timed-release encryption. *Cryptology ePrint Archive*, 2021.
42. Y. Ma, J. Woods, S. Angel, A. Polychroniadou, and T. Rabin. Flamingo: Multi-round single-server secure aggregation with applications to private federated learning. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 477–496. IEEE, 2023.
43. G. Malavolta and S. A. K. Thyagarajan. Homomorphic time-lock puzzles and applications. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I*, pages 620–649. Springer, 2019.
44. D. Malkhi and P. Szalachowski. Maximal extractable value (mev) protection on a dag. In *4th International Conference on Blockchain Economics, Security and Protocols*, page 1, 2023.
45. Y. Manevich and A. Akavia. Cross chain atomic swaps in the absence of time via attribute verifiable timed commitments. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 606–625. IEEE, 2022.
46. R. J. McEliece and D. V. Sarwate. On sharing secrets and reed-solomon codes. *Communications of the ACM*, 24(9):583–584, 1981.

47. L. Medley, A. F. Loe, and E. A. Quaglia. Sok: Delay-based cryptography. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, pages 169–183. IEEE, 2023.
48. K. Pietrzak. Simple verifiable delay functions. In *10th innovations in theoretical computer science conference (itcs 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
49. I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.
50. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. 1996.
51. A. Rondelet and Q. Kilbourn. Threshold encrypted mempools: Limitations and considerations. *arXiv preprint* arXiv:2307.10878, 2023.
52. A. Ruiz and J. L. Villar. Publicly verifiable secret sharing from paillier's cryptosystem. In *WEWoRC 2005–Western European Workshop on Research in Cryptology*. Gesellschaft für Informatik eV, 2005.
53. B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Annual International Cryptology Conference*, pages 148–164. Springer, 1999.
54. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
55. M. A. Specter, S. Park, and M. Green. {KeyForge}:{Non-Attributable} email from {Forward-Forgeable} signatures. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1755–1773, 2021.
56. S. Srinivasan, J. Loss, G. Malavolta, K. Nayak, C. Papamanthou, and S. A. Thyagarajan. Transparent batchable time-lock puzzles and applications to byzantine consensus. In *IACR International Conference on Public-Key Cryptography*, pages 554–584. Springer, 2023.
57. S. A. K. Thyagarajan, A. Bhat, G. Malavolta, N. Döttling, A. Kate, and D. Schröder. Verifiable timed signatures made practical. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1733–1750, 2020.
58. S. A. K. Thyagarajan, G. Castagnos, F. Laguillaumie, and G. Malavolta. Efficient cca timed commitments in class groups. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2663–2684, 2021.
59. S. A. K. Thyagarajan, T. Gong, A. Bhat, A. Kate, and D. Schröder. Opensquare: Decentralized repeated modular squaring service. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3447–3464, 2021.
60. B. Wesolowski. Efficient verifiable delay functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 379–407. Springer, 2019.
61. H. Zhang, L.-H. Merino, Z. Qu, M. Bastankhah, V. Estrada-Galiñanes, and B. Ford. F3b: A low-overhead blockchain architecture with per-transaction front-running protection. In *5th Conference on Advances in Financial Technologies*, 2023.

Security Against Physical Attacks

Formal Definition and Verification for Combined Random Fault and Random Probing Security

Sonia Belaïd[1], Jakob Feldtkeller[2](✉), Tim Güneysu[2,3], Anna Guinet[2], Jan Richter-Brockmann[2], Matthieu Rivain[1], Pascal Sasdrich[2], and Abdul Rahman Taleb[1]

[1] CryptoExperts, Paris, France
{sonia.belaid,matthieu.rivain}@cryptoexperts.com
[2] Ruhr University Bochum, Horst Görtz Institute for IT Security, Bochum, Germany
{jakob.feldtkeller,tim.gueneysu,anna.guinet,jan.richter-brockmann, pascal.sasdrich}@rub.de
[3] DFKI, Bremen, Germany

Abstract. In our highly digitalized world, an adversary is not constrained to purely digital attacks but can monitor or influence the physical execution environment of a target computing device. Such side-channel or fault-injection analysis poses a significant threat to otherwise secure cryptographic implementations. Hence, it is important to consider additional adversarial capabilities when analyzing the security of cryptographic implementations besides the default black-box model. For side-channel analysis, this is done by providing the adversary with knowledge of some internal values, while for fault-injection analysis the capabilities of the adversaries include manipulation of some internal values.

In this work, we extend probabilistic security models for physical attacks, by introducing a *general random probing model* and a *general random fault model* to enable security analysis with arbitrary leakage and fault distributions, as well as the combination of these models. Our aim is to enable a more accurate modeling of low-level physical effects. We then analyze important properties, such as the impact of adversarial knowledge on faults and compositions, and provide tool-based formal verification methods that allow the security assessment of design components. These methods are introduced as extension of previous tools VERICA and IronMask which are implemented, evaluated and compared.

Keywords: Physical Security · Random Probing Model · Random Fault Model · Combined Analysis · VERICA · IronMask

1 Introduction

Advanced cryptographic schemes are typically examined within the common black-box model, which assumes that the internal values of the scheme remain

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15490, pp. 167–200, 2025.
https://doi.org/10.1007/978-981-96-0941-3_6

concealed and protected from any adversary. However, over the past 25 years, physical attacks that exploit the actual implementation of cryptographic algorithms have challenged these assumptions. In particular, Side-Channel Analysis (SCA) exploits dependencies between processed values and physical execution characteristics such as timing behavior [32], instantaneous power consumption [33], or electromagnetic emanations [26], to conclude information about a secret. Similarly, Fault Injection Analysis (FIA) manipulates the physical execution environment to create a faulty intermediate state such that the resulting output gives a hint about the processed secret. Common fault injection methods include clock and voltage glitching [19,49], targeted electromagnetic (EM) pules [16,21], or focused laser beams [47]. Both attack vectors question a different assumption of the cryptographic black-box model: SCA that internal values are *confidential* and hidden from the adversary and FIA that the internal values have *integrity* and cannot be manipulated by the adversary. However, in a real attack scenario, the adversary is not restricted to performing only one of the two attack types. Indeed, first practical attack for Combined Analysis (CA), i.e., the combination of FIA and SCA, are emerging [3,14,41–44].

Constructing effective countermeasures against such physical attacks requires a deep understanding of attack properties and leakage behavior. To this end, the research community endeavors to theoretically model the leakage emanating from the victim device which is susceptible to exploitation by the adversary. In particular for SCA, in the famous d-probing model introduced by Ishai, Sahai, and Wagner [31], the leakage is modeled by the exact values of d internal variables of the adversary's choice. A victim device is then deemed d-probing secure if any such set of d intermediate variables is statistically independent on the processed secret. While this probing model facilitates security proofs, it sometimes fails to closely reflect the reality of embedded devices. For instance, it does not capture horizontal attacks [6], which exploit repeated manipulation of variables within an execution. As a consequence, the community is beginning to focus on more realistic leakage models, like the ϵ-random probing model [2,8,31]. In this model, the exact value carried by each wire of the circuit is leaked with probability p to the adversary. The security is then determined by the probability ϵ to obtain a secret-dependent probe combination. This model tightly reduces to the security in the practical noisy leakage model [20,36], where each variable leaks a noisy function of its value. Nevertheless, the random probing model is still insufficient in modeling low-level physical effects because it assigns the same independent probability p to each intermediate variable whereas the underlying noise is likely to be different and the independence assumption not verified in practice.

For modeling of FIA the adversary is given the ability to manipulate a set of intermediate variables in a predefined way and then their impact on the system output is analyzed. Models from the literature often take inspiration from probing models for SCA. Specifically, the k-fault model [30] allows the adversary to manipulate up to k intermediate values, while the random fault model [18] defines a fault probability q and manipulates each internal value with probability q. In both models, the adversary wins when they get a faulty output that cannot be detected or corrected, whereas the random fault model determines the probability μ with which this happens. There again, while beneficial for a first approximation of physical security, the current models come with significant abstractions. In particular, the deterministic k-fault model cannot capture the imprecise nature of physical attacks due to the probabilistic fault behavior, while the random fault model fails to precisely model low-level physical characteristics of fault injection due to the same independent probability for each fault. Naturally, models for CA combine the capabilities of the individual models. All those models allow the analysis of certain types of secret-dependent leakages and enable pre-silicon evaluation.

Contribution. In this work, we provide a generalization of the probabilistic models for physical attacks by considering arbitrary leakage and fault distributions. In particular, we introduce a *general random probing model*, a *general random fault model*, and a *general random combined model* in Sect. 3, where the occurrence of a probe and a fault is not restricted to the same independent probability and instead arbitrary distributions can be chosen to model low-level circuit and noise effects. All our models, but especially our random combined model, are analyzed for interesting properties. Specifically, we analyze the impact of the adversarial knowledge on the injected faults on combined security. Since the resulting analysis complexity is very high, we introduce notions for compositions for all three models in Sect. 4. For our general random probing model, this is a straightforward extension of an existing notion for the ϵ-random probing model (details are given in the extended version [9]) For the other two models, to the best of our knowledge, this is the first attempt at composition under probabilistic leakage and fault behavior. We then continue to investigate the formal verification of the proposed combined composition notions for gadgets. Specifically, we introduce two methods of tool-based analysis using VERICA [38] and IronMask [10]. We explain how to extend these tools to verify the proposed notions and present the implementation details in Sect. 5. In particular, our extension of VERICA can analyze arbitrary circuits with exact precision (up to some threshold) but is comparably slow. In contrast, our extension of IronMask is much faster but restricted to (N)LR gadgets [10] with intermediate corrections and performs further approximations. We finally provide an extensive evaluation and comparison of the tools in Sect. 6.

2 Preliminaries

In the following, we give a short overview of the used notation and circuit model. Afterwards, we provide the basic concepts required to understand the contribution of this paper. Specifically, we introduce countermeasures for SCA and FIA and discuss security proofs via simulation.

2.1 Notation

Throughout the paper, we use a sans-serif font for functions (e.g., f) and an upper-case calligraphic font for sets (e.g., $\mathcal{S}$). We designate $\overline{\mathcal{S}}$ the complement of a set $\mathcal{S}$, $|\mathcal{S}|$ its cardinality, and $\mathbb{D}\mathcal{S}$ a discrete probability distribution defined over the event set $\mathcal{S}$. Further, we name $\prod_i \mathbb{D}\mathcal{S}_i$ the joint probability distribution of independent distributions $\mathbb{D}\mathcal{S}_i$ where the joint probability is computed as the multiplication of the individual probabilities, i.e., $\mathsf{Pr}[a \cap b] = \mathsf{Pr}[a] \cdot \mathsf{Pr}[b]$, and with $\equiv$ the equality of distributions. To simplify the notation for n-times replication we denote with $\mathbb{V}_n = \{(0)^n, (1)^n\}$ the set that contains the zero vector $(0)^n$ and the one vector $(1)^n$ of size n. Other notations will be introduced throughout the paper where necessary.

2.2 Circuit Model

In this work, we model a circuit C as a *Directed Acyclic Graph (DAG)* $\mathcal{C} = \{\mathcal{G}, \mathcal{W}\}$, where vertices $g \in \mathcal{G}$ represent logical gates and edges $w \in \mathcal{W}$ represent wires carrying a Boolean value and connecting individual gates. We restrict the set of combinational gates to $\mathcal{G}_{\mathsf{c}} = \{\mathsf{inv}, \mathsf{and}, \mathsf{xor}, \mathsf{or}\}$ and the set of memory gates to $\mathcal{G}_{\mathsf{m}} = \{\mathsf{reg}\}$. Further, we define a set of input and output gates $\mathcal{G}_{\mathsf{io}} = \{\mathsf{in}, \mathsf{out}\}$, where in produces and out absorbs a Boolean value, and a set of probabilistic gates $\mathcal{G}_{\mathsf{rand}} = \{\mathsf{rand}\}$, where rand produces a uniform-random Boolean value. Finally, we define the set of constant gates $\mathcal{G}_{\mathsf{const}} = \{\mathsf{zero}, \mathsf{one}\}$, where each gate produces the respective Boolean value. Hence, each gate is from the set $\mathcal{G}_{\mathsf{all}} = \mathcal{G}_{\mathsf{c}} \cup \mathcal{G}_{\mathsf{m}} \cup \mathcal{G}_{\mathsf{io}} \cup \mathcal{G}_{\mathsf{rand}} \cup \mathcal{G}_{\mathsf{const}}$. Given this, each gate implements a deterministic or probabilistic Boolean function $\mathsf{f_g} : \mathbb{F}_2^h \to \mathbb{F}_2$, with $0 \leq h \leq 2$, of the respective functionality. Further, let $\mathcal{G}_{\mathsf{f}} = \{\mathbb{F}_2^h \to \mathbb{F}_2 \mid h \leq 2\}$ be the set of all possible Boolean functions for this purpose. With respect to the fan-out of wires, we consider two different scenarios depending on the analysis of hardware or software. For hardware (cf. Sect. 5.2), we create a copy of the wire for each gate that is connected, i.e., for a wire with fan-out n we create n copies. Here a gate output can be connected to any number of copies of the same wire. For software (cf. Sect. 5.3), we say that each gate can have at most one output and introduce a special copy operation that outputs the input twice. Hence, to represent a wire with fan-out n we require $2n - 1$ wires to construct a tree of copys.

2.3 Countermeasures

Masking. A popular countermeasure against SCA is *Boolean masking* [13,27], due to its sound formal foundation. The core idea is to split a secret $x \in \mathbb{F}_2$ into a vector $\langle x_0, \ldots, x_{s-1} \rangle \in \mathbb{F}_2^s$, with $x_i \in \mathbb{F}_2$, such that $x = \bigoplus_{i=0}^{s-1} x_i$ and each subset $\{x_i \mid i \in [0, s-1]\}$ with cardinality smaller than s is statistically independent of x. We refer to a component x_i as a *share* of x with *share index* i. Similarly, a circuit is transformed to a *masked circuit*, which operates over shares of the inputs. In this paper, we assume the initial encoding enc of inputs and the final decoding dec of outputs (generating shares or recreating the secret, respectively) not as part of the masked circuit.

Replication. A popular countermeasure against FIA is the replication of the circuit in combination with a majority function for error correction purposes. In particular, a value $x \in \mathbb{F}_2$ is replicated to a vector $\langle x_0, \ldots, x_{n-1} \rangle \in \mathbb{F}_2^n$ with $n = 2k+1$, such that $\forall i, j \in [0, n-1] : x_i = x_j$. Then, up to k faults can be corrected with a majority function maj. Likewise, a circuit is replicated n times, where every replication operates on a unique set of value replications.

2.4 Security Proofs via Simulation

Security proofs in the context of SCA are often conducted based on *simulation*. For this, two worlds are introduced. The first world represents a real implementation, while the second world is made trivially secure by removing the secret that an adversary tries to learn. If an adversary is not able to distinguish between the two worlds then the view of the adversary is proven to be independent of the secret. The proof works by construction of a simulator that recreates the distribution of the observed values without access to any secret.

However, in this work, we do not require perfect simulation but allow the simulator to be wrong with a small probability. In particular, we require a simulator to create a distribution that is ϵ close to the observed distribution. We say that any two probability distributions $\mathbb{D}_1$ and $\mathbb{D}_2$ are ϵ-close ($\mathbb{D}_1 \approxeq_\epsilon \mathbb{D}_2$) if their statistical distance is upper-bounded by ϵ, i.e.,

$$\frac{1}{2} \sum_x |\mathsf{Pr}_{\mathbb{D}_1}[x] - \mathsf{Pr}_{\mathbb{D}_2}[x]| \leq \epsilon.$$

3 Security Model

We start our contribution by defining probabilistic security models for SCA, FIA, and CA. For this, we build on existing models but capture a more general attack scenario.

3.1 General Random Probing Security

Adversary Model. We introduce a new generalization of the random probing model [2,8,31] called *general random probing model*. Here, a probing adversary $\mathcal{A}_p$ can invoke a circuit C multiple times and on each invocation, the exact values of a random subset of wires in C are leaked to $\mathcal{A}_p$. We denote the leaking combination of wires, i.e., the subset of the wires of the circuit that is given to $\mathcal{A}_p$, with $\tilde{\mathcal{W}} \subseteq \mathcal{W}$. Let $\mathcal{W}^\infty = \{\tilde{\mathcal{W}} \subseteq \mathcal{W}\}$ be the set of all wire combinations in C and $\mathbb{D}\mathcal{W}^\infty$ an arbitrary discrete probability distribution defined over $\mathcal{W}^\infty$. Further, we define the following two functions to first select a random element from $\mathcal{W}^\infty$ and then determine the values carried by the selected wires for a given input:

$\texttt{LeakingWires}(C, \mathbb{D}\mathcal{W}^\infty)$: The leaking-wire sampler selects for a given circuit C a wire combination $\tilde{\mathcal{W}}$ with probability $\mathsf{Pr}_{\mathbb{D}\mathcal{W}^\infty}[\tilde{\mathcal{W}}]$.

$\texttt{AssignWires}(C, \tilde{\mathcal{W}}, x)$: The assign-wire sampler takes a fixed input x for C and outputs the values assigned to the wires $w \in \tilde{\mathcal{W}}$ as a tuple in $\mathbb{F}_2^{|\tilde{\mathcal{W}}|}$. If C is probabilistic so is $\texttt{AssignWires}()$.

Then, the view of $\mathcal{A}_p$ is formally defined as the *random probing leakage* $\mathcal{L}_{\tilde{\mathcal{W}}}(C, x)$, which is given by the random experiment

$$\tilde{\mathcal{W}} \leftarrow \texttt{LeakingWires}(C, \mathbb{D}\mathcal{W}^{\infty}),$$
$$\mathcal{L}_{\tilde{\mathcal{W}}}(C, x) \leftarrow \texttt{AssignWires}(C, \tilde{\mathcal{W}}, x).$$

Security Definition. Intuitively, a circuit C is secure in our model if the view of $\mathcal{A}_p$ can be simulated with high probability without access to the secret, i.e., there exists a simulator Sim that recreates the distribution of the leaking wires $\tilde{\mathcal{W}}$ without knowledge of the secret, such that the failure probability of Sim is bounded by some (small) ϵ. A more formal definition is given in Definition 1. Throughout this work, we consider enc to be some encoding function, e.g., defined by Boolean masking (cf. Sect. 2.3).

Definition 1 (General Random Probing Security). *A circuit C is said to be* $(\mathbb{D}\mathcal{W}^{\infty}, \epsilon)$-random probing secure *with respect to an encoding* enc *if there exists a simulator* Sim *such that for all inputs x:*

$$\mathsf{Sim}(C, \tilde{\mathcal{W}}) \approx_{\epsilon} \mathcal{L}_{\tilde{\mathcal{W}}}(C, \mathsf{enc}(x)).$$

The required simulator Sim can be constructed by returning a *simulation failure* $\perp$ whenever the exact distribution $\mathcal{L}_{\tilde{\mathcal{W}}}(C, \mathsf{enc}(x))$ cannot be recreated without access to x and ensuring that

$$\Pr[\mathsf{Sim}(C, \tilde{\mathcal{W}}) = \perp] = \epsilon,$$

and, conditioned to the event $\mathsf{Sim}(C, \tilde{\mathcal{W}}) \neq \perp$,

$$\mathsf{Sim}(C, \tilde{\mathcal{W}}) \equiv \mathcal{L}_{\tilde{\mathcal{W}}}(C, \mathsf{enc}(x)).$$

Relation to Random Probing Model. The *random probing model* [2,8,31] is a specific instance of the above-defined general random probing model. Specifically, the probability distribution $\mathbb{D}\mathcal{W}^{\infty}$ is selected such that each wire $w \in \mathcal{W}$ leaks with the same probability p independent of all other wires.

As shown by Duc et *al.* [20], the random probing model can be seen as an intermediate model between the noisy leakage model [36] and the d-threshold probing model [31]. Here, the assumption of the mutually independent leakage probability p for all wires can be traced to the assumption of equal and mutually independent noise in the noisy leakage model. However, the latter does not necessarily hold in practice, neither in hardware nor in software [7]. By defining the leakage over an arbitrary discrete probability distribution $\mathbb{D}\mathcal{W}^{\infty}$, we can model scenarios where the noise of two wires is not independent (e.g., because they operate in parallel with the same background computation) and, hence, the leakage probability of the two wires is dependent. Similarly, dependencies in the leakage probability can also be caused by physical defaults such as glitches [34,37] and couplings [15,37] or by shared structures like the Power Distribution Network (PDN) [45].

The introduction of the arbitrary discrete probability distribution $\mathbb{D}\mathcal{W}^\infty$ also allows two different wire combinations of one wire $\{w_i\}$ and $\{w_j\}$ to have different leakage probabilities p_i and p_j, respectively. With that, it enables more fine-grained modeling for the contribution of individual wires to the occurring leakage of wire combinations (by a different weight for the wires w_i and w_j to occur). For example, an EM probe usually does not capture the entire circuit but only a subset of neighboring circuit elements. Hence, a subset of wires does not contribute to the leakage at all and can be modeled by wire combinations with zero probability. Modeling this correctly can lead to place-and-route algorithms that take EM probing into account and minimize the observable leakage. Other differences in the leakage probability may be caused by the difference in the driver strengths of individual gates, or even by the type of operation a wire is used for.

By introducing this general model, we hope to fuel research into the nature of physical leakage by looking specifically into the leakage dependencies and contributions of different circuit structures. However, due to the computational blow-up, we will stick to mutually independent leakage probabilities in the practical implementation of our evaluation tools (cf. Sect. 5).

3.2 General Random Fault Security

Adversary Model. We use the adversary model proposed by Feldtkeller et *al.* [24] that is based on a fault model from Richter-Brockmann et *al.* [40]. Here, a faulting adversary can invoke a circuit C multiple times, where on each invocation, a random subset of gates is manipulated according to a specified fault transformation. More specifically, a fault consists of a *fault location* $g \in \mathcal{G}$, i.e., a gate of the circuit, and a *fault transformation* $\tau : \mathcal{G}_\mathsf{f} \to \mathcal{G}_\mathsf{f}$, i.e., a transformation of the Boolean function a gate implements, where the fault model restricts the allowed transformations. Popular fault transformations are, e.g., $\tau_\mathsf{set}(g) = \mathsf{one}$, $\tau_\mathsf{reset}(g) = \mathsf{zero}$, or $\tau_\mathsf{flip}(g) = \mathsf{inv}(g)$. Hence, a fault is a pair $f = (g, \tau)$ and we denote by $\mathcal{F}$ the set of all possible faults, by $\mathcal{F}^\infty = \{\tilde{\mathcal{F}} \subseteq \mathcal{F}\}$ the set of all possible fault combinations $\tilde{\mathcal{F}}$ with distinct locations (i.e., each $g \in \mathcal{G}$ occurs at most once in $\tilde{\mathcal{F}}$), and by $\mathbb{D}\mathcal{F}^\infty$ an arbitrary distribution defined over $\mathcal{F}^\infty$. We define the following function to select a fault combination for a single circuit invocation:

`AssignFaultGates`$(C, \mathbb{D}\mathcal{F}^\infty)$: For a given circuit C, the faulty-gate sampler selects a fault combination $\tilde{\mathcal{F}}$ with probability $\mathsf{Pr}_{\mathbb{D}\mathcal{F}^\infty}[\tilde{\mathcal{F}}]$ and outputs the modified circuit $C^{\tilde{\mathcal{F}}}$ that we refer to as a *faulty circuit*.

Note that $\tilde{\mathcal{F}}$ can be empty and, if $\mathsf{Pr}_{\mathbb{D}\mathcal{F}^\infty}[\emptyset] > 0$, then the sampler can output the original circuit $C = C^\emptyset$. For the sake of simplicity, it still falls within the definition of a faulty circuit.

The original adversary model introduced by Feldtkeller et *al.* provides the adversary with a correct and a faulty output of the circuit C. For our purposes, we define the leakage of faults by the correctness of the output, which is a conservative but popular choice for fault security [4,17,38,39]. For that, we define a decoding or correction gadget for the context of faulty circuits:

G^D : The decoding gadget realizes a function such that, given an input with at most k bit faults, outputs a corrected result.

Then, we define the leakage by the correctness of the output of the faulty circuit. More formally, we define the *random fault leakage* $\mathcal{L}_{\tilde{\mathcal{F}}}(C, x)$ as the output of the random experiment, with

$$C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{F}^{\infty}) ,$$
$$\mathcal{L}_{\tilde{\mathcal{F}}}(C) \leftarrow \begin{cases} 0 & \text{if } \forall x : C(x) = G^D(C^{\tilde{\mathcal{F}}}(x)) , \\ 1 & \text{else.} \end{cases}$$

In contrast to $\mathcal{L}_{\tilde{\mathcal{W}}}(C, \mathsf{enc}(x))$, the random fault leakage is not a distribution but, for each fault combination $\tilde{\mathcal{F}}$, a constant Boolean value. With this, we define the view of the adversary $\mathcal{A}_f$, after injecting a random fault into a circuit C, as $\mathcal{L}_{\tilde{\mathcal{F}}}(C)$. Note, that we focus on correction-based countermeasures here. However, detection-based countermeasures can be treated analogously by considering the result secure if a fault was detected correctly.

As with the general random probing model, this adversary model allows the modeling of a wide range of different adversarial capabilities [24]. For example, the set of fault combinations $\mathcal{F}^{\infty}$ can be set to register combinations with long computation paths to model clock glitches. Similarly, it can be set to a set of adjacent gates to model a laser attack. Similarly, a distribution of faults $\mathbb{D}\mathcal{F}^{\infty}$ with a small variance can be used to model an adversary with precise faulting capabilities, while a broader distribution can be used for a more dispersed fault behavior.

Security Definition. We extend the above adversary models by providing an appropriate definition for security. Intuitively, we say a fault combination leads to an insecure circuit behavior if there exists some input assignment that cannot be corrected at the output. This is a very conservative assumption, in that it assumes an adversary who can exploit every effective fault at the output to gain full knowledge of the secret, and is popular in the literature [4,17,38,39]. To account for the random behavior of our adversary model, we say a circuit is *random fault secure* if the probability that the adversary will get exploitable information is bounded by some (small) μ.

Definition 2 (General Random Fault Security). *A circuit C is $(\mathbb{D}\mathcal{F}^\infty, \mu)$-random fault secure with respect to a decoding G^D if:*

$$\mathsf{Pr}[\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 1] \leq \mu\,,$$

where $\mathcal{L}_{\tilde{\mathcal{F}}}()$ is computed from the random experiment

$$C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{F}^\infty).$$

In this definition, the decoding gadget required for the computation of the fault leakage $\mathcal{L}_{\tilde{\mathcal{F}}}(C)$ cannot be faulted by $\mathcal{A}_f$. This is symmetric to the encoding enc in random probing security, which is not probed, and can be justified by the fact that a fault in a final correction can only leak the output of the circuit. Note, however, that any correction implemented within the circuit is subject to faults in our model.

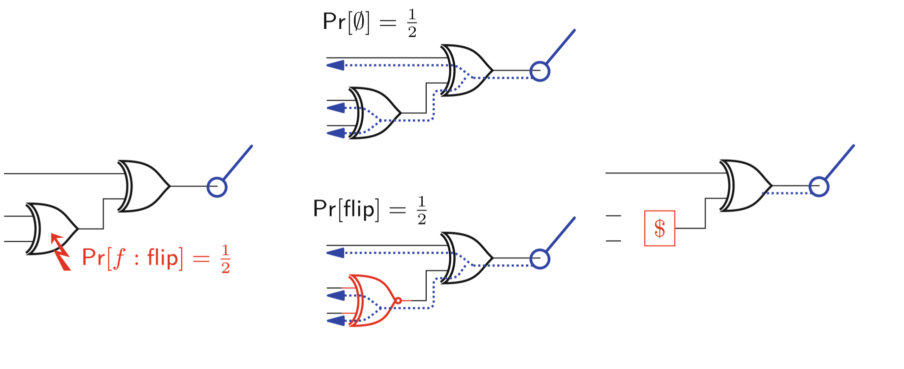
Relation to Random Fault Model. Dhooghe and Nikova proposed in [18] a *random fault model* that is inspired by the random probing model. Note that their model differs from our proposal in two key features: (i) They consider faults in wires allowing only the fault transformations set, reset, and flip. In contrast, we model faults by a transformation of gate functions which allows a wide range of possible fault scenarios, including set, reset, and flip [40]. (ii) They consider an adversary that has precise control over the location of the fault where the occurrence of the fault is randomly determined by an independent probability κ. In contrast, we allow an arbitrary distribution over fault combinations. Hence, different fault locations can be dependent and the fault type can be uncertain as well. This allows a wide range of possible adversarial scenarios [24]. Therefore, our new model is a generalization of the previously proposed random fault model.

3.3 General Random Combined Security

Adversary Model. We now introduce a model for an adversary that can both inject faults and place probes simultaneously. As such, the resulting *combined* adversary will have the capability to manipulate a random set of gates of a circuit. Then, both the exact values of a random subset of wires and the correctness of the circuit are leaked to the adversary. To formally capture the view of the adversary, we define the *random combined leakage* $\mathcal{L}_{\tilde{\mathcal{W}},\tilde{\mathcal{F}}}(C, x)$ as the output of the random experiment

$$C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{F}^\infty)\,,$$
$$\tilde{\mathcal{W}} \leftarrow \texttt{LeakingWires}(C^{\tilde{\mathcal{F}}}, \mathbb{D}\mathcal{W}^\infty)\,,$$
$$\mathcal{L}_{\tilde{\mathcal{W}},\tilde{\mathcal{F}}}(C, x) \leftarrow \texttt{AssignWires}(C^{\tilde{\mathcal{F}}}, \tilde{\mathcal{W}}, x) \parallel \mathcal{L}_{\tilde{\mathcal{F}}}(C)\,.$$

Again, we assume a correction-based countermeasure for simplicity. When detection is used, it is important to include the detection flag in the random combined leakage (which then needs to be simulated alongside the leaking wires).



(a) Fault scenario with a probabilistic flip fault. **(b)** View of the simulator with known faulty circuit. **(c)** View of the simulator with known fault distribution $\mathbb{D}\mathcal{F}^{\infty}$.

Fig. 1. Difference when knowing the injected fault versus knowing only the fault distribution. When knowing the fault the circuit is deterministic (for deterministic inputs) and a probe on the output propagates to all inputs. When only the fault distribution is known, the circuit is probabilistic (even with deterministic inputs) stopping the propagation of probes.

Considering the SCA aspect of CA, we require for security that the leaking wires $\tilde{\mathcal{W}}$ can be simulated without knowledge of the secret. However, for simulation, the adversarial knowledge of the faulty circuit makes a difference. Imagine the scenario in Fig. 1, where a flip fault is injected into a xor chain with probability $\frac{1}{2}$. If the adversary knows the faulty circuit $C^{\tilde{\mathcal{F}}}$ (and, hence, $C^{\tilde{\mathcal{F}}}$ is given to the simulator) then all deterministic inputs to the xor chain are required to simulate the probe at the end in both cases (non-faulty/faulty). In contrast, if the adversary does not know $C^{\tilde{\mathcal{F}}}$ (and only the fault distribution $\mathbb{D}\tilde{\mathcal{F}}^{\infty}$ is given to the simulator) then the fault randomizes the intermediate value and the output of the final xor can be simulated by a uniform random value. Hence, the fault effectively works as a mask refreshing. For this, we introduce two different combined adversaries: one without and one with knowledge of the faulty circuit.

i) Unknown-Fault Random Combined Adversary. Our first adversary $\mathcal{A}_{uc}$ is the combination of $\mathcal{A}_p$ and $\mathcal{A}_f$ without knowledge of the randomly chosen fault combination $\tilde{\mathcal{F}}$. Specifically, the adversary has no access to the faulty circuit $C^{\tilde{\mathcal{F}}}$ and the corresponding simulator receives just the circuit C and the fault distribution $\mathbb{D}\mathcal{F}^{\infty}$ as input. Hence, the view of $\mathcal{A}_{uc}$ is defined by $\mathcal{L}_{\tilde{\mathcal{W}},\tilde{\mathcal{F}}}(C, x)$ and the effects of fault injection and probing are interleaved, i.e., we analyze the circuit in Fig. 1c.

ii) Known-Fault Random Combined Adversary. The second adversary $\mathcal{A}_{kc}$ is the combination of $\mathcal{A}_p$ and $\mathcal{A}_f$ with additional knowledge of the faulty circuit $C^{\tilde{\mathcal{F}}}$ (or equivalently, the selected fault combination $\tilde{\mathcal{F}}$). For this, the corresponding simulator has access to $C^{\tilde{\mathcal{F}}}$ for the simulation of the leaking wires. With this, the view of $\mathcal{A}_{kc}$ is defined by $\mathcal{L}_{\tilde{\mathcal{W}},\tilde{\mathcal{F}}}(C, x) \parallel \tilde{\mathcal{F}}$. Since the faulty circuit is known, we can analyze $\mathcal{L}_{\tilde{\mathcal{W}}}(C^{\tilde{\mathcal{F}}}, x)$ and $\mathcal{L}_{\tilde{\mathcal{F}}}(C)$ independently, i.e., after the fault injection, we only consider one of the two circuits in Fig. 1b.

When comparing the two adversaries, it becomes apparent that $\mathcal{A}_{uc}$ is the more realistic adversary model for CA because, in a real-world circuit, the adversary usually does not know the exact effect of an injected fault. However, the uncertainty about the faults makes the analysis of combined security much more complex, due to the reciprocal effects of faults and probes. Fortunately, we can show that any circuit secure against $\mathcal{A}_{kc}$ is also secure against $\mathcal{A}_{uc}$. For this, we see $\mathcal{A}_{kc}$ mostly as a useful abstraction for analysis, allowing a clear path to security verification.

In this sense, we can make a further distinction in the knowledge an adversary has about the effect of fault injection. Specifically, we can separate the knowledge about the fault location and its effect. For this, we use the known-fault adversary $\mathcal{A}_{kc}$ in combination with a fault transformation to a probabilistic gate function f_g, i.e., f_g has an internal random tape that impacts the output of the gate. While this pushes the analysis closer to $\mathcal{A}_{uc}$, it is important to evaluate the correctness of the circuit for all values of the random tape, i.e., $\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 0$ if the output can be corrected for all inputs and random values of probabilistic gate functions.

Our model also allows us to go in the opposite direction by empowering $\mathcal{A}_{kc}$ with direct control over the injected fault. Hence, we can model a *chosen-fault* combined adversary as the adversary $\mathcal{A}_{kc}$ that can freely choose the fault distribution $\mathbb{D}\mathcal{F}^\infty$ of a (restrict) set of fault combinations $\mathcal{F}^\infty$. In most cases, this will lead to an adversary that directly chooses the location and transformation of the injected fault, since any uncertainty will reduce the advantage of the adversary, which removes the random nature of the model. Note, if the set $\mathcal{F}^\infty$ is restricted to the set of all gate combinations with up to k faults, this model is equivalent to the popular threshold fault model [1,4,30,39,46]. However, since the semantics of the security definition do not change (we do not care how the fault distribution is generated), we do not introduce a specific adversary model for this case.

Security Definition. We provide security definitions for the setting of combined adversaries. Intuitively, we say that a circuit is combined-secure if the view of the corresponding adversary can be simulated and the output can be corrected with high probability. For this, we introduce a new security parameter γ that represents the advantage of the adversary, i.e., the probability that the adversary gains some knowledge about the secret. In the combined setting, the adversary wins if either there exists some fault leakage ($\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 1$) or the leaking wires in the faulty circuit cannot be simulated without knowledge of the secret. To overcome the dependencies between those two events, we only conduct the simulation of the leaking wires if there is no fault leakage. Hence, we get the following two parameters (where we usually choose the lowest possible value):

$$\mu \geq \mathsf{Pr}[\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 1] \,,$$

$$\epsilon \geq \mathsf{Pr}[\mathsf{Sim}(C, \tilde{\mathcal{W}}) \equiv \perp | \; \mathcal{L}_{\tilde{\mathcal{F}}}(C) = 0] \,.$$

Here, μ is defined exactly as in Definition 2, while ϵ is the probability that the simulation fails knowing that $\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 0$. We then express the advantage of the combined adversary by

$$\gamma \geq \mu + (1 - \mu)\epsilon \,.$$

To compute ϵ, we introduce a new subset of the fault combination $\mathcal{B} \subseteq \mathcal{F}^{\infty}$, such that $\mathcal{B}$ captures all fault combinations that always yield a result that can be corrected, i.e., $\mathcal{B} = \{\tilde{\mathcal{F}} \in \mathcal{F}^{\infty} \mid \mathcal{L}_{\tilde{\mathcal{F}}}(C) = 0\}$. Then, we define the distribution $\mathbb{D}\mathcal{B}$ as the scaled distribution $\mathbb{D}\mathcal{F}^{\infty}$ conditioned to the event $\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 0$.

We start with the security definition for the adversary $\mathcal{A}_{uc}$, where the random combined leakage needs to be simulated. In a sense, this is the most natural definition as the adversary has no knowledge about the circuit transformation caused by the injected fault.

Definition 3 (Unknown-Fault Random Combined Security). *A circuit C is* $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma)$-Unknown-Fault Random Combined Secure ($\mathrm{RCS}_{\mathrm{UF}}$) *with respect to an value encoding* enc *and a error decoding G^D if there exists some $\mu, \epsilon \leq 1$ such that C is $(\mathbb{D}\mathcal{F}^{\infty}, \mu)$-random fault secure with respect to G^D, there exists a simulator* Sim *such that for all inputs x:*

$$\mathsf{Sim}(C, \mathbb{D}\mathcal{F}^{\infty}) \approx_{\epsilon} \mathcal{L}_{\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{B}}(C, \mathsf{enc}(x)) \qquad \textit{and} \qquad \mu + (1 - \mu)\epsilon \leq \gamma \;,$$

where $\mathcal{L}_{\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{B}}()$ is computed from the random experiment

$$\tilde{\mathcal{W}} \leftarrow \texttt{LeakingWires}(C, \mathbb{D}\mathcal{W}^{\infty}) \,,$$
$$C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{B}) \,.$$

While simple, this definition is difficult to analyze due to the interleaving of probes and faults within the simulator. Therefore, we provide a second security definition tailored to $\mathcal{A}_{kc}$. Because this adversary knows the faulty circuit $C^{\tilde{\mathcal{F}}}$, the interleaving of probes and faults is eliminated. Hence, the analysis gets simpler and we will later show that security in this model implies security with unknown faults.

Definition 4 (Known-Fault Random Combined Security). *A circuit C is* $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma)$-Known-Fault Random Combined Secure ($\mathrm{RCS}_{\mathrm{KF}}$) *with respect to an value encoding* enc *and an error decoding G^D if there exists some $\mu, \epsilon \leq 1$ such that C is $(\mathbb{D}\mathcal{F}^{\infty}, \mu)$-random fault secure with respect to G^D, there exists a simulator* Sim *such that for all inputs x it holds that*

$$C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{B}) \,,$$
$$\mathsf{Sim}(C^{\tilde{\mathcal{F}}}) \approx_{\epsilon} \mathcal{L}_{\tilde{\mathcal{W}}}(C^{\tilde{\mathcal{F}}}, x),$$

and

$$\mu + (1 - \mu)\epsilon \leq \gamma \,,$$

where $\mathcal{L}_{\tilde{\mathcal{W}}}()$ is computed from the random experiment

$$\tilde{\mathcal{W}} \leftarrow \texttt{LeakingWires}(C^{\tilde{\mathcal{F}}}, \mathbb{D}\mathcal{W}^{\infty}) \,.$$

Reduction Between Security Definitions. It is evident that the adversary $\mathcal{A}_{kc}$ has more knowledge about the probed circuit structure than $\mathcal{A}_{uc}$. Hence, it seems reasonable that RCS_{UF} is the more general security notion and we can reduce its security to RCS_{KF}. In other words, if a circuit is RCS_{KF}, then it is always RCS_{UF}.

In particular, we can show that if there exists a simulator Sim_{KF} for the adversary $\mathcal{A}_{kc}$ then we can always construct a simulator Sim_{UF} for the adversary $\mathcal{A}_{uc}$ with at most the failure probability of Sim_{KF}. For that, consider that RCS_{KF} requires the simulation for a given but randomly chosen faulty circuit $C^{\tilde{\mathcal{F}}}$. Hence, we can compute the failure probability of Sim_{KF} as the sum of the failure probability for each faulty circuit weighted by the probability of that circuit to occur, i.e.,

$$\epsilon_{kf} = \sum_{\tilde{\mathcal{F}}} \mathsf{Pr}[\tilde{\mathcal{F}}]\epsilon_{kc}^{\tilde{\mathcal{F}}}, \tag{1}$$

where $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ is the failure probability of Sim_{KF} given the faulty circuit $C^{\tilde{\mathcal{F}}}$. We can then construct a simulator that randomly selects a faulty circuit and calls Sim_{KF} for the wire simulation. Note, that standalone fault security is not affected by the choice of the adversary. We show a more formal argumentation below.

Theorem 1. *Let C be a circuit that is $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma_{kc})$-$RCS_{KF}$. Then C is $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma_{uc})$-$RCS_{UF}$ with $\gamma_{uc} \leq \gamma_{kc}$.*

Proof. Let C be a $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma_{kc})$-$\text{RCS}_{\text{KF}}$ circuit. Then, by definition of RCS_{KF}, C is $(\mathbb{D}\mathcal{F}^{\infty}, \mu)$-random fault secure and there exists a simulator Sim_{KF} with failure probability ϵ_{kc} for the probing leakage $\mathcal{L}_{\tilde{\mathcal{W}}}(C^{\tilde{\mathcal{F}}}, x)$ such that $\gamma_{kc} \geq \mu + (1-\mu)\epsilon_{kc}$.

Now we construct a simulator Sim_{UF} for RCS_{UF} out of Sim_{KF}. This is possible because Sim_{UF} has to simulate the same events as Sim_{KF} since the set of fault combinations with $\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 1$ remains untouched by the knowledge of the adversary. The simulator Sim_{UF} is constructed by the following two steps:

1. Call $C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{B})$,
2. Return $\mathsf{Sim}_{KF}(C^{\tilde{\mathcal{F}}}, \tilde{\mathcal{W}})$.

When this simulator does not fail it is a perfect simulation of the leaking wires because it draws $\tilde{\mathcal{F}}$ from the correct distribution and Sim_{KF} produces a perfect simulation for any faulty circuit $C^{\tilde{\mathcal{F}}}$ (if it does not fail). Given this, the failure probability of Sim_{UF} is given by $\epsilon_{uf} = \sum_{\tilde{\mathcal{F}}} \mathsf{Pr}[\tilde{\mathcal{F}}]\epsilon_{kc}^{\tilde{\mathcal{F}}}$, where $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ is the failure probability of Sim_{KF} given the faulty circuit $C^{\tilde{\mathcal{F}}}$. Hence, we have $\epsilon_{uc} = \epsilon_{kc}$ (see Eq. 1) for this simulator (however, there could be a simulator with a smaller failure probability). Together with $(\mathcal{F}^{\infty}, \mu)$-random fault security of C, it follows that C is $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma_{uc})$-$\text{RCS}_{\text{UF}}$ with $\gamma_{uc} \leq \gamma_{kc}$. $\square$

In the remainder of this paper, we will mostly focus on RCS_{KF} for the sake of simplicity.

Related Work on the Random Combined Model. In addition to the random fault model, Dhooghe and Nikova propose a combined version of the random probing and random fault model [18]. Their model does not leak the exact occurring fault to the adversary (similar to $\mathrm{RCS}_{\mathrm{UF}}$). However, since they use their version of the random fault model and the traditional random probing model, the combination has the same limitations as the individual models. Our model introduces a new security parameter γ that captures the overall advantage of the combined adversary. Notably, the parameters of our combined model can be chosen to align with the definitions provided by Dhooghe and Nikova. While they propose this model in an appendix, they do not perform any analysis or further investigations into it.

Related Work on the Combined Attacks. In the literature, several combined attacks have been proposed [14,41–44,48] that exploit both fault injection and side-channel leakage. These attacks leverage additional SCA leakage caused by conditional fault propagation or the disruption of masking schemes through fault injection. Our model captures these attacks, as simulating the corresponding probes is not possible without access to the secret, reflecting the inherent vulnerabilities. For instance, works [42, Section 4.2] and [44] describe combined attacks on gadgets involving logical AND operations protected by countermeasures like DOMREP [29]. The strategy typically involves injecting a fault into one share of a sensitive variable a, which is multiplied by all shares of another sensitive variable b. This causes a conditional fault propagation where effective faults propagate only if $b = 1$, and, hence, the observation of the effectiveness of the fault is sufficient to learn the secret b. With our new combined model, this attack can be captured using a single fault (set, reset, or bit-flip), and two probes placed in different replications or a single probe in the correction module that combines two replications. By probing the correction module, we can observe the fault's propagation based on whether b is zero (when the fault does not propagate) or one (when it does), enabling the extraction of sensitive information about b. In response to this vulnerability, DOMREP-II [35] was introduced, which performs a masked correction such that a single probe cannot determine the fault's effectiveness.

4 Compositional Notions

Analyzing entire circuits for their security is often prohibitively complex. As a result, the research community focuses on the construction of so-called *gadgets*, i.e., small circuits implementing a small function (often single binary operations) in a secure manner such that security is guaranteed even under composition [5,8,12,17,25,38]. To abstractly argue about the security via composition, we first define a *composability notion*, which defines the properties a gadget must fulfill, usually, by restricting the propagation of leakage or faults. Second, we outline and prove the conditions of composition, i.e., how the gadgets can be securely combined, in a *composition theorem*.

For composition in the general random probing model, we can extend a notion from the ϵ-random probing model to arbitrary leakage distributions. In particular, we extend the notion of Random Probing Composability (RPC) [8] such that there is an arbitrary but independent leakage distribution for each gadget. This independence of leakage distributions between gadgets is required to preserve the advantages of the gadget-based approach, namely the independent analysis of gadgets. The definition and compositional properties are straightforward extensions of the original statements. In particular, the random set of leaking wires is extended by a bounded set of output wires, which together are only allowed to provide information about a bounded set of input shares. Then, the failure probability ϵ is determined by the maximum failure probability over all possible leaking output combinations. We provide more details in the extended version [9] and continue with composition in the general random fault model.

4.1 Composition in the Random Fault Model

In contrast to the random probing model, to the best of our knowledge, there exists no prior work on composition in the random fault model. Due to the duality of probes and faults, we can adapt the notation from the random probing to the random fault case. However, instead of adding probes on outputs we now consider additional faults on inputs and check for correctness not simulation.

To properly argue about the composition we need to specify the used countermeasure (similar to masking for probing). Hence, in the following, we will only consider circuits secured by simple repetition against a fault adversary. In particular, we will use $2k + 1$ repetitions such that a majority vote can be used for the correction of up to k faults. Then, for composition, we consider an adversary that can inject a random fault combination into the gadget where additionally a tuple $\mathcal{I}'$ of arbitrarily but bounded sets of inputs are potentially manipulated by faults. More specifically, for each input with index i, there is a bounded set $\mathcal{I}'_i$ that contains the replication indices of inputs that may be affected by a fault. Here, we consider each replication in $\mathcal{I}'_i$ to be a unique and independent input, to account for all possible distribution changes due to a fault in previous parts of the composed circuit. Hence, a fault on an input wire can be seen as an arbitrary change in the value distribution over $\mathbb{F}_2$. Then, a gadget is composable, if the output can be corrected with high probability.

To formally define our notion of composition in the presence of a random fault adversary, we start by defining Restricted Random Fault Composability (RRFC), which states the required property for a fixed tuple of input faults $\mathcal{I}'$.

Definition 5 (Restricted Random Fault Composability). *Let $n = 2k+1$ and $\mathcal{I}' = (\mathcal{I}'_i)_{i \in [h]}$ be a tuple of sets such that $\mathcal{I}'_i \subseteq [n]$, for all i: $|\mathcal{I}'_i| \leq k$. A gadget $G : (\mathbb{F}_2^n)^h \to (\mathbb{F}_2^n)^m$ is $(\mathcal{I}', k, \mathbb{D}\mathcal{F}^\infty, \mu)$-RRFC if there exists a deterministic algorithm* `ReplicationSelect` *such that the random experiment*

$$G^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(G, \mathbb{D}\mathcal{F}^{\infty}),$$
$$\mathcal{O}' := (\mathcal{O}'_1, \dots, \mathcal{O}'_m) \leftarrow \texttt{ReplicationSelect}(G^{\tilde{\mathcal{F}}}, \mathcal{I}')$$
$$\mathcal{L}_{\mathcal{I}',\tilde{\mathcal{F}}}(G) \leftarrow \begin{cases} 0 & \textit{if } \forall i \leq m : |\mathcal{O}'_i| \leq k\,, \\ 1 & \textit{else.} \end{cases}$$

yields

$$\mathsf{Pr}[\mathcal{L}_{\mathcal{I}',\tilde{\mathcal{F}}}(G) = 1] \leq \mu$$
$$G^{\tilde{\mathcal{F}}}(x')|_{\overline{\mathcal{O}'}} \equiv G(x)|_{\overline{\mathcal{O}'}}$$

for all inputs $x \in \mathbb{V}_n^h$ and faulty inputs $x' \in (\mathbb{F}_2^n)^h$ with $x|_{\overline{\mathcal{I}'}} = x'|_{\overline{\mathcal{I}'}}$.

Then, a gadget supports Random Fault Composability (RFC) with some bound μ if for all possible tuples $\mathcal{I}'$ the RRFC failure probability is bounded by μ.

Definition 6 (Random Fault Composability). *A gadget $G : (\mathbb{F}_2^n)^h \to (\mathbb{F}_2^n)^m$ with $n = 2k+1$ is $(k, \mathbb{D}\mathcal{F}^{\infty}, \mu)$-RFC if for all tuples of sets $\mathcal{I}' = (\mathcal{I}'_i)_{i \in [h]}$ with $\forall i : \mathcal{I}'_i \subseteq [n]$ and $|\mathcal{I}'_i| \leq k$ the gadget G is $(\mathcal{I}', k, \mathbb{D}\mathcal{F}^{\infty}, \mu_{\mathcal{I}'})$-RRFC and* $\max_{\mathcal{I}'}\{\mu_{\mathcal{I}'}\} \leq \mu$.

While we split the definition of RFC into two parts (for reasons that become apparent when we look at composition under a combined adversary) it is easy to see that there is a close symmetry with the definition of General Random Probing Composability (GRPC). Where GRPC goes through all possible tuples of output probes $\mathcal{O}$, RFC goes through all possible tuples of input faults. Where GRPC restricts the number of shares per input the simulator can use for a successful simulation, RFC restricts the number of replications per output that can be affected by a fault.

We now show that gadgets supporting RFC can be composed arbitrarily. Under the assumption of mutually independent fault distributions for each gadget, the failure probability increases linearly with the number of gadgets in the circuit.

Theorem 2. *Let C be a circuit constructed by composition of $(k, \mathbb{D}\mathcal{F}_i^{\infty}, \mu_i)$-RFC gadgets G_i, for $i \in \{1, \dots, |C|\}$. Then, C is $(\prod_{i=1}^{|C|} \mathbb{D}\mathcal{F}_i^{\infty}, 1 - \prod_{i=1}^{|C|}(1-\mu_i))$-random fault secure.*

Proof. Let the faulty circuit be $C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \prod_{i=1}^{|C|} \mathbb{D}\mathcal{F}_i^{\infty})$, with $\tilde{\mathcal{F}}$ the set of faults selected with $\mathsf{Pr}_{\prod_i \mathbb{D}\mathcal{F}_i^{\infty}}[\tilde{\mathcal{F}}]$. We can divide $\tilde{\mathcal{F}}$ into $|C|$ disjoint fault sets $\tilde{\mathcal{F}}_i \subseteq \tilde{\mathcal{F}}$ such that $\tilde{\mathcal{F}}_i$ belongs to G_i and is selected with $\mathsf{Pr}_{\mathbb{D}\mathcal{F}_i^{\infty}}[\tilde{\mathcal{F}}_i]$.

We go iteratively through the gadgets, starting with the gadgets only connected to the inputs of C. Let G_i be such a gadget. Then, by definition of $(k, \mathbb{D}\mathcal{F}_i^{\infty}, \mu_i)$-RFC, the gadget G_i is $(\mathcal{I}'_{G_i}, k, \mathbb{D}\mathcal{F}_i^{\infty}, \mu_i)$-RRFC for the tuple of empty sets $\mathcal{I}'_{G_i} = (\mathcal{I}'_{G_i,j} = \emptyset)_{j \in [h]}$. Hence, there exists a tuple of sets $\mathcal{O}'_{G_i}$ such that $\mathsf{Pr}[\mathcal{L}_{\mathcal{I}'_{G_i},\tilde{\mathcal{F}}}(G) = 1] \leq \mu$ and $G_i^{\tilde{\mathcal{F}}_i}(x)|_{\overline{\mathcal{O}'_{G_i}}} \equiv G_i(x)|_{\overline{\mathcal{O}'_{G_i}}}$.

We continue with the child gadgets, i.e., gadgets that have inputs connected to outputs of the just handled gadgets and (potentially) inputs of the circuit. Let G_j be such a gadget. Again, we can create a tuple of sets $\mathcal{I}'_{G_j}$ out of the respective tuples $\mathcal{O}'_{G_i}$ of the parent gadgets G_i. Because of $(k, \mathbb{DF}_j^{\infty}, \mu_j)$-RFC there exists tuple of sets $\mathcal{O}'_{G_j}$ such that $\mathsf{Pr}[\mathcal{L}_{\mathcal{I}'_{G_j}, \tilde{\mathcal{F}}}(G) = 1] \leq \mu$ and $G_j^{\tilde{\mathcal{F}}_j}(x)|_{\overline{\mathcal{O}'_{G_j}}} \equiv G_j(x)|_{\overline{\mathcal{O}'_{G_j}}}$. We repeat this process until we reach the outputs of C.

Since we have $n = 2k+1$ replications, we can construct a decoding gadget G^D for C by computing the majority of the output wires $w_i^1, \ldots, w_i^n$ for all $i \in [m]$. This decoding gadget will correct an output value as long as the number of faulty replications is smaller or equal to k, which is always true if none of the gadgets G_i, for $i = 1, \ldots |C|$, fail with respect to RFC. The probability that at least one of $|C|$ gadgets fail is $1 - \prod_{i=1}^{|C|}(1-\mu_i)$. Hence, $\mathsf{Pr}[C(x) \equiv G^D(C^{\tilde{\mathcal{F}}}(x))] \leq 1 - \prod_{i=1}^{|C|}(1-\mu_i)$, which shows random fault security of C. $\square$

Note, that the above composition loses some tightness in μ because a failure of RFC in one gadget does not necessarily mean that the entire circuit is insecure, e.g., if some faults cancel each other out in a later gadget.

Relation to Fault-Secure Composition. To the best of our knowledge, this is the first work establishing a compositional property in the random fault model. However, in the threshold fault model [30], where an adversary can place up to k faults, composition is already discussed and notions usually have a strong symmetry to notions in the Ishai-Sahai-Wagner (ISW) probing model. In particular, (Strong) Non-Accumulation ((S)NA) [17] (later refined to Fault (Strong) Non-Interference (F-(S)NI) [38]) restricts the number of faults that can propagate to the output of a gadget, while Fault-Isolating Non-Interference (FINI) [25] limits fault propagation within so-called redundancy domains. Both variants ensure that the number of faults in the replication of a single value does not exceed the threshold that allows correction (detection) of faults. In this sense, our proposed notion has some similarity with (S)NA/F-(S)NI, in that it restricts the number of output replications that are allowed to be affected by a fault. However, similar to the contrast between (G)RPC and (Strong) Non-Interference ((S)NI), the number of allowed faulty outputs is not dependent on the amount of injected faults. Also, we use the same symmetry between faults and probes for the definition of our notion of composition.

4.2 Composition in the Random Combined Model

Finally, we provide a compositional statement for the random combined model, i.e., under an adversary with both faulting and probing capabilities. Here, we consider composition under an adversary that knows the injected fault and use the reduction from Sect. 3.3 for the adversary with unknown faults. We leave the tighter compositional statement in the setting with unknown faults for future work. In principle, the following notion is a combination of the two notions for the individual cases, however, with subtle differences.

First, we count faults in inputs and outputs per input and output share. More specifically, the tuple of indices for potentially faulty inputs has now a set for each share of each input, i.e., $\mathcal{I}' = ((\mathcal{I}'_{i,j})_{i\in[s]})_{j\in[h]}$, where j is the index of the input and i the share index. The same holds for the tuple of potential faulty outputs $\mathcal{O}' = ((\mathcal{O}'_{i,j})_{i\in[s]})_{j\in[m]}$, which is constructed by `ReplicationSelect`. Hence, we bound the number of allowed faults per input and output share by k.

Second, and in contrast, probes on outputs are extended to all replications of the probed value[1]. Hence, the tuple $\mathcal{O} = (\mathcal{O}_i)_{i\in[m]}$, where each $\mathcal{O}_i$ contains the probed share indices of the i'th output, does not change. However, the meaning of $j \in \mathcal{O}_i$ changes in so far as now all replications of the j'th share of the i'th output need to be simulated. This is necessary to allow for a wide range of gadget implementations where there are potential interdependencies between different replications, e.g., via a correction module, that allows probe propagation across replications [23]. Similarly, the simulator gets access to all replications of the shares indicated in the tuple $\mathcal{I} = (\mathcal{I}_i)_{i\in[h]}$.

Third, we define the failure probability for probe simulation $\epsilon_{\mathcal{I}'}$ in dependence on the tuple of faulty-input indices $\mathcal{I}'$. This is analogous to the definition of RRFC (Definition 5) and allows us to iterate over all possible tuples $\mathcal{I}'$ for our combined composition. In particular, this enables a precise definition of the conditions for Random Probing Composition under Faults (RPCUF) under a given tuple of potentially faulty inputs $\mathcal{I}'$ and a random fault combination $\tilde{\mathcal{F}} \in \mathcal{F}^\infty$.

Definition 7 (Random Probing Composition under Faults). *Let $\mathcal{I}' = ((\mathcal{I}'_{i,j})_{i\in[s]})_{j\in[h]}$ be a tuple such that $\forall i,j : \mathcal{I}'_{i,j} \subseteq [n]$ and $|\mathcal{I}'_{i,j}| \leq k$. A gadget $G : ((\mathbb{F}_2^n)^s)^h \to ((\mathbb{F}_2^n)^s)^m$ with $n = 2k+1$ is $(\mathcal{I}', \mathbb{D}\mathcal{F}^\infty, d, \mathbb{D}\mathcal{W}^\infty, \epsilon)$-RPCUF if there exists a deterministic algorithm* `ShareSelect` *and a probabilistic simulator* Sim *such that for all faulty inputs $x' \in ((\mathbb{F}_2^n)^s)^h$, for which there exists an $x \in (\mathbb{V}_n^s)^h$ with $x|_{\overline{\mathcal{I}'}} = x'|_{\overline{\mathcal{I}'}}$, and every tuple of sets $\mathcal{O} = (\mathcal{O}_1, \ldots, \mathcal{O}_m)$, with $\forall i : \mathcal{O}_i \subseteq [s]$ and $|\mathcal{O}_i| \leq d$, the random experiment*

$$G^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(G, \mathbb{D}\mathcal{F}^\infty)$$
$$\tilde{\mathcal{W}} \leftarrow \texttt{LeakingWires}(G^{\tilde{\mathcal{F}}}, \mathbb{D}\mathcal{W}^\infty)$$
$$\mathcal{I} := (\mathcal{I}_1 \ldots \mathcal{I}_h) \leftarrow \texttt{ShareSelect}(\tilde{\mathcal{W}}, \mathcal{O})$$
$$out \leftarrow \mathsf{Sim}(x'|_{\mathcal{I}}, \mathcal{I}, \mathcal{I}', \mathcal{O}, \tilde{\mathcal{W}})$$

yields

$$\mathsf{Pr}[(|\mathcal{I}_1| > d) \vee \ldots \vee (|\mathcal{I}_h| > d)] \leq \epsilon$$
$$out \equiv (\texttt{AssignWires}(G^{\tilde{\mathcal{F}}}, \tilde{\mathcal{W}}, x'), y'|_{\mathcal{O}})$$

with $y' \leftarrow G^{\tilde{\mathcal{F}}}(x')$.

[1] The same procedure should be used when analyzing replicated circuits for stand-alone GRPC.

Fourth, and similar to Sect. 3.3, we only check for side-channel security if the gadget is fault secure to keep the failure probabilities for probe simulation $\epsilon_{\mathcal{I}'}$ and correctness $\mu_{\mathcal{I}'}$ independent. For this, we again introduce a subset of fault combinations $\mathcal{B}_{\mathcal{I}'} \subseteq \mathcal{F}^{\infty}$, such that $\mathcal{B}_{\mathcal{I}'}$ captures all fault combinations that, in combination with potential faults on the inputs with indices in $\mathcal{I}'$, lead to a gadget output that can be corrected, i.e., $\mathcal{B}_{\mathcal{I}'} = \{\tilde{\mathcal{F}} \in \mathcal{F}^{\infty} \mid \mathcal{L}_{\mathcal{I}',\tilde{\mathcal{F}}}(G) = 0\}$. Note, that we define this set to be dependent on the tuple $\mathcal{I}'$. The corresponding distribution $\mathbb{D}\mathcal{B}_{\mathcal{I}'}$ is defined as the scaled distribution $\mathbb{D}\mathcal{F}^{\infty}$ conditioned on the event $\mathcal{L}_{\mathcal{I}',\tilde{\mathcal{F}}}(G) = 0$.

With this, we say that a gadget supports Known-Fault Random Combined Composability (RCC$_{\text{KF}}$) if, for any tuple $\mathcal{I}'$, the gadget supports RRFC and RPCUF with negligible advantage for the adversary. To again compute a unified failure probability we chose the maximum combined failure probability (i.e., the probability that either RRFC fails or RPCUF fails under the condition that RRFC holds) over all tuples $\mathcal{I}'$.

Definition 8 (Known-Fault Random Combined Composability). *A gadget $G : ((\mathbb{F}_2^n)^s)^h \rightarrow ((\mathbb{F}_2^n)^s)^m$ is $(d, k, \mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma)$-RCC$_{KF}$ if for all tuples of sets $\mathcal{I}' = ((\mathcal{I}'_{i,j})_{i\in[s]})_{j\in[h]}$, such that $\forall i, j : \mathcal{I}'_{i,j} \subseteq [n]$ and $|\mathcal{I}'_{i,j}| \leq k$, there exists some $\mu_{\mathcal{I}'}, \epsilon_{\mathcal{I}'} \leq 1$ such that the gadget G is $(\mathcal{I}', k, \mathbb{D}\mathcal{F}^{\infty}, \mu_{\mathcal{I}'})$-RRFC and $(\mathcal{I}', \mathbb{D}\mathcal{B}_{\mathcal{I}'}, d, \mathbb{D}\mathcal{W}^{\infty}, \epsilon_{\mathcal{I}'})$-RPCUF and it holds that $\max_{\mathcal{I}'}\{\mu_{\mathcal{I}'} + (1-\mu_{\mathcal{I}'})\epsilon_{\mathcal{I}'}\} \leq \gamma$.*

Under this definition, we can compose any RCC$_{\text{KF}}$ gadgets as long as each output of a gadget is only used once as input to another gadget. This restriction comes from the composition under probes. Then, under the assumption of mutually independent fault and probing distributions per gadget, the combined failure probability increases linearly in the number of gadgets in the circuit.

Theorem 3. *Let C be a circuit constructed by composition of gadgets G that are $(d, k, \mathbb{D}\mathcal{W}_i^{\infty}, \mathbb{D}\mathcal{F}_i^{\infty}, \gamma_i)$-RCC$_{KF}$, for $i \in \{1, \ldots, |C|\}$, such that each output of G_i is used as input of at most one other gadget G_j or as output of C. Then, C is*

$$(\prod_{i=1}^{|C|} \mathbb{D}\mathcal{W}_i^{\infty}, \prod_{i=1}^{|C|} \mathbb{D}\mathcal{F}_i^{\infty}, 1 - \prod_{i=1}^{|C|}(1 - \gamma_i))\text{-}RCC_{KF}.$$

Intuitively, the proof follows the lines of the compositional statements for stand-alone probing and faulting. Specifically, we first go from inputs to outputs through the circuit to construct the respective tuples of potentially faulty inputs to each gadget to show composition under faults. Then, we go backward, i.e., from outputs to inputs, through the gadgets and use the fact that the set of faulty inputs is bounded in case of a fault combination that can be corrected to show random probing security under faults.

Proof. Let the faulty circuit be $C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \prod_{i=1}^{|C|} \mathbb{D}\mathcal{F}_i^{\infty})$, with $\tilde{\mathcal{F}}$ the set of faults selected with $\mathsf{Pr}_{\prod_i \mathbb{D}\mathcal{F}_i^{\infty}}[\tilde{\mathcal{F}}]$. We can divide $\tilde{\mathcal{F}}$ into $|C|$ disjoint fault sets $\tilde{\mathcal{F}}_i \subseteq \tilde{\mathcal{F}}$ such that $\tilde{\mathcal{F}}_i$ belongs to G_i and is selected with $\mathsf{Pr}_{\mathbb{D}\mathcal{F}_i^{\infty}}[\tilde{\mathcal{F}}_i]$.

Further, let $\tilde{\mathcal{W}}$ be the set of leaking wires of C selected with $\mathsf{Pr}_{\prod_i \mathbb{D}\mathcal{W}_i^{\infty}}[\tilde{\mathcal{W}}]$. We can divide $\tilde{\mathcal{W}}$ into $|C|$ disjoint parts $\tilde{\mathcal{W}}_i \subseteq \tilde{\mathcal{W}}$, each belonging to the gadget G_i such that each $\tilde{\mathcal{W}}_i$ was selected with $\mathsf{Pr}_{\mathbb{D}\mathcal{W}_i^{\infty}}[\tilde{\mathcal{W}}_i]$.

Since each gadget G_i is $(\mathcal{I}', k, \mathbb{D}\mathcal{F}^{\infty}, \mu_{\mathcal{I}'})$-RRFC for all tuples $\mathcal{I}'$ it follows with Theorem 2 that the circuit is random fault secure with $\mu = 1 - \prod_{i=1}^{|C|}(1-\mu_i)$.

Let $\mathcal{B} = \{\tilde{\mathcal{F}} \mid \tilde{\mathcal{F}} = \bigcup_{i=1}^{|C|} \tilde{\mathcal{F}}_i, \forall \mathcal{I}'_{G_i} : \mathcal{L}_{\mathcal{I}'_{G_i}, \tilde{\mathcal{F}}_i}(G_i) = 0\}$ be the set of faults considered secure in Theorem 2, i.e., all fault combinations that can be corrected at the output of the respective gadgets. We denote by $\mathcal{B}_{\mathcal{I}'_{G_i}} = \{\tilde{\mathcal{F}}_i \in \mathcal{F}^{\infty} \mid \mathcal{L}_{\mathcal{I}'_{G_i}, \tilde{\mathcal{F}}_i}(G_i) = 0\} \subseteq \mathcal{B}$ the set of fault combinations that can be corrected in a gadget G_i under faults in the input indices in $\mathcal{I}'_{G_i}$, with $\forall j : |\mathcal{I}'_{G_i,j}| \leq k$.

We now go backward through the circuit, starting with gadgets connected to the outputs of C, considering only faults $\tilde{\mathcal{F}} \in \mathcal{B}$. Let G_i be a gadget only connected to outputs of C. By RPCUF of G_i, we can construct a simulator Sim_{G_i} that requires a subset of inputs (defined by the tuple $\mathcal{I}_{G_i}$) for the simulation of the wires in $\tilde{\mathcal{W}}_i$ under the faults $\tilde{\mathcal{F}}_i$ and with faulty inputs with indices in $\mathcal{I}'_{G_i}$. Let the failure probability of Sim_{G_i} be ϵ_i.

We continue with the parent gadgets, i.e., gadgets that have outputs connected to the inputs of just handled gadgets and (potentially) outputs of the circuit. Let G_j be such a gadget. By RPCUF of G_j, we can construct a simulator Sim_{G_j} that requires a subset of inputs (defined by the tuple $\mathcal{I}_{G_j}$) for the simulation of the wires in $\tilde{\mathcal{W}}_j$ and the output wires defined by the tuples $\mathcal{I}_{G_i}$ of the child gadgets G_i under the faults $\tilde{\mathcal{F}}_j$ and with faulty inputs with indices in $\mathcal{I}'_{G_j}$. In particular, each gadget output with index ℓ is only used once in the circuit and we use the corresponding set $\mathcal{I}_{G_i,\ell'}$ of the child gadget as $\mathcal{O}_{G_j,\ell}$. Again, we denote the failure probability of Sim_{G_i} by ϵ_j. We repeat this process until we reach the inputs of C.

Given this, we can construct the simulator Sim for C by composition of the gadget simulators Sim_{G_i}. The failure probability ϵ of Sim is the probability that at least one simulator Sim_{G_i} fails, i.e., $\epsilon = 1 - \prod_{i=1}^{|C|}(1-\epsilon_i)$. Therefore, with the random fault security advantage $\mu = 1 - \prod_{i=1}^{|C|}(1-\mu_i)$, we have

$$\begin{aligned}\mu + (1-\mu)\epsilon &= 1 - \prod_{i=1}^{|C|}(1-\mu_i) + \prod_{i=1}^{|C|}(1-\mu_i)(1 - \prod_{i=1}^{|C|}(1-\epsilon_i)) \\ &= 1 - \prod_{i=1}^{|C|}(1-\mu_i)(1-\epsilon_i) \leq 1 - \prod_{i=1}^{|C|}(1-\gamma_i),\end{aligned}$$

with $\gamma_i = \mu_i + (1-\mu_i)\epsilon_i$ being the combined failure probability of each gadget G_i. It follows $(\prod_{i=1}^{|C|} \mathbb{D}\mathcal{W}_i^{\infty}, \prod_{i=1}^{|C|} \mathbb{D}\mathcal{F}_i, 1 - \prod_{i=1}^{|C|}(1-\gamma_i))$-RCS$_{\text{KF}}$ of C. □

Similar to Theorem 2, the above composition loses some tightness by only considering fault combinations that can be corrected after each gadget. While the set $\mathcal{B}$ gets bigger, and hence ϵ increases, when considering all fault combinations that can be corrected at the output of C, the overall γ gets smaller. The reason is that the respective fault combination is entirely captured in μ of the above argument, however, it would only be partially captured (multiplied by some $\epsilon \leq 1$) when considering the tighter definition of $\mathcal{B}$. Hence, the provided γ is indeed a upper bound of the combined failure probability.

Relation to Combined-Secure Composition. As with RFC, to the best of our knowledge, this is the first work discussing compositional properties in the context of the random combined model. However, again in the threshold model, several compositional notions were discussed. Most of them are a combination of compositional notions for probing and faulting, respectively. In this respect, (Strong) Non-Interference Non-Accumulation ((S)NINA) [17] (later refined to Combined (Strong) Non-Interference (C-(S)NI) [38]) is the combination of (S)NA with (S)NI, and Combined-Isolating Non-Interference (CINI) [25] is the combination of FINI and Probe-Isolating Non-Interference (PINI), respectively. In the context of polynomial masking, Berndt et *al.* [11] coined the notion of fault-resilient (S)NI, which is the usual (S)NI notion that is invariant to fault injection. Our proposed compositional notation has some similarities with (S)NINA/C-(S)NI in that it is a combination of stand-alone faulting and probing composition and the underlying stand-alone notions relate to the respective stand-alone notions of (S)NA and (S)NI.

An interesting direction for future research into the composition in the random combined model is the notion of expansion (as considered in the random probing model) and the investigation of gadgets where the random probing security is invariant to faults.

5 Automatic Verification of Protected Implementations

We implement the verification of the new random combined security notion (and its composability variants) for cryptographic circuits by extending the verification tools VERICA [38] and IronMask [10]. At the state of the art, VERICA can be employed for combined hardware security verification considering the glitch-extended probing model [22] and the zeta fault-injection model [40], while IronMask is able to efficiently verify various (random, glitch-extended) probing security notions by relying on an algebraic characterization for specific gadgets.

In our work, we first aim to establish a common foundation for the practical verification of circuits under the new security models. Subsequently, we provide detailed insights into our extensions, denoted as VERICA^{+}[2] and IronMask^{+}[3], enabling the verification of combined security properties in cryptographic circuits.

[2] The code for VERICA^{+} can be accessed at https://github.com/Chair-for-Security-Engineering/VERICA/tree/verica%2B.

[3] The code for IronMask^{+} can be accessed at https://github.com/CryptoExperts/IronMask.

5.1 Verification of the Generalized Security Models

We provide explicit formulas and practical verification choices for the computation of the simulation and correction failure probabilities in the general random probing model, the general random fault model, and the known-fault random combined security. These then serve as a basis for the extensions VERICA$^+$ and IronMask$^+$. For our implementations, we assume independent leakage and fault probabilities (p_w and q_f) for wires w and faults f to simplify the specification.

General Random Probing Security. The probability of a leaking wire combination $\tilde{\mathcal{W}}$ is the product of leakage probabilities of each wire $w \in \tilde{\mathcal{W}}$ times the product of $(1 - p_{w'})$ of the remaining wires $w' \in \mathcal{W} \setminus \tilde{\mathcal{W}}$. Consequently, let ϵ be the simulation failure probability such that

$$\epsilon = \sum_{i=1}^{|\mathcal{W}|} \sum_{\tilde{\mathcal{W}} \in \mathcal{W}^{\infty}_{\#i}} \prod_{w \in \tilde{\mathcal{W}}} p_w \prod_{w' \in \mathcal{W} \setminus \tilde{\mathcal{W}}} (1 - p_{w'}) , \tag{2}$$

where $\mathcal{W}^{\infty}_{\#i} \subseteq \mathcal{W}^{\infty}$ is the set of wire combinations of exactly i wires that lead to a simulation failure, i.e., $\tilde{\mathcal{W}}^{\infty}_{\#i} = \{\tilde{\mathcal{W}} \in \mathcal{W}^{\infty} \mid |\tilde{\mathcal{W}}| = i \text{ and } \mathsf{Sim}(C, \tilde{\mathcal{W}}) = \perp\}$. The above equation generalizes the computation of the simulation failure probability in the random probing security definition from Belaïd et *al.* [8]. Precisely, Eq. 2 is only equivalent to the latter if we consider that all wires leak with the same probability p.

Practically, a circuit may be too large for exhaustively checking all wire combinations. Therefore, VERICA$^+$ and IronMask$^+$ compute the outer sum of Eq. 2 only up to a threshold α, i.e., for $i \in [1, \alpha]$. For any combinations of more than α wires, we can either consider that (1) they do not result in a simulation failure ($\forall i \in]\alpha, |\mathcal{W}|] , \mathcal{W}^{\infty}_{\#i} = \emptyset$) to obtain a lower bound $\epsilon_{\min}$, or (2) they all lead to a simulation failure ($\forall i \in]\alpha, |\mathcal{W}|] , \tilde{\mathcal{W}}^{\infty}_{\#i} = \{\tilde{\mathcal{W}} \in \mathcal{W}^{\infty} \mid |\tilde{\mathcal{W}}| = i\}$) to obtain an upper bound $\epsilon_{\max}$. Those lower and upper bound computations follow the same method used for random probing security by Belaïd et *al.* in [8].

General Random Fault Security. The probability of a fault combination $\tilde{\mathcal{F}}$ is the product of the individual fault probabilities of the said combination times the product of $(1 - q_{f'})$ for all faults f' not present in it. Let μ be the correction failure probability defined by

$$\mu = \sum_{i=1}^{|\mathcal{F}|} \sum_{\tilde{\mathcal{F}} \in \mathcal{F}^{\infty}_{\#i}} \prod_{f \in \tilde{\mathcal{F}}} q_f \prod_{f' \in \mathcal{F} \setminus \tilde{\mathcal{F}}} (1 - q_{f'}) , \tag{3}$$

where $\mathcal{F}^{\infty}_{\#i} \subseteq \mathcal{F}^{\infty}$ is the set of fault combinations of exactly i faults that cannot be corrected (such that $\tilde{\mathcal{F}} \in \mathcal{F}^{\infty}_{\#i}$ iff $\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 1$).

For practical reasons, Eq. 3 is computed for up to β faults during security verification; the outer sum thus reduces to $\sum_{i=1}^{\beta}$. Similarly to the previous models, we derive the lower bound $\mu_{\min}$ and the upper bound $\mu_{\max}$ of μ by assuming that any combination of more than β faults can be, respectively, corrected or not.

Known-Fault Random Combined Security. For simplicity, we restrict ourselves to $\mathrm{RCS_{KF}}$ (cf. Definition 4) in the practical tool implementations. We then rely on the discussed security reduction for $\mathrm{RCS_{UF}}$ (cf. Sect. 3.3) and leave room for a tighter $\mathrm{RCS_{UF}}$ security analysis for future work. That said, three parameters are reported for $\mathrm{RCS_{KF}}$:

1. the correction failure probability μ as described in Eq. 3,
2. the known-fault simulation failure probability ϵ_{kf} if fault combinations can be corrected, such that

$$\epsilon_{kf} = \frac{1}{1-\mu} \sum_{i=1}^{|\mathcal{F}|} \sum_{\tilde{\mathcal{F}} \in \mathcal{B}_{\#i}} \epsilon_{kc}^{\tilde{\mathcal{F}}} \prod_{f \in \tilde{\mathcal{F}}} q_f \prod_{f' \in \mathcal{F} \setminus \tilde{\mathcal{F}}} (1 - q_{f'}) \,, \tag{4}$$

   where $\mathcal{B}_{\#i}$ is the set of fault combinations of i faults that can be corrected and $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ is the known-corrected-fault simulation failure probability (Eq. 2). The above formula is obtained by observing that for any $\tilde{\mathcal{F}} \in \mathcal{B}$, we have $\Pr_{\mathbb{D}_{\mathcal{B}}}[\tilde{\mathcal{F}}] = \frac{1}{1-\mu} \Pr_{\mathbb{D}_{\mathcal{F}^\infty}}[\tilde{\mathcal{F}}] = \frac{1}{1-\mu} \prod_{f \in \tilde{\mathcal{F}}} q_f \prod_{f' \in \mathcal{F} \setminus \tilde{\mathcal{F}}} (1 - q_{f'})$.
3. and the advantage of the combined adversary γ_{kc}:

$$\gamma_{kc} = \mu + \epsilon_{kf} \cdot (1 - \mu) \,. \tag{5}$$

Again, due to practical limitations, we compute the lower and upper bounds of μ up to β faults and the lower and upper bounds of $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ up to α wires, if fault combinations can be corrected. Then, the lower and upper bounds of ϵ_{kf} are derived from the ones of $\epsilon_{kc}^{\tilde{\mathcal{F}}}$. Note, however, that those bounds on $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ are related to the upper bound of μ. The reason is, that we cannot compute the respective $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ for $\tilde{\mathcal{F}} \in \mathcal{B}_{\#i}$ with $i > \beta$, since this would mean to iterate over all faulty circuits with more than β faults. Then, the lower and upper bounds of γ_{kc} are computed from the corresponding bounds of both ϵ_{kf} and μ. The respective bounds of those three parameters are then returned.

Compositional Notions. We provide algorithms in the extended version [9] to describe how to perform the security verification of the compositional notions from Sect. 4, which have been implemented VERICA$^+$ and IronMask$^+$.

5.2 Extension of VERICA Verification

We briefly present an optimization to count leaking wire combinations that fail simulations in the (general) random probing model in VERICA$^+$ with binomial trees, to provide a tighter lower bound of the simulation failure probability. Moreover, we outline additional considerations to assess the fault probability for the general random fault and known-fault random combined security. More details are given in the extended version [9].

A Tighter Lower Bound in the (General) Random Probing Model. VERICA$^+$ implements a novel approach, relying on the traversal of binomial trees, to identify larger wire combinations containing a failing wire combination detected by the tool, in a deterministic way. It also does not include redundant combinations or use extra memory. In short, VERICA$^+$ finds failing wire combinations in a depth-first manner for up to α wires, $\alpha \in [1, |\mathcal{W}|]$, contrary to the naive breadth-first approach of VERICA. It results in the computation of a tighter lower bound of the simulation failure probability in the (general) random probing model, than previously mentioned in Sect. 5.1 and in [8], while concurrently providing an upper bound.

The core insight is that adding wires to a failing wire combination will result in another simulation failure. Thus, by representing the intermediate wires of a circuit as nodes of a binomial tree, VERICA$^+$ identifies the larger wire combinations containing the failing one. Succinctly, VERICA$^+$ traverses a binomial tree whose nodes represent the intermediate wires of a circuit, in pre-order fashion, from the root to a chosen depth α. During the traversal, VERICA$^+$ updates the set of leaking wires to verify. Depending on the outcome of the verification, VERICA$^+$ updates the computation of the lower and upper bounds of the simulation failure probability, in the considered security model. The latter values are returned at the end of the binomial tree traversal.

Probability of a Fault Combination. VERICA$^+$ makes two assumptions to compute the probability of a non-corrected fault combination $\tilde{\mathcal{F}}$ for the evaluation of the correction failure probability μ. Firstly, VERICA$^+$ allows a different number of fault transformations τ per gate g, which are equally likely to occur (*assumption* 1). Secondly, it assumes that up to one fault transformation τ is injected per gate and per verification analysis (*assumption* 2). As a result, to compute the probability of a fault combination, VERICA$^+$ just iterates over the gates in the circuit, distinguishing between faulted and non-faulted ones.

5.3 Extension of IronMask Verification

IronMask verifies the implementation of (N)LR-gadgets. In essence, the gadgets accept up to two inputs, each potentially undergoing linear refreshing, which are then processed by a non-linear function (homogeneous multi-linear form) whose output may also be linearly refreshed. As argued in [10], a wide majority of masking gadgets existing in the literature are (N)LR. Our extension not only supports these gadgets but also accommodates more complex ones, termed C(N)LR-gadgets (for Correction blocks). These C(N)LR-gadgets are constructed by parallel replication of (N)LR-gadgets and incorporate intermediate non-linear correction blocks (e.g., implementing the majority function), with gates that may be affected by *set* or *reset* faults.

To handle C(N)LR-gadgets, we introduce a novel verification methodology that preserves the tool's performance, at the expense of sometimes allowing false positives. This trade-off is necessary to benefit from the efficient tool's verification paradigm, while allowing more complex constructions. This methodology operates under the assumption that each output of a correction block remains correct as long as no fault occurs along its corresponding path within the block. Initially, the symbolic expression of each wire that depends on at least one output copy of a correction block, unaffected by a fault within the block, is modified to its correct symbolic expression (i.e., a hypothetical input of the correction block without faults along its path). Then, in the IronMask$^+$ verification process, output or internal variables from correction blocks impacted by faults are substituted with all the $2k+1$ corresponding inputs of their correction block. While IronMask provides complete verification without false positives, our new methodology may produce artificial attack paths by conservatively replacing some variables with all their dependencies. This approach enhances the attacker's potential reach by granting access to sets of variables instead of their functions.

IronMask$^+$ includes a Python script to generate correctable faulty scenarios for a circuit. It takes a circuit description, a fault type (*set* or *reset*), and the numbers of faults and tolerated faults (usually linked to the number of replications, $2k + 1$). The script evaluates circuit outputs in SageMath using symbolic computation with randoms and input shares. If any output share has more than k faulty copies, it is deemed uncorrectable. Faulty input shares become new symbolic variables, while faulty randoms are compared to a circuit with identical random faults (per [38]). Further details are provided in the full version [9].

6 Evaluation

We select five different gadgets for experiments in the new known-fault random combined model (see Definitions 4 and 8) all designed for security against one fault and one probe. First, we consider the DOMREP [29] gadget, which is a simple replication of a masked multiplication gadget according to the Domain-Oriented Masking (DOM) [28] principle. To get a reasonable analysis, we add a majority vote at the gadget's output to account for a correction that is required at some point in the circuit. Since DOMREP-based implementations were shown to be vulnerable to combined attacks (which the original design did not claim to protect against, cf. Sect. 3), a new variant, DOMREP-II [35], was recently introduced. This version uses two independent bits of randomness for mask refreshing and a masked correction module. To ensure a more accurate analysis, we once again add the masked correction at the gadget's output. As a third design, we analyze the SININA gadget originally proposed for software in [17] and modified for hardware in [38]. This gadget is similarly based on a replicated DOM multiplication, however, with corrections introduced after the mask refreshing and before the compression. Fourth, we analyze HPC_1^C gadget from [25], which is a replicated DOM gadget with an additional refresh and correction of one of the shared inputs. Finally, the CPC_1 gadget from [23] is a variant of the HPC_1^C gadget with additional corrections after the mask refreshing in the multiplication. Note that the DOMREP, SININA, and HPC_1^C gadgets were shown to be insecure under composition [23,38,42,44]. Our case studies aim to highlight the strengths and limitations of VERICA$^+$ and IronMask$^+$, and give a qualitative comparison between the gadgets. We intentionally do not conduct an extensive security analysis of the gadgets under realistic leakage and fault distributions, as determining these distributions is left for future work. Nevertheless, such an analysis would only affect the absolute failure probabilities, without altering the qualitative expressiveness of the data.

Both tables presenting our results follow the same structure. The first column identifies the evaluated gadget, while the second and third columns show the number of gates (FIA locations) and the number of wires (SCA locations), respectively. The type of fault transformation is denoted by τ, while p represents the random probing leakage probability, and q denotes the random fault probability for each gate. In this setting, faults and leakages on different wires are independent, and the probabilities are equal across wires, except when only a subset of SCA and FIA locations is selected, where some probabilities are set to zero for efficiency reasons. The remaining columns display our results: $\mu_{\min}$ and $\mu_{\max}$ denote the bounds on the attacker's advantage in the random fault model or for RFC, while $\epsilon_{\min}$ and $\epsilon_{\max}$ correspond to the bounds in the random probing model or for RPCUF. For the computation, we selected thresholds of $\alpha = 2$ for the number of probes and $\beta = 2$ for the number of faults. Lastly, $\gamma_{\min}$ and $\gamma_{\max}$ indicate the overall attacker advantage in the RCS_{KF} or RCC_{KF} models. Each table specifies whether the bounds are computed for general security or for composition. In general, a lower probability reflects a more secure implementation, and when the lower and upper bounds are close, we have a reasonable approximation of the real probability and, hence, a meaningful analysis. Note that the bounds on ϵ correlate to the bounds on $\mu_{\max}$. We first present our results in terms of security parameters and verification timings for each tool individually. Lastly, we run the same experiments on both tools to validate correctness and compare timings in specific scenarios.

6.1 Results on VERICA$^+$

All our experiments for VERICA$^+$ have been conducted on a machine equipped with two AMD EPYC 7742 64-Core processors and 512 GB memory. This allowed us to perform all experiments on 256 threads. The analysis results and performance of VERICA$^+$ for RCC_{KF} of the five gadgets under different scenarios can be found in Table 1. We first observe that, under the same fault and leakage conditions, the CPC_1 gadget is more secure than the other four gadgets. This is expected since CPC_1 is shown composable in the threshold combined model (CINI [25]), while three of the others have shown to be flawed under composition in the same model [23,38,42,44] and the fifth (DOMREP-II) seems to provide

Table 1. $\mathrm{RCC}_{\mathrm{KF}}$ analysis of gadgets on VERICA^{+}.

Design	**#Loc.**		**Model**			**Probabilities**			**Time**
	FIA	SCA	τ	q	p	$\mu_{\min}/\mu_{\max}$	$\epsilon_{\min}/\epsilon_{\max}$	$\gamma_{\min}/\gamma_{\max}$	t
DOMREP [29]	67	114	*set*	2^{-10}	2^{-10}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	2.1 min
			set	2^{-15}	2^{-15}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	2.1 min
			reset	2^{-10}	2^{-10}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	2.1 min
			reset	2^{-15}	2^{-15}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	2.1 min
			flip	2^{-10}	2^{-10}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	2.0 min
			flip	2^{-15}	2^{-15}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	2.0 min
DOMREP-II [35]	298	444	*set*	2^{-10}	2^{-10}	9.96e-1/1.00e-0	0.00e-0/0.00e-0	9.96e-1/1.00e-0	22.8 h
			set	2^{-15}	2^{-15}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	22.6 h
			reset	2^{-10}	2^{-10}	9.96e-1/1.00e-0	0.00e-0/0.00e-0	9.96e-1/1.00e-0	22.6 h
			reset	2^{-15}	2^{-15}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	22.8 h
			flip	2^{-10}	2^{-10}	9.96e-1/1.00e-0	0.00e-0/0.00e-0	9.96e-1/1.00e-0	24.2 h
			flip	2^{-15}	2^{-15}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	24.4 h
SININA [17,38]	122	198	*set*	2^{-10}	2^{-10}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	46.0 min
			set	2^{-15}	2^{-15}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	46.1 min
			reset	2^{-10}	2^{-10}	9.99e-1/9.99e-1	2.99e-4/4.52e-2	9.99e-1/9.99e-1	48.2 min
			reset	2^{-15}	2^{-15}	1.00e-0/1.00e-0	2.44e-4/1.34e-3	1.00e-0/1.00e-0	48.1 min
			flip	2^{-10}	2^{-10}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	45.0 min
			flip	2^{-15}	2^{-15}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	44.9 min
HPC_1^C [25]	104	180	*set*	2^{-10}	2^{-10}	5.68e-2/5.69e-2	1.61e-1/1.61e-1	2.09e-1/2.09e-1	46.6 min
			set	2^{-15}	2^{-15}	1.83e-3/1.83e-3	5.48e-3/5.48e-3	7.30e-3/7.30e-3	46.4 min
			reset	2^{-10}	2^{-10}	6.41e-2/6.43e-2	1.61e-1/1.61e-1	2.15e-1/2.15e-1	40.5 min
			reset	2^{-15}	2^{-15}	2.07e-3/2.07e-3	5.48e-3/5.48e-3	7.54e-3/7.54e-3	40.6 min
			flip	2^{-10}	2^{-10}	6.41e-2/6.42e-2	1.61e-1/1.61e-1	2.15e-1/2.15e-1	39.1 min
			flip	2^{-15}	2^{-15}	2.07e-3/2.07e-3	5.48e-3/5.48e-3	7.54e-3/7.54e-3	39.0 min
CPC_1 [23]	98	174	*set*	2^{-10}	2^{-10}	5.13e-2/5.14e-2	8.19e-2/8.21e-2	1.29e-1/1.29e-1	53.3 min
			set	2^{-15}	2^{-15}	1.65e-3/1.65e-3	2.65e-3/2.65e-3	4.29e-3/4.29e-3	52.6 min
			reset	2^{-10}	2^{-10}	6.78e-2/6.79e-2	8.17e-2/8.19e-2	1.44e-1/1.44e-1	40.9 min
			reset	2^{-15}	2^{-15}	2.19e-3/2.19e-3	2.65e-3/2.65e-3	4.84e-3/4.84e-3	40.8 min
			flip	2^{-10}	2^{-10}	6.96e-2/6.97e-2	8.18e-2/8.20e-2	1.46e-1/1.46e-1	38.8 min
			flip	2^{-15}	2^{-15}	2.26e-3/2.26e-3	2.65e-3/2.65e-3	4.90e-3/4.90e-3	38.5 min

no further advantage. The discrepancy between the CPC_1 and the HPC_1^C gadget is mainly caused by the probing factor, while the discrepancy to the other gadgets is caused entirely by the faulting factor. The reason is, that for those gadgets, there exist some input-fault combinations that cannot be corrected properly (even without internal faults), specifically, when there are k faults in all the inputs. Those input-fault combinations are always considered insecure and, hence, dominate the failure probability for RFC because we determine the maximum over the input faults. This also explains why some of the ϵ bounds are zero. Since those bounds are related to $\mu_{\max}$, which is one, there is no side-channel evaluation performed. In this case, any internal faults can at most make the gadget more secure (by compensating the impact of input faults) and hence, the respective μ gets larger with smaller fault probability q.

To get a more in depth analsis, we analysed the DOMREP and DOMREP-II gadgets with (masked) correction for $\mathrm{RCS}_{\mathrm{KF}}$ i.e., without composition. The DOMREP-II gadget is intended to be more secure as it prevents a known flaw in DOMREP. However, we find that the opposite is the case. Specifically, for

Table 2. $\mathrm{RCS}_{\mathrm{KF}}$ analysis of the CPC_1 gadget [23] on VERICA$^+$, with computation threshold for probes $\alpha = 2$ and for faults $\beta = 2$, where only a subset of gates and wires are faulted and potentially leaked to the adversary.

Location	**#Loc.**		**Model**			**Probabilities**			**Time**
	FIA	SCA	τ	q	p	$\mu_{\min}/\mu_{\max}$	$\epsilon_{\min}/\epsilon_{\max}$	$\gamma_{\min}/\gamma_{\max}$	t
all; all	98	174	*reset*	2^{-10}	2^{-10}	7.40e-4/8.72e-4	2.44e-3/3.00e-3	3.18e-3/3.87e-3	13.6 s
reg; rep. index 0	18	106	*reset*	2^{-10}	2^{-10}	5.07e-5/5.15e-5	8.89e-4/1.03e-3	9.39e-4/1.08e-3	0.8 s
reg; cone of c_0^0	18	103	*reset*	2^{-10}	2^{-10}	5.07e-5/5.15e-5	7.50e-4/8.76e-4	8.00e-4/9.27e-4	0.8 s

$\tau = \mathit{flip}$, $q = 2^{-10}$, and $p = 2^{-10}$, we get $\gamma = [1.88\text{e-}3, 2.07\text{e-}3]$ for DOMREP and $\gamma = [1.79\text{e-}2, 3.05\text{e-}2]$ for DOMREP-II. Other fault scenarios provide similar results. This additional vulnerability of DOMREP-II can be explained through the application of the masked correction. Specifically, the use of DOM gadgets in the correction means that by faulting a share of the input a a fault at the gadget output is conditioned on the input b (similar to the attack explained in Sect. 3 - last paragraph). The effectiveness of the respective fault at the output can be learned by the adversary through two probes, which are available with a certain probability. Hence, while DOMREP-II indeed prevents the specific attack as indented it introduced new vulnerabilities on the way. The practical exploitation of this leakage remains an avenue for future research.

With respect to performance, we can see that VERICA$^+$ can estimate the failure probabilities, by enumerating all fault and probe combinations with up to two faults and probes, for four of the gadgets in less than one hour. This is only possible because of the massive parallelization exploited by VERICA$^+$. For the fifth gadget (DOMREP-II) the analysis requires nearly an entire day due to much larger designs because of the masked correction. As can be seen in Table 2 the evaluation of $\mathrm{RCS}_{\mathrm{KF}}$ is much faster. Specifically, the CPC_1 gadget can be analyzed within seconds. This speedup is possible because $\mathrm{RCC}_{\mathrm{KF}}$ needs to perform essentially the same analysis as for $\mathrm{RCS}_{\mathrm{KF}}$, however, for all input-fault and output-probe combinations.

In addition, we showcase the ability of VERICA$^+$ to analyze a design under different fault and leakage probabilities (where each location is faulted or leaks with independent probabilities). Specifically, we analyze the CPC_1 gadget [23] where we restrict the fault locations to registers only and the location of leaking wires to either all wires of the first replication (*rep. index 0*) or all wires that influence the output c_0^0 (*cone of* c_0^0). This could resemble an adversary that uses clock glitching for fault injection and an EM probe for side-channel analysis. Then, the two cases can be seen as representations of two possible layouts of the chip design, such that either all values of one replication index or all the values that influence the same output value are placed in close proximity. To give some context, we also provide in Table 2 the results of an analysis where all gates are faulted and all wires can leak. As can be seen, this allows a tighter analysis for specific attack scenarios, potentially enabling a reduction of the implementation cost if the attacker can be restricted in the possible fault and leakage distributions.

Table 3. $\mathrm{RCC_{KF}}$ analysis of gadgets on IronMask$^+$.

	#Loc.		Model			Probabilities			Time
Design	FIA	SCA	τ	q	p	$\mu_{\min}/\mu_{\max}$	$\epsilon_{\min}/\epsilon_{\max}$	$\gamma_{\min}/\gamma_{\max}$	t
DOMREP [29]	127	203	*set*	2^{-10}	2^{-10}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	16min 11s
			set	2^{-15}	2^{-15}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	16min 10s
			reset	2^{-10}	2^{-10}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	16min 24s
			reset	2^{-15}	2^{-15}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	16min 24s
SININA [17,38]	230	364	*set*	2^{-10}	2^{-10}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	3h 12min 53s
			set	2^{-15}	2^{-15}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	3h 33min 52s
			reset	2^{-10}	2^{-10}	9.99e-1/1.00e-0	1.18e-4/1.09e-1	9.99e-1/1.00e-0	3h 21min 59s
			reset	2^{-15}	2^{-15}	1.00e-0/1.00e-0	1.22e-4/3.30e-3	1.00e-0/1.00e-0	3h 13min 49s
HPC_1^C [25]	188	322	*set*	2^{-10}	2^{-10}	3.68e-2/3.76e-2	9.93e-1/9.97e-1	9.93e-1/9.97e-1	23h 54min 38s
			set	2^{-15}	2^{-15}	1.16e-3/1.16e-3	1.00e-0/1.00e-0	1.00e-0/1.00e-0	23h 55min 18s
			reset	2^{-10}	2^{-10}	8.53e-2/8.60e-2	9.95e-1/1.00e-0	9.96e-1/1.00e-0	23h 32min 20s
			reset	2^{-15}	2^{-15}	2.68e-3/2.68e-3	1.00e-0/1.00e-0	1.00e-1/1.00e-0	23h 33min 10s
CPC_1 [23]	182	316	*set*	2^{-10}	2^{-10}	8.89e-2/8.95e-2	1.62e-1/1.66e-1	2.37e-1/2.41e-1	21h 42min 38s
			set	2^{-15}	2^{-15}	2.93e-3/2.93e-3	5.66e-3/5.66e-3	8.57e-3/8.57e-3	21h 41min 48s
			reset	2^{-10}	2^{-10}	8.89e-2/8.96e-2	1.62e-1/1.65e-1	2.36e-1/2.40e-1	20h 33min 24s
			reset	2^{-15}	2^{-15}	2.93e-3/2.93e-3	5.63e-3/5.63e-3	8.54e-3/8.54e-3	20h 33min 15s

6.2 Results on IronMask$^+$

All experiments for IronMask$^+$ have been conducted on a single thread of a machine equipped with an AMD Ryzen Threadripper PRO 7995WX processor with 96 cores (192 threads) and 512GB of RAM. The analysis outcomes and performance results of IronMask$^+$ for $\mathrm{RCC_{KF}}$ across various scenarios are presented in Table 3. Unlike VERICA$^+$, IronMask$^+$ cannot evaluate DOMREP-II, as it does not conform to the C(N)LR structure described in Sect. 5.3. Specifically, the gadget employs a masked correction mechanism using DOM gadgets, which deviates from the majority-vote structure required by IronMask$^+$.

In this extension, we exclusively focus on *set* and *reset* faults, deferring the inclusion of other fault types for future work. It is notable that IronMask$^+$ performs quite efficiently (especially given its execution on a single thread), thanks to its evaluation of symbolic variables with efficient rules. At the same time, this allows for some tolerance of false positives, as detailed in Sect. 5. Specifically, although the bounds on the final γ advantages are pretty close to VERICA$^+$ for the CPC gadget, IronMask$^+$ outputs higher bounds on ϵ for HPC_1^C. This is because with probed output and faulty input shares, we quickly multiply the failures when considering all the copies. Furthermore, note that the increase in the lower bound of ϵ as p decreases (e.g., for HPC_1^C), is solely a consequence of the method used which restricts the execution to computing failures of size at most two, whereas the correct ϵ would actually increase with p.

Regarding performance, the four computations involving different probabilities under identical settings (gadget, fault type) can be partially generated concurrently. For example, when considering the CPC_1 gadget [23] for *set* faults, the selection of faulty circuits that can be corrected takes around 9 min. Subsequently, identifying failure tuples within these correctable circuits requires around 20 h, 43 min. Both of these operations are common to any combination of faults and probe probabilities. Finally, computing security advantages based on the probabilities of faults and probes requires 50 to 52 min.

6.3 Tool Comparison

We provide an extensive comparison between our two tool implementations in the full version [9]. In general, both tools yield the same bounds for μ, with only minor differences in the computed ϵ, which is expected due to variations in computational tightness. This reinforces our confidence in the correctness of both implementations. In general, VERICA$^+$ achieves tighter computations of failure probabilities, while IronMask$^+$ leverages design-specific structures for efficient computations, leading to higher performance. Thus, the tools offer different trade-offs between accuracy and efficiency.

7 Conclusion

In this work, we extended and refined probabilistic models for physical attacks with general leaking and fault probabilities. Specifically, we examined the random probing model, the random fault model, and their combinations. For the resulting models, we explored key properties, such as the impact of adversarial knowledge and composition. Additionally, we developed two tool-based analysis methods that enable the security assessment of design components. Ultimately, we hope our model will stimulate further research into the low-level physical effects of leakage and their expressions in probability distributions.

Acknowledgments. This work was co-funded by the European Union (ERC, AMAskZONE, 101077506). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. In addition, the work described in this paper has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972 and through the project CAVE (510964147), and by the German Federal Ministry of Education and Research BMBF through the project 6GEM (16KISK038), DI-SIGN-HEP (16KIS2073), and DI-OWAS (16ME0967).

References

1. Aghaie, A., Moradi, A., Rasoolzadeh, S., Shahmirzadi, A.R., Schellenberg, F., Schneider, T.: Impeccable circuits. IEEE Trans. Computers **69**(3), 361–376 (2020)
2. Ajtai, M.: Secure computation with information leaking to an adversary. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 715–724. ACM Press (Jun 2011). https://doi.org/10.1145/1993636.1993731
3. Amiel, F., Villegas, K., Feix, B., Marcel, L.: Passive and active combined attacks: Combining fault attacks and side channel analysis. In: FDTC 2007: Vienna, Austria. pp. 92–102 (2007)

4. Arribas, V., Wegener, F., Moradi, A., Nikova, S.: Cryptographic Fault Diagnosis using VerFI. In: HOST 2020. pp. 229–240. IEEE (2020)
5. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 116–129. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978427
6. Battistello, A., Coron, J.S., Prouff, E., Zeitoun, R.: Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 23–39. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53140-2_2
7. Belaïd, S., Cassiers, G., Mutschler, C., Rivain, M., Roche, T., Standaert, F., Taleb, A.R.: Towards achieving provable side-channel security in practice. IACR Cryptol. ePrint Arch. p. 1198 (2023)
8. Belaïd, S., Coron, J.S., Prouff, E., Rivain, M., Taleb, A.R.: Random probing security: Verification, composition, expansion and new constructions. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 339–368. Springer, Cham (Aug 2020). https://doi.org/10.1007/978-3-030-56784-2_12
9. Belaïd, S., Feldtkeller, J., Güneysu, T., Guinet, A., Richter-Brockmann, J., Rivain, M., Sasdrich, P., Taleb, A.R.: Formal Definition and Verification for Combined Random Fault and Random Probing Security. IACR Cryptol. ePrint Arch. p. 757 (2024)
10. Belaïd, S., Mercadier, D., Rivain, M., Taleb, A.R.: IronMask: Versatile verification of masking security. In: 2022 IEEE Symposium on Security and Privacy. pp. 142–160. IEEE Computer Society Press (May 2022). https://doi.org/10.1109/SP46214.2022.9833600
11. Berndt, S., Eisenbarth, T., Faust, S., Gourjon, M., Orlt, M., Seker, O.: Combined fault and leakage resilience: Composability, constructions and compiler. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Santa Barbara, CA, USA. LNCS, vol. 14083, pp. 377–409. Springer (2023)
12. Cassiers, G., Standaert, F.: Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference. IEEE Trans. Inf. Forensics Secur. **15**, 2542–2555 (2020)
13. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer (1999)
14. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M.: Passive and Active Combined Attacks on AES: Combining Fault Attacks and Side Channel Analysis. In: FDTC 2010, Santa Barbara, California, USA. pp. 10–19 (2010)
15. De Cnudde, T., Bilgin, B., Gierlichs, B., Nikov, V., Nikova, S., Rijmen, V.: Does coupling affect the security of masked implementations? In: Guilley, S. (ed.) COSADE 2017. LNCS, vol. 10348, pp. 1–18. Springer, Cham (Apr 2017). https://doi.org/10.1007/978-3-319-64647-3_1
16. Dehbaoui, A., Dutertre, J., Robisson, B., Tria, A.: Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES. In: FDTC 2012. pp. 7–15. IEEE Computer Society (2012)
17. Dhooghe, S., Nikova, S.: My gadget just cares for me - how NINA can prove security against combined attacks. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 35–55. Springer, Cham (Feb 2020). https://doi.org/10.1007/978-3-030-40186-3_3

18. Dhooghe, S., Nikova, S.: The random fault model. In: Carlet, C., Mandal, K., Rijmen, V. (eds.) SAC 2023, Fredericton, Canada. LNCS, vol. 14201, pp. 191–212. Springer (2023)
19. Dobraunig, C., Eichlseder, M., Groß, H., Mangard, S., Mendel, F., Primas, R.: Statistical ineffective fault attacks on masked AES with fault countermeasures. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 315–342. Springer, Cham (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_11
20. 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, Berlin, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_24
21. Dumont, M., Lisart, M., Maurine, P.: Electromagnetic Fault Injection : How Faults Occur. In: FDTC 2019. pp. 9–16. IEEE (2019)
22. Faust, S., Grosso, V., Pozo, S.M.D., Paglialonga, C., Standaert, F.: Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model. IACR TCHES **2018**(3), 89–120 (2018)
23. Feldtkeller, J., Güneysu, T., Moos, T., Richter-Brockmann, J., Saha, S., Sasdrich, P., Standaert, F.: Combined private circuits - combined security refurbished. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023, Copenhagen, Denmark. pp. 990–1004. ACM (2023)
24. Feldtkeller, J., Güneysu, T., Schaumont, P.: Quantitative fault injection analysis. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Guangzhou, China. LNCS, vol. 14441, pp. 302–336. Springer (2023)
25. Feldtkeller, J., Richter-Brockmann, J., Sasdrich, P., Güneysu, T.: CINI MINIS: Domain isolation for fault and combined security. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 1023–1036. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560614
26. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Çetin Kaya., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Berlin, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44709-1_21
27. Goubin, L., Patarin, J.: DES and differential power analysis (the "duplication" method). In: Koç, Çetin Kaya., Paar, C. (eds.) CHES'99. LNCS, vol. 1717, pp. 158–172. Springer, Berlin, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48059-5_15
28. Groß, H., Mangard, S., Korak, T.: Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In: ACM TIS@CCS 2016. p. 3. ACM (2016)
29. Gruber, M., Probst, M., Karl, P., Schamberger, T., Tebelmann, L., Tempelmeier, M., Sigl, G.: DOMREP-An Orthogonal Countermeasure for Arbitrary Order Side-Channel and Fault Attack Protection. IEEE Trans. Inf. Forensics Secur. **16**, 4321–4335 (2021)
30. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Berlin, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_19
31. 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, Berlin, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_27

32. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 104–113. Springer, Berlin, Heidelberg (Aug 1996). https://doi.org/10.1007/3-540-68697-5_9
33. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 388–397. Springer, Berlin, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_25
34. Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked CMOS gates. In: Menezes, A. (ed.) CT-RSA 2005, San Francisco, CA, USA. LNCS, vol. 3376, pp. 351–365. Springer (2005)
35. Probst, M., Brosch, M., Gruber, M., Sigl, G.: DOMREP II. In: IEEE HOST 2024, Tysons Corner, VA, USA. pp. 112–121. IEEE (2024)
36. 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, Berlin, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_9
37. Renauld, M., Standaert, F., Veyrat-Charvillon, N., Kamel, D., Flandre, D.: A formal study of power variability issues and side-channel attacks for nanoscale devices. In: Paterson, K.G. (ed.) EUROCRYPT 2011, Tallinn, Estonia. LNCS, vol. 6632, pp. 109–128. Springer (2011)
38. Richter-Brockmann, J., Feldtkeller, J., Sasdrich, P., Güneysu, T.: VERICA - verification of combined attacks automated formal verification of security against simultaneous information leakage and tampering. IACR TCHES **2022**(4), 255–284 (2022). https://doi.org/10.46586/tches.v2022.i4.255-284
39. Richter-Brockmann, J., Rezaei Shahmirzadi, A., Sasdrich, P., Moradi, A., Güneysu, T.: FIVER – Robust Verification of Countermeasures against Fault Injections. IACR TCES **2021**(4), 447–473 (2021)
40. Richter-Brockmann, J., Sasdrich, P., Güneysu, T.: Revisiting Fault Adversary Models - Hardware Faults in Theory and Practice. IEEE Trans. Computers pp. 1 – 14 (2022)
41. Roche, T., Lomné, V., Khalfallah, K.: Combined Fault and Side-Channel Attack on Protected Implementations of AES. In: CARDIS 2011, Leuven, Belgium. pp. 65–83 (2011)
42. Saha, S., Bag, A., Jap, D., Mukhopadhyay, D., Bhasin, S.: Divided we stand, united we fall: Security analysis of some SCA+SIFA countermeasures against SCA-enhanced fault template attacks. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part II. LNCS, vol. 13091, pp. 62–94. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92075-3_3
43. Saha, S., Jap, D., Breier, J., Bhasin, S., Mukhopadhyay, D., Dasgupta, P.: Breaking Redundancy-Based Countermeasures with Random Faults and Power Side Channel. In: FDTC 2018, Amsterdam, The Netherlands. pp. 15–22 (2018)
44. Saha, S., Ravi, P., Jap, D., Bhasin, S.: Non-Profiled Side-Channel Assisted Fault Attack: A Case Study on DOMREP. In: DATE 2023. pp. 1–6. IEEE, Antwerp, Belgium (2023)
45. Schellenberg, F., Gnad, D.R.E., Moradi, A., Tahoori, M.B.: Remote inter-chip power analysis side-channel attacks at board-level. In: Bahar, I. (ed.) ICCAD 2018, San Diego, CA, USA. p. 114. ACM (2018)
46. Shahmirzadi, A.R., Rasoolzadeh, S., Moradi, A.: Impeccable Circuits II. In: DAC 2020. pp. 1–6. IEEE (2020)

47. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Kaliski Jr., B.S., Koç, Çetin Kaya., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 2–12. Springer, Berlin, Heidelberg (Aug 2003). https://doi.org/10.1007/3-540-36400-5_2
48. Yao, Y., Yang, M., Patrick, C., Yuce, B., Schaumont, P.: Fault-assisted side-channel analysis of masked implementations. In: 2018 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2018, Washington, DC, USA, April 30 - May 4, 2018. pp. 57–64. IEEE Computer Society (2018). https://doi.org/10.1109/HST.2018.8383891, https://doi.org/10.1109/HST.2018.8383891
49. Zussa, L., Dutertre, J., Clédière, J., Tria, A.: Power supply glitch induced faults on FPGA: An in-depth analysis of the injection mechanism. In: IOLTS 2013. pp. 110–115. IEEE (2013)

Leakage-Resilient Incompressible Cryptography: Constructions and Barriers

Kaartik Bhushan[1(✉)], Rishab Goyal[2], Venkata Koppula[3], Varun Narayanan[4], Manoj Prabhakaran[1], and Mahesh Sreekumar Rajasree[3]

[1] Indian Institute of Technology, Bombay, India
kbhushan@cse.iitb.ac.in , manojmp@iitb.ac.in
[2] University of Wisconsin-Madison, Madison, USA
rishab@cs.wisc.edu
[3] Indian Institute of Technology, Delhi, India
kvenkata@cse.iitd.ac.in
[4] University of California, Los Angeles, USA

Abstract. We introduce Leakage-Resilient Incompressible cryptography, which simultaneously addresses two variants of side-channel attacks that have been tackled in theoretical cryptography. Leakage-resilience seeks to provide security against an adversary who learns a part of the secret-key and the entire ciphertext or signature; conversely, incompressible cryptography provides security against an adversary who learns the entire secret-key, but only a part of the ciphertext or signature. However, constructions in either of these security models can fail against an attack in the other model. In this work, we define a new model of security that subsumes both leakage-resilient cryptography and incompressible cryptography, and we present several non-trivial positive and negative results.

On the positive side, first we present a transformation from incompressible symmetric-key encryption (SKE) to leakage-resilient incompressible SKE in the information-theoretic setting. Next, as one of our main results, we construct a leakage-resilient incompressible public-key encryption (PKE), combining an incompressible SKE and a new primitive that we call leakage-resilient non-committing key encapsulation mechanism (LR-NC-KEM). While an incompressible SKE suitable for use in both these constructions already exists in the literature (Dziembowski, CRYPTO 2006), we present a new construction with better

K. Bhushan was supported by the Prime Minister's Research Fellowship by the Government of India.
R. Goyal was supported by OVCRGE at UW-Madison, and funding from the Wisconsin Alumni Research Foundation
V. Koppula was supported by the Pankaj Gupta Young Faculty Fellowship at IIT Delhi, and IIT Bombay Trust Lab Early Career Award.
V. Narayanan was supported by NSF Grants CNS-2246355, CCF-2220450, and CNS-2001096.
M. Prabhakaran was supported by IIT Bombay Trust Lab and Algorand Centre of Excellence by Algorand Foundation.

parameters, using an appropriate notion of invertible extractors; this leads to corresponding improvements in the final parameters we obtain in these constructions. We also design a leakage-resilient incompressible signature scheme.

On the negative side, we show barriers to significantly improving the parameters we obtain, by showing impossibility of basing the security of such improved schemes on blackbox reductions.

Apart from the general framework and the specific results we obtain, some of the intermediate tools that we define and instantiate, like LR-NC-KEM and invertible extractors, may be of independent interest.

Keywords: leakage-resilient · incompressible encryption · impossibility · black-box reductions

1 Introduction

Traditionally, cryptography relies on each honest user having an ideally secure computation platform. If a user's computing system does not provide an ideally secure platform, the user is essentially considered to be corrupted. However, this severely limits the security guarantees cryptography can provide, as computing systems are seldom ideal. In particular, existence of various practical side-channel and key-infiltration attacks routinely undermine security of numerous deployed cryptosystems.

From a theoretical perspective, this seems to be a hopelessly bleak situation, especially when all the users rely on non-ideal computing systems. However, a long line of research in cryptography has studied the problem of providing full-fledged security guarantees even in non-ideal settings. Our work is inspired by two such popular formulations considered today – leakage resilience, and incompressible cryptography.

Leakage Resilient Cryptography. As demonstrated by several practical side-channel attacks (such as [22] presenting one of the earlier ones), an adversary may be able to extract partial information (*leakage*) about the secret keys of an, otherwise, honest user. Since traditional cryptographic schemes provide *no guarantees at all* once such leakage takes place, the area of *leakage-resilient cryptography* [1] was developed to protect from such side-channel attacks. It is a very rich area of cryptography that has received a lot of attention over the last two decades.

One of the prominent examples is (leakage resilient) public-key encryption. It allows an adversary to learns the public-key pk as well as a leakage on the secret-key sk in the form of $f_{\mathsf{leak}}(\mathsf{sk})$. Here f_{leak} is any adversarial chosen function, with the only constraint on f_{leak} that its output length is limited. This ensures that leakage resilient public-key encryption is not trivially impossible, as we still require semantic security of the encryption system to hold even in presence of the secret-key leakage.

Incompressible Cryptography. Side-channel attacks are not the only means by which an adversary can learn some information about the secret keys. Key infiltration attacks, where the adversary is able to learn the entire secret key, are another common security threat. In such scenarios, it is unclear whether one could ask for any meaningful notion of security, but prior works have demonstrated that one may still ask for security of past communication. Consider encryption[1] as an example, suppose an adversary who knows that, at some point in the future, it will learn the secret key. Such an adversary can store as much of the prior (encrypted) communication as possible, and plan to decrypt the stored ciphertexts once it has the secret key. A mitigating factor in such a scenario is the vast amounts of ciphertexts such an adversary will need to store (anticipating key compromise in the future). Alternately, consider a scenario where the ciphertexts need to be "exfiltrated" from a partially secure system via malware, large amounts of communication could be detected and truncated. However, if the adversary can compress the ciphertexts (and later recover some digest of the message when the key is obtained), then even with the limits on storage or exfiltration bandwidth, secrecy of the messages will be compromised.

Again, this concern can be addressed by enhancing the cryptographic guarantees of the cryptosystems used. An early variant of this problem was introduced by Rivest [25] (in a somewhat different context); it was explicitly studied by Dziembowski [12] in the secret-key setting, and more recently studied in the public-key setting by Guan, Wichs, and Zhandry [19]. Guan *et al.* called this *incompressible encryption*, and we adopt the same terminology in this work. Informally, incompressible public-key encryption states that no attacker can learn any information about plaintext from any arbitrarily compressed ciphertext, even if it obtains the secret key, as long as the ciphertext is compressed before receiving the key. That is, even given secret key sk, an attacker cannot learn anything about an encrypted message from $f_{\mathsf{comp}}(\mathsf{ct})$, where f_{comp} is any adversarially chosen function with a limited output size (which disallows storing the entire ciphertext).

Leakage Resilience and Incompressibility are Duals. Leakage resilient encryption and incompressible encryption can be viewed as *duals* of each other. In the former, the adversary gets partial information about the secret key and the full ciphertext. In the latter, it gets the full secret key, but can only retain partial information about the ciphertext. Indeed, connections between these two domains have been studied, and techniques from one domain have led to progress in the other. However, quite apart from the similarity of techniques, we note that the attacks underlying both these settings can simultaneously take place: *When the ciphertexts are being compressed for storage/exfiltration, leakage on the key may already be available to the adversary.* We ask:

[1] In the case of key-agreement, the threat due to compromise of long-terms keys is addressed using ephemeral keys, to yield forward security. However, this does not apply to scenarios like encrypted e-mails and disk encryption, involving a long-term decryption key.

Can such combined threats be tackled and if so, how efficiently?

To address this, we initiate the study of *leakage resilient incompressible cryptography.* While the central focus of this work is on encryption (public-key and symmetric-key), we also study signatures in this setting. We briefly recall that a leakage resilient signature scheme says that an adversary must not be able forge a signature, even after receiving leakage on the signing key. While incompressible signature prevent replay attacks, where the attacker must not be able to forge a signature, even it receives an arbitrary (yet bounded) leakage on any number of signatures. We formally explain this further in the technical overview.

Measuring Efficacy of Leakage Resilient Incompressible Cryptography. Inspired by prior works [1,2,4,5,8–10,16,17,19,20,24] on leakage resilience and incompressible cryptography, we consider a natural combination of known efficiency metrics for cryptosystems in the leakage resilient incompressible setting.

Let us start by reviewing the commonly used efficiency metrics for encryption. For leakage resilience encryption, a common approach is to measure the amount of leakage that is permitted without loosing semantic security. It is formally capture as the *leakage-rate*, which is the ratio of total number of secret-key bits that can be leaked divided by the secret-key length, i.e. $\frac{|f_{\mathsf{leak}}(\mathsf{sk})|}{|\mathsf{sk}|}$. For incompressible encryption, a well-known metric is the ciphertext-rate. It measures the maximum amount of message that can be packed in a single ciphertext, and formally defined as the ratio of the message length and the ciphertext size, i.e. $\frac{|m|}{|\mathsf{ct}|}$. In our work, we also study another efficiency metric for incompressible encryption, that we call *compression-rate*. It is defined as the ratio of the total number of bits about a ciphertext that can be stored divided by the ciphertext length, i.e. $\frac{|f_{\mathsf{comp}}(\mathsf{ct})|}{|\mathsf{ct}|}$.

There are three main metrics – ciphertext-rate, compression-rate, and leakage-rate – that we want to optimize while designing leakage resilient incompressible encryption schemes. Ideally, we want to design encryption schemes where all three rates are close to 1: this would give a scheme that is as efficient as possible in terms of the ciphertext size, while semantic security will hold even when an attacker learns a the secret key and the ciphertext almost entirely. Thus, the central question that we study in this work is:

Q1. *Can we build leakage-resilient incompressible PKE with ciphertext-rate, compression-rate, and leakage-rate approaching* 1*?*

We already have encryption schemes that can tolerate as much as $1 - o(1)$ fraction of the secret-key leakage (i.e., leakage-rate is almost 1) [1–3,7,21]. Moreover, we also have incompressible encryption schemes with ciphertext-rate and compression-rate of 1 [4,17,19,20]. Thus, it is quite natural to ask can we design a unified encryption scheme that achieves all these rates optimally.

Furthermore, towards our broader goal of studying other forms of leakage-resilient incompressible cryptosystems, we also study a similar question for signature schemes:

Q2. *Can we build leakage-resilient incompressible signature schemes with* $1 - o(1)$ *leakage-rate and optimal communication (message + signature size)?*

1.1 Our Contributions

Our first main contribution is establishing a barrier to Q1! We show that leakage-resilient incompressible PKE where all three rates are close to 1 is impossible, as long as we want to prove security via a black-box reduction to a secure cryptographic game [15,28]. In light of the above barrier, we ask a natural question which is – *What is the best ciphertext-rate, compression-rate, and leakage-rate that can we can provably achieve for leakage-resilient incompressible encryption?* Our next contribution is to design encryption schemes from standard cryptographic assumptions that achieve near-optimal rates. As an additional contribution, we also design a leakage-resilient incompressible signature scheme from standard assumptions. Below we summarize our main results in two main categoriess – lower bounds and positive results.

Lower Bounds: We prove that any leakage-resilient incompressible PKE scheme with ciphertext-rate $1 - o(1)$, compression-rate $1 - o(1)$ and leakage-rate $1 - o(1)$ cannot be proven secure via a black-box reduction to a cryptographic game [15,28]. Formally, we prove the following.

Theorem 1. *Let* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *be a PKE scheme for arbitrary message length* $\ell_{\mathtt{msg}}$ *with ciphertext length* $\ell_{\mathtt{ct}} = \ell_{\mathtt{msg}} + o(\ell_{\mathtt{msg}})$ *and secret key length* $\ell_{\mathtt{key}}$*, and possesses almost perfect correctness and deterministic decryption. If* $\ell_{\mathtt{leak}} = \ell_{\mathtt{key}} - o(\ell_{\mathtt{key}})$ *and the size of compressed state* $\ell_{\mathtt{st}} = \ell_{\mathtt{msg}} - o(\ell_{\mathtt{msg}})$*, then there is no secure cryptographic game* $\mathcal{G}$ *such that there is a black-box reduction that proves leakage-resilient incompressible encryption security of* PKE *from the security of* $\mathcal{G}$*.*

We note that if we restrict the adversary's leakage functions to bit-leakage functions (where the adversary can learn some bits of the secret key), then we can achieve ciphertext-rate-1 (we discuss this formally in the full version). The proof of the above theorem builds heavily on a new lower bound that we prove for incompressible PKE.

*Incompressible Encryption with Ciphertext-Rate-*1*.* As mentioned earlier, we have constructions for incompressible PKE with ciphertext-rate-1 [4,17,19]. However, all constructions with provable security in the standard model have *large* secret keys, i.e. the key size grows with the message size. Guan *et al.* [19] conjectured that any ciphertext-rate $1 - o(1)$ and compression-rate $1 - o(1)$ incompressible PKE scheme with '*short*' key size cannot be proven secure using a black-box reduction to a secure cryptographic game [28]. In this work, we formally prove this conjecture. Formally, we prove:

Theorem 2. *Let* $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *be an incompressible PKE scheme for arbitrary message length* $\ell_{\mathtt{msg}}$ *with ciphertext length* $\ell_{\mathtt{ct}} = \ell_{\mathtt{msg}} + o(\ell_{\mathtt{msg}})$*, secret key length* $\ell_{\mathtt{key}}$*, state size* $\ell_{\mathtt{st}} = \ell_{\mathtt{msg}} - o(\ell_{\mathtt{msg}})$ *and possesses almost perfect correctness and deterministic decryption. Then, if* $\ell_{\mathtt{key}} \leq \ell_{\mathtt{msg}}(1 - \Omega(1))$*, there is no secure cryptographic game* $\mathcal{G}$ *such that there is a black-box reduction that derives the security of* PKE *from the security of* $\mathcal{G}$*.*

Positive Results: We show that any information theoretic incompressible secret key encryption scheme [11] can be transformed into an information-theoretic leakage-resilient incompressible SKE scheme. Naturally, given the information-theoretic promise, this scheme has large secret keys. When instantiated using the incompressible SKE scheme of Dziembowski [11], the resulting leakage-resilient incompressible SKE scheme has ciphertext-rate $\frac{1}{3}$ and compression-rate $\frac{1}{3}$.

Further, we present an alternate information-theoretic incompressible SKE scheme which improves upon the one in [11] in terms of the ciphertext-rate and compression-rate (but uses a larger secret-key). Using this to instantiate our leakage-resilient incompressible SKE we obtain a ciphertext-rate and compression-rate of $\frac{1}{2}$. This is optimal information-theoretically, since the sum of the ciphertext-rate and compression-rate can be at most 1.

Theorem 3. *There exists a leakage-resilient ciphertext-rate-*$\frac{1}{2}$ *incompressible SKE scheme with compression-rate* $\frac{1}{2}$ *and leakage-rate* $1 - o(1)$ *that is secure against unbounded adversaries.*

Next, we move on to leakage-resilient incompressible public-key encryption. Following a recent result by Goyal *et al.* [17], which lifts incompressible SKE to incompressible PKE, we note that we can generically lift any leakage-resilient incompressible SKE to leakage-resilient incompressible PKE. But this yields a $o(1)$ ciphertext-rate. In this work, we optimize this significantly to design a leakage-resilient incompressible PKE scheme with ciphertext-rate $\frac{1}{2}$.

Theorem 4. *Assuming the hardness of* DDH *along with* DCR *problem*[2]*, there exists a ciphertext-rate* $\frac{1}{2}$*, compression-rate* $\frac{1}{2}$ *and leakage-rate* $1 - o(1)$ *leakage-resilient incompressible PKE.*

For this construction, we introduce a new cryptographic primitive called *leakage-resilient non-committing encryption* (LR-NCE) which may be of independent interest (see Sect. 6). All of our results surrounding incompressible encryption, together with existing results are included in Table 1 and Table 2.

Leakage-Resilient Incompressible Signatures. Finally, we also design a leakage-resilient incompressible signature scheme (see the full version), following the template of Guan *et al.* [19]. Formally, we show that:

[2] The same set of assumptions were needed by Branco *et al.* [4] to construct ct-rate-1 incompressible PKE.

Table 1. Constructions for Regular Incompressible Encryption. In the information-theoretic setting, we show an incompressible SKE scheme with ciphertext-rate $\frac{1}{2}$. We conjecture that this rate is optimal for information-theoretic incompressible SKE. In the computational setting, prior works showed incompressible SKE/PKE with ciphertext-rate 1 and having large secret keys. In the last row, we discuss certain barriers for provably secure incompressible SKE/PKE.

Primitive	Ciphertext-Rate	Compression-Rate	Secret-Key Size	Note
IT. SKE	$\frac{1}{3}$	$\frac{1}{3}$	$\max(\ell_{\mathtt{st}}, \ell_{\mathtt{msg}}) + \mathsf{poly}(\lambda)$	[12]
IT. SKE	$\frac{1}{2}$	$\frac{1}{2}$	$4 \cdot \max(\ell_{\mathtt{st}}, \ell_{\mathtt{msg}})$	Section 5.1
SKE	$1 - o(1)$	$1 - o(1)$	$\max(\ell_{\mathtt{st}}, \ell_{\mathtt{msg}})(1 + o(1))$	[4]
PKE	$1 - o(1)$	$1 - o(1)$	$\ell_{\mathtt{msg}}(1 + o(1)) \cdot \mathsf{poly}(\lambda)$	[4]
PKE/SKE	$1 - o(1)$	$1 - o(1)$	$\ell_{\mathtt{msg}}(1 - \Omega(1))$	Section 8.1 (Barrier)

Table 2. Constructions for Leakage-Resilient Incompressible Encryption. In the first two rows, we discuss parameters that can be achieved for leakage-resilient incompressible encryption. The third row lists parameters for leakage-resilient incompressible encryption which cannot be achieved with black-box reductions. Finally, in the last row, we present optimal parameters that can be achieved when the leakage is restricted to a subset of bits of the secret key.

Primitive	Ciphertext-Rate	Compression-Rate	Leakage-Rate	Leakage Function	Reference
IT. SKE	$\frac{1}{2}$	$\frac{1}{2}$	$1 - o(1)$	General	Section 5
PKE	$\frac{1}{2}$	$\frac{1}{2}$	$1 - o(1)$	General	Section 7
PKE/SKE	$1 - o(1)$	$1 - o(1)$	$1 - o(1)$	General	Section 8.2 (Barrier)
PKE/SKE	$1 - o(1)$	$1 - o(1)$	$1 - o(1)$	Bits	full version

Theorem 5. *Assuming the existence of incompressible encoding scheme with encoding rate of* $1 - o(1)$ *and leakage-resilient public key signature scheme with leakage-rate of* $1 - o(1)$*, there exists leakage resilient incompressible signature scheme with signature-rate of* $1 - o(1)$ *and leakage-rate of* $1 - o(1)$*.*

2 Technical Overview

This section provides a high-level summary of the paper's results, both positive and negative.

2.1 Impossibility Results

In this work, we introduce two black-box separation results, demonstrating that specific incompressible encryption primitives cannot be proven secure under any standard assumptions using a black-box reduction. We accomplish this by leveraging the notion of simulatable attacks introduced by Wichs [28]. A simulatable attack against the security of a primitive involves two entities: an inefficient adversary $\mathcal{A}$ that achieves a high probability of winning the security game, and an efficient simulator Sim capable of mimicking $\mathcal{A}$. Notably, we require that Sim can convincingly emulate $\mathcal{A}$ for any efficient reduction R with oracle access to Sim, i.e., $|\Pr[R^{\mathcal{A}}(1^\lambda) = 1] - \Pr[R^{\mathsf{Sim}}(1^\lambda) = 1]| = \text{negl}(\lambda)$.

Simulatable Attack on Ciphertext Rate-1 Incompressible PKE with Small Keys. Assuming that the reader is familiar with the definition of incompressibile security game (see Sect. 3.2), we describe our inefficient reduction $\mathcal{A}$, which has two hardcoded functions, g and h, as follows:

1. On input a public key pk, $\mathcal{A}_0$ returns a pair of messages $g(\mathsf{pk}) = (m_0, m_1)$ along with $\mathsf{aux} = g(\mathsf{pk})$.
2. On input a ciphertext $\mathsf{ct}^*, \mathsf{aux}$, $\mathcal{A}_1$ returns $h(\mathsf{ct}^*)$.
3. On input $(\mathsf{pk}, \mathsf{sk}, \mathsf{st}, \mathsf{aux})$, $\mathcal{A}_2$ checks whether $\mathsf{aux} = g(\mathsf{pk})$. If not, it returns $\perp$. Else, it computes $C = \{\mathsf{ct} | h(\mathsf{ct}) = \mathsf{st}\}$ and $M = \{m | m = \mathsf{IncPKE.Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}), \mathsf{ct} \in C\}$. It checks whether $(m_0, m_1) \cap M = \emptyset$ or $(m_0, m_1) \subseteq M$. If this check passes, it returns $\perp$. Else, if $m_0 \in M$, it returns 0. Otherwise, it returns 1.

Now, let's understand why $\mathcal{A}$ wins the incompressible security game with high probability. In the security game, the challenger faithfully generates the challenge ciphertext ct^*, an encryption of m_b where b is the random bit challenge. This implies that $m_b \in M$, and the only case where $\mathcal{A}$ would lose the game is if $m_{1-b} \in M$. Since h is a random function, $|h^{-1}(\mathsf{st})|$ has a size close to average, i.e., $2^{|\mathsf{ct}^*|}/2^{|\mathsf{st}|}$. Given the rate-1 incompressible scheme ($|\mathsf{ct}^*| = |m| + o(|m|)$ and $|\mathsf{st}| = |m| - o(|m|)$), we have $|C| \approx o(|m|)$ and the set is randomly chosen. Thus, the probability that one of the messages decrypts to m_{1-b} is approximately $1/2^{|m| - o(|m|)}$ which is negligible in the message size.

Now, let's present our simulator Sim which maintains two databases Q_0, Q_1:

1. On input a public key pk, Sim first checks whether an entry $(\mathsf{pk}, (m_0, m_1))$ is in Q_0. If so, it outputs $(m_0, m_1, \mathsf{aux} = (m_0, m_1))$. If not, it randomly generates a pair of messages (m_0, m_1), stores $(\mathsf{pk}, (m_0, m_1))$ in Q_0, and outputs $(m_0, m_1, \mathsf{aux} = (m_0, m_1))$.
2. On input a ciphertext ct^* and auxiliary information aux, Sim checks whether an entry $(\mathsf{ct}^*, \mathsf{st})$ is in Q_1. If so, it outputs st. If not, it randomly generates st, stores $(\mathsf{ct}^*, \mathsf{st})$ in Q_1, and outputs st.
3. On input $(\mathsf{pk}, \mathsf{sk}, \mathsf{st}, \mathsf{aux})$, Sim checks whether $(\mathsf{pk}, (m_0, m_1))$ is present in Q_0. If not, it returns $\perp$. Else, it computes $C = \{\mathsf{ct} | (\mathsf{ct}, \mathsf{st}) \in Q_1)\}$ and $M = \{m | m = \mathsf{IncPKE.Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}), \mathsf{ct} \in C\}$. It checks whether $(m_0, m_1) \cap M = \emptyset$ or $(m_0, m_1) \subseteq M$. If this check passes, it returns $\perp$. Else, if $m_0 \in M$, it returns 0. Otherwise, it returns 1.

The differences between $\mathcal{A}$ and Sim are that Sim generates random responses to the queries it receives and maintains a log of them. In the third type of query, it searches its database Q_1 to construct the set C. These differences lead to the following bad events that the reduction may use to distinguish $\mathcal{A}$ from Sim:

1. On input $(\mathsf{pk}, \mathsf{sk}, \mathsf{st}, \mathsf{aux})$, Sim may have early aborted because $(\mathsf{pk}, (m_0, m_1))$ wasn't present in Q_0. However, for the same input, $\mathcal{A}$ must have returned a bit b. This event happens when the reduction was able to guess the value of $h(\mathsf{ct}^*)$ without querying its oracle. Using simple guessing probability, one can show that this event is negligible in the size of the message.
2. On input $(\mathsf{pk}, \mathsf{sk}, \mathsf{st}, \mathsf{aux})$, Sim may have early aborted because $(m_0, m_1) \cap M = \emptyset$ whereas, for the same input, $\mathcal{A}$ must have returned a bit b. This event happens when the reduction was able to guess the value of $h(\mathsf{ct}^*)$ without querying its oracle such that ct^* decrypts to either m_0 or m_1. Simple guessing probability cannot be applied directly because the reduction may cleverly guess a *bad* secret key that decrypts the majority of the ciphertext to one of m_0 and m_1. However, using the fact that the secret key size is small, we argue that the number of such messages is negligible for every secret key. So, for a randomly chosen message m_0 and m_1, the messages are *good*, in the sense that around $2^{|\mathsf{ct}|}/2^{|m|}$ ciphertexts decrypt to m_0 and m_1. With this guarantee, we can apply simple guessing probability bounds to show that this event happening is negligible.
3. On input $(\mathsf{pk}, \mathsf{sk}, \mathsf{st}, \mathsf{aux})$, Sim may have returned a bit b whereas, for the same input, $\mathcal{A}$ must have returned $\perp$. We show that this scenario is very similar to the previous one and again use guessing probability bounds.

Since, the above events occurs with negligible probability, there does not exists any efficient reduction that can distinguish between $\mathcal{A}$ and Sim. Therefore, we have a simulatable attack against the security of a rate-1 incompressible scheme with small keys.

Simulatable Attack on Ciphertext Rate-1 Incompressible PKE with Leakage Rate-1. Assuming that the reader is familiar with the definition of leakage-resilient incompressibile security game (see Sect. 4.1), we describe our inefficient reduction $\mathcal{A}$, which has three hardcoded functions, f, g and h, as follows:

1. On input a secret key sk, $\mathcal{A}_0$ returns the truncation of sk to $|\mathsf{sk}| - o(|\mathsf{sk}|)$ bits.
2. On input a leakage z, $\mathcal{A}_1$ returns a pair of messages $(m_0, m_1) = g(z)$ and $\mathsf{aux} = (z, g(z))$.
3. On input a ciphertext $\mathsf{ct}^*, \mathsf{aux}$, $\mathcal{A}_1$ returns $h(\mathsf{ct}^*)$.
4. On input $(\mathsf{sk}, \mathsf{st}, \mathsf{aux} = (z, m_0, m_1))$, $\mathcal{A}_2$ checks whether $z = f(\mathsf{sk})$ and $(m_0, m_1) = g(z)$. If not, it returns $\perp$. Else, it computes $C = \{\mathsf{ct} | h(\mathsf{ct}) = \mathsf{st}\}$ and $M = \{m | m = \mathsf{IncPKE.Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}), \mathsf{ct} \in C\}$. It checks whether $(m_0, m_1) \cap M = \emptyset$ or $(m_0, m_1) \subseteq M$. If this check passes, it returns $\perp$. Else, if $m_0 \in M$, it returns 0. Otherwise, it returns 1.

Similar to the argument presented in the previous section, we can show that $\mathcal{A}$ wins the leakage-resilient incompressible game with very high probability. Now, let's present our simulator Sim which maintains two databases Q_0, Q_1, Q_2:

1. On input a secret key sk, Sim first checks whether an entry (sk, z) is in Q_0. It so, it outputs z, else it randomly generates z. It then stores (sk, z) in Q_0 and returns z.
2. On input a leakage z, Sim first checks whether an entry $(z, (m_0, m_1))$ is in Q_1. If so, it outputs $(m_0, m_1, \mathsf{aux} = (m_0, m_1))$. If not, it randomly generates a pair of messages (m_0, m_1), stores $(\mathsf{pk}, (m_0, m_1))$ in Q_1, and outputs $(m_0, m_1, \mathsf{aux} = (z, m_0, m_1))$..
3. On input a ciphertext $\mathsf{ct}^*, \mathsf{aux}$, Sim checks whether an entry $(\mathsf{ct}^*, \mathsf{st})$ is in Q_2. If so, it outputs st. If not, it randomly generates st, stores $(\mathsf{ct}^*, \mathsf{st})$ in Q_2, and outputs st.
4. On input $(\mathsf{sk}, \mathsf{st}, \mathsf{aux} = (z, m_0, m_1))$, Sim checks whether (sk, z) is present in Q_0 and (z, m_0, m_1) is present in Q_1. If not, it returns $\perp$. Else, it computes $C = \{\mathsf{ct} | (\mathsf{ct}, \mathsf{st}) \in Q_1\}$ and $M = \{m | m = \mathsf{IncSKE.Dec}(\mathsf{pk}, \mathsf{st}, \mathsf{ct}), \mathsf{ct} \in C\}$. It checks whether $(m_0, m_1) \cap M = \emptyset$ or $(m_0, m_1) \subseteq M$. If this check passes, it returns $\perp$. Else, if $m_0 \in M$, it returns 0. Otherwise, it returns 1.

Similar to the previous section, note that the distinctions between $\mathcal{A}$ and Sim lie in the fact that Sim generates random responses to the queries it receives and keeps a log of them. In the fourth type of query, it searches its database Q_2 to construct the set C. These differences give rise to bad events similar to those in the previous section. We address most of them in a similar manner, with the exception that, rather than relying on small key information, we leverage the advantage of substantial leakage of the secret key to argue indistinguishability.

Comparison with [28]. While our techniques for the above separation results are inspired by prior work [28], there are key technical differences which required new non-trivial ideas. At a high level, one of the main differences is that prior separation results [28] were for cryptosystems such as one-way functions or entropy-generators/condensers. Abstractly, all these cryptographic objects were just different types of one-way functions with some special properties (such as correlated input security, leakage-resilience etc.). Therefore, the objects were defined w.r.t. only a 'public' function key, and there was not a concept of a 'secret' key. Informally, thus meant that the primitives for which impossibilities existed did not have any real *correctness* property, but it was efficiency/succinctness and security. This meant that to show the separation result, they just had to design an inefficient (simulatable) attacker on the security of the system by leveraging the desired level of efficiency/succinctness for the object.

Unfortunately, it is unclear how this simple strategy can be directly extended to the encryption setting, since now there is also a 'secret' key which brings in a new (and much needed) correctness guarantee. Thus, our inefficient (simulatable) attacker must find a way to leverage both efficiency/succinctness and correctness. Clearly, if we do not use correctness while designing our inefficient (simulatable) attack, then we cannot prove the separation result since one could trivially design an *"incorrect"* encryption scheme that achieves optimal efficiency/succinctness as well as security. Thus, one of our key technical contributions in coming up with the above separation result is that we had to find a new approach to also leverage correctness along with succinctness. We believe our approach of generalizing the [28] black-box separation techniques to simultaneously leverage correctness and succinctness together to break security will have more applications in the future.

2.2 Positive Results

In this work, we present the first leakage resilient incompressible encryption schemes, both in the secret key and the public key setting as well as signature schemes.

LR Incompressible SKE Secure Against Unbounded Adversaries. At a high level, our leakage-resilient incompressible SKE scheme uses an incompressible SKE scheme for encryption, but applies some preprocessing of the secret key to protect against leakage. Specifically, in our scheme, the secret key is represented by a large random string sk. To encrypt a message m, a random seed s is sampled, and a truly random string $\mathsf{inc.sk}$ is derived by applying the extractor to sk using the seed s. Subsequently, $\mathsf{inc.sk}$ serves as the secret key for the incompressible SKE scheme to encrypt the message m to produce $\mathsf{inc.ct}$. The final ciphertext for the message is $(s, \mathsf{inc.ct})$. To decrypt, $\mathsf{inc.sk}$ is recovered using the secret key sk and s. Then, the decoder of the incompressible SKE scheme is invoked with secret key $\mathsf{inc.sk}$ and ciphertext $\mathsf{inc.ct}$.

To provide the intuition behind the leakage-resilient incompressible security of our scheme, we draw a parallel to the role of the extractor in leakage resilient encryption schemes: even if a substantial portion of the secret key sk is leaked to the adversary, if sk retains sufficient entropy after the leakage $f(\mathsf{sk})$, the resulting string $\mathsf{inc.sk}$ obtained by extraction is statistically close to uniform. Consequently, an unbounded adversary cannot learn any information about the secret key used in the inner incompressible SKE scheme. We should, however, exercise caution while arguing security against the adversary who eventually observes the leakage $f(\mathsf{sk})$, the compression of the ciphertext chosen adaptively based of the leakage, and the secret key sk. For instance, in the security game, the adversary can force the challenger to commit to the secret key and message using a hash functions of its choice while computing the leakage and compression making it impossible to efficiently and retroactively resample the secret key consistent with the commitments. In our proof, we carry out this resampling using a computationally unbounded program to make the secret key independent of the inner incompressible SKE scheme, and then reduce the security to the security of the inner incompressible SKE scheme. This necessitates incompressible SKE scheme to guarantee security against unbounded adversary.

Our construction inherits its compression rate from the inner the incompressible SKE scheme. Hence, instantiating it with the scheme proposed by Dziembowski [12] gives a sub-optimal ciphertext rate 1/3. To overcome this, we build our own incompressible SKE scheme with unbounded security and ciphertext rate 1/2.

Information-Theoretic Incompressible SKE with Ciphertext-Rate-$\frac{1}{2}$. In Dziembowski's scheme, the secret key sk serves as a seed for a randomness extractor.

To encrypt a message m, a large random string R is sampled and applied to the extractor (with seed sk) to produce k. This is then used in a *one-time-pad* operation on the message m to produce the ciphertext $(R, k \oplus m)$. We tweak this construction using seeded strong invertible extractor to get rid of the second component of the ciphertext *viz.* $k \oplus m$. The secret key for our construction is the seed s for a strong invertible extractor Ext, and a random string k of the same length as the message (which coincides with the output length of the invertible extractor). The ciphertext of a message m is sampled $\mathsf{ct} = \mathsf{Ext}_s^{-1}(m \oplus k)$, where Ext^{-1} is the inverter for the extractor that samples a random R such that $\mathsf{Ext}_s(R) = m \oplus k$. To decode, $m \oplus k$ is extracted from the ciphertext R using the seed s in the secret key using Ext; m is then recovered using k provided in the secret key. To argue security, we observe that, by the property of the inverter, and the fact that k is random, $(R, s, m \oplus k)$ is distributed identically as in the experiment in which R and s are sampled uniformly, and $m \oplus k$ is extracted from R using s using Ext. After this step, our security argument follows that in scheme proposed in [12].

LR Incompressible PKE from LR Non-committing Encryption. We present a hybrid construction for LR incompressible PKE that combines a (statistically secure) incompressible SKE with a leakage-resilient non-committing key encapsulation mechanism (NC-KEM). Essentially, a leakage-resilient NC-KEM is a stronger variant of NC-KEM which can handle leakage from the secret key. We formally define this primitive in Sect. 4.2 and present a construction based on hash proof systems in Sect. 6.

The public key pk and secret key sk of the PKE scheme will be the public and secret keys of the NC-KEM scheme. Using NC-KEM scheme, a fresh secret key $\mathsf{inc.sk}$ is generated by computing $(\mathsf{inc.sk}, \mathsf{nce.ct}) \leftarrow \mathsf{Encaps}(\mathsf{pk})$. The message m is then encrypted using $\mathsf{inc.sk}$ to produce $\mathsf{inc.ct} \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, m)$ and the final ciphertext is $\mathsf{ct} = (\mathsf{nce.ct}, \mathsf{inc.ct})$. To argue the scheme's security, we first set the NC-KEM to simulation mode. Note that the simulator does not have to be efficient because the leakage functions provided by the adversaries may not be efficiently computable (see Sect. 4.2 and Sect. 6). Thus, we can no longer use computationally secure incompressible SKE schemes because of these inefficient simulators will make the reduction inefficient. So we'll use a statistically secure incompressible SKE scheme to make our proof work. For more details, refer Sect. 7.

LR Incompressible PKE Against Bits-Leakage from Incompressible PKE. We present a transformation that converts any incompressible public-key encryption (PKE) scheme into an LR incompressible PKE secure against leakage functions that output a subset of the secret key bits. This is achieved using a reconstructable probabilistic encoding (RPE). An ℓ-RPE is an encoding where, for any input x and subset S with $|S| \leq \ell$, the partial encoding $\mathsf{Enc}(x)_S$ is perfectly indistinguishable from the uniform distribution. Additionally, there exists a reconstructing algorithm that, given x, a partial codeword $\tilde{w}$, and subset

S with $|\tilde{w}| = |S| \leq \ell$, outputs a complete codeword w from the distribution $\mathsf{Enc}(x)$ such that $w_S = \tilde{w}$.

The transformation is straightforward. The public key of the scheme is the public key $\mathsf{inc.pk}$ of an incompressible PKE scheme, while the secret key is the RPE of the secret key $\mathsf{RPE.Enc(inc.sk)}$. Any adversary capable of winning the LR incompressible game for this scheme can break the underlying incompressible PKE scheme. This is because the reduction can respond to queries with random values and later reconstruct a valid encoding for $\mathsf{inc.sk}$ during the second phase using the reconstructing algorithm of the RPE scheme. For more detail, see the full version.

LR Incompressible PKE from LR Incompressible SKE. We give a high level description of how to bootstrap any LR incompressible SKE scheme along with any regular PKE scheme to an LR incompressible PKE scheme (against general leakage) using *deferred encryption* technique.

The public key of the leakage-resilient incompressible scheme consists of a set of $2n$ public keys, denoted as $\left\{\mathsf{pk}_{i,b}\right\}_{i,b}$. On the other hand, the secret key of the scheme includes the secret key k of a leakage-resilient incompressible SKE scheme along with n PKE secret keys corresponding to the bits of k, i.e., $\{\mathsf{sk}_{i,k_i}\}_i$. To encrypt a message m, the encryptor garbles the encryption circuit of the leakage-resilient incompressible SKE scheme with the message m and hardcoded randomness r, i.e., $\mathsf{IncSKE.Enc}(\cdot, m; r)$. This garbling process yields a garbled circuit $\hat{C}$ along with $2n$ labels, given by $\{\mathsf{lab}_{i,b}\}_{i,b}$. Subsequently, each label $\mathsf{lab}_{i,b}$ is encrypted under its corresponding PKE public key $\mathsf{pk}_{i,b}$ to generate a set of ciphertexts, denoted as $\{\mathsf{ct}_{i,b}\}_{i,b}$. The ciphertext consists of the garbled circuit $\hat{C}$ along with the set of ciphertexts $\{\mathsf{ct}_{i,b}\}_{i,b}$. For decryption, the decryptor simply decrypts $\{\mathsf{ct}_{i,k_i}\}_i$ using the corresponding secret keys $\{\mathsf{sk}_{i,k_i}\}_i$. This enables the decryptor to acquire the labels $\{\mathsf{lab}_{i,k}\}_i$, which can be utilized to evaluate the garbled circuit $\hat{C}$ and derive an incompressible SKE ciphertext $\mathsf{inc.ct}$. By utilizing the incompressible SKE key k, the decryptor can perform the final decryption to obtain the original message m.

The above approach postpones the actual encryption of the message m until the decryption phase. That is, the actual $\mathsf{inc.ct}$ is generated only when the secret key is acquired. Using this fact, in the security game, we are able to equivocate the challenge ciphertext to appear as an encryption under any key k during the second phase. It's important to emphasize that we need to be careful to make sure that the leaking of the secret keys $\{\mathsf{sk}_{i,k_i}\}_i$ does not cause problems for the equivocation process. This aspect is appropriately addressed in our proof. For completeness, we provide our LR incompressible PKE construction in the full version of our paper.

LR Incompressible Signatures. We define leakage-resilient incompressible signatures by adding the leakage-resilience property to the security definition of incompressible signatures as mentioned in [19]. The authors of that paper constructed incompressible signatures using a standard signature scheme along with an incompressible encoding scheme (defined in [23]). We provide a similar construction of leakage-resilient incompressible signatures using a leakage-resilient signature scheme along with an incompressible encoding scheme. For more detail, see the full version.

3 Preliminaries

Throughout this paper, we will use λ to denote the security parameter and $\text{negl}(\cdot)$ to denote a negligible function in the input. We will use the short-hand notation PPT for "probabilistic polynomial time". We use the notation $\mathcal{A}^{\mathcal{B}(\cdot)}$ to denote that the algorithm $\mathcal{A}$ can issue multiple queries to its oracle $\mathcal{B}$ and receives the outputs of $\mathcal{B}$ executed on these queries.

For any finite set X, $x \leftarrow X$ denotes the process of picking an element x from X uniformly at random. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from the distribution $\mathcal{D}$. We will use the notation $\approx_\epsilon$ and $\equiv$ to denote that two distributions are ϵ-close and identical respectively. For any natural number $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For any two binary string x and y, $x||y$ denotes the concatenation of x and y. For ℓ-length binary string x, x_i denotes the i^{th} bit of x.

For any function h and a set X that is a subset of the domain of h, we use the short-hand notation $h(X)$ to denote the set $\{h(x)\}_{x \in X}$.

3.1 Extractors

Definition 1. *For two random variables X and Y taking values in $\mathscr{U}$, their statistical distance is $\triangle(X, Y) = \max_{T \subseteq \mathscr{U}} |\Pr[X \in T] - \Pr[Y \in T]|$. We say that X and Y are ϵ-close if $\triangle(X, Y) \leq \epsilon$.*

Definition 2. *An extractor $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^m$ is a strong (k, ϵ)-extractor if for every distribution X on $\{0,1\}^n$ such that $H_\infty(X) \geq k$, $(U_d, \mathsf{Ext}(U_d, X))$ is ϵ-close to (U_d, U_m). Here, U_d denotes the uniform distribution on the set $\{0,1\}^d$ and the min-entropy $H_\infty(X) = \min_x \left\{\log \frac{1}{\Pr[X=x]}\right\}$.*

Chapter 6 of [27] describes many randomness extractors out of which we will be using the following.

Theorem 6 (Corollary 6.40, [27]). *For every $n \in \mathbb{N}, k \in [0, n]$ and $\epsilon > 0$, there is an explicit strong (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^m$ with $m = k - O(\log(1/\epsilon))$ and $d = O(\log k \cdot \log(n/\epsilon))$.*

We require $\epsilon = 2^{-\lambda}$ for our applications. We can obtain the following corollary by substituting this value in the above theorem.

Corollary 1. *Let $\lambda \in \mathbb{N}$. For every $n \in \mathbb{N}$ and $k \in [0, n]$ such that $n = \mathsf{poly}(\lambda)$ and $k = \mathsf{poly}(\lambda)$, there is an explicit strong $(k, 2^{-\lambda})$-extractor $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^m$ with $m = k - O(\lambda)$ and $d = \tilde{O}(\lambda)$ where $\tilde{O}$ hides all logarithmic factors.*

3.2 Incompressible Public Key Encryption

An incompressible public key encryption scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is a tuple of PPT algorithms with the following properties. For any security parameter λ and message-length parameter $\ell_{\mathtt{msg}}$, there exists sets $\mathcal{M}_{\ell_{\mathtt{msg}}}$ (message space), $\mathcal{SK}_{\lambda,\ell_{\mathtt{msg}}}$ (secret-key space), $\mathcal{PK}_{\lambda,\ell_{\mathtt{msg}}}$ (public-key space), and $\mathcal{C}_{\lambda,\ell_{\mathtt{msg}}}$ (ciphertext space), from which the inputs and outputs for these algorithms are taken.

- $\mathsf{Setup}\left(1^{\lambda}, 1^{\ell_{\mathtt{msg}}}\right)$: The setup algorithm takes as input the security parameter λ, the message size $1^{\ell_{\mathtt{msg}}}$ and outputs a public key $\mathsf{pk} \in \mathcal{PK}_{\lambda,\ell_{\mathtt{msg}}}$ and a secret key $\mathsf{sk} \in \mathcal{SK}_{\lambda,\ell_{\mathtt{msg}}}$.
- $\mathsf{Enc}(\mathsf{pk}, m)$: The encryption algorithm takes as input a public key $\mathsf{pk} \in \mathcal{PK}_{\lambda,\ell_{\mathtt{msg}}}$ and a message $m \in \mathcal{M}_{\ell_{\mathtt{msg}}}$ and outputs a ciphertext $\mathsf{ct} \in \mathcal{C}_{\lambda,\ell_{\mathtt{msg}}}$.
- $\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct})$: The deterministic decryption algorithm takes $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{PK}_{\lambda,\ell_{\mathtt{msg}}} \times \mathcal{SK}_{\lambda,\ell_{\mathtt{msg}}}$ and $\mathsf{ct} \in \mathcal{C}_{\lambda,\ell_{\mathtt{msg}}}$ as inputs, and outputs either a message $m \in \mathcal{M}_{\ell_{\mathtt{msg}}}$ or $\perp$.

Correctness. For correctness, we require that there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, for all polynomials $\ell_{\mathtt{msg}} = \ell_{\mathtt{msg}}(\lambda)$, for all $m \in \mathcal{M}_{\ell_{\mathtt{msg}}}$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}\left(1^{\lambda}, 1^{\ell_{\mathtt{msg}}}\right)$,

$$\Pr_r[\mathsf{Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}) = m \mid \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m; r)] = 1 - \mathrm{negl}(\lambda).$$

Incompressible PKE Security. Consider the following experiment with an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$. Let $\ell_{\mathtt{st}} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ be a function that denotes the adversary's long-term storage capacity as a function of the security parameter and message size.

- **Initialization Phase:** $\mathcal{A}_0$ on input 1^{λ}, outputs $1^{\ell_{\mathtt{msg}}}$. The challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{\ell_{\mathtt{msg}}})$ and sends pk to $\mathcal{A}_0$.
- **Challenge Phase:** $\mathcal{A}_0$ outputs two message m_0, m_1, along with an auxiliary information aux. The challenger randomly chooses $b \in \{0, 1\}$ and computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{pk}, m_b)$ and sends it to $\mathcal{A}_1$.
- **First Response Phase:** $\mathcal{A}_1$ computes a state st such that $|\mathsf{st}| \leq \ell_{\mathtt{st}}(\lambda, \ell_{\mathtt{msg}})$.
- **Second Response Phase:** $\mathcal{A}_2$ receives $(\mathsf{pk}, \mathsf{sk}, \mathsf{aux}, \mathsf{st})$ and outputs b'. $\mathcal{A}$ wins the experiment if $b = b'$

Definition 3. *A PKE scheme is said to be $\ell_{\mathtt{st}}$-incompressible secure if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* negl *such that for all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathcal{A} \textit{ wins in the above experiment}] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

Two important parameters of an incompressible encryption scheme are ciphertext-rate and compression-rate defined as follows.

Definition 4 (Ciphertext-Rate). *Let $\ell_{\texttt{ct}}(\lambda, \ell_{\texttt{msg}})$ denote the size of ciphertext (as a function of the security parameter λ and the message size $\ell_{\texttt{msg}}$). The ciphertext-rate of a encryption scheme is defined as the ratio of the size of the message $\ell_{\texttt{msg}}$ to the size of the ciphertext $\ell_{\texttt{ct}}(\lambda, \ell_{\texttt{msg}})$ as $\ell_{\texttt{msg}}$ tends to infinity, or more precisely,*

$$\textsf{ct-rate}(\lambda) = \liminf_{\ell_{\texttt{msg}} \to \infty} \frac{\ell_{\texttt{msg}}}{\ell_{\texttt{ct}}(\lambda, \ell_{\texttt{msg}})}.$$

Definition 5 (Compression-Rate). *The compression rate of an incompressible encryption scheme is defined as the ratio of the size of the state $\ell_{\texttt{st}}(\lambda, \ell_{\texttt{msg}})$ to the size of the ciphertext as $\ell_{\texttt{msg}}$ tends to infinity, or more precisely,*

$$\textsf{comp-rate}(\lambda) = \liminf_{\ell_{\texttt{msg}} \to \infty} \frac{\ell_{\texttt{st}}(\lambda, \ell_{\texttt{msg}})}{\ell_{\texttt{ct}}(\lambda, \ell_{\texttt{msg}})}.$$

Guan *et al.* [19] gave the first scheme with optimal ciphertext-rate and compression-rate using indistinguishable obfuscators (iO) [13,14,26]. Later, Branco *et al.* [4] gave a construction from standard assumptions.

Theorem 7 ([4]). *Assuming the hardness of* LWE *problem or* DDH *along with* DCR *problem, for any $\ell_{\texttt{st}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, there exists a $\ell_{\texttt{st}}$-incompressible PKE scheme such that* $\textsf{ct-rate}(\lambda) = 1$ *and* $\textsf{comp-rate}(\lambda) = 1$.

Depending on the number of ciphertexts that decrypts to a particular message using a given public-secret key, we categorise messages as good or bad. Intuitively, we would expect each message to possess an equal proportion of the ciphertexts. If the proportion significantly exceeds the average, we label it as 'bad'. Additionally, we also categorise public-secret keys as good or bad based on the number of bad messages associated with it. The proof of the impossibility results will heavily rely on these definitions.

Definition 6. *Let $\lambda \in \mathbb{N}, \ell_{\texttt{msg}} \in \mathbb{N}, \epsilon \in (0,1)$ and $(\textsf{pk}, \textsf{sk}) \in \mathcal{PK}_{\lambda,\ell_{\texttt{msg}}} \times \mathcal{SK}_{\lambda,\ell_{\texttt{msg}}}$. A message $m \in \mathcal{M}_{\lambda,\ell_{\texttt{msg}}}$ is said to be ϵ-**bad** with respect to $(\textsf{pk}, \textsf{sk})$ if*

$$\Pr_{\textsf{ct} \leftarrow \mathcal{C}_\lambda}[\textsf{Dec}(\textsf{pk}, \textsf{sk}, \textsf{ct}) = m] > 1/2^{(\epsilon/2)\cdot \ell_{\texttt{msg}}}$$

*A message $m \in \mathcal{M}_{\lambda,\ell_{\texttt{msg}}}$ is said to be ϵ-**good** with respect to $\textsf{pk} \in \mathcal{PK}_{\lambda,\ell_{\texttt{msg}}}$ if it is not ϵ-bad with respect to any $(\textsf{pk}, \textsf{sk})$ where $\textsf{sk} \in \mathcal{SK}_{\lambda,\ell_{\texttt{msg}}}$. A public-secret key pair $(\textsf{pk}, \textsf{sk}) \in \mathcal{PK}_{\lambda,\ell_{\texttt{msg}}} \times \mathcal{SK}_{\lambda,\ell_{\texttt{msg}}}$ is said to be ϵ-**bad** if*

$$\Pr_{m \leftarrow \mathcal{M}_{\lambda,\ell_{\texttt{msg}}}}[m \textit{ is bad wrt } (\textsf{pk}, \textsf{sk})] > 1/2^{(1-\epsilon/2)\cdot \ell_{\texttt{msg}}}$$

4 Leakage-Resilient Incompressible and Non-committing Schemes: Definitions

4.1 Leakage-Resilient Incompressible Encryption

The syntax for leakage-resilient SKE/PKE is similar to (regular) SKE/PKE. Let $\ell_{\texttt{st}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and $\ell_{\texttt{leak}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be functions denoting the adversary's long-term storage and leakage, as a function of the security parameter and the message length. In the security game, we will consider two adversaries $\mathcal{A}_1, \mathcal{A}_2$. The first adversary $\mathcal{A}_1$, after receiving leakage on the secret key, will be provided with the complete challenge ciphertext, and it produces a compressed digest. The second adversary $\mathcal{A}_2$ is provided with the public and the secret key along with the compressed challenge ciphertext which was created by $\mathcal{A}_1$. Since the definitions of leakage-resilient incompressible SKE security and its PKE version are similar, we present only the SKE version.

Leakage-Resilient-Incompressible SKE Security.
Let $\mathsf{SKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be an SKE scheme which, for any value of the security parameter $\lambda \in \mathbb{N}$ and message-length $\ell_{\mathtt{msg}} \in \mathbb{N}$, has message space $\mathcal{M}_{\ell_{\mathtt{msg}}}$, secret key space $\mathcal{SK}_{\lambda,\ell_{\mathtt{msg}}}$ and ciphertext space $\mathcal{C}_{\lambda,\ell_{\mathtt{msg}}}$. Consider the following experiment with an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

- **Initialization Phase:** $\mathcal{A}_1$ on input 1^λ, outputs an upper bound on the message size $\ell_{\mathtt{msg}}$. The challenger runs $\mathsf{sk} \leftarrow \mathsf{Setup}\left(1^\lambda, 1^{\ell_{\mathtt{msg}}}\right)$.
- **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}_1$ is allowed to make a query f. The challenger computes $f(\mathsf{sk})$ and returns it to $\mathcal{A}_1$.
- **Challenge Phase:** $\mathcal{A}_1$ outputs two messages m_0, m_1, along with an auxiliary information aux. The challenger randomly chooses $b \in \{0,1\}$ and computes a ciphertext $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{sk}, m_b)$ and sends it to $\mathcal{A}_1$.
- **First Response Phase:** $\mathcal{A}_1$ computes a state st such that $|st| \leq \ell_{\mathtt{st}}(\lambda, \ell_{\mathtt{msg}})$.
- **Second Response Phase:** $\mathcal{A}_2$ receives (sk, aux, st). Finally, $\mathcal{A}_2$ outputs b'. $\mathcal{A}$ wins the experiment if $b = b'$.

Definition 7. *An SKE scheme is said to be $(\ell_{\mathtt{st}}, \ell_{\mathtt{leak}})$-leakage-resilient incompressible secure if for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function* $\text{negl}(\cdot)$ *such that* $\Pr[\mathcal{A}$ *wins in the above experiment*$] = \frac{1}{2} + \text{negl}(\lambda)$ *provided $|f(\mathsf{sk})| \leq \ell_{\mathtt{leak}}(\lambda, \ell_{\mathtt{msg}})$ where f is the query made by the first adversary $\mathcal{A}_1$ in the pre-challenge query phase.*

Definition 8 (Leakage-Rate). *Let $\ell_{\mathtt{key}}(\lambda, \ell_{\mathtt{msg}})$ denote the size of the secret key output by* $\mathsf{Setup}\left(1^\lambda, 1^{\ell_{\mathtt{msg}}}\right)$. *The leakage rate of an $(\ell_{\mathtt{st}}, \ell_{\mathtt{leak}})$-leakage-resilient incompressible secure is defined as the ratio between $\ell_{\mathtt{leak}}$ and the size of the secret key as $\ell_{\mathtt{msg}}$ tends to infinity, i.e.,*

$$\mathsf{sk\text{-}rate}(\lambda) = \liminf_{\ell_{\mathtt{msg}} \to \infty} \frac{\ell_{\mathtt{leak}}(\lambda, \ell_{\mathtt{msg}})}{\ell_{\mathtt{key}}(\lambda, \ell_{\mathtt{msg}})}.$$

4.2 Leakage-Resilient Non-committing Key Encapsulation Mechanism

In this section, we define the leakage resilient non-committing key encapsulation mechanism (NC-KEM). A leakage-resilient NC-KEM $\mathsf{NCE} = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps}, \mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ with secret key space $\left\{\mathcal{SK}_{\lambda,\ell_{\mathtt{key}}}\right\}_{\lambda,\ell_{\mathtt{key}}}$, public key space $\left\{\mathcal{PK}_{\lambda,\ell_{\mathtt{key}}}\right\}_{\lambda,\ell_{\mathtt{key}}}$ and key space $\left\{\mathcal{K}_{\lambda,\ell_{\mathtt{key}}}\right\}_{\lambda,\ell_{\mathtt{key}}}$ consists of the following algorithms, of which the first three are probabilistic polynomial time.

- $\mathsf{Setup}\left(1^\lambda, 1^{\ell_{\mathsf{key}}}\right)$: The setup algorithm takes as input the security parameter 1^λ and encapsulation key size $1^{\ell_{\mathsf{key}}}$ and outputs a public key pk and a secret key sk.
- $\mathsf{Encaps}(\mathsf{pk})$: The encapsulation algorithm takes as input a public key pk, produces a key $k \in \{0,1\}^{\ell_{\mathsf{key}}}$ and a ciphertext ct.
- $\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$: The decapsulation algorithm takes as input a secret key sk and a ciphertext ct and outputs $k \in \{0,1\}^{\ell_{\mathsf{key}}}$.
- $\mathsf{Sim}_1\left(1^\lambda, 1^{\ell_{\mathsf{key}}}\right)$: The first simulator algorithm takes as input the security parameter 1^λ and encapsulation key size $1^{\ell_{\mathsf{key}}}$ and outputs a public key pk, a ciphertext ct and a state state_1.
- $\mathsf{Sim}_2(\mathsf{state}_1, f)$: The second inefficient[3] simulator algorithm takes as input a state state_1 and a function $f : \mathcal{SK}_{\lambda,\ell_{\mathsf{key}}} \to \{0,1\}^{\ell_{\mathsf{leak}}}$ and outputs a string $r \in \{0,1\}^{\ell_{\mathsf{leak}}}$ and a state state_2.
- $\mathsf{Sim}_3(\mathsf{state}_2, k)$: The third inefficient[4] simulator algorithm takes as input a state state_2 and a string $k \in \{0,1\}^{\ell_{\mathsf{key}}}$ and outputs a secret key sk.

Correctness. For correctness, we require that for all $\lambda \in \mathbb{N}, \ell_{\mathsf{key}} \in \mathbb{N}$ and $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{Setup}\left(1^\lambda, 1^{\ell_{\mathsf{key}}}\right)$,

$$\Pr_r[\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct}) = k \mid (k, \mathsf{ct}) = \mathsf{Encaps}(\mathsf{pk}; r)] = 1 - \text{negl}(\lambda)$$

Leakage-Resilient Non-committing Security. Consider the following two experiments with an adversary $\mathcal{A}$.

Real World:

- **Initialization Phase:** $\mathcal{A}$ on input 1^λ, outputs the key size $1^{\ell_{\mathsf{key}}}$. The challenger computes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{key}}})$ and sends pk to $\mathcal{A}$.
- **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}$ is allowed to make a query f. The challenger returns $r = f(\mathsf{sk})$ to $\mathcal{A}$.
- **Challenge Phase:** The challenger computes $(k, \mathsf{ct}) \leftarrow \mathsf{Encaps}(\mathsf{pk})$ and returns $(\mathsf{sk}, k, \mathsf{ct})$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b.

Simulated World:

- **Initialization Phase:** $\mathcal{A}$ on input 1^λ, outputs the key size $1^{\ell_{\mathsf{key}}}$. The challenger computes $(\mathsf{pk}, \mathsf{ct}, \mathsf{state}_1) \leftarrow \mathsf{Sim}_1(1^\lambda, 1^{\ell_{\mathsf{key}}})$ and sends pk to $\mathcal{A}$.
- **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}$ is allowed to make a query f. The challenger computes $(r, \mathsf{state}_2) \leftarrow \mathsf{Sim}_2(\mathsf{state}_1, f)$ and returns r to $\mathcal{A}$.
- **Challenge Phase:** The challenger randomly samples $k \leftarrow \{0,1\}^{\ell_{\mathsf{key}}}$, computes $\mathsf{sk} \leftarrow \mathsf{Sim}_3(\mathsf{state}_2, k)$, and returns $(\mathsf{sk}, k, \mathsf{ct})$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b.

[3] f may not be an efficiently computable function.

[4] Even if the function f is efficiently computable, it may not be efficiently invertible. The third simulator must generate a (simulated) secret key sk such that $f(\mathsf{sk})$ must match with the output of the second simulator. See Definition 9.

Let p_{real} and p_{sim} be the probabilities with which $\mathcal{A}$ outputs 0 in the real world and simulated world, respectively.

Definition 9. *A $\ell_{\mathtt{leak}}$-LR-NC-KEM scheme is said to be secure if for all* PPT *adversaries $\mathcal{A}$, there exists a negligible function* $\text{negl}(\cdot)$ *such that, for all $\lambda \in \mathbb{N}$,*

$$|p_{\text{real}} - p_{\text{sim}}| = \text{negl}(\lambda),$$

provided $|f(\mathsf{sk})| \leq \ell_{\mathtt{leak}}$ where f is the query made by the adversary $\mathcal{A}$ in the pre-challenge query phase.

4.3 Leakage-Resilient Incompressible Signatures

An $\ell_{\mathtt{leak}}$-leakage-resilient incompressible signature scheme Sig with a signing-key space $\{0,1\}^{\ell_{\mathtt{key}}}$, message space $\{0,1\}^{\ell_{\mathtt{msg}}}$ and signature space $\{0,1\}^{\ell_{\mathtt{sign}}}$, consists of a tuple of PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, with the following syntax:

- $\mathsf{Gen}(1^\lambda, 1^{\ell_{\mathtt{st}}})$: The generation algorithm takes as input the security parameter 1^λ and compression size $1^{\ell_{\mathtt{st}}}$ and outputs $(\mathsf{vk}, \mathsf{sk})$, where vk is the verification key and sk is the signing key.
- $\mathsf{Sign}(\mathsf{sk}, m)$: The signing algorithm takes as input the signing key sk and a message m, and outputs a signature σ.
- $\mathsf{Verify}(\mathsf{vk}, \sigma)$: The verification algorithm takes as input the verification key vk and a signature σ, and outputs a message m' or $\perp$.

The leakage-rate of the scheme is defined as $\ell_{\mathtt{leak}}/\ell_{\mathtt{key}}$. The signature-rate is defined as $\ell_{\mathtt{msg}}/\ell_{\mathtt{sign}}$. The compression-rate is defined as $\ell_{\mathtt{st}}/\ell_{\mathtt{sign}}$.
Correctness. For all security parameters $\lambda, \ell_{\mathtt{st}} \in \mathbb{N}$, all message lengths $\ell_{\mathtt{msg}}$, and all messages $m \in \{0,1\}^{\ell_{\mathtt{msg}}}$, over the randomness of $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, 1^{\ell_{\mathtt{st}}})$, and $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, it holds that

$$\Pr[\mathsf{Verify}(\mathsf{vk}, \sigma) \neq m] \leq \text{negl}(\lambda).$$

Security. Consider the following experiment for adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

Leakage-Resilient Incompressible Unforgeability Experiment $\mathsf{Expt}^{\text{LeakIncompSig}}_{\mathcal{A},\mathsf{Sig}}(\lambda, \ell_{\mathtt{leak}})$*:*

1. **Initialization Phase:** On input 1^λ, the adversary $\mathcal{A}_1$ specifies the state size $\ell_{\mathtt{st}}$. The challenger samples $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, 1^{\ell_{\mathtt{st}}})$ and provides vk to $\mathcal{A}_1$.
2. **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}_1$ is allowed to make polynomially many leakage queries. For j^{th} query f_j, the challenger provides $f_j(\mathsf{sk})$ to $\mathcal{A}_1$, as long as $\sum_j |f_j(\cdot)| \leq \ell_{\mathtt{leak}}$. At the end of the last round, $\mathcal{A}_1$ provides auxiliary information aux to the challenger.
3. **Challenge Phase:** For polynomially many rounds, $\mathcal{A}_1$ requests for signatures on messages. In the i^{th} query, it queries for message $m_i \in \{0,1\}^{\ell_{\mathtt{msg}}}$, and the challenger computes $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, m_i)$ and sends σ_i back to $\mathcal{A}_1$.

4. **First Response Phase:** $\mathcal{A}_1$ sends a state st such that $|\mathsf{st}| \leq \ell_{\mathtt{st}}$.
5. **Second Response Phase:** $\mathcal{A}_2$ gets the verification key vk, the auxiliary information aux, and the state st, and outputs a signature σ'. If $\mathsf{Verify}(\mathsf{vk}, \sigma')$ output $\perp$, then the challenger outputs 0. Otherwise, it outputs 1.

Definition 10 ($\ell_{\mathtt{leak}}$-Leakage-Resilient Incompressible Unforgeability). *A signature scheme* Sig *is said to be* $\ell_{\mathtt{leak}}$*-leakage-resilient incompressible unforgeable if for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, all* $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{Expt}^{\mathrm{LeakIncompSig}}_{\mathcal{A},\mathsf{Sig}}(\lambda, \ell_{\mathtt{leak}}) = 1] \leq \mathrm{negl}(\lambda).$$

5 Leakage-Resilient Incompressible SKE Against Unbounded Adversaries

In this section, we present a leakage-resilient incompressible SKE. We first improve the current state-of-the-art with regard to information-theoretic incompressible SKE. Next, we show how to transform any information-theoretic incompressible SKE scheme into an information-theoretic leakage-resilient incompressible SKE scheme.

5.1 Information-Theoretic Incompressible SKE with Ciphertext-Rate $\frac{1}{2}$

In this section, we build an incompressible SKE scheme with ciphertext-rate $\frac{1}{2}$ and having security against unbounded adversaries. The construction using *invertible seeded extractors*. These are extractors for which, given a seed and an output, a random sample from the source randomness can be efficiently sampled. A stronger version of this notion was introduced by Cheraghchi *et al.* [6].

Definition 11. *A strong* (k, ϵ)*-extractor* $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ *is said to be* ν*-invertible if there exists a probabilistic polynomial (in* d *and* n*) time algorithm* Ext^{-1} *with the following properties.* Ext^{-1} *takes an input a seed* $s \in \{0,1\}^d$ *and a string* $o \in \{0,1\}^m$ *and outputs a string* $R \in \{0,1\}^n$.

1. ***Inversion.*** *The probability that* Ext^{-1} *fails to output a string* R *such that* $\mathsf{Ext}_s(R) = o$ *is bounded by* ν*, i.e.,*

$$\Pr\left[\mathsf{Ext}_s(R) \neq o \;\middle|\; o \leftarrow \{0,1\}^m, s \leftarrow \{0,1\}^d, R \leftarrow \mathsf{Ext}_s^{-1}(o)\right] \leq \nu$$

2. ***Uniformity.*** *The distribution* $(s, R, \mathsf{Ext}_s(R))$ *is* ν*-close to* $(s, \mathsf{Ext}_s^{-1}(o), o)$*, i.e.,*

$$\triangle\left(\left(s, \mathsf{Ext}_s^{-1}(o), o \;\middle|\; \begin{matrix} o \leftarrow \{0,1\}^m \\ s \leftarrow \{0,1\}^d \end{matrix}\right), \left(s, R, \mathsf{Ext}_s(R) \;\middle|\; \begin{matrix} R \leftarrow \{0,1\}^n \\ s \leftarrow \{0,1\}^d \end{matrix}\right)\right) \leq \nu$$

Our Incompressible SKE Scheme. Let $\mathsf{Ext} : \{0,1\}^{\ell_{\mathtt{key}}} \times \{0,1\}^{\ell_{\mathtt{ct}}} \to \{0,1\}^{\ell_{\mathtt{msg}}}$ be a ν-invertible strong (k, ϵ)-extractor with inverter Ext^{-1} where $k \geq \ell_{\mathtt{ct}} - \ell_{\mathtt{st}}$. Our incompressible SKE scheme is as follows.

– $\mathsf{Setup}(1^\lambda, 1^{\ell_{\mathtt{msg}}}, 1^{\ell_{\mathtt{st}}})$: The setup algorithm generates $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$ where $\mathsf{sk}_1 \leftarrow \{0,1\}^{\ell_{\mathtt{key}}}$ and $\mathsf{sk}_2 \leftarrow \{0,1\}^{\ell_{\mathtt{msg}}}$.
– $\mathsf{Enc}(\mathsf{sk}, m)$: Let $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$ be the secret key and $m \in \{0,1\}^{\ell_{\mathtt{msg}}}$ be a plaintext message. The encryption algorithm computes $\mathsf{ct} = \mathsf{Ext}^{-1}_{\mathsf{sk}_1}(m \oplus \mathsf{sk}_2)$, i.e., ct is the output of the inverter on inputs sk_1 and $m \oplus \mathsf{sk}_2$. It outputs ct.
– $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Let $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$, the decryption algorithm outputs $\mathsf{Ext}_{\mathsf{sk}_1}(\mathsf{ct}) \oplus \mathsf{sk}_2$.

Correctness: We will show that the decoder is incorrect with probability at most ν. Let $m \in \{0,1\}^{\ell_{\mathtt{msg}}}$ a message and $\mathsf{ct} = \mathsf{Ext}^{-1}(sk_1, m \oplus \mathsf{sk}_2)$ be a ciphertext encrypting m. By the definition of Dec, decoding is correct if and only if $\mathsf{Ext}_{\mathsf{sk}_1}(\mathsf{ct}) = m \oplus \mathsf{sk}_2$. Since sk_2 is uniformly chosen from the same domain, $\mathsf{sk}_2 \oplus m$ is also uniformly distributed. Further, sk_1 is chosen uniformly from $\{0,1\}^{\ell_{\mathtt{key}}}$. From the inversion property of Ext^{-1},

$$\Pr\left[\mathsf{Ext}_{\mathsf{sk}_1}(\mathsf{ct}) \neq m \oplus \mathsf{sk}_2\right] = \Pr\left[\mathsf{Ext}_{\mathsf{sk}_1}(\mathsf{Ext}^{-1}_{\mathsf{sk}_1}(\mathsf{sk}_2 \oplus m)) \neq m \oplus \mathsf{sk}_2\right] \leq \nu.$$

Hence, $\Pr\left[\mathsf{Dec}(\mathsf{Enc}(\mathsf{sk}, m)) \neq m\right] \leq \nu$, for all m.

Theorem 8. *Assuming* $\mathsf{Ext} : \{0,1\}^{\ell_{\mathtt{key}}} \times \{0,1\}^{\ell_{\mathtt{ct}}} \to \{0,1\}^{\ell_{\mathtt{msg}}}$ *is a ν-invertible strong (k, ϵ)-extractor where $\ell_{\mathtt{ct}} - \ell_{\mathtt{st}} \geq k$ and both $\epsilon = \epsilon(\lambda)$ and $\nu = \nu(\lambda)$ are negligible functions, then the above construction is an incompressible SKE scheme secure against unbounded adversaries. The ciphertext-rate and compression-rate are equal to $\frac{1}{2}$, and the scheme has secret keys of size $4 \cdot \max(\ell_{\mathtt{st}}, \ell_{\mathtt{msg}})$.*

Due to space constraints, the proof of this theorem is included in the full version.

5.2 Generic Transformation

Our Construction. Let $\mathsf{IncSKE} = (\mathsf{IncSKE.Setup}, \mathsf{IncSKE.Enc}, \mathsf{IncSKE.Dec})$ be an incompressible secret-key encryption scheme for arbitrary message length $\ell_{\mathtt{msg}}$ with ciphertext length $\ell^{\mathsf{IncSKE}}_{\mathsf{ct}}$ and secret key length $\ell^{\mathsf{IncSKE}}_{\mathsf{key}}$. Also, the $\mathsf{IncSKE.Setup}$ algorithm generates a secret key by sampling a truly random string. Let $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^{\ell_{\mathtt{key}}} \to \{0,1\}^{\ell^{\mathsf{IncSKE}}_{\mathsf{key}}}$ be a strong $(\ell^{\mathsf{IncSKE}}_{\mathsf{key}} + O(\lambda), \mathrm{negl}(\lambda))$-extractor where $d = \mathsf{poly}(\lambda)$[5]. Then, our leakage-resilient incompressible SKE scheme is as follows.

– $\mathsf{Setup}(1^\lambda, 1^{\ell_{\mathtt{msg}}}, 1^{\ell_{\mathtt{st}}})$: The setup algorithm generates a random string $\mathsf{sk} \leftarrow \{0,1\}^{\ell_{\mathtt{key}}}$[6] and outputs it.

[5] The parameters are chosen from Corollary 1.
[6] where $\ell_{\mathtt{key}}$ must be greater $\ell^{\mathsf{IncSKE}}_{\mathsf{key}} + O(\lambda)$. This is due to the fact that the extractor requires a string from the randomness source that has $\ell^{\mathsf{IncSKE}}_{\mathsf{key}} + O(\lambda)$ bits of entropy.

– $\mathsf{Enc}(\mathsf{sk}, m)$: Let sk be the secret key and $m \in \{0,1\}^{\ell_{\mathsf{msg}}}$ be a message. The encryption algorithm first samples a random seed $s \leftarrow \{0,1\}^d$ and then computes $k := \mathsf{Ext}_s(\mathsf{sk})$. It returns $\mathsf{ct} := (s, \mathsf{IncSKE.Enc}(k, m))$.
– $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Let sk be the secret key and $\mathsf{ct} = (s', \mathsf{ct}')$ be a ciphertext. The decryption algorithm first computes $k := \mathsf{Ext}_{s'}(\mathsf{sk})$ and then decrypts ct' using k, i.e., $m = \mathsf{IncSKE.Dec}(k, \mathsf{ct})$. It returns m.

Correctness. An encryption of a message m under a secret key sk using the above scheme is $(\mathsf{ct}_0, \mathsf{ct}_1) = (s, \mathsf{IncSKE.Enc}(k, m))$ where s is randomly generated and $k = \mathsf{Ext}_s(\mathsf{sk})$. To decrypt this ciphertext using sk, the decryption algorithm first computes $\mathsf{Ext}_{\mathsf{ct}_0}(\mathsf{sk})$ which is equal to k. Then, it decrypts ct_1 using k by computing $m' := \mathsf{IncSKE.Dec}(k, \mathsf{ct}_1)$. From the correctness of IncSKE, we have $m = m'$.

Ciphertext-Rate. The size of the ciphertext in the above scheme is $\mathsf{poly}(\lambda) + \ell_{\mathsf{ct}}^{\mathsf{IncSKE}}$ because there exists strong extractor with $d = \mathsf{poly}(\lambda)$ (see Theorem 6). Therefore, the ciphertext-rate is $\left(\frac{\mathsf{poly}(\lambda)}{\ell_{\mathsf{msg}}} + \frac{\ell_{\mathsf{ct}}^{\mathsf{IncSKE}}}{\ell_{\mathsf{msg}}}\right)^{-1}$ which is equal to the ciphertext-rate of the incompressible SKE as ℓ_{msg} becomes arbitrarily larger than $\mathsf{poly}(\lambda)$.

Compression-Rate. The compression size admissible in the above scheme is equal to the compression size of IncSKE. This is because the security relies on the incompressibility of IncSKE (see ??). Therefore, the compression rate of the above scheme is

$$\frac{\ell_{\mathsf{ct}}^{\mathsf{IncSKE}} \cdot \mathtt{CompRate}^{\mathsf{IncSKE}}}{\mathsf{poly}(\lambda) + \ell_{\mathsf{ct}}^{\mathsf{IncSKE}}} = \frac{\mathtt{CompRate}^{\mathsf{IncSKE}}}{\frac{\mathsf{poly}(\lambda)}{\ell_{\mathsf{ct}}^{\mathsf{IncSKE}}} + 1}$$

where $\mathtt{CompRate}^{\mathsf{IncSKE}}$ is the compression rate of IncSKE. This becomes equal to the compression-rate of the incompressible SKE as $\ell_{\mathsf{ct}}^{\mathsf{IncSKE}}$ can be arbitrarily larger than $\mathsf{poly}(\lambda)$.

Leakage-Rate. From the correctness of the randomness extractor, the amount of leakage of the secret key permissible is at most $\ell_{\mathtt{key}} - (\ell_{\mathtt{key}}^{\mathsf{IncSKE}} + O(\lambda))$. This implies that the leakage rate is

$$\frac{\ell_{\mathtt{key}} - (\ell_{\mathtt{key}}^{\mathsf{IncSKE}} + O(\lambda))}{\ell_{\mathtt{key}}} = 1 - \frac{\ell_{\mathtt{key}}^{\mathsf{IncSKE}} + O(\lambda)}{\ell_{\mathtt{key}}}$$

By setting $\ell_{\mathtt{key}} = \omega(\ell_{\mathtt{key}}^{\mathsf{IncSKE}} + O(\lambda))$, the above scheme attains a leakage-rate of $1 - o(1)$.

Theorem 9. *Let* $\mathsf{IncSKE} = (\mathsf{IncSKE.Setup}, \mathsf{IncSKE.Enc}, \mathsf{IncSKE.Dec})$ *be an incompressible secret-key encryption scheme secure against unbounded adversaries with the* $\mathsf{IncSKE.Setup}$ *algorithm generating a secret key by sampling a truly random string. Let* Ext *be a strong average min-entropy extractor. Then, the above scheme is a leakage-resilient incompressible secret-key encryption scheme secure against unbounded adversaries with leakage-rate 1, ciphertext and compression rates equal to rates of* IncSKE*.*

The proof for the above theorem is provided in the full version.

6 Leakage-Resilient NC-KEM from Hash Proof Systems

In this section, we show that LR-PKE construction by Naor and Segev [24] is a leakage-resilient NC-KEM scheme (see Sect. 4.2).

Our Construction. Let $\mathsf{HPS} = (\mathsf{HPS.Setup}, \mathsf{HPS.Encaps}, \mathsf{HPS.Decaps})$ be a HPS that outputs a encapsulated key of length ℓ using a language $\mathcal{L} \subset X$. Let $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^\ell \to \{0,1\}^{\ell_{\mathtt{key}}}$ be a strong $(\ell_{\mathtt{key}} + O(\lambda), \mathrm{negl}(\lambda))$-extractor where $d = \mathsf{poly}(\lambda)$[7]. Then, the leakage-resilient NC-KEM scheme where the leakage size is $\ell - (\ell_{\mathtt{key}} + O(\lambda))$[8] is as follows.

- $\mathsf{Setup}(1^\lambda, 1^{\ell_{\mathtt{key}}})$: The setup algorithm generates $(\mathsf{hps.pk}, \mathsf{hps.sk}) \leftarrow \mathsf{HPS.Setup}(1^\lambda, 1^\ell)$[9] and sets $\mathsf{pk} = \mathsf{hps.pk}$ and $\mathsf{sk} = \mathsf{hps.sk}$.
- $\mathsf{Encaps}(\mathsf{pk})$: The encryption algorithm first samples a random seed $s \leftarrow \{0,1\}^d$ and a string-witness pair $(x, w) \leftarrow \mathcal{L}$. It computes secret key $k := \mathsf{Ext}_s(\mathsf{HPS.Encaps}(\mathsf{pk}, x, w))$ and the ciphertext $\mathsf{ct} := (x, s)$. It returns (k, ct)
- $\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$: Let sk be the secret key and $\mathsf{ct} = (x, s)$ be a ciphertext. The decryption algorithm computes $k := \mathsf{Ext}_s(\mathsf{HPS.Decaps}(\mathsf{sk}, x))$ and outputs k.
- $\mathsf{Sim}_1(1^\lambda, 1^{\ell_{\mathtt{key}}})$: The simulator generates $(\mathsf{hps.pk}, \mathsf{hps.sk}) \leftarrow \mathsf{HPS.Setup}(1^\lambda, 1^\ell)$, and sets $\mathsf{pk} = \mathsf{hps.pk}$. Further samples $s \leftarrow \{0,1\}^d, R \leftarrow \{0,1\}^\ell$ and $x \leftarrow X \backslash \mathcal{L}$. It sets $\mathsf{ct} = (x, s)$, *resamples* sk conditioned on pk, x and R, and outputs $\mathsf{state}_1 = (\mathsf{pk}, \mathsf{sk}, \mathsf{ct})$.
  We elaborate on resampling of sk: Define $\mathcal{D}$ as the distribution
  $$\mathcal{D} = \left(\mathsf{pk}, \mathsf{sk}, x, \mathsf{HPS.Decaps}(\mathsf{sk}, x) \,\middle|\, (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HPS.Setup}(1^\lambda, 1^\ell), x \leftarrow X \setminus \mathcal{L}\right).$$
  Let P be a potentially inefficient program (a stochastic channel) such that $\tilde{\mathcal{D}} \equiv \mathcal{D}$ when
  $$\tilde{\mathcal{D}} = \left(\mathsf{pk}, \mathsf{sk}, x, y \,\middle|\, (\mathsf{pk}, \mathsf{sk}', x, y) \leftarrow \mathcal{D}, \mathsf{sk} \leftarrow P(\mathsf{pk}, x, y)\right). \tag{1}$$
  By resampling of sk conditioned on pk, x and R, we mean $\mathsf{sk} \leftarrow P(\mathsf{pk}, x, R)$.
- $\mathsf{Sim}_2(\mathsf{state}_1, f)$: Let $\mathsf{state}_1 = (\mathsf{pk}, \mathsf{sk}, \mathsf{ct})$. The simulator outputs $f(\mathsf{sk})$ and sets $\mathsf{state}_2 = (\mathsf{state}_1, f(\mathsf{sk}), f)$.
- $\mathsf{Sim}_3(\mathsf{state}_2, k)$: Let $\mathsf{state}_2 = ((\mathsf{pk}, \mathsf{sk}, \mathsf{ct}), f(\mathsf{sk}), f)$. Sim_3 resamples and outputs sk'' conditioned on $\mathsf{pk}, f(\mathsf{sk}), \mathsf{ct}$ and k. We elaborate on resampling of sk'': For any f, define $\mathcal{D}'_f$ as
  $$\mathcal{D}'_f = \left(\mathsf{pk}, \mathsf{sk}, f(\mathsf{sk}), \mathsf{ct}, k \,\middle|\, \begin{array}{r} (\mathsf{pk}, \mathsf{sk}') \leftarrow \mathsf{HPS.Setup}(1^\lambda, 1^\ell) \\ s \leftarrow \{0,1\}^d, R \leftarrow \{0,1\}^\ell, x \leftarrow X \setminus \mathcal{L} \\ \mathsf{sk} \leftarrow P'(\mathsf{pk}, x, R), k = \mathsf{Ext}_s(R), \mathsf{ct} = (x, s) \end{array}\right).$$

[7] The parameters are chosen from Corollary 1.

[8] In order to achieve the smoothness property, we require that the size of the secret key of the HPS to be large.

[9] Here, we include 1^ℓ in the setup algorithm of HPS.

Let P'_f be a potentially inefficient program (a stochastic channel) such that $\tilde{\mathcal{D}}'_f \equiv \mathcal{D}'_f$ when

$$\tilde{\mathcal{D}}'_f = \left(\mathsf{pk}, \mathsf{sk}'', f(\mathsf{sk}), \mathsf{ct}, k \middle| \begin{array}{l} (\mathsf{pk}, \mathsf{sk}, f(\mathsf{sk}), \mathsf{ct}, k) \leftarrow \mathcal{D}'_f \\ \mathsf{sk}'' \leftarrow P'_f(\mathsf{pk}, f(\mathsf{sk}), \mathsf{ct}, k) \end{array} \right). \qquad (2)$$

By resampling of sk'' conditioned on $(\mathsf{pk}, f(\mathsf{sk}), \mathsf{ct}, k)$, we mean $\mathsf{sk}'' \leftarrow P'_f(\mathsf{pk}, \mathsf{ct}, f(\mathsf{sk}))$.

Correctness. A ciphertext in the above scheme is $\mathsf{ct} = (x, s)$ where s is a truly random string and x is a random string from language $\mathcal{L}$ with a witness w. The key generated in the encapsulation algorithm is $k = \mathsf{Ext}_s(\mathsf{HPS.Encaps}(\mathsf{pk}, x, w))$ whereas in the decapsulation algorithm it is $k = \mathsf{Ext}_s \mathsf{HPS.Decaps}(\mathsf{sk}, x)$ where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HPS.Setup}(1^\lambda, 1^{\ell_{\mathsf{key}}})$. These two are the same (except with negligible probability) due to the correctness of the HPS system, i.e., $\mathsf{HPS.Encaps}(\mathsf{pk}, x, w) = \mathsf{HPS.Decaps}(\mathsf{sk}, x)$.

Parameters. The size of a ciphertext in the above scheme is $|x| + |s|$ where x is a string from the language $\mathcal{L}$ and s is a seed for the extractor. From Corollary 1, we get $|s| = \mathsf{poly}(\lambda)$, therefore, the size of a ciphertext is $|x| + \mathsf{poly}(\lambda)$.

The amount of secret key leakage allowed is $\ell - (\ell_{\mathsf{key}} + O(\lambda))$. Therefore, the leakage rate is $\frac{\ell - (\ell_{\mathsf{key}} + O(\lambda))}{|\mathsf{hps.sk}|}$.

Theorem 10. *Assuming* $\mathsf{HPS} = (\mathsf{HPS.Setup}, \mathsf{HPS.Encaps}, \mathsf{HPS.Decaps})$ *is a secure HPS and* $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^\ell \to \{0,1\}^{\ell_{\mathsf{key}}}$ *be a strong extractor, the above scheme is a leakage-resilient non-committing key-encapsulation mechanism.*

The proof for the above theorem is provided in the full version. Using Corollary 1, we require $\ell \geq \ell_{\mathsf{key}} + O(\lambda)$. In our full version, we present an HPS such that we have $|x| = \ell^{3/4}$ and $|\mathsf{hps.sk}| = \ell$. Therefore, the leakage rate is $\frac{\ell - (\ell_{\mathsf{key}} + O(\lambda))}{\ell}$ which approaches to $1 - o(1)$ by setting $\ell = \omega(\ell_{\mathsf{key}} + O(\lambda))$. Using the above theorem, we get the following corollary.

Corollary 2. *Assuming the hardness of* DDH *problem, there exists a leakage-resilient NC-KEM such that for* $\lambda \in \mathbb{N}, \ell_{\mathsf{key}} \in \mathbb{N}$*, the size of the ciphertext is* $|\mathsf{ct}| = |x| + |s| = \ell_{\mathsf{key}}^{3/4} + \mathsf{poly}(\lambda)$. *Also, the leakage-rate is* $1 - o(1)$.

7 Leakage-Resilient Incompressible PKE

In this section, we present a construction of leakage-resilient incompressible PKE scheme from leakage-resilient NC-KEM scheme and regular incompressible SKE scheme. We emphasize that the incompressible SKE scheme must be secure against unbounded adversaries.[10]

[10] All existing incompressible SKE schemes that is secure against unbounded adversaries generate the secret keys by sampling uniformly at random from a specific domain.

Our Construction. Let $\mathsf{NCE} = (\mathsf{NCE.Setup}, \mathsf{NCE.Encaps}, \mathsf{NCE.Decaps}, \mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ be a $\ell_{\mathtt{leak}}$-LR-NC-KEM (see Sect. 4.2 for the definition, and Sect. 6 for construction of LR-NC-KEM) and $\mathsf{IncSKE} = (\mathsf{IncSKE.Setup}, \mathsf{IncSKE.Enc}, \mathsf{IncSKE.Dec})$ be an incompressible SKE scheme. Then, our LR incompressible PKE scheme has the following algorithms.

- $\mathsf{Setup}\left(1^\lambda, 1^{\ell_{\mathtt{msg}}}, 1^{\ell_{\mathtt{st}}}\right)$: The setup algorithm generates the $\ell_{\mathtt{leak}}$-LR-NCE parameters. In order to do this, it must first set the size of NC-KEM key (as a function of $\lambda, \ell_{\mathtt{msg}}$ and $\ell_{\mathtt{st}}$). Let $\ell_{\mathsf{key}}^{\mathsf{IncSKE}}$ denote the size of the secret keys of the incompressible SKE scheme when we run $\mathsf{IncSKE.Setup}\left(1^\lambda, 1^{\ell_{\mathtt{st}}}\right)$.
  The setup algorithm samples $(\mathsf{nce.pk}, \mathsf{nce.sk}) \leftarrow \mathsf{NCE.Setup}\left(1^\lambda, 1^{\ell_{\mathsf{key}}^{\mathsf{IncSKE}}}\right)$ and sets $\mathsf{pk} = \mathsf{nce.pk}$ and $\mathsf{sk} = \mathsf{nce.sk}$ as public key and secret key.
- $\mathsf{Enc}(\mathsf{pk}, m)$: For public key pk and message $m \in \{0,1\}^{\ell_{\mathtt{msg}}}$, the encryption algorithm samples $(\mathsf{inc.sk}, \mathsf{nce.ct}) \leftarrow \mathsf{NCE.Encaps}(\mathsf{pk})$ and $\mathsf{inc.ct} \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, m)$.[11]
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Let sk be a secret key and $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$ be a ciphertext. The decryption algorithm first computes $k \leftarrow \mathsf{NCE.Decaps}(\mathsf{sk}, \mathsf{ct}_0)$ and then decrypts ct_1 using k, i.e., $m = \mathsf{IncSKE.Dec}(k, \mathsf{ct}_1)$. It returns m.

Correctness. A ciphertext for a message m in the above scheme is $(\mathsf{nce.ct}, \mathsf{inc.ct})$ where $(\mathsf{inc.sk}, \mathsf{nce.ct}) \leftarrow \mathsf{NCE.Encaps}(\mathsf{pk})$ and $\mathsf{inc.ct} \leftarrow \mathsf{IncSKE.Enc}(\mathsf{inc.sk}, m)$. The decryption algorithm first computes $k \leftarrow \mathsf{NCE.Decaps}(\mathsf{sk}, \mathsf{ct})$. By the correctness of NCE, $k = \mathsf{inc.sk}$. Then, the algorithm computes $m' = \mathsf{IncSKE.Dec}(k, \mathsf{ct}_1) = \mathsf{IncSKE.Dec}(\mathsf{inc.sk}, \mathsf{inc.ct})$. By the correctness of the incompressible SKE scheme, $m = m'$.

Ciphertext-rate, Compression-rate and Leakage-rate. The size of a ciphertext is $|\mathsf{nce.ct}| + |\mathsf{inc.ct}|$. Therefore, the ciphertext rate is $(|\mathsf{nce.ct}| + |\mathsf{inc.ct}|)/\ell_{\mathtt{msg}}$. The compression size of the above scheme is equal to $\ell_{\mathtt{st}}/(|\mathsf{nce.ct}| + |\mathsf{inc.ct}|)$. The leakage rate of the above scheme is the same as the leakage rate of the LR-NC-KEM scheme.

Theorem 11. *Let* $\mathsf{NCE} = (\mathsf{NCE.Setup}, \mathsf{NCE.Encaps}, \mathsf{NCE.Decaps}, \mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ *be a secure* $\ell_{\mathtt{leak}}$*-LR-NC-KEM and* $\mathsf{IncSKE} = (\mathsf{IncSKE.Setup}, \mathsf{IncSKE.Enc}, \mathsf{IncSKE.Dec})$ *be an incompressible SKE scheme secure against unbounded adversaries. Then, the above is a* $\ell_{\mathtt{leak}}$*-leakage-resilient incompressible PKE scheme.*

Proof (Proof sketch). The proof proceeds via a sequence of hybrid experiments.

- H_0. This corresponds to the leakage-resilient incompressible PKE security game.
- H_1. In this hybrid, the challenger uses the simulation mode of NCE. It generates the public key pk and $\mathsf{nce.ct}$ (first part of the challenge ciphertext) at

[11] Here, the secret key of IncSKE is assumed to be uniform. This is the case for all known constructions with unbounded security. For the more general case, we can use the encapsulated key as the randomness for $\mathsf{IncSKE.Setup}$.

the beginning the game using Sim_1, responds to the query f using Sim_2 and generates $\mathsf{inc.sk}$ uniformly at random. Finally, generates the secret key sk in the second phase using Sim_3. Note that H_0 and H_1 are indistinguishable due to NCE security.

– Exploiting the fact that $\mathsf{inc.sk}$ is sampled uniformly, we observe that no unbounded adversary $\mathcal{A}$ can win in game H_1 with significant advantage due the security of the incompressible SKE scheme. This is because using the simulators, we can construct a (inefficient) adversary $\mathcal{B}$ that wins the incompressible security game for IncSKE by simulating the game for $\mathcal{A}$. $\square$

The detailed proof of security is provided in the full version.

We present a construction for LR-NC-KEM using a hash proof system in Sect. 6, which achieves ciphertext size equal to $(\ell_{\mathtt{key}}^{\mathsf{IncSKE}})^{3/4}$ (see Corollary 2). Combining this with our ct-rate-$\frac{1}{2}$ incompressible SKE having $\ell_{\mathtt{key}}^{\mathsf{IncSKE}} = 4\ell_{\mathtt{msg}}$ (see ??), we obtain Theorem 4 using the above theorem.

We also present a transformation from any LR Incompressible SKE to an LR Incompressible PKE in the full version following similar techniques to [17,18].

8 Impossibility Results for Rate-1 Leakage-Resilient Incompressible Encryption Schemes

In this section, we explore barriers towards achieving rate-1 leakage-resilient incompressible encryption schemes. We show that it is impossible to base their security on a standard assumption in a black-box manner. These barriers are shown using the framework of simulatable attacks introduced by Wichs [28]. The following definitions of cryptographic games, cryptographic properties and simulatable attacks are from [28] with some minor simplifications.

Definition 12 (Cryptographic Game). *A cryptographic game $\mathcal{G}$ is defined by a (possibly inefficient) random system $\mathscr{C}$ called the challenger. When given as input the security parameter λ (in unary), the challenger $\mathscr{C}$ interacts with some attacker $\mathcal{A}$ (possibly for multiple rounds) and outputs a bit b, which is taken to be the output of the game $\mathcal{G}$. We use the shorthand $b = \langle \mathcal{A}(1^\lambda), \mathscr{C}(1^\lambda)\rangle$ to denote this process. The advantage of $\mathcal{A}$ in the game $\mathcal{G}$ is given by $\mathsf{Adv}_{\mathcal{G}}^{\mathcal{A}}(\lambda) = \Pr[\langle \mathcal{A}(1^\lambda), \mathscr{C}(1^\lambda)\rangle = 1] - \frac{1}{2}$. A cryptographic game $\mathcal{G}$ is secure if the advantage $\mathsf{Adv}_{\mathcal{G}}^{\mathcal{A}}$ is negligible for every PPT adversary $\mathcal{A}$.*

We say that an attacker $\mathcal{A}$ successfully breaks the security of $\mathcal{G}$ if the advantage $\mathsf{Adv}_{\mathcal{G}}^{\mathcal{A}}(\lambda)$ is non-negligible in λ. Below, we give a more general definition that captures other kind of cryptographic scenarios.

Definition 13 (Cryptographic Property). *Let $\mathscr{A}$ denote the set of all adversaries (efficient or inefficient). A cryptographic property $\mathcal{C}$ is defined as a function $\mathcal{C} : \mathscr{A} \times \mathbb{N} \to [0, 1]$ such that for an attacker $\mathcal{A} \in \mathscr{A}$ and security parameter $\lambda \in \mathbb{N}, \mathcal{C}(\mathcal{A}, \lambda) = \mathsf{Adv}_{\mathcal{C}}^{\mathcal{A}}(\lambda)$ denotes the advantage of $\mathcal{A}$ in breaking the property. We say that $\mathcal{C}$ is secure if for all PPT attackers $\mathcal{A}$, the advantage $\mathcal{C}(\mathcal{A}, \lambda) = \mathsf{Adv}_{\mathcal{C}}^{\mathcal{A}}(\lambda)$ is negligible in λ.*

Most proofs in cryptography have the form of a black-box reduction and this is what we define below. We have changed the corresponding definition from [28] by not limiting successful oracles $\mathcal{A}$ to have advantage greater than 1/2 in breaking the property $\mathcal{C}$, while also not guaranteeing that the advantage of the reduction $\mathcal{R}^{\mathcal{A}}$ in breaking the game $\mathcal{G}$ would be some noticeable function, but intead only something non-negligible.

Definition 14 (Black Box Reduction). *Let $\mathcal{C}$ be some cryptographic property and $\mathcal{G}$ be a cryptographic game. A black-box reduction deriving the security of $\mathcal{C}$ from that of $\mathcal{G}$ is an oracle PPT machine $\mathcal{R}^{(\cdot)}$, for all (possibly inefficient, non-uniform) attackers $\mathcal{A}_\lambda$ with some non-negligible advantage $\mathsf{Adv}_{\mathcal{C}}^{\mathcal{A}_\lambda}(\lambda)$, we have that $\mathsf{Adv}_{\mathcal{G}}^{\mathcal{R}^{\mathcal{A}_\lambda}}(\lambda)$ is also some non-negligible function in λ.*

We finally define a simulatable attack, a notion developed by [28] for proving black-box separation results. The technique involves constructing an inefficient but successful attacker $\mathcal{A}$ for a cryptographic property $\mathcal{C}$, while also providing an efficient simulator Sim such that no oracle machine can distinguish access to $\mathcal{A}$ from access to Sim. The efficient simulator Sim in itself would usually not correspond to a valid attack on $\mathcal{C}$ since it would fail to satisfy some desired properties needed from a valid attacker. This could be used to show the impossibility of a black-box reduction proof for $\mathcal{C}$.

Definition 15 (Simulatable Attack). *A simulatable attack on a cryptographic property $\mathcal{C}$ consists of*

1. *an ensemble of (possibly inefficient) stateless non-uniform attackers $\{\mathcal{A}_{\lambda,h}\}_{\lambda\in\mathbb{N},h\in\mathcal{H}_\lambda}$ where $\mathcal{H}_\lambda$ are some finite sets of functions, and*
2. *a stateful PPT simulator Sim.*

We require the following two properties to hold:

- ***Inefficient Attack:*** *For each $\lambda \in \mathbb{N}, h \in \mathcal{H}_\lambda$, the inefficient attacker $\mathcal{A}_{\lambda,h}$ breaks the security of $\mathcal{C}$ with advantage $\mathsf{Adv}_{\mathcal{C}}^{\mathcal{A}_{\lambda,h}(1^\lambda)} = 1 - \text{negl}(\lambda)$.*
- ***Simulatability:*** *There exists a negligible function $\text{negl}(\cdot)$ such that for $\lambda \in \mathbb{N}$ and every (possibly inefficient) oracle machine $\mathcal{R}^{(\cdot)}$ making at most $q = q(\lambda)$ queries to its oracle, it holds that*

$$\left| \Pr_{h \xleftarrow{\$} \mathcal{H}_\lambda, \mathcal{R}} [\mathcal{R}^{\mathcal{A}_{\lambda,h}}(1^\lambda) = 1] - \Pr_{\mathcal{R},\mathsf{Sim}} [\mathcal{R}^{\mathsf{Sim}}(1^\lambda) = 1] \right| = \mathsf{poly}(q(\lambda)) \cdot \text{negl}(\lambda),$$

Wichs [28] proved the following theorem that shows how the existence of a simulatable attack against a property $\mathcal{C}$ implies impossibility of reducing the security of $\mathcal{C}$ to any secure cryptographic game $\mathcal{G}$ in a black-box manner.

Theorem 12 ([28]). *If there exists a simulatable attack against a cryptographic notion $\mathcal{C}$ and there is a black-box reduction showing the security of $\mathcal{C}$ from the security of some cryptograhic game $\mathcal{G}$, then $\mathcal{G}$ is not secure.*

8.1 Impossibility Result for Incompressible PKE with Short Keys

As a stepping stone, we first show an analogous results for rate-1 incompressible PKE without any leakage. We show that for any rate-1 incompressible PKE scheme with short (that is, sublinear in the message size) secret keys, the security cannot be based on a standard assumption in a black-box manner. This resolves an open question from [19].

Theorem 13. *Let $\ell_{\mathtt{st}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any function (denoting the adversary's long-term storage as a function of the security parameter and message length). Let $\mathsf{IncPKE} = (\mathsf{IncPKE.Setup}, \mathsf{IncPKE.Enc}, \mathsf{IncPKE.Dec})$ be an $\ell_{\mathtt{st}}$-incompressible PKE scheme with ciphertext-rate $1 - o(1)$, compression-rate $1 - o(1)$, secret-key-rate $1 - \epsilon$ for some constant ϵ, and possesses almost perfect correctness and deterministic decryption. Then, there is no secure cryptographic game $\mathcal{G}$ such that there is a black-box reduction that derives the security of IncPKE from the security of $\mathcal{G}$.*

Proof. To establish the above theorem, it is enough to demonstrate a *simulatable attack* (see Definition 15) against the incompressible security of IncPKE when secret-key-rate is $(1 - \epsilon)$ where $\epsilon > 0$ is a constant, and $\ell_{\mathtt{st}}(\lambda, \ell_{\mathtt{msg}}) \leq \ell_{\mathtt{msg}}$ (since we want compression-rate should be 1). Recall that a simulatable attack involves an inefficient adversary against the primitive, and an efficient stateful simulation of it. We present these two below.

Inefficient Adversary: Fix any security parameter λ. Let $\ell_{\mathtt{msg}} = \ell_{\mathtt{msg}}(\lambda)$ be a sufficiently large polynomial (the exact polynomial is specified at the end of our proof), and let $\ell_{\mathtt{st}} = S(\lambda, \ell_{\mathtt{msg}})$. Let $\ell_{\mathtt{pk}}$ (resp. $\ell_{\mathtt{key}}$) denote the size of public key (resp. secret-key size) output by $\mathsf{IncPKE.Setup}\left(1^{\lambda}, 1^{\ell_{\mathtt{msg}}}\right)$. Let G be the set of all functions from $\{0,1\}^{\ell_{\mathtt{pk}}} \to \{0,1\}^{\ell_{\mathtt{msg}}} \times \{0,1\}^{\ell_{\mathtt{msg}}}$ and H is the set of all functions from $\{0,1\}^{\ell_{\mathtt{ct}}} \to \{0,1\}^{\ell_{\mathtt{st}}}$. We define our inefficient adversaries (see Definition 15) as follows. The adversaries are parameterized by functions $g \in G$ and $h \in H$ (the adversaries are also parameterized by λ, however we are skipping the dependence on λ for notational brevity).

$$\{\mathcal{A}^{g,h} = (\mathcal{A}_0^{g,h}, \mathcal{A}_1^{g,h}, \mathcal{A}_2^{g,h})\}_{g \in G, h \in H}$$

where

1. $\mathcal{A}_0^{g,h}$: On input pk, it computes $(m_0, m_1) = g(\mathsf{pk})$ outputs (m_0, m_1, aux) where $\mathsf{aux} = (m_0, m_1)$.
2. $\mathcal{A}_1^{g,h}$: On input $(\mathsf{pk}, \mathsf{ct}^*, \mathsf{aux})$, it returns $h(\mathsf{ct}^*)$.
3. $\mathcal{A}_2^{g,h}$: On input $(\mathsf{pk}, \mathsf{sk}, \mathsf{aux}, \mathsf{st})$, it computes $(m_0, m_1) = g(\mathsf{pk})$. It checks if $\mathsf{aux} = (m_0, m_1)$, returning $\perp$ if the check fails, and continuing otherwise. It then, by brute force, constructs the set $C_{\mathcal{A}}^{h,\mathsf{st}} = \{\mathsf{ct} \mid h(\mathsf{ct}) = \mathsf{st}\}$, and further generates $M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{pk},\mathsf{sk}} = \left\{m \mid m = \mathsf{IncPKE.Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}), \mathsf{ct} \in C_{\mathcal{A}}^{h,\mathsf{st}}\right\}$. If $\{m_0, m_1\} \cap M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{pk},\mathsf{sk}} = \emptyset$ or $\{m_0, m_1\} \subseteq M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{pk},\mathsf{sk}}$, then return $\perp$. Else, if $m_0 \in M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{pk},\mathsf{sk}}$, then return 0. Else, return 1.

We will demonstrate that the adversary mentioned above wins the incompressible PKE game with a probability negligibly far from 1. This proof crucially relies on the fact that the scheme possesses a ciphertext-rate of 1.

Lemma 1. *There exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$,

$$\mathsf{Pr}_{g,h}[\mathcal{A}^{g,h} \textit{ wins Incompressible PKE game}] = 1 - \text{negl}(\lambda).$$

Proof. First, we observe that, with very high probability, for a faithfully generated ciphertext ct^* for message m_b, we have $m_b \in M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{pk},\mathsf{sk}}$ with all but negligible probability. This follows from almost perfect correctness.

Hence, the only potential case with non-negligible probability where $\mathcal{A}^{g,h}$ fails to output b in the Incompressible PKE game is if $m_{1-b} \in M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{pk},\mathsf{sk}}$.

To bound the probability of $m_{1-b} \in M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{pk},\mathsf{sk}}$, note that m_0, m_1, b are chosen uniformly at random and are independent of $h, \mathsf{st}, \mathsf{pk}, \mathsf{sk}$. Therefore, $M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{pk},\mathsf{sk}}$ is independent of m_{1-b}, and the probability of $m_{1-b} \in M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{pk},\mathsf{sk}}$ is equal to $|M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{pk},\mathsf{sk}}|/2^{\ell_{\mathsf{msg}}}$. Further, $|M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{pk},\mathsf{sk}}| \leq |C_{\mathcal{A}}^{h,\mathsf{st}}|$ and $|C_{\mathcal{A}}^{h,\mathsf{st}}| = 2^{o(\ell_{\mathsf{msg}})}$, except with negligible probability (the proof of this is a standard probability argument, which we defer to the full version). Hence the above probability is upper bounded by $2^{o(\ell_{\mathsf{msg}})}/2^{\ell_{\mathsf{msg}}}$, which is negligible in λ.

Simulator: We will now present the efficient simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2)$ which uses two tables Q_0 and Q_1 to simulate the random functions g and h, and behaves as follows.

1. Sim_0 : On input pk, it checks whether an $(\mathsf{pk}, \mathsf{aux} = (m_0, m_1))$ is present in Q_0. If it does, then it returns (m_0, m_1, aux). Else, it randomly generates m_0, m_1, stores (pk, m_0, m_1) in Q_0, and outputs (m_0, m_1, aux) where $\mathsf{aux} = (m_0, m_1)$.
2. Sim_1 : On input $(\mathsf{pk}, \mathsf{ct}^*, \mathsf{aux})$, it checks whether an entry of the form $(\mathsf{ct}^*, \mathsf{st})$ is present in Q_1. If it is, then it returns st. Else, it randomly generates st, stores $(\mathsf{ct}^*, \mathsf{st})$ in Q_1, and returns st.
3. Sim_2 : On input $(\mathsf{pk}, \mathsf{sk}, \mathsf{aux}, \mathsf{st})$, it first checks whether there exists an entry $(\mathsf{pk}, m_0,\ m_1)$ in Q_0 and $\mathsf{aux} = (m_0, m_1)$. If not, it returns $\perp$. Else, it computes $C_{\mathsf{Sim}} = \left\{\mathsf{ct} \mid (\mathsf{ct}, \mathsf{st}) \in Q_1\right\}$. It then generates $M_{\mathsf{Sim}} = \{m \mid m = \mathsf{IncPKE.Dec}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}), \mathsf{ct} \in C_{\mathsf{Sim}}\}$. If $\{m_0, m_1\} \cap M_{\mathsf{Sim}} = \emptyset$ or $\{m_0, m_1\} \subseteq M_{\mathsf{Sim}}$, then return $\perp$. Else, if $m_0 \in M_{\mathsf{Sim}}$, then return 0. Else, return 1.

To establish the indistinguishability between $\mathcal{A}$ and Sim by any efficient reduction, we introduce a series of intermediate adversaries. Each subsequent adversary in the sequence either maintains a log of its computations or incorporates additional conditions for aborting executions compared to its predecessor. We show that these intermediate adversaries are statistically indistinguishable by any reduction making a polynomially bounded number of queries, and prove that the final adversary is identical to Sim. In the full version, we prove the following lemma.

Lemma 2. *For any oracle-machine R making polynomially many queries to its oracle, there exists a negligible function* negl*, such that for all $\lambda \in \mathbb{N}$,*

$$\left|\mathsf{Pr}_{g,h}\left[R^{\mathcal{A}^{g,h}}(1^{\lambda}) \to 1\right] - \mathsf{Pr}_{\mathsf{Sim}}\left[R^{\mathsf{Sim}}(1^{\lambda}) \to 1\right]\right| = \mathrm{negl}(\lambda).$$

From Lemma 1, we get that $\mathcal{A}$ wins the Incompressible PKE security game with probability negligibly close to 1 (i.e. $1 - \mathrm{negl}(\lambda)$) and from Lemma 2, we have shown that the probability of any efficient reduction distinguishing $\mathcal{A}$ from Sim is $\mathrm{negl}(\lambda)$. Therefore, we have established a simulatable attack against the incompressible security of IncPKE.

8.2 Impossibility Result for Leakage-Resilient Incompressible SKE

In this section, we demonstrate the impossibility of basing the security of a leakage-resilient *symmetric* key encryption scheme with ciphertext-rate 1, that can tolerate *both* leakage-rate and compression-rate of 1, on standard assumptions in a blackbox manner.

Theorem 14. *Let $\ell_{\mathtt{st}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and $\ell_{\mathtt{leak}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any functions (denoting the long-term storage bound and the leakage bound respectively). Let $\mathsf{IncSKE} = (\mathsf{IncSKE.Setup}, \mathsf{IncSKE.Enc}, \mathsf{IncSKE.Dec})$ be an incompressible SKE scheme with almost perfect correctness, deterministic decryption and ciphertext-rate, storage-rate and leakage-rate all being $1 - o(1)$. Then there is no secure cryptographic game $\mathcal{G}$ such that there is a black-box reduction that derives the security of IncSKE from the security of $\mathcal{G}$.*

Proof. To establish the above theorem, we will demonstrate the existence of a simulatable attack (see Definition 15) against the leakage-resilient incompressible security of IncSKE.

Inefficient Adversary: Let $\lambda \in \mathbb{N}$ be any security parameter. Below, we skip the dependence on λ when it is clear from the context. Let $\ell_{\mathtt{msg}} = \ell_{\mathtt{msg}}(\lambda)$ be a sufficiently large polynomial (to be fixed later in our proof), $\ell_{\mathtt{st}} = \ell_{\mathtt{st}}(\lambda, \ell_{\mathtt{msg}})$, $\ell_{\mathtt{leak}} = \ell_{\mathtt{leak}}(\lambda, \ell_{\mathtt{msg}})$ and $\ell_{\mathtt{key}} = \ell_{\mathtt{key}}(\lambda, \ell_{\mathtt{msg}})$. Let F be the set of all functions from $\{0,1\}^{\ell_{\mathsf{key}}}$ to $\{0,1\}^{\ell_{\mathsf{leak}}}$, G is the set of all functions from $\{0,1\}^{\ell_{\mathsf{leak}}} \to \{0,1\}^{\ell_{\mathsf{msg}}} \times \{0,1\}^{\ell_{\mathsf{msg}}}$ and H is the set of all functions from $\{0,1\}^{\ell_{\mathsf{ct}}} \to \{0,1\}^{\ell_{\mathsf{st}}}$. Our adversaries are parameterized by functions in F, G, H (and the security parameter λ, which we skip for notational brevity). We define our inefficient adversaries (see Definition 15) as follows.

$$\{\mathcal{A}^{f,g,h} = (\mathcal{A}_0^{f,g,h}, \mathcal{A}_1^{f,g,h}, \mathcal{A}_2^{f,g,h}, \mathcal{A}_3^{f,g,h}\}_{f,g,h \in F \times G \times H}$$

1. $\mathcal{A}_0^{f,g,h}$: On input sk, it outputs $f(\mathsf{sk})$.
2. $\mathcal{A}_1^{f,g,h}$: On input $(1^{\lambda}, z)$, it compute $(m_0, m_1) = g(z)$ and outputs $(m_0, m_1, \mathsf{aux} = (z, m_0, m_1))$.
3. $\mathcal{A}_2^{f,g,h}$: On input $(\mathsf{ct}^*, \mathsf{aux})$, it returns $h(\mathsf{ct}^*)$.

4. $\mathcal{A}_3^{f,g,h}$: On input $(\mathsf{sk}, \mathsf{aux} = (z, m_0, m_1), \mathsf{st})$, it checks whether $f(\mathsf{sk}) = z$ and $(m_0, m_1) = g(z)$. If the test fails, it returns $\perp$. Else, it brute forces over all ct to construct $C_{\mathcal{A}}^{h,\mathsf{st}} = \{\mathsf{ct} \mid h(\mathsf{ct}) = \mathsf{st}\}$. It then generates $M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{sk}} = \left\{m \mid m = \mathsf{IncSKE.Dec}(\mathsf{sk}, \mathsf{ct}),\ \mathsf{ct} \in C_{\mathcal{A}}^{h,\mathsf{st}}\right\}$. If $(m_0, m_1) \cap M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{sk}} = \emptyset$ or $(m_0, m_1) \subseteq M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{sk}}$, then return $\perp$. Else, if $m_0 \in M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{sk}}$, then return 0. Else, return 1.

We will demonstrate that the adversary mentioned above wins the leakage-resilient incompressible SKE game with a probability close 1. This proof crucially relies on the fact that the scheme possesses a ciphertext-rate of 1.

Lemma 3. *There exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$,

$$\mathsf{Pr}_{f,g,h}\left[\mathcal{A}^{f,g,h} \textit{ wins LR-Incompressible SKE game}\right] = 1 - \mathrm{negl}(\lambda).$$

Proof. Observe that the only case when $\mathcal{A}^{f,g,h}$ fails to output b in the Incompressible SKE game is if both m_0 and m_1 are in $M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{sk}}$ using the following argument. By assuming almost perfect correctness, we have $\forall \lambda, m$,

$$\mathsf{Pr}_{r,\mathsf{sk}\leftarrow\mathsf{Setup}(1^\lambda,1^{\ell_{\mathsf{st}}})}[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m \mid \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m; r)] = 1 - \mathrm{negl}(\lambda)$$

This implies that with very high probability for a faithfully generated ciphertext ct^* for message m_b, we have $m_b \in M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{sk}}$. To bound the probability of $m_{1-b} \in M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{sk}}$, where m_{1-b} is randomly generated, note that m_0 and m_1 are chosen uniformly at random and are independent of $h, \mathsf{st}, \mathsf{sk}$. Therefore, $M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{sk}}$ is independent of m_{1-b}. Given that $|M_{\mathcal{A}}^{h,\mathsf{st},\mathsf{sk}}| \le |C_{\mathcal{A}}^{h,\mathsf{st}}| = 2^{o(\ell_{\mathsf{msg}})}$, we can upper bound the probability as $2^{o(\ell_{\mathsf{msg}})}/2^{\ell_{\mathsf{msg}}}$ using the union bound. In other words, the probability is at most $1/2^{\ell_{\mathsf{msg}}-o(\ell_{\mathsf{msg}})}$, which becomes negligible in λ for $\ell_{\mathsf{msg}} \ge \lambda$.

We will now present the efficient simulator.

Simulator:
Let us consider the following simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ which maintains T_0, T_1, T_2 tables as follows.

1. Sim_0 : On input sk, it checks whether an (sk, z) is present in Q_0. If it does, then it returns z. Else, it randomly generates z, stores (sk, z) in Q_0 and returns z.
2. Sim_1 : On input $(1^\lambda, z)$, it checks whether an $(z, (m_0, m_1))$ is present in Q_1. If it does, then it returns $(m_0, m_1, \mathsf{aux} = (z, m_0, m_1))$. Else, it randomly generates (m_0, m_1), stores $(z, (m_0, m_1))$ in Q_1 and returns $(m_0, m_1, \mathsf{aux} = (z, m_0, m_1))$.
3. Sim_2 : On input $(\mathsf{ct}^*, \mathsf{aux})$, it checks whether an $(\mathsf{ct}^*, \mathsf{st})$ is present in Q_2. If it does, then it returns st. Else, it randomly generates st, stores $(\mathsf{ct}^*, \mathsf{st})$ in Q_2 and returns st.

4. Sim_3 : On input $(\mathsf{sk}, \mathsf{aux}, \mathsf{st})$, it checks whether (sk, z) is present in Q_0 and $(z, (m_0, m_1))$ is present in Q_1. If the test fails, it returns $\perp$. Else, it first checks whether there exists an entry $(\mathsf{ct}, \mathsf{st})$ in Q_2. If not, it returns $\perp$.
Else, $C_{\mathsf{Sim}} = \{\mathsf{ct} \mid (\mathsf{ct}, \mathsf{st}) \in Q_2\}$. It then generates $M_{\mathsf{Sim}} = \Big\{m \mid m = \mathsf{IncSKE.Dec}(\mathsf{sk}, \mathsf{ct}),\ \mathsf{ct} \in C_{\mathsf{Sim}}\Big\}$. If $(m_0, m_1) \cap M_{\mathsf{Sim}} = \emptyset$ or $(m_0, m_1) \subseteq M_{\mathsf{Sim}}$, then return $\perp$. Else, if $m_0 \in M_{\mathsf{Sim}}$, then return 0. Else, return 1.

To establish the indistinguishability between $\mathcal{A}$ and Sim by any efficient reduction, we introduce a series of intermediate adversaries. Each subsequent adversary in the sequence either maintains a log of its computations or incorporates additional conditions for aborting executions compared to its predecessor. We show that these intermediate adversaries are indistinguishable by any efficient reduction and prove that the final adversary is identical to Sim.

Lemma 4. *For any oracle-machine R making polynomially many queries to its oracle, there exists a negligible function* negl*, such that for all $\lambda \in \mathbb{N}$,*

$$\left|\mathsf{Pr}_{f,g,h}\left[R^{\mathcal{A}^{f,g,h}}(1^{\ell_{\mathsf{msg}}}) \rightarrow 1\right] - \mathsf{Pr}_{rand(\mathsf{Sim})}\left[R^{\mathsf{Sim}}(1^{\ell_{\mathsf{msg}}}) \rightarrow 1\right]\right| = \mathrm{negl}(\lambda).$$

The proof of the above lemma is provided in the full version of our paper. From Lemma 3, we have shown that $\mathcal{A}$ wins the LR Incompressible SKE security game with probability negligibly close to 1 (i.e., $1 - \mathrm{negl}(\lambda)$) and from Lemma 4, we have shown that the probability of any efficient reduction distinguishing $\mathcal{A}$ from Sim is $\mathrm{negl}(\lambda)$. Therefore, we have established a simulatable attack against IncSKE.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous Hardcore Bits and Cryptography against Memory Attacks. In: Reingold, O. (ed.) Theory of Cryptography. pp. 474–495. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_28
2. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability: (or: Quadratic residuosity strikes back). In: Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30. pp. 1–20. Springer (2010)
3. Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous ibe, leakage resilience and circular security from new assumptions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 535–564. Springer (2018)
4. Branco, P., Döttling, N., Dujmović, J.: Rate-1 Incompressible Encryption from Standard Assumptions. In: Kiltz, E., Vaikuntanathan, V. (eds.) Theory of Cryptography. pp. 33–69. Lecture Notes in Computer Science, Springer Nature Switzerland, Cham (2022). https://doi.org/10.1007/978-3-031-22365-5_2

5. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19. pp. 453–469. Springer (2000)
6. Cheraghchi, M., Didier, F., Shokrollahi, A.: Invertible extractors and wiretap protocols. IEEE Transactions on Information Theory **58**(2), 1254–1274 (2011)
7. Dachman-Soled, D., Gordon, S.D., Liu, F.H., O'Neill, A., Zhou, H.S.: Leakage resilience from program obfuscation. Journal of Cryptology **32**, 742–824 (2019)
8. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16. pp. 613–631. Springer (2010)
9. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 621–630 (2009)
10. Dodis, Y., Sahai, A., Smith, A.: On Perfect and Adaptive Security in Exposure-Resilient Cryptography. In: Pfitzmann, B. (ed.) Advances in Cryptology — EUROCRYPT 2001. pp. 301–324. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_19
11. Dziembowski, S.: Intrusion-Resilience Via the Bounded-Storage Model. In: Halevi, S., Rabin, T. (eds.) Theory of Cryptography. pp. 207–224. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11681878_11
12. Dziembowski, S.: On Forward-Secure Storage. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006. pp. 251–270. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11818175_15
13. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
14. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM Journal on Computing **45**(3), 882–929 (2016)
15. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Proceedings of the forty-third annual ACM symposium on Theory of computing. pp. 99–108 (2011)
16. Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Robustness of the learning with errors assumption (2010)
17. Goyal, R., Koppula, V., Rajasree, M.S., Verma, A.: Incompressible functional encryption. Cryptology ePrint Archive (2024)
18. Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II (2016)
19. Guan, J., Wichs, D., Zhandry, M.: Incompressible Cryptography. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022. pp. 700–730. Lecture Notes in Computer Science, Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-06944-4_24

20. Guan, J., Wichs, D., Zhandry, M.: Somewhere Randomness Extraction and Security against Bounded-Storage Mass Surveillance (2023), https://eprint.iacr.org/2023/409, report Number: 409
21. Hajiabadi, M., Kapron, B.M., Srinivasan, V.: On generic constructions of circularly-secure, leakage-resilient public-key encryption schemes. In: Public-Key Cryptography–PKC 2016, pp. 129–158. Springer (2016)
22. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19. pp. 388–397. Springer (1999)
23. Moran, T., Wichs, D.: Incompressible Encodings. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. pp. 494–523. Lecture Notes in Computer Science, Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_17
24. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. pp. 18–35. Springer (2009)
25. Rivest, R.L.: All-or-nothing encryption and the package transform. In: Fast Software Encryption: 4th International Workshop, FSE'97 Haifa, Israel, January 20–22 1997 Proceedings 4. pp. 210–218. Springer (1997)
26. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC. pp. 475–484 (2014)
27. Vadhan, S.P.: Pseudorandomness. Foundations and Trends® in Theoretical Computer Science **7**(1-3), 1–336 (2012). https://doi.org/10.1561/0400000010, http://dx.doi.org/10.1561/0400000010
28. Wichs, D.: Barriers in cryptography with weak, correlated and leaky sources. In: Kleinberg, R.D. (ed.) Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013. pp. 111–126. ACM (2013). https://doi.org/10.1145/2422436.2422451, https://doi.org/10.1145/2422436.2422451

Cryptanalysis on Symmetric-Key Schemes

The First Practical Collision for 31-Step SHA-256

Yingxin Li[1,2], Fukang Liu[2], Gaoli Wang[1](✉), Xiaoyang Dong[3,5], and Siwei Sun[4,6]

[1] Shanghai Key Laboratory of Trustworthy Computing, School of Cryptology, Software Engineering Institute, East China Normal University, Shanghai, China
glwang@sei.ecnu.edu.cn

[2] Institute of Science Tokyo, Tokyo, Japan
liu.f.ad@m.titech.ac.jp

[3] Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing, China
xiaoyangdong@tsinghua.edu.cn

[4] School of Cryptology, University of Chinese Academy of Sciences, Beijing, China
sunsiwei@ucas.ac.cn

[5] Zhongguancun Laboratory, Beijing, China

[6] State Key Laboratory of Cryptology, Beijing 100878, China

Abstract. SHA-256 is a hash function standardized by NIST and has been widely deployed in real-world applications, e.g., Bitcoin. Recently, an improved collision attack on 31-step SHA-256 was proposed by Li-Liu-Wang at EUROCRYPT 2024, whose time and memory complexity are $2^{49.8}$ and 2^{48}, respectively. Such a result indicates that we are close to a practical collision attack on 31-step SHA-256, and that the current bottleneck is the memory complexity. To overcome such an obstacle, we develop a novel memory-efficient attack in this paper, which allows us to find the first practical colliding message pair for 31-step SHA-256 in only 1.2 h with 64 threads and negligible memory. This technique is general and Li-Liu-Wang's collision attack on 31-step SHA-512 can also be significantly improved, i.e., the time and memory complexity can be improved by a factor of $2^{20.9}$ and $2^{42.1}$, respectively. Although we have set a new record in the practical collision attack on SHA-256, which improves the previous best practical attack published at EUROCRYPT 2013 by 3 steps, the attack is still far from threatening the security of SHA-256 since it has 64 steps in total. On the other hand, our new attack shows that nearly half of full SHA-256 can be practically cracked now, and it should be viewed as a major progress in the cryptanalysis of SHA-256 since 2013.

Keywords: practical collisions · SHA-256 · SHA-512 · meet-in-the-middle technique

1 Introduction

Since the breakthrough made by Wang et al. in collision attacks [17–20] on MD-SHA hash family including MD4, MD5, SHA-0, SHA-1, notable progress has also been made in the preimage and collision attacks on SHA-2 family. Regarding the preimage attacks on SHA-2, they are mainly based on the meet-in-the-middle (MitM) technique. Although this technique can lead to a preimage attack reaching a much larger number of attacked steps, its time complexity is usually close to brute force, and hence it is most of academic interests. For example, the best preimage attack on SHA-256 can reach up to 45 steps [10], while its time and memory complexity are $2^{255.5}$ and 2^6, respectively.

For the collision attacks on SHA-2, there is however another challenge, which is to search for a valid differential characteristic that can be used to construct colliding message pairs. The first effort towards solving this challenge was made by Mendel et al. at ASIACRYPT 2011 [14], who improved the original guess-and-determine (GnD) technique for SHA-1 [2], and proposed a novel method to search for SHA-2 characteristics and conforming message pairs simultaneously. With this tool for SHA-2, the practical collision attack could reach 27 steps of SHA-256, which significantly improved the previous practical collision attack reaching 24 steps published at SAC 2008 [9]. Since the invention of this tool for SHA-2, it has been gradually improved in a series of follow-up works, which has been used to significantly improve the (free-start and semi-free-start) collision attacks on SHA-2 [4,7,15].

Despite many years of efforts to understand the collision resistance of SHA-2, the best collision attack on SHA-256 could only reach 31 steps and it was first achieved at EUROCRYPT 2013 [15]. However, this attack is far from being practical, as it has time and memory complexity of $2^{65.5}$ and 2^{34}, respectively. A decade later at EUROCRYPT 2024 [11], this collision attack on 31-step SHA-256 was improved for the first time, mainly benefiting from novel MILP/SAT/SMT-based techniques to search for signed differential characteristics [11,13]. Although the improved collision attack has time and memory complexity of $2^{49.8}$ and 2^{48}, respectively, it still cannot be turned into a practical attack due to the high memory complexity.

Indeed, both the collision attacks [11,15] on 31-step SHA-256 are based on the idea to convert a semi-free-start (SFS) collision into a collision with a MitM technique developed at EUROCRYPT 2013 [15]. However, this MitM technique has its own limitation, since the time complexity denoted by $\mathcal{T}$ and memory complexity denoted by $\mathcal{M}$ have to satisfy $\mathcal{M} \times \mathcal{T} \approx 2^{32\times 3} = 2^{96}$ for SHA-256, and the best time-memory tradeoff has almost been achieved in [11]. For the latest 31-step attack in [11], even if we can reduce the memory complexity by increasing the time complexity accordingly, it is still difficult to find a practical colliding message pair in reasonable time with limited computation resources. This is the main motivation of this work, i.e., can we develop a memory-efficient technique to convert SFS collisions into collisions for 31-step SHA-256 such that we can overcome the obstacle caused by $\mathcal{M} \times \mathcal{T} \approx 2^{96}$?

Like the block cipher AES, SHA-256 is obviously of great importance in many real-world applications. For AES, it is common to see some new progress in the cryptanalysis of round-reduced versions almost every few years. In particular, since the multiple-of-8 property for 5-round AES was found [8], different new strategies have been proposed for the key-recovery attacks on 5-round AES. In particular, there is a practical key-recovery attack on 5-round AES with time complexity 2^{24} [1], which was further reduced to $2^{16.5}$ [6]. Despite the same importance as AES, there is no new progress in the collision attack on SHA-256 for nearly 10 years, and a practical collision attack on 31-step SHA-256 is still beyond reach according to the latest results published at EUROCRYPT 2024. To solve this challenge, we may need some new insight into SHA-256 and develop novel techniques.

1.1 Brief Description of SHA-256 and SHA-512

SHA-2 is a family of hash functions including SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256 [5]. All these hash functions are based on the well-known Merkle-Damgård construction [3,16]. Here, we only focus on our targets SHA-256 and SHA-512. Especially, SHA-256 and SHA-512 are almost the same, except the size of the internal state words and the total number of steps.

To compute the hash value of an arbitrary-length message with SHA-256 or SHA-512, it is first padded such that the length in bits becomes a multiple of $16n$, where $n = 32$ for SHA-256 and $n = 64$ for SHA-512. Then, the padded message of $16n\ell$ bits ($\ell > 0$) will be divided into ℓ groups of $16n$-bit message blocks $M_0, M_1, \ldots, M_{l-1}$. Finally, the compression function of SHA-256 or SHA-512 denoted by $f_{\mathtt{com}}$ is used to compress all these message blocks one by one. More specifically, the input of $f_{\mathtt{com}}$ consists of $M_i \in \mathbb{F}_2^{16n}$ and a chaining variable $CV_i \in \mathbb{F}_2^{8n}$, and its output is a new chaining variable $CV_{i+1} \in \mathbb{F}_2^{8n}$, i.e.,

$$CV_{i+1} = f_{\mathtt{com}}(CV_i, M_i),\ 0 \leq i \leq \ell - 1.$$

In this way, the final output CV_ℓ is the hash value of the message M. It should be emphasized that CV_0 is a fixed $8n$-bit constant.

SFS Collisions and Collisions. For a semi-free-start collision attack, the attackers need to find a pair (a, b) and (a, b') where $a \in \mathbb{F}_2^{8n}$ and $b, b' \in \mathbb{F}_2^{16n}$ such that

$$b \neq b' \text{ and } f_{\mathtt{com}}(a, b) = f_{\mathtt{com}}(a, b').$$

For a collision attack using j message blocks, the attackers need to find $(M_0, M_1, \ldots, M_{j-1})$ and $(M'_0, M'_1, \ldots, M'_{j-1})$ such that

$$\begin{aligned} CV_{i+1} &= f_{\mathtt{com}}(CV_i, M_i),\ 0 \leq i \leq j-1, \\ CV'_{i+1} &= f_{\mathtt{com}}(CV'_i, M'_i),\ 0 \leq i \leq j-1 \text{ and } CV_0 = CV'_0. \end{aligned}$$

where

$$(M_0, M_1, \ldots, M_{j-1}) \neq (M'_0, M'_1, \ldots, M'_{j-1}) \text{ and } CV_j = CV'_j,$$

In our new attacks on SHA-256 and SHA-512, we use only two message blocks. More specifically, we aim to find message blocks (M_0, M_1) and (M_0, M'_1) such that

$$M_1 \neq M'_1 \text{ and } CV_2 = CV'_2.$$

Since M_0 is the same in the two messages (M_0, M_1) and (M_0, M_1'), $CV_1 = CV_1'$ always holds and their values depend on M_0.

Brief Description of the Compression Function. Let us further dive into the details of $f_{\mathtt{com}}$ here. $f_{\mathtt{com}}$ is composed of r steps where $r = 64$ for SHA-256 and $r = 80$ for SHA-512. In addition, the two inputs of $f_{\mathtt{com}}(CV, M)$, i.e., $CV \in \mathbb{F}_2^{8n}$ and $M \in \mathbb{F}_2^{16n}$, are both further divided into several blocks of n bits each. Specifically, we have

$$\begin{aligned} CV &= (A_{-4}, A_{-3}, A_{-2}, A_{-1}, E_{-4}, E_{-3}, E_{-2}, E_{-1}) \in (\mathbb{F}_2^n)^8, \\ M &= (W_0, W_1, \ldots, W_{15}) \in (\mathbb{F}_2^n)^{16}. \end{aligned}$$

First, $(W_0, \ldots, W_{15})$ are used to generate $(W_{16}, \ldots, W_{r-1})$ with the following equation, where $\boxplus$ denotes the addition over the ring $\mathbb{Z}_{2^n}$, i.e., modulo 2^n:

$$W_i = \sigma_1(W_{i-2}) \boxplus W_{i-7} \boxplus \sigma_0(W_{i-15}) \boxplus W_{i-16},\ 16 \le i \le r-1.$$

In the above equation,

$$\begin{aligned} \sigma_0(x) &= (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3), \\ \sigma_1(x) &= (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10). \end{aligned}$$

are used in SHA-256, while

$$\begin{aligned} \sigma_0(x) &= (x \ggg 1) \oplus (x \ggg 8) \oplus (x \gg 7), \\ \sigma_1(x) &= (x \ggg 19) \oplus (x \ggg 61) \oplus (x \gg 6). \end{aligned}$$

are used for SHA-512. Note that $\gg$ and $\ggg$ denote *shift right* and *rotate right*, respectively.

With $(W_0, \ldots, W_{r-1})$, it is now easy to describe $f_{\mathtt{com}}$, which simply iterates a step function r times. At the i-th $(0 \le i \le r-1)$ step, the step function is specified as follows:

$$\begin{aligned} E_i &= A_{i-4} \boxplus E_{i-4} \boxplus \Sigma_1(E_{i-1}) \boxplus \mathrm{IF}(E_{i-1}, E_{i-2}, E_{i-3}) \boxplus K_i \boxplus W_i, \\ A_i &= E_i \boxminus A_{i-4} \boxplus \Sigma_0(A_{i-1}) \boxplus \mathrm{MAJ}(A_{i-1}, A_{i-2}, A_{i-3}). \end{aligned}$$

where

$$\begin{aligned} \mathrm{IF}(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus z, \\ \mathrm{MAJ}(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z), \end{aligned}$$

and K_i is a pre-defined constant. As for the functions Σ_0 and Σ_1, they are different in SHA-256 and SHA-512. Specifically,

$$\begin{aligned} \Sigma_0(x) &= (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22), \\ \Sigma_1(x) &= (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25) \end{aligned}$$

are used in SHA-256, while

$$\Sigma_0(x) = (x \ggg 28) \oplus (x \ggg 34) \oplus (x \ggg 39),$$
$$\Sigma_1(x) = (x \ggg 14) \oplus (x \ggg 18) \oplus (x \ggg 41)$$

are used in SHA-512.

After performing the step function for r times, the final output of $f_{\mathtt{com}}$ is computed as

$$(A_{r-1} \boxplus A_{-1}, \ldots, A_{r-4} \boxplus A_{-4}, E_{r-1} \boxplus E_{-1}, \ldots, E_{r-4} \boxplus E_{-4}) \in (\mathbb{F}_2^n)^8.$$

1.2 Technical Overview of Our New Attack

We propose a novel memory-efficient technique to convert SFS collisions into collisions for 31-step SHA-256. As already stated, we will use two message blocks (M_0, M_1) to construct collisions.

Previous MitM Technique. At a high level, in Mendel et al.'s MitM technique for the 31-step collision attack [15], three n-bit words (A_{-1}, A_{-2}, A_{-3}) in CV_1 are required to match simultaneously. Specifically, when processing the second message block M_1, precompute some valid solutions of

$$(A_{-3}, \ldots, A_{12}, E_1, \ldots, E_{12}, W_5, \ldots, W_{12}),$$

and store them in a table. Then, use the first message block M_0 to compute CV_1, and check whether the tuple (A_{-1}, A_{-2}, A_{-3}) in this computed CV_1 is stored in the table. If a match is found, with some additional techniques, a collision can be found for 31-step SHA-256.

Roughly speaking, the time complexity[1] $\mathcal{T}$ and memory complexity $\mathcal{M}$ of this technique have to satisfy $\mathcal{T} \times \mathcal{M} \approx 2^{3n}$, and the optimal $\mathcal{T}$ is achieved when $\mathcal{M} \approx 2^{1.5n}$.

Our New Approach. Different from the above technique, our memory-efficient technique allows us to first consider the match in only one word A_{-1}, and then check the validity of the remaining 2 words (A_{-2}, A_{-3}) on-the-fly. This might be misinterpreted as a simple extension of Mendel et al.'s MitM attack [15] since we turn to considering the match in only one word. We emphasize that Mendel et al.'s attack has been proposed for more than 10 years. Although SHA-256 is as popular and important as AES and ChaCha, no one has even tried to improve this attack framework or question the optimality of this technique. In this sense, it is indeed unclear in the literature whether converting the simultaneous match in 3 words into the match in only one word for the collision attack on SHA-256 is possible. We are the first to challenge the optimality of this old but important

[1] We first ignore some other factors affecting the overall complexity of the attack.

technique, and demonstrate that it can indeed be significantly improved if more details of the differential characteristics are taken into account.

More specifically, when processing the second message block M_1, we will only precompute some valid solutions of

$$(A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{12}, W_7, \ldots, W_{12}),$$

and store them in a table. Then, use the first message block M_0 to compute CV_1, and check whether the corresponding A_{-1} in this computed CV_1 is stored in the table. If it is, check the validity of (A_{-2}, A_{-3}) with other simple calculations.

Denote by $\mathcal{T}_{\texttt{unit}}$ the average time complexity to precompute 1 valid value of $(A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{12}, W_7, \ldots, W_{12})$. In addition, denote the probability that (A_{-2}, A_{-3}) is valid by $\mathcal{P}$ when checked on-the-fly. In this case, the new time complexity $\mathcal{T}$ and memory complexity $\mathcal{M}$ of our technique will satisfy

$$\mathcal{T} = \mathcal{T}_{\texttt{unit}} \times \mathcal{M} + \frac{2^n}{\mathcal{M}} \times \frac{1}{\mathcal{P}}. \tag{1}$$

According to Eq. 1, it is easy to see why we can significantly reduce the memory complexity while the time complexity is still kept low. For example, we may choose $\mathcal{M} = 2^{0.5n}$, and then the time complexity becomes

$$\mathcal{T} = \mathcal{T}_{\texttt{unit}} \times 2^{0.5n} + 2^{0.5n} \times \frac{1}{\mathcal{P}}.$$

If

$$\frac{1}{\mathcal{P}} < 2^n \text{ and } T_{\texttt{unit}} < 2^n,$$

we can achieve the optimal time complexity $2^{1.5n}$ in the previous MitM technique using only $2^{0.5n}$ memory, resulting in an improvement by a factor of 2^n. Similarly, if $T_{\texttt{unit}}$ and $\frac{1}{\mathcal{P}}$ are small enough, we can even achieve the time complexity below $2^{1.5n}$.

Motivation Behind this Approach. According to the above explanation, it is easy to observe that $\mathcal{P}$ has a significant impact on the overall complexity even if $\mathcal{T}_{\texttt{unit}}$ is negligible.[2] The motivation to develop this new technique comes from our observation on the new 31-step differential characteristic for SHA-256 found in [11]. Specifically, it is very sparse, which makes $\mathcal{P}$ quite large. However, in Mendel et al.'s original collision attack on 31-step SHA-256, the differential characteristic is quite dense, resulting in a very small value of $\mathcal{P}$. We are thus motivated to develop new techniques to efficiently find collisions by exploiting the sparsity of the differential characteristic, and we finally find the above new approach.

This also indicates that the sparsity of signed differential characteristics also plays an important role to improve the collision attacks. While the sparsity was mainly used to speed up the search for valid SHA-2 characteristics in [11], we

[2] $\mathcal{T}_{\texttt{unit}}$ is usually very small.

further show that it can also be exploited to speed up the collision attack. This also further demonstrates the advantage of MILP/SAT/SMT-based tools [11,13] to search for signed differential characteristics since optimizing the sparsity of a differential characteristic is relatively easy with them.

Applications. We have applied our new technique to improve the collision attacks on 31-step SHA-256 and SHA-512, respectively. Specifically, we can find a practical colliding message for 31-step SHA-256 in only 1.2 h with 64 threads and negligible memory, as shown in Table 3. For 31-step SHA-512, by searching for a new differential characteristic, the time complexity is reduced from $2^{115.6}$ to $2^{94.7}$, and the memory complexity is significantly reduced from $2^{77.3}$ to $2^{35.2}$. A summary of the collision attacks on SHA-2 family is given in Table 1.

Organization. This paper is organized as follows. First, we briefly recall the signed difference and how to find collisions using a signed differential characteristic in Sect. 2. Then, we explain Mendel et al.'s MitM attack framework in Sect. 3. Our new memory-efficient collision attacks on 31-step SHA-256 and SHA-512 are shown in Sect. 4 and Sect. 5, respectively. Finally, this paper is concluded in Sect. 6.

Table 1. Summary of (SFS) collision attacks on SHA-2.

State size	Hash size	Attack type	Steps	Time ($\mathcal{T}$)	Memory ($\mathcal{M}$)	References	Year
256	All	collision	28	*practical*		[15]	2013
			31	$2^{65.5}$	2^{34}	[15]	2013
			31	$2^{49.8}$	2^{48}	[11]	2023
			31	***practical***		Sect. 4	**2024**
		SFS collision	38	*practical*		[15]	2013
			39	*practical*		[11]	2023
512	All	collision	27	*practical*		[4]	2015
			28	*practical*		[11]	2023
			31	$2^{115.6}$	$2^{77.3}$	[11]	2023
			31	$\mathcal{T} \times \mathcal{M} = 2^{129.9}$ $(1 \leq \mathcal{M} \leq 2^{35.2})$		Sect. 5	**2024**
		SFS collision	38	*practical*		[7]	2014
			39	*practical*		[4]	2015

2 Preliminaries

In this section, we briefly recall Wang et al.'s techniques [17–20] to construct collisions for the MD-SHA hash family by using a signed differential characteristic.

2.1 Signed Difference

Different from the XOR difference that is commonly used in the differential cryptanalysis of block ciphers, the signed difference is mostly used in the collision attacks on the MD-SHA hash family, though it can also be used to analyze ciphers involving modular additions and Boolean functions [12].

For the XOR difference, we are given two n-bit words

$$x = (x[n-1], \ldots, x[0]) \in \mathbb{F}_2^n,\ x' = (x'[n-1], \ldots, x'[0]) \in \mathbb{F}_2^n,$$

and the XOR difference between (x, x') is

$$\varDelta x = (\varDelta x[n-1], \ldots, \varDelta x[0]) = x \oplus x' = (x'[n-1] \oplus x[n-1], \ldots, x'[0] \oplus x[0]).$$

Note that throughout this paper $x[i]$ denotes the i-th bit of x where the index is counted from 0.

For signed difference, the relations between x and x' are more refined. Specifically, if $\varDelta x[i] = 1$, we need to further specify how this bit is changed, i.e., is it changed from 1 to 0, or is it from 0 to 1? In other words, we not only need to capture the fact that the bit is flipped, but also need to trace how the bit is flipped. Indeed, this is equivalent to adding a condition on $x[i]$ apart from its XOR difference $\varDelta x[i]$.

If $\varDelta x[i] = 0$, although this bit remains the same in (x, x'), we may again want to impose a condition on $x[i]$, e.g., $x[i] = 0$ or $x[i] = 1$. Of course, it is still possible there is no extra condition on $x[i]$ even if $\varDelta x[i] = 0$. In this case, this bit is called a free bit, which means that this bit can take either 0 or 1.

To describe the signed difference between (x, x') as stated above, we use the notation $\nabla x = (\nabla x[n-1], \ldots, \nabla x[0])$, which is defined as follows:

$$\nabla x[i] = \begin{cases} \texttt{n} & (x[i]=0, x'[i]=1) \\ \texttt{u} & (x[i]=1, x'[i]=0) \\ \texttt{=} & (x[i]=x'[i] \in \{0,1\}) \\ \texttt{0} & (x[i]=x'[i]=0) \\ \texttt{1} & (x[i]=x'[i]=1) \end{cases} \Leftrightarrow \nabla x[i] = \begin{cases} \texttt{n} & (\varDelta x[i]=1, x[i]=0) \\ \texttt{u} & (\varDelta x[i]=1, x[i]=1) \\ \texttt{=} & (\varDelta x[i]=0, x[i] \in \mathbb{F}_2) \\ \texttt{0} & (\varDelta x[i]=0, x[i]=0) \\ \texttt{1} & (\varDelta x[i]=0, x[i]=1) \end{cases} \quad (2)$$

2.2 Finding Collisions Using Signed Differential Characteristics

The goal of a collision attack is to find two messages M and M' whose hash value is the same. To search for a colliding message pair (M, M'), i.e., $f_{\texttt{com}}(CV, M) =$

$f_{\mathtt{com}}(CV, M')$ where $M \neq M'$ and CV is fixed, it is common to first search for a signed difference characteristic.

More speficially, let the internal state words when compressing M and M' be $(A_0, \ldots, A_{r-1}, E_0, \ldots, E_{r-1})$ and $(A'_0, \ldots, A'_{r-1}, E'_0, \ldots, E'_{r-1})$, respectively. Moreover, let the corresponding message words generated from M and M' be $(W_0, \ldots, W_{r-1})$ and $(W'_0, \ldots, W'_{r-1})$. Then, a signed differential characteristic for r steps of SHA-256 (or SHA-512) is a valid solution of

$$(\nabla A_0, \ldots, \nabla A_{r-1}),\ (\nabla E_0, \ldots, \nabla E_{r-1}),\ (\nabla W_0, \ldots, \nabla W_{r-1}).$$

Note that we emphasize the word *valid* here, because the signed difference does not only specify

$$(\varDelta A_0, \ldots, \varDelta A_{r-1}),\ (\varDelta E_0, \ldots, \varDelta E_{r-1}),\ (\varDelta W_0, \ldots, \varDelta W_{r-1}),$$

but also imposes extra conditions on some bits of

$$(A_0, \ldots, A_{r-1}),\ (E_0, \ldots, E_{r-1}),\ (W_0, \ldots, W_{r-1})$$

such that the bits with non-zero difference can be flipped in the desired direction, i.e., the signed differential characteristic can hold. It should be emphasized that the forms of these conditions cannot be fully described by Eq. 2, as we may also have conditions on $(A_{j_1}[i_1], A_{j_2}[i_2])$ or $(E_{j_1}[i_1], E_{j_2}[i_2])$ or $(W_{j_1}[i_1], W_{j_2}[i_2])$, e.g., $A_{j_1}[i_1] \neq A_{j_2}[i_2]$. These conditions are called the *2-bit conditions*. With this in mind, it is easy to understand the meaning of *valid*, i.e., it refers to that there should be a valid M such that all these conditions on

$$(A_0, \ldots, A_{r-1}),\ (E_0, \ldots, E_{r-1}),\ (W_0, \ldots, W_{r-1})$$

can hold when processing $f_{\mathtt{com}}(CV, M)$. As already studied in [14], searching for a valid signed differential characteristic for SHA-256 is challenging, as contradictions in the conditions easily occur. Fortunately, with the latest SAT-based tools [11], we can search for a possibly valid one in a relatively easy and convenient way, i.e., we can detect as many contradictions as possible in the search, though not all of them are taken into account. For convenience, we call the conditions to make the signed differential characteristic hold the *differential conditions.*

Indeed, once a possibly valid signed differential characteristic is found, we start the so-called *message modification phase*, which is to practically find a concrete M such that the all differential conditions can be fulfilled when processing $f_{\mathtt{com}}(CV, M)$. Once it is found, we immediately get a collision since there must be

$$f_{\mathtt{com}}(CV, M) = f_{\mathtt{com}}(CV, M'),$$

where

$$(W'_0, \ldots, W'_{15}) = (W_0 \oplus \varDelta W_0, \ldots, W_{15} \oplus \varDelta W_{15})$$

and $(\varDelta W_0, \ldots, \varDelta W_{15})$ have been given in the signed differential characteristic.

For example, the differential characteristic used for our collision attack on 31-step SHA-256 is given in Table 2. Apart from the conditions imposed by the signed difference, i.e., which bit should take 0 or 1, there are also 2-bit conditions displayed in Table 4 and Table 5.

3 Previous Collision Attacks on 31-Step SHA-256 and SHA-512

With Wang et al.'s techniques, the common procedure to find a collision consists of 2 steps, as specified below:

Step 1: Find a valid signed differential characteristic.
Step 2: Find a message M that can make all the differential conditions hold.

As valid 31-step differential characteristics have been found for SHA-256 and SHA-512, the problem left is how to efficiently perform the message modification to make all differential conditions hold. Before showing our new approach, we first recall how to find collisions for 31-step SHA-256 with the MitM technique [15] to convert a SFS collision into a collision.

3.1 Finding Collisions with the MitM Technique [15]

For a 31-step differential characteristic of the shape similar to Table 2, Mendel et al. proposed to use two message blocks (M_0, M_1) to find a collision with a MitM technique [15], as detailed below:

Step 1: Based on the used 31-step differential characteristic, pre-compute 2^{γ} valid solutions of

$$(A_{-3}, \ldots, A_{12}, E_1, \ldots, E_{12}, W_5, \ldots, W_{12}).$$

Store these tuples in a table denoted by $\texttt{TAB}_1$.

Step 2: Test $2^{96-\gamma}$ arbitrary first message blocks M_0 and get $2^{96-\gamma}$ values of

$$CV_1 = (A_{-4}, A_{-3}, A_{-2}, A_{-1}, E_{-4}, E_{-3}, E_{-2}, E_{-1}).$$

Check $\texttt{TAB}_1$, find a match between (A_{-3}, A_{-2}, A_{-1}).

Step 3: If a match is found, retrieve the corresponding

$$(A_{-3}, \ldots, A_{12}), (E_1, \ldots, E_{12}), (W_5, \ldots, W_{12})$$

stored in $\texttt{TAB}_1$. Then, determine other message words $(W_0, \ldots, W_4)$ such that the retrieved (A_0, E_1, E_2, E_3) can also be consistent with those computed from CV_1 and $(W_0, \ldots, W_4)$.

To make A_0 consistent, we can compute E_0 satisfying

$$A_0 = E_0 \boxminus A_{-4} \boxplus \Sigma_0(A_{-1}) \boxplus \mathrm{MAJ}(A_{-1}, A_{-2}, A_{-3}).$$

Then, to make this E_0 consistent, we compute W_0 satisfying

$$E_0 = A_{-4} \boxplus E_{-4} \boxplus \Sigma_1(E_{-1}) \boxplus \mathrm{IF}(E_{-1}, E_{-2}, E_{-3}) \boxplus K_0 \boxplus W_0.$$

To make (E_1, E_2, E_3) consistent, we can directly compute (W_1, W_2, W_3) satisfying the following 3 equations:

$$\begin{aligned}
E_3 &= A_{-1} \boxplus E_{-1} \boxplus \Sigma_1(E_2) \boxplus \mathrm{IF}(E_2, E_1, E_0) \boxplus K_3 \boxplus W_3,\\
E_2 &= A_{-2} \boxplus E_{-2} \boxplus \Sigma_1(E_1) \boxplus \mathrm{IF}(E_1, E_0, E_{-1}) \boxplus K_2 \boxplus W_2,\\
E_1 &= A_{-3} \boxplus E_{-3} \boxplus \Sigma_1(E_0) \boxplus \mathrm{IF}(E_0, E_{-1}, E_{-2}) \boxplus K_1 \boxplus W_1.
\end{aligned}$$

Step 4: At this step, $(W_0, \ldots, W_{12})$, (A_{-3}, A_{12}) and $(E_{-3}, \ldots, E_{12})$ have been fixed. The purpose of this step is to find (W_{13}, W_{14}, W_{15}) to fulfill the remaining uncontrolled conditions on $(E_{13}, E_{14}, E_{15}, W_{16}, W_{18})$. If no valid (W_{13}, W_{14}, W_{15}) exist, this step fails and go to Step 2. Denote the success probability by $2^{-\beta}$.

3.2 More Details for Step 1

At Step 1, we pre-compute some solutions of

$$(A_{-3}, \ldots, A_{12}, E_1, \ldots, E_{12}, W_5, \ldots, W_{12})$$

that can satisfy the differential conditions on them, by only considering

$$E_i = A_{i-4} \boxplus E_{i-4} \boxplus \Sigma_1(E_{i-1}) \boxplus \mathrm{IF}(E_{i-1}, E_{i-2}, E_{i-3}) \boxplus K_i \boxplus W_i, \text{ for } 5 \leq i \leq 12,$$

and

$$A_i = E_i \boxminus A_{i-4} \boxplus \Sigma_0(A_{i-1}) \boxplus \mathrm{MAJ}(A_{i-1}, A_{i-2}, A_{i-3}), \text{ for } 1 \leq i \leq 12.$$

This is can be simply automated with a SAT/SMT-based tool. Specifically, we first write the constraints to describe these calculations for $(A_1, \ldots, A_{12})$ and $(E_5, \ldots, E_{12})$, which only requires us to model the value transitions through the modular addition $\boxplus$ and Boolean functions $\Sigma_1, \mathrm{IF}, \Sigma_0, \mathrm{MAJ}$ with some constraints. Then, we further add the constraints to describe the differential conditions to the model. Finally, by using the off-the-shelf solvers, we can easily enumerate solutions satisfying all these constraints. Indeed, the model to describe the value transitions for SHA-2 has been implemented in [11] with SAT/SMT, and we can directly use this tool for Step 1.

3.3 More Details for Step 4

Since

$$\begin{aligned}
W_{16} &= \sigma_1(W_{14}) \boxplus W_9 \boxplus \sigma_0(W_1) \boxplus W_0,\\
W_{18} &= \sigma_1(W_{16}) \boxplus W_{11} \boxplus \sigma_0(W_3) \boxplus W_2,
\end{aligned}$$

where $(W_0, W_1, W_2, W_3, W_9, W_{11})$ have all been fixed at Step 4, we could only use the degrees of freedom of W_{14} to fulfill the differential conditions on (W_{16}, W_{18}).

As there are also differential conditions on W_{14} itself, it is possible that there does not exist valid W_{14} to make all the conditions on (W_{14}, W_{16}, W_{18}) hold for the fixed $(W_0, W_1, W_2, W_3, W_9, W_{11})$. This explains why Step 4 can only succeed with a certain probability $2^{-\beta}$.

If there is a valid W_{14}, fulfilling the remaining conditions on (E_{13}, E_{14}, E_{15}) with (W_{13}, W_{15}) is almost trivial and can be efficiently finished. For completeness, we denote the complexity to find valid (W_{13}, W_{15}) by $\mathcal{T}_{\texttt{last}}$, and it can be simply estimated as $\mathcal{T}_{\texttt{last}} = 2^{\texttt{con}_{14}}$, where $\texttt{con}_{14}$ denotes the number of bit conditions on E_{14}.

3.4 Complexity Analysis

Denote the time complexity to pre-compute 2^γ solutions of

$$(A_{-3}, \ldots, A_{12}, E_1, \ldots, E_{12}, W_5, \ldots, W_{12})$$

by $\mathcal{T}_{\texttt{start}}$. Then, the memory complexity of this attack is 2^γ, and the time complexity is

$$\mathcal{T}_{\texttt{start}} + 2^{96-\gamma+\beta} + 2^\beta \cdot \mathcal{T}_{\texttt{last}}.$$

For the original attack on 31-step SHA-256 [15], there are

$$\gamma \approx 34,\ \beta \approx 3.5,\ \mathcal{T}_{\texttt{start}} \approx 2^{34},\ \mathcal{T}_{\texttt{last}} < 2^{32}$$

and hence the time and memory complexity are $2^{65.5}$ and 2^{34}, respectively.

For the new attack on 31-step SHA-256 in [11], there are

$$\gamma \approx 48,\ \beta \approx 1.3,\ \mathcal{T}_{\texttt{start}} = 2^{48},\ \mathcal{T}_{\texttt{last}} < 2^{32}$$

resulting in time and memory complexity of $2^{49.8}$ and 2^{48}, respectively.

For the new attack on 31-step SHA-512 in [11], there are

$$\gamma \approx 77.3,\ \beta \approx 0.9,\ \mathcal{T}_{\texttt{start}} = 2^{77.3},\ \mathcal{T}_{\texttt{last}} < 2^{64}$$

resulting in time and memory complexity of $2^{115.6}$ and $2^{77.3}$.

4 New Memory-Efficient Collision Attacks

The latest collision attack [11] on 31-step SHA-256 has time complexity of $2^{49.3}$ and memory complexity of 2^{48}. This indicates that a practical collision attack is possible if the memory complexity can be further reduced. This is however quite challenging since this attack is already optimal under the existing attack framework.

Our new insight is that instead of simultaneously matching (A_{-1}, A_{-2}, A_{-3}), we can indeed first consider to match only A_{-1}, and then check the validity of (A_{-3}, A_{-2}) on-the-fly. By such a novel perspective, we can significantly reduce the memory complexity. Accordingly, the attack procedure also becomes different from the previous MitM technique, which roughly consists of the pre-processing phase and the matching phase.

Table 2. The differential characteristic for 31-step SHA-256

i	∇A_i	∇E_i	∇W_i
-4	================================	================================	
-3	================================	================================	
-2	================================	================================	
-1	================================	================================	
0	================================	================================	================================
1	================================	================================	================================
2	================================	================================	================================
3	================================	==========================10====	================================
4	================================	============0===0=========01===0	================================
5	===================n=unnnnnnn=n=	000111010001111110nu=11111unnnu1	================nuuu=======0=uu=
6	========n======================u	101011=11==0n0==u11110==1110011n	==========u=====u===u======n===u
7	===u===n==n========n=========n=u	un0u1100n=01u11111001u1=n110u10n	=u=u=======n=====n=nu=n=====nun=
8	=============================n==	1u01un0u0=1=1=11n=00u0=001001u0=	=u=nn==========u===u===u==1=====
9	================================	01100001110=0=010===00=11101u0=1	================u==========1=u==
10	================u============u==	=1n1uuuuu0100=1un0=10unnnnnnn010	================================
11	================================	=01u1010uu1==11100===1000001n=0=	================================
12	================================	==110001=11====1n====0011110n=0=	================================
13	================================	===0====01======1===============	================================
14	================================	================u===========0u==	================================
15	================================	================0============1==	================================
16	================================	================1============1==	=============unnnunnnnnnnnnnnn==
17	================================	================================	================================
18	================================	================================	==============1=n=0==========n==
19	================================	================================	================================
20	================================	================================	================================
21	================================	================================	================================
22	================================	================================	================================
23	================================	================================	================================
24	================================	================================	================================
25	================================	================================	================================
26	================================	================================	================================
27	================================	================================	================================
28	================================	================================	================================
29	================================	================================	================================
30	================================	================================	================================

4.1 New Pre-processing Phase

Instead of pre-computing valid solutions of

$$(A_{-3}, \ldots, A_{12}, E_1, \ldots, E_{12}, W_5, \ldots, W_{12})$$

as in the previous attacks, we pre-compute valid solutions of

$$(A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{12}, W_7, \ldots, W_{12})$$

in the memory-efficient attack.

Generating Starting Points. Specifically, we first find valid solutions of

$$(A_1, \ldots, A_{12}, E_5, \ldots, E_{12}, W_9, \ldots, W_{12})$$

by only considering

$$E_i = A_{i-4} \boxplus E_{i-4} \boxplus \Sigma_1(E_{i-1}) \boxplus \mathrm{IF}(E_{i-1}, E_{i-2}, E_{i-3}) \boxplus K_i \boxplus W_i, \text{ for } 9 \leq i \leq 12.$$
$$A_i = E_i \boxminus A_{i-4} \boxplus \Sigma_0(A_{i-1}) \boxplus \mathrm{MAJ}(A_{i-1}, A_{i-2}, A_{i-3}), \text{ for } 5 \leq i \leq 12.$$

Among them, we only choose $\mathcal{N}_{\texttt{start}}$ solutions with distinct

$$(A_1, \ldots, A_4, E_5, \ldots, E_8)$$

and call them *starting points*, i.e., $(A_1, \ldots, A_4, E_5, \ldots, E_8)$ are distinct in these starting points.

Such a phase can be efficiently done with a SAT-based tool for simplicity if only a few starting points are required, which is the case of our practical collision attack on 31-step SHA-256. For our attack on SHA-512, we will then use a dedicated method.

Finding Valid $(A_{-1}, A_0, E_3, E_4, W_7, W_8)$ from Each Starting Point. For each obtained starting point, we can first exhaust all possible (W_8, E_4) satisfy the following relation:

$$E_8 = A_4 \boxplus E_4 \boxplus \Sigma_1(E_7) \boxplus \mathrm{IF}(E_7, E_6, E_5) \boxplus K_8 \boxplus W_8. \tag{3}$$

Assuming that there are n_1 and n_2 bit conditions on W_8 and E_4, respectively, a trivial method to find all valid (W_8, E_4) has time complexity of

$$\min(2^{n-n_1}, 2^{n-n_2}),$$

i.e., either exhausting valid W_8 or valid E_4. The expected number of valid pairs is $2^{n-n_1-n_2}$.

For each valid pair (W_8, E_4) satisfying Eq. 3, we then compute the corresponding A_0 according to the following relation:

$$A_4 = E_4 \boxminus A_0 \boxplus \Sigma_0(A_3) \boxplus \mathrm{MAJ}(A_3, A_2, A_1). \tag{4}$$

For each valid tuple (W_8, E_4, A_0), we similarly exhaust all possible (E_3, W_7) satisfying

$$E_7 = A_3 \boxplus E_3 \boxplus \Sigma_1(E_6) \boxplus \mathrm{IF}(E_6, E_5, E_4) \boxplus K_7 \boxplus W_7, \tag{5}$$

and compute the corresponding A_{-1} according to the following relation:

$$A_3 = E_3 \boxminus A_{-1} \boxplus \Sigma_0(A_2) \boxplus \mathrm{MAJ}(A_2, A_1, A_0), \tag{6}$$

Assuming that there are n_3 and n_4 conditions on W_7 and E_3, respectively, we can expect to obtain $2^{n-n_3-n_4}$ possible (A_{-1}, E_3, W_7) from each valid (W_8, E_4, A_0) with a naive method of time complexity

$$\min(2^{n-n_3}, 2^{n-n_4}).$$

According to the above method, it is expected to obtain about

$$\mathcal{N}_{\texttt{start}} \cdot 2^{2n-n_1-n_2-n_3-n_4}$$

valid tuples of

$$(A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{12}, W_7, \ldots, W_{12})$$

from $\mathcal{N}_{\texttt{start}}$ starting points. Store all these valid tuples in a table denoted by $\texttt{TAB}_2$. Denote the time complexity to obtain one starting point by $\mathcal{T}_{\texttt{sat}}$. Then, the time complexity of the pre-processing phase denoted by $\mathcal{T}_{\texttt{pre}}$ can be estimated as

$$\mathcal{T}_{\texttt{pre}} = \mathcal{N}_{\texttt{start}}(\mathcal{T}_{\texttt{sat}} + \min(2^{n-n_1}, 2^{n-n_2}) + 2^{n-n_1-n_2} \times \min(2^{n-n_3}, 2^{n-n_4})). \quad (7)$$

The memory complexity denoted by $\mathcal{M}$ is

$$\mathcal{M} = \mathcal{N}_{\texttt{start}} \cdot 2^{2n-n_1-n_2-n_3-n_4}. \quad (8)$$

4.2 New Matching Phase

After the above pre-processing phase, perform the following attack procedure to find a collision with two message blocks M_0 and M_1.

Step 1: Try an arbitrary M_0, and get the corresponding chaining input

$$CV_1 = (A_{-4}, A_{-3}, A_{-2}, A_{-1}, E_{-4}, E_{-3}, E_{-2}, E_{-1})$$

for the second message block M_1.

Step 2: **Checking A_{-1}:** Search for a match with A_{-1} in $\texttt{TAB}_2$. If there is no match, go to Step 1. Otherwise, for each associated value of

$$(A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{12}, W_7, \ldots, W_{12}),$$

move to Step 3. If all these values are considered, move to Step 1.

Step 3: **Checking (A_{-3}, A_{-2}):** Compute (E_0, E_1, E_2) to make the associated (A_0, A_1, A_2) consistent with those computed from this CV_1 according to the following 3 equations:

$$\begin{aligned} A_0 &= E_0 \boxminus A_{-4} \boxplus \Sigma_0(A_{-1}) \boxplus \mathrm{MAJ}(A_{-1}, A_{-2}, A_{-3}), \\ A_1 &= E_1 \boxminus A_{-3} \boxplus \Sigma_0(A_0) \boxplus \mathrm{MAJ}(A_0, A_{-1}, A_{-2}), \\ A_2 &= E_2 \boxminus A_{-2} \boxplus \Sigma_0(A_1) \boxplus \mathrm{MAJ}(A_1, A_0, A_{-1}). \end{aligned}$$

Then, compute (W_4, W_5, W_6) to make the associated (E_4, E_5, E_6) also consistent with those compted from this CV_1:

$$\begin{aligned} E_4 &= A_0 \boxplus E_0 \boxplus \Sigma_1(E_3) \boxplus \mathrm{IF}(E_3, E_2, E_1) \boxplus K_4 \boxplus W_4, \\ E_5 &= A_1 \boxplus E_1 \boxplus \Sigma_1(E_4) \boxplus \mathrm{IF}(E_4, E_3, E_2) \boxplus K_5 \boxplus W_5, \\ E_6 &= A_2 \boxplus E_2 \boxplus \Sigma_1(E_5) \boxplus \mathrm{IF}(E_5, E_4, E_3) \boxplus K_6 \boxplus W_6. \end{aligned}$$

If all the conditions on $(E_0, E_1, E_2, W_4, W_5, W_6)$ hold, move to Step 4. Otherwise, move to Step 2 and try another associated value of

$$(A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{12}, W_7, \ldots, W_{12}).$$

Step 4: At this step, for the associated value of

$$(A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{12}, W_7, \ldots, W_{12}),$$

(A_0, A_1, A_2, A_3) and (E_4, E_5, E_6, E_7) must be consistent with those computed from the new CV_1, and $(W_4, W_5, W_6, E_0, E_1, E_2, E_3)$ are also fixed. The remaining task is to compute (W_0, W_1, W_2, W_3) such that this fixed (E_0, E_1, E_2, E_3) can also be consistent with those computed from the new CV_1, as shown below:

$$\begin{aligned}
E_3 &= A_{-1} \boxplus E_{-1} \boxplus \Sigma_1(E_2) \boxplus \mathrm{IF}(E_2, E_1, E_0) \boxplus K_3 \boxplus W_3,\\
E_2 &= A_{-2} \boxplus E_{-2} \boxplus \Sigma_1(E_1) \boxplus \mathrm{IF}(E_1, E_0, E_{-1}) \boxplus K_2 \boxplus W_2,\\
E_1 &= A_{-3} \boxplus E_{-3} \boxplus \Sigma_1(E_0) \boxplus \mathrm{IF}(E_0, E_{-1}, E_{-2}) \boxplus K_1 \boxplus W_1,\\
E_0 &= A_{-4} \boxplus E_{-4} \boxplus \Sigma_1(E_{-1}) \boxplus \mathrm{IF}(E_{-1}, E_{-2}, E_{-3}) \boxplus K_0 \boxplus W_0.
\end{aligned}$$

Step 5: The last step is the same as in the previous MitM attack. Specifically, use the degrees of freedom in (W_{13}, W_{14}, W_{15}) to fulfill the remaining uncontrolled conditions on $(E_{13}, E_{14}, E_{15}, W_{16}, W_{18})$. If this step fails, move to Step 2 and try another associated value of

$$(A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{12}, W_7, \ldots, W_{12}).$$

Otherwise, a collision is found. Abusing notation, we denote the success probability of this step by $2^{-\beta}$.

Remark 1. We explain here why it is necessary to let $\mathcal{N}_{\mathtt{start}}$ be the number of starting points with distinct $(A_1, \ldots, A_4, E_5, \ldots, E_8)$. On the one hand, according to the pre-processing phase, finding valid $(A_{-1}, A_0, E_3, E_4, W_7, W_8)$ only requires the knowledge of $(A_1, \ldots, A_4, E_5, \ldots, E_8)$. On the other hand, at the matching phase, checking the validity of (A_{-3}, A_{-2}) only requires the knowledge of $(A_0, A_1, A_2, E_3, \ldots, E_6)$ and CV_1. Hence, the same $(A_1, \ldots, A_4, E_5, \ldots, E_8)$ will lead to the same set of valid $(A_{-1}, A_0, E_3, E_4, W_7, W_8)$, which cannot effectively increase the size of $\mathtt{TAB}_2$. As a result, the matching probability will not be improved even if we have more than 1 different valid values of

$$(A_1, \ldots, A_{12}, E_5, \ldots, E_{12}, W_9, \ldots, W_{12})$$

with the same $(A_1, \ldots, A_4, E_5, \ldots, E_8)$. However, it is also obvious that if we allow such valid values with the same $(A_1, \ldots, A_4, E_5, \ldots, E_8)$ but different W_{11}, the cost of the last step of the matching phase can be amortized. The reason is simple. Once (A_{-3}, A_{-2}, A_{-1}) pass the test, i.e., Step 3 of the matching phase, we can then use the degrees of freedom of (W_{14}, W_{11}) rather than only W_{14} to fulfill all the differential conditions on (W_{16}, W_{18}). Hence, we only expect to have 2^{β} different W_{11} for each $(A_1, \ldots, A_4, E_5, \ldots, E_8)$ to amortize the cost of the last step. Since $\beta \in \{1.3, 0.9\}$ in our attacks on SHA-256 and SHA-512 is quite small, such a strategy can only slightly improve the attack, and we thus ignore this to make the analysis simpler.

4.3 Complexity Analysis

Denote the total number of conditions on $(E_0, E_1, E_2, W_4, W_5, W_6)$ by $\mathcal{N}_{\texttt{pro}}$. Since there are no conditions on $(W_0, \ldots, W_3)$, for each

$$(CV_1, A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{12}, W_7, \ldots, W_{12})$$

after a match is found, it is valid with probability

$$\mathcal{P} = 2^{-\mathcal{N}_{\text{pro}}-\beta}.$$

If $\mathcal{N}_{\texttt{test}}$ different M_0 are tested, we then expect to test on average

$$\mathcal{N}_{\texttt{test}} \times \frac{\mathcal{N}_{\texttt{start}} \cdot 2^{2n-n_1-n_2-n_3-n_4}}{2^n}$$

different values of

$$(CV_1, A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{12}, W_7, \ldots, W_{12})$$

for Step 3–5. Hence, it is required to have

$$\mathcal{N}_{\texttt{test}} \times \frac{\mathcal{N}_{\texttt{start}} \cdot 2^{2n-n_1-n_2-n_3-n_4}}{2^n} \geq \frac{1}{\mathcal{P}} = 2^{\mathcal{N}_{\text{pro}}+\beta}$$

in order to find a collision, i.e.,

$$\mathcal{N}_{\texttt{test}} \geq \frac{2^{n+\mathcal{N}_{\text{pro}}+\beta}}{\mathcal{N}_{\texttt{start}} \cdot 2^{2n-n_1-n_2-n_3-n_4}} = \frac{2^{\mathcal{N}_{\text{pro}}+\beta+n_1+n_2+n_3+n_4-n}}{\mathcal{N}_{\texttt{start}}}.$$

The time complexity of the matching phase is thus estimated as

$$\mathcal{T}_{\texttt{match}} = \frac{2^{\mathcal{N}_{\text{pro}}+\beta+n_1+n_2+n_3+n_4-n}}{\mathcal{N}_{\texttt{start}}} + 2^{\mathcal{N}_{\text{pro}}+\beta}. \tag{9}$$

Taking the cost of the pre-processing phase into account, i.e., Eq. 7 and Eq. 8, the time and memory complexity of our new attacks are $\mathcal{T}_{\texttt{pre}} + \mathcal{T}_{\texttt{match}}$ and $\mathcal{N}_{\texttt{start}} \cdot 2^{2n-n_1-n_2-n_3-n_4}$, respectively.

4.4 A Practical Colliding Message Pair for 31-Step SHA-256

For the 31-step differential characteristic in Table 2, where all the 2-bit conditions are listed in Table 4 and Table 5, we have

$$n = 32,\ n_1 = 19,\ n_2 = 5,\ n_3 = 7,\ n_4 = 23,\ \mathcal{N}_{\texttt{pro}} = 27,\ \beta \approx 1.3.$$

Hence, the expected time complexity of our attack is

$$\mathcal{T}_{\texttt{pre}} + \mathcal{T}_{\texttt{match}} = \mathcal{N}_{\texttt{start}} \cdot (\mathcal{T}_{\texttt{sat}} + 2^{17}) + \frac{2^{50.3}}{\mathcal{N}_{\texttt{start}}} + 2^{28.3},$$

and the expected memory complexity is $\mathcal{N}_{\texttt{start}} \cdot 2^{10}$.

The Actual Cost. In our experiments, we first generate $\mathcal{N}_{\texttt{start}} = 200$ starting points with the SAT-based tool. Then, based on these starting points, we obtain in total $2^{19.8}$ possible

$$(A_{-1}, \ldots, A_{12}, E_3, \ldots, E_{12}, W_7, \ldots, W_{12})$$

at the pre-processing phase. Note that this number is expected to be $200 \times 2^{10} = 204800$ if all bit conditions on (E_3, E_4, W_7, W_8) are independent. Experiments indicate that this number is larger. Therefore, the memory complexity of our attack is $2^{19.8}$, and the complexity of the matching phase becomes

$$\frac{2^{32+27+1.3}}{2^{19.8}} + 2^{28.3} \approx 2^{40.5}.$$

By performing the matching phase, we find a practical colliding message pair in about 1.2 h using 64 threads[3], as shown Table 3.

Table 3. The colliding message pair for 31 steps of SHA-256

M	8ce3f805	5c401aed	579e5f7f	bc3116cb	ca189b3c	eb75f04c	958f0a0e	7760b082
	dcd5027d	32260ad6	7b12b659	eee66518	ad7f88dd	f8ad20bb	7ae40ffd	21609249
M_1	9abdeb1b	1f195f41	5a7210c1	55614f13	a2269dd1	be888a61	359257d4	adf3737b
	9f0484a6	eb830a58	66add94a	9669232d	45271fa5	b8f69585	428bbce3	0703b904
M_1'	9abdeb1b	1f195f41	5a7210c1	55614f13	a2269dd1	be887a67	35b2dfc5	fde32975
	c70595a6	eb838a5c	66add94a	9669232d	45271fa5	b8f69585	428bbce3	0703b904
hash	ff558659	2977dd01	54638843	35f8de84	a3336841	f4f476f2	7c571548	f7025605

Remark 2. It should be remarked that our technique is general and can also be applied to the 31-step differential characteristic in [15]. However, this 31-step differential characteristic is quite dense, and there are too many conditions on (W_4, W_5, W_6), which make our new technique have a much higher complexity. This is also further demonstrates the advantage to use MILP/SAT/SMT-based tools to search for signed differential characteristics [11,13] since we can easily optimize the uncontrolled part.

5 Improved Collision Attack on 31-Step SHA-512

The above method to improve the collision attack on 31-step SHA-256 can be trivially applied to SHA-512. However, there seems to be room to further improve

[3] We were very lucky to obtain a colliding message pair in seconds when first running the attack. But this was too special. Therefore, we ran more experiments and then obtained a new colliding message pair in 1.2 h.

Table 4. W_i conditions for SHA-256

	conditions
W_5	$W_5[3] \neq W_5[31]$, $W_5[23] = W_5[19]$, $W_5[22] = W_5[18]$, $W_5[19] \neq W_5[30]$, $W_5[23] \neq W_5[8]$, $W_5[17] \neq W_5[28]$, $W_5[16] \neq W_5[27]$, $W_5[26] = W_5[11]$, $W_5[24] \neq W_5[9]$, $W_5[18] \neq W_5[29]$, $W_5[25] \neq W_5[10]$.
W_6	$W_6[19] = W_6[30]$, $W_6[22] \neq W_6[7]$, $W_6[10] = W_6[6]$, $W_6[8] \neq W_6[19]$.
W_7	$W_7[7] \neq W_7[24]$, $W_7[23] \neq W_7[19]$, $W_7[22] \neq W_7[18]$, $W_7[31] = W_7[16]$, $W_7[23] = W_7[8]$, $W_7[18] \neq W_7[29]$, $W_7[17] = W_7[13]$, $W_7[16] = W_7[27]$, $W_7[22] \neq W_7[7]$, $W_7[25] = W_7[10]$, $W_7[13] = W_7[24]$, $W_7[15] \neq W_7[26]$, $W_7[7] = W_7[18]$, $W_7[19] = W_7[15]$.
W_8	$W_8[19] = W_8[4]$, $W_8[2] \neq W_8[13]$, $W_8[9] \neq W_8[26]$, $W_8[17] = W_8[13]$, $W_8[31] = W_8[10]$, $W_8[1] \neq W_8[29]$, $W_8[29] \neq W_8[25]$, $W_8[7] = W_8[24]$, $W_8[6] = W_8[23]$, $W_8[20] \neq W_8[31]$, $W_8[19] \neq W_8[15]$, $W_8[0] = W_8[11]$.
W_9	$W_9[13] \neq W_9[30]$, $W_9[23] \neq W_9[19]$, $W_9[26] \neq W_9[11]$, $W_9[9] \neq W_9[20]$.
W_{16}	$W_{16}[1] = W_{16}[26]$, $W_{16}[23] \neq W_{16}[25]$, $W_{16}[25] \neq W_{16}[27]$, $W_{16}[24] \neq W_{16}[26]$, $W_{16}[0] \neq W_{16}[25]$, $W_{16}[22] \neq W_{16}[24]$, $W_{16}[21] \neq W_{16}[23]$, $W_{16}[20] \neq W_{16}[22]$, $W_{16}[19] \neq W_{16}[21]$.
W_{18}	$W_{18}[4] = W_{18}[27]$, $W_{18}[2] \neq W_{18}[25]$, $W_{18}[22] = W_{18}[24]$.

the previous differential characteristic for 31-step SHA-512 in order to take full advantage of our new technique. Different from the attack on SHA-256, we do not expect a practical collision attack on 31-step SHA-512 because it is still challenging to obtain a valid and sparse differential characteristic for 31-step SHA-512 with the SAT/SMT-based tool. We leave this as an open problem, and expect that our technique can be useful for the practical collision attack on 31 steps of SHA-512 in the future. What we could do is to optimize the differential characteristic as far as we can. The new 31-step differential characteristic is shown in Table 6. To demonstrate that the new 31-step differential characteristic is valid, we provide a SFS collision instance shown in Table 7.

As can be observed from Eq. 9, there are many factors affecting the complexity of the matching phase. Specifically, if $n_1 + n_2 + n_3 + n_4$ and $\mathcal{N}_{\texttt{pro}} + \beta$ are small, and there are also many available starting points, i.e., $\mathcal{N}_{\texttt{start}}$ can be large, the collision attack can be much more efficient. However, searching a valid differential characteristic for SHA-512 is time-consuming, and we could not take all these factors into account when writing the SAT model. After several different trials, we eventually choose to minimize the number of conditions on E_3 and E_4 so that $n_1 + n_2 + n_3 + n_4$ can be small. The corresponding new 31-step differential characteristic is displayed in Table 6.

Table 5. The A_i and E_i conditions for SHA-256

	conditions
A_3	$A_3[1] = A_4[1]$, $A_3[3] = A_4[3]$, $A_3[4] \neq A_4[4]$, $A_3[12] \neq A_4[12]$, $A_3[7] \neq A_4[7]$, $A_3[8] = A_4[8]$, $A_3[9] \neq A_4[9]$, $A_3[10] = A_4[10]$, $A_3[6] \neq A_4[6]$, $A_3[5] = A_4[5]$.
A_4	$A_4[1] = A_6[1]$, $A_4[3] \neq A_6[3]$, $A_4[4] = A_6[4]$, $A_4[23] \neq A_5[23]$, $A_4[7] \neq A_6[7]$, $A_4[8] \neq A_6[8]$, $A_4[9] \neq A_6[9]$, $A_4[10] = A_6[10]$, $A_4[0] = A_5[0]$, $A_4[5] = A_6[5]$, $A_4[6] = A_6[6]$, $A_4[12] \neq A_6[12]$.
A_5	$A_5[31] \neq A_5[19]$, $A_5[30] \neq A_5[18]$, $A_5[29] \neq A_5[17]$, $A_5[28] \neq A_5[16]$, $A_5[26] \neq A_5[14]$, $A_5[25] \neq A_5[13]$, $A_5[21] \neq A_5[0]$, $A_5[20] = A_5[31]$, $A_5[18] \neq A_5[29]$, $A_5[17] \neq A_5[28]$, $A_5[16] \neq A_5[27]$, $A_5[17] = A_5[26]$, $A_5[15] = A_5[26]$, $A_5[13] = A_5[24]$, $A_5[23] = A_5[0]$, $A_5[21] \neq A_5[30]$, $A_5[19] \neq A_5[28]$, $A_5[18] \neq A_5[27]$, $A_5[15] = A_5[24]$, $A_5[14] = A_5[23]$, $A_5[19] \neq A_5[30]$, $A_5[16] \neq A_5[25]$, $A_5[27] \neq A_5[15]$, $A_5[20] = A_5[29]$.
A_6	$A_5[2] \neq A_6[2]$, $A_5[21] \neq A_6[21]$, $A_5[24] \neq A_6[24]$, $A_5[28] = A_6[28]$, $A_6[2] = A_6[11]$, $A_6[21]! = A_6[9]$, $A_6[11] \neq A_6[20]$, $A_6[3] = A_6[14]$.
A_7	$A_7[4] = A_7[15]$, $A_7[11] \neq A_7[20]$, $A_7[7] \neq A_7[16]$, $A_7[23] \neq A_7[11]$, $A_7[14] \neq A_7[25]$, $A_7[13] = A_7[1]$, $A_7[10] = A_7[30]$, $A_7[8] = A_7[19]$, $A_7[13] \neq A_7[22]$, $A_7[4] = A_7[15]$, $A_7[5] = A_7[17]$, $A_7[13] \neq A_7[22]$, $A_5[23] = A_7[23]$.
A_8	$A_8[23] \neq A_8[11]$, $A_8[14] = A_8[25]$, $A_8[13] \neq A_8[22]$, $A_6[12] = A_8[12]$, $A_6[24] = A_8[24]$, $A_6[28] = A_8[28]$, $A_6[21] \neq A_8[21]$, $A_7[23] = A_8[23]$.
A_9	$A_8[0] \neq A_9[0]$, $A_8[12] = A_9[12]$, $A_8[21] = A_9[21]$, $A_8[24] = A_9[24]$, $A_8[28] = A_9[28]$, $A_8[15] = A_9[15]$.
A_{10}	$A_{10}[4] \neq A_{10}[24]$, $A_{10}[23] = A_{10}[11]$, $A_{10}[26] \neq A_{10}[3]$, $A_{10}[27] \neq A_{10}[6]$, $A_{10}[25] = A_{10}[14]$, $A_{10}[13] \neq A_{10}[22]$.
A_{11}	$A_9[15] = A_{11}[15]$, $A_9[2] = A_{11}[2]$.
A_{12}	$A_{11}[15] = A_{12}[15]$, $A_{11}[2] = A_{12}[2]$.
E_3	$E_3[1] = E_4[1]$, $E_3[2] = E_4[2]$, $E_3[3] = E_4[3]$, $E_3[12] = E_4[12]$, $E_3[13] = E_4[13]$.
E_7	$E_7[8] \neq E_7[22]$.
E_8	$E_8[9] \neq E_8[22]$, $E_8[0] = E_8[14]$.
E_{10}	$E_{10}[31] \neq E_{10}[18]$, $E_{10}[31] \neq E_{10}[13]$.
E_{12}	$E_{12}[30] \neq E_{12}[17]$, $E_{12}[20] \neq E_{12}[2]$.
E_{14}	$E_{14}[29] = E_{14}[10]$, $E_{14}[28] \neq E_{14}[1]$, $E_{14}[29] \neq E_{14}[16]$, $E_{14}[7] \neq E_{14}[21]$.

Table 6. The differential characteristic for 31-step SHA-512

i	∇A_i	∇E_i	∇W_i
-4	==	==	
-3	==	==	
-2	==	==	
-1	==	==	
0	==	==	==
1	==	==	==
2	==	==	==
3	==	==1======0===========0======0=====0==0======1====1=1=======0====	==
4	==	0=001010100=1========0====0=1=00==1==1=0====0===1001==0==0=1==1=	==
5	=nunuuuuuu======u======u===unnnu===u=n======u=====nn===unnnn====	10n011101n01010=u0=01unn1011un0n10un0n0101=1u=010n1u=110=0un1110	==n======n=====un======u=====u=nunnnnu======u=====nn=======u====
6	========n=======u================un========u=====unnnn====nuunn=	n11nunnunnu1u10u11101n0100u100uu0100100n0=10u010u0un11u11u1u11n1	n0u011=01u=====0n====10n====un=============n====uu===u=u=nuuuuu=
7	=n======nunnu==u=u=========u==uu=n=u========nnn====n=====u=u====	uu1n011nu1nuu1n1nu01nn0n1u1n111n0001u1111100110u0n0101110n0n101n	=u=============n============u====u==unn==n====u=======n=0=1====u
8	===u	10010111000100u110101u0n00n0=000=n=0010n=1nu1un1n1n01001n1101100	n=====nu=====u=====nu=====n======u=====u======u==========0======
9	========================n=======================================	10n0==001=101111010=01unn01100u0=000u1u10=11=0100100=1==10=1=001	============u============n============n========================n
10	============u===========nu============n======================nuu	==11==1===010=00u0==101=0n0==10==11110n11=1u=100000=0n==1111=nuu	==
11	==	=0u1======un0=n=nn===11001u==un====unnn0n==n=========u==u=00=100	==
12	==	=000======00n=1=11===10==u1==00===1010u11==1=========1==1==1=11u	==
13	==	==1=======110===01=======11==11====1111=1==0=========0==1======1	==
14	==	============u============n============n========================n	==
15	==	============0============0============0========================0	==
16	==	============1============1============1========================1	=======unnnnn====nuuuuuuuu=======nuuuuu==========11====nuuuuuuuu
17	==	==	==
18	==	==	============n============u============u========================u
19	==	==	==
20	==	==	==
21	==	==	==
22	==	==	==
23	==	==	==
24	==	==	==
25	==	==	==
26	==	==	==
27	==	==	==
28	==	==	==
29	==	==	==
30	==	==	==

Table 7. The SFS colliding message pair for 31 steps of SHA-512

CV	c063f6b37f43a1d8	0bb4a9c28ec8ebbf	719325677a147967	e87e837a5a274200
	a536e4afe540d5e8	6425fc52b3568687	f3e944647335886d	e132ce8175d55506
M	015cdb1cb4a8c188	807399b1649a04c3	ea97e34008081810	30a030e8f2399184
	c000010231aa3780	20c0ba3979163000	8ca8b53683f11a40	99438b3407fc9622
	a6c8f225183d6a33	d5e316641e7c1855	9225d8d423ee4b4c	a23f1fc91747798d
	a97700a8603f3df1	c077cb3a0e772512	34221f18d5728600	b8810e0098b8aa5d
M'	015cdb1cb4a8c188	807399b1649a04c3	ea97e34008081810	30a030e8f2399184
	c000010231aa3780	00813b3c851e0010	2ce8343a83e1df3ed	9428b3c49be9423
	25ccea05593f6a33	d5eb16241c7c1854	9225d8d423ee4b4c	a23f1fc91747798d
	a97700a8603f3df1	c077cb3a0e772512	34221f18d5728600	b8810e0098b8aa5d
hash	76b60192018bae4b	bb93482fb546f9c5	6e9394a42ff48f7a	de8654fd2a5f2a70
	8ee9dee6bbe2059e	54a70df6e7a436a5	199bcc4c6ff152a5	ebcf4b290e40ff80

For this new 31-step differential characteristic, we have

$$n = 64,\ n_1 = 20,\ n_2 = 29,\ n_3 = 36,\ n_4 = 16,\ \mathcal{N}_{\texttt{pro}} = 65,\ \beta \approx 0.9.$$

Therefore, the expected time complexity of the new collision attack on 31-step SHA-256 becomes

$$\mathcal{T}_{\texttt{pre}} + \mathcal{T}_{\texttt{match}} = \mathcal{N}_{\texttt{start}} \cdot (\mathcal{T}_{\texttt{sat}} + 2^{43}) + \frac{2^{101.9}}{\mathcal{N}_{\texttt{start}}} + 2^{65.9}, \tag{10}$$

and the expected memory complexity is $\mathcal{N}_{\texttt{start}} \cdot 2^{27}$.

Simple Improvement and the Expected Complexity. Since we can simply find a starting point with the SAT-based tool, using $\mathcal{N}_{\texttt{start}} = 1$, we have already significantly improved the previous attack. Specifically, we can reduce the time complexity from $2^{115.6}$ to $2^{102.9}$, and meanwhile reduce the memory complexity from $2^{77.3}$ to 2^{29}.

Improving the Attack Using All Starting Points. Obviously, according to Eq. 10, the time complexity can be significantly reduced if more available starting points can be used. Hence, we are motivated to enumerate all possible starting points for the new 31-step differential characteristic, i.e., finding all possible $(A_1, \ldots, A_4, E_4, \ldots, E_8)$ that can lead to valid solutions of

$$(A_1, \ldots, A_{12}, E_5, \ldots, E_{12}, W_9, \ldots, W_{12}).$$

As the first step, we list all the 2-bit conditions for this 31-step differential characteristic, as shown in Table 8, Table 9, Table 10 and Table 11. We find that there are 34 free bits in A_8. However, by a clever test, we find that there is indeed only one value of A_8 that can lead to valid solutions of $(A_1, \ldots, A_{12}, E_5, \ldots, E_{12},$

$W_9, \ldots, W_{12}$). Specifically, if $A_8[i]$ is a free bit, we fix it to 0 or 1, and then call the solver to output a valid solution by modelling the value transitions. If it returns UNSAT for $A_8[i] = 0$ (resp. $A_8[i] = 1$), it means $A_8[i] = 1$ (resp. $A_8[i] = 0$) should hold. By testing each free bit of A_8 in this way, we find that A_8 can only take 1 value.

Moreover, by the exhaustive search, we also find that there are only 4194304 valid solutions of $(A_5, \ldots, A_8)$ that can fulfill their differential conditions.

For each valid $(A_5, \ldots, A_8)$, we then exhaust all possible E_9 and compute A_9 using

$$A_9 = E_9 \boxminus A_5 \boxplus \Sigma_0(A_8) \boxplus \mathrm{MAJ}(A_8, A_7, A_6).$$

In this way, we can obtain all 64256 valid values of $(A_5, \ldots, A_9, E_9)$.

Next, for each valid $(A_5, \ldots, A_9, E_9)$, exhaust all possible E_8 and compute A_4 using

$$A_8 = E_8 \boxminus A_4 \boxplus \Sigma_0(A_7) \boxplus \mathrm{MAJ}(A_7, A_6, A_5).$$

Then, in total of 1344 valid $(A_4, \ldots, A_9, E_8, E_9)$ are obtained.

Next, we similarly exhaust all possible E_7, and obtain all 112 valid values of $(A_3, \ldots, A_9, E_7, \ldots, E_9)$. Finally, exhaust all possible (E_6, E_5), and compute valid values of (A_2, A_1, W_9) using

$$\begin{aligned}
A_6 &= E_6 \boxminus A_2 \boxplus \Sigma_0(A_5) \boxplus \mathrm{MAJ}(A_5, A_4, A_3), \\
A_5 &= E_5 \boxminus A_1 \boxplus \Sigma_0(A_4) \boxplus \mathrm{MAJ}(A_4, A_3, A_2), \\
E_9 &= A_5 \boxplus E_5 \boxplus \Sigma_1(E_8) \boxplus \mathrm{IF}(E_8, E_7, E_6) \boxplus K_9 \boxplus W_9.
\end{aligned}$$

With the above method, we find in total 224 valid values of

$$(A_1, \ldots, A_9, E_5, \ldots, E_9, W_9)$$

in practical time. Among them, only 4 valid values can ensure that there is a valid solution of $(A_{10}, \ldots, A_{12}, E_{10}, \ldots, E_{12}, W_{10}, \ldots, W_{12})$ such that

$$\begin{aligned}
E_i &= A_{i-4} \boxplus E_{i-4} \boxplus \Sigma_1(E_{i-1}) \boxplus \mathrm{IF}(E_{i-1}, E_{i-2}, E_{i-3}) \boxplus K_i \boxplus W_i, \text{ for } 10 \le i \le 12, \\
A_i &= E_i \boxminus A_{i-4} \boxplus \Sigma_0(A_{i-1}) \boxplus \mathrm{MAJ}(A_{i-1}, A_{i-2}, A_{i-3}), \text{ for } 10 \le i \le 12.
\end{aligned}$$

Moreover, among these 4 valid values, $(A_1, \ldots, A_4, E_5, \ldots, E_8)$ are distinct. Hence, the number of useful starting points for our attack is 4.

Constructing $\mathtt{TAB}_2$ Using All 4 Starting Points. For each starting point, as described at the pre-processing phase, we can use a naive exhaustive search to find all possible $(A_{-1}, A_0, E_3, E_4, W_7, W_8)$ and store them in $\mathtt{TAB}_2$. The expected time complexity of this naive method to enumerate $(A_{-1}, A_0, E_3, E_4, W_7, W_8)$ for all 4 starting points is $4 \times (2^{64-29} + 2^{64-20-29} \times 2^{64-36}) \approx 2^{45}$. Moreover, we expect the size of $\mathtt{TAB}_2$ to be $4 \times 2^{128-20-29-36-16} = 2^{29}$. However, these numbers are computed under the assumption that all the differential conditions on (E_3, E_4, W_7, W_8) are linearly independent, which may not hold in practice. Therefore, we also performed an exhaustive search to enumerate all these $(A_{-1}, A_0, E_3, E_4, W_7, W_8)$ for all the 4 starting points. To improve the efficiency

Table 8. The $A_3 - A_6$ conditions for SHA-512

	conditions
A_3	$A_3[i] = A_4[i], i \in \{5, 6, 7, 12, 13, 19, 26, 33, 36, 54, 58, 59, 60\}$ $A_3[i] \neq A_4[i], i \in \{4, 8, 28, 32, 34, 35, 40, 47, 55, 56, 57, 61, 62\}$
A_4	$A_4[i] = A_5[i], i \in \{2, 3, 10, 11, 29, \}$ $A_4[i] \neq A_5[i], i \in \{1, 14, 20\}$ $A_4[i] = A_6[i], i \in \{7, 8, 58, 59, 60, 61, 62, \}$ $A_4[i] \neq A_6[i], i \in \{6, 19, 26, 32, 33, 34, 35, 36, 40, 54, 56, 57\}$
A_5	$A_5[27] \neq A_5[38], A_5[17] = A_5[23], A_5[53] = A_5[11], A_5[46] = A_5[51],$ $A_5[25] = A_5[31], A_5[15] = A_5[21], A_5[52] \neq A_5[63], A_5[41] \neq A_5[46],$ $A_5[24] \neq A_5[30], A_5[18] = A_5[23], A_5[46] = A_5[52], A_5[39] \neq A_5[44],$ $A_5[23] = A_5[29], A_5[11] \neq A_5[16], A_5[45] = A_5[51], A_5[38] = A_5[43],$ $A_5[21] = A_5[27], A_5[10] \neq A_5[15], A_5[44] = A_5[50], A_5[30] = A_5[41],$ $A_5[20] = A_5[31], A_5[41] \neq A_5[52], A_5[43] = A_5[49], A_5[25] = A_5[30],$ $A_5[63] = A_5[10], A_5[11] = A_5[0].$
A_6	$A_6[23] = A_6[34], A_6[6] = A_6[59], A_6[41] = A_6[52], A_6[16] = A_6[21],$ $A_6[22] \neq A_6[33], A_6[0] \neq A_6[58], A_6[36] = A_6[42], A_6[53] \neq A_6[58],$ $A_6[26] = A_6[31], A_6[7] = A_6[18], A_6[63] = A_6[57], A_6[35] = A_6[40],$ $A_6[18] \neq A_6[24], A_6[6] \neq A_6[17], A_6[56] \neq A_6[62], A_6[34] \neq A_6[39],$ $A_6[19] \neq A_6[24], A_6[7] = A_6[60], A_6[54] = A_6[60], A_6[17] \neq A_6[23],$ $A_6[18] \neq A_6[23], A_6[9] = A_6[62], A_6[49] = A_6[60], A_6[44] \neq A_6[50],$ $A_6[17] \neq A_6[22], A_6[8] \neq A_6[61], A_6[9] \neq A_6[15], A_6[8] \neq A_6[19],$ $A_6[0] = A_6[6].$ $A_5[i] = A_6[i], i \in \{46, 48, 51, 52, 53\}$ $A_5[i] \neq A_6[i], i \in \{17, 18, 30\}$

of the exhaustive search, instead of using the naive method, we use a tree-based searching structure and perform the depth-first search. The details are explained below.

For a given $(A_1, \ldots, A_4, E_4, \ldots, E_8)$ in the starting point, we first need to enumerate all valid (E_4, W_8) satisfying their differential conditions and the following equation:

$$E_8 = A_4 \boxplus E_4 \boxplus \Sigma_1(E_7) \boxplus \mathrm{IF}(E_7, E_6, E_5) \boxplus K_8 \boxplus W_8.$$

Let

$$C = E_8 \boxminus (A_4 \boxplus E_4 \boxplus \Sigma_1(E_7) \boxplus \mathrm{IF}(E_7, E_6, E_5) \boxplus K_8).$$

We will then have

$$E_4 \boxplus W_8 = C,$$

Table 9. W_i conditions for SHA-512

	conditions
W_5	$W_5[55] \neq W_5[62]$, $W_5[42] = W_5[49]$, $W_5[33] = W_5[39]$, $W_5[39] = W_5[38]$, $W_5[38] \neq W_5[37]$, $W_5[37] = W_5[36]$, $W_5[36] = W_5[35]$, $W_5[7] = W_5[14]$, $W_5[5] = W_5[11]$, $W_5[11] \neq W_5[10]$.
W_6	$W_6[33] \neq W_6[39]$, $W_6[29] \neq W_6[36]$, $W_6[27] \neq W_6[33]$, $W_6[27] = W_6[26]$. $W_6[22] = W_6[21]$, $W_6[13] = W_6[19]$, $W_6[17] \neq W_6[16]$, $W_6[9] = W_6[16]$. $W_6[7] = W_6[13]$, $W_6[13] = W_6[12]$, $W_6[12] = W_6[11]$.
W_7	$W_7[57] \neq W_7[63], W_7[37] = W_7[36], W_7[23] \neq W_7[29], W_7[11] \neq W_7[18]$, $W_7[56] = W_7[63], W_7[29] \neq W_7[36], W_7[29] = W_7[28], W_7[10] \neq W_7[16]$, $W_7[55] = W_7[61], W_7[28] \neq W_7[34], W_7[21] \neq W_7[28], W_7[16] \neq W_7[15]$, $W_7[55] \neq W_7[54], W_7[34] \neq W_7[33], W_7[18] = W_7[24], W_7[3] \neq W_7[10]$, $W_7[42] \neq W_7[49], W_7[33] = W_7[32], W_7[24] = W_7[23], W_7[2] = W_7[8]$, $W_7[41] = W_7[47], W_7[32] = W_7[31], W_7[16] \neq W_7[23], W_7[42] \neq W_7[41]$, $W_7[24] \neq W_7[31], W_7[15] = W_7[21]$.
W_8	$W_8[51] = W_8[58], W_8[49] \neq W_8[55], W_8[38] \neq W_8[45], W_8[36] = W_8[42]$, $W_8[31] = W_8[38], W_8[23] = W_8[29], W_8[18] \neq W_8[25], W_8[11] = W_8[18]$, $W_8[10] \neq W_8[16]$.
W_9	$W_9[57] \neq W_9[63], W_9[58] \neq W_9[57], W_9[45] = W_9[52], W_9[44] \neq W_9[50]$, $W_9[44] \neq W_9[45], W_9[32] = W_9[39], W_9[31] \neq W_9[37], W_9[32] = W_9[31]$, $W_9[19] = W_9[26], W_9[18] \neq W_9[24]$.
W_{16}	$W_{16}[12] \neq W_{16}[63], W_{16}[11] \neq W_{16}[62], W_{16}[10] \neq W_{16}[61], W_{16}[9] \neq W_{16}[60]$, $W_{16}[50] \neq W_{16}[59], W_{16}[49] = W_{16}[58], W_{16}[48] = W_{16}[57], W_{16}[63] \neq W_{16}[50]$, $W_{16}[62] \neq W_{16}[49], W_{16}[61] \neq W_{16}[48], W_{16}[60] \neq W_{16}[47], W_{16}[59] = W_{16}[37]$, $W_{16}[58] \neq W_{16}[36], W_{16}[57] \neq W_{16}[35], W_{16}[50] \neq W_{16}[37], W_{16}[49] \neq W_{16}[36]$, $W_{16}[48] \neq W_{16}[35], W_{16}[47] \neq W_{16}[34], W_{16}[24] = W_{16}[33], W_{16}[23] \neq W_{16}[32]$, $W_{16}[22] \neq W_{16}[31], W_{16}[24] = W_{16}[11], W_{16}[23] \neq W_{16}[10], W_{16}[22] \neq W_{16}[9]$, $W_{16}[21] = W_{16}[63], W_{16}[20] \neq W_{16}[62], W_{16}[19] \neq W_{16}[61]$.
W_{18}	$W_{18}[9] = W_{18}[60], W_{18}[60] \neq W_{18}[47], W_{18}[47] \neq W_{18}[34], W_{18}[3] = W_{18}[12]$, $W_{16}[22] = W_{16}[9]$.

where C is a known constant. Therefore, we can enumerate valid (E_4, W_8) with the depth-first search from the least significant bit to the most significant bit. The recursive procedure can be simply described as follows. Once a valid $(E_4[i], W_8[i])$ is found, move to the search for valid $(E_4[i+1], W_8[i+1])$, where $0 \leq i \leq 63$. As there are conditions on $(E_4[i], W_8[i])$, if no valid $(E_4[i], W_8[i])$ exists,

Table 10. The $A_7 - A_{12}$ conditions for SHA-512

	conditions
A_7	$A_7[27] = A_7[38], A_7[11] = A_7[22], A_7[57] \neq A_7[63], A_7[11] \neq A_7[0],$ $A_7[26] \neq A_7[37], A_7[45] \neq A_7[56], A_7[10] = A_7[15], A_7[14] = A_7[8],$ $A_7[25] \neq A_7[31], A_7[13] = A_7[24], A_7[39] = A_7[44], A_7[44] = A_7[50],$ $A_7[24] = A_7[35], A_7[43] \neq A_7[49], A_7[60] \neq A_7[1], A_7[56] = A_7[3],$ $A_7[21] = A_7[27], A_7[41] \neq A_7[47], A_7[59] \neq A_7[0], A_7[1] \neq A_7[7],$ $A_7[47] = A_7[58], A_7[58] = A_7[63], A_7[37] \neq A_7[43], A_7[7] \neq A_7[13],$ $A_7[49] \neq A_7[60], A_7[42] \neq A_7[47], A_7[35] \neq A_7[41], A_7[61] \neq A_7[2],$ $A_7[38] \neq A_7[43], A_7[34] = A_7[39], A_7[24] \neq A_7[29], A_7[59] \neq A_7[1],$ $A_5[i] = A_7[i], i \in \{8, 34, 35, 40, 56, 60\},$ $A_6[i] \neq A_7[i], i \in \{0, 26, 57, 58, 59, 61\},$ $A_5[i] = A_7[i], i \in \{1, 2, 3, 20\}$ $A_5[i] \neq A_7[i], i \in \{10, 11, 14, 29\},$
A_8	$A_8[53] = A_8[59], A_8[58] = A_8[5], A_8[6] \neq A_8[11],$ $A_6[i] = A_8[i], i \in \{6, 17, 18, 32, 33, 46, 48, 51, 52, 53, 62\}$ $A_6[i] \neq A_8[i], i \in \{19, 30, 36, 54, \}$ $A_7[i] = A_8[i], i \in \{1, 2, 3, 5, 10, 11, 14, 29, 39\}$ $A_7[i] \neq A_8[i], i \in \{13, 20, 47\}$
A_9	$A_9[45] \neq A_9[50], A_9[33] = A_9[44], A_9[28] \neq A_9[34],$ $A_7[i] \neq A_9[i], i \in \{0\}$ $A_8[i] = A_9[i], i \in \{1, 2, 6, 12, 17, 18, 19, 25, 30, 32, 33, 36, 38, 46, 48, 52, 53, 54, 62\}$ $A_8[i] \neq A_9[i], i \in \{4, 51, 55, 28\}$
A_{10}	$A_{10}[27] \neq A_{10}[33], A_{10}[45] = A_{10}[56], A_{10}[31] \neq A_{10}[36], A_{10}[59] \neq A_{10}[6],$ $A_{10}[40] \neq A_{10}[46], A_{10}[19] = A_{10}[30], A_{10}[45] = A_{10}[50], A_{10}[58] = A_{10}[5],$ $A_{10}[14] \neq A_{10}[21], A_{10}[55] \neq A_{10}[61], A_{10}[44] = A_{10}[49], A_{10}[12] = A_{10}[7],$ $A_{10}[54] = A_{10}[60], A_{10}[33] = A_{10}[44], A_{10}[53] = A_{10}[59], A_{10}[13] \neq A_{10}[8],$ $A_{10}[32] \neq A_{10}[43], A_{10}[28] = A_{10}[34], A_{10}[57] = A_{10}[62], A_{10}[6] = A_{10}[11],$ $A_{10}[60] \neq A_{10}[7].$
A_{11}	$A_9[i] \neq A_{11}[i], i \in \{25, 38, 51\}$ $A_9[i] = A_{11}[i], i \in \{0, 1, 2\}$
A_{12}	$A_{11}[i] \neq A_{12}[i], i \in \{39\}$ $A_{11}[i] = A_{12}[i], i \in \{0, 1, 2, 25, 38, 39, 51\}$

Table 11. The $E_5 - E_{14}$ conditions for SHA-512

	conditions
E_3	$E_3[47] = E_4[47], E_3[41] = E_4[41], E_3[40] = E_4[40], E_3[34] = E_4[34],$ $E_3[28] = E_4[28], E_3[5] = E_4[5].$
E_4	$E_4[48] = E_5[48].$
E_9	$E_9[44] = E_9[3], E_9[31] \neq E_9[54].$
E_{11}	$E_{11}[57] = E_{11}[16], E_{11}[56] \neq E_{11}[15], E_{11}[48] = E_{11}[11], E_{11}[50] = E_{11}[9],$ $E_{11}[45] = E_{11}[8], E_{11}[44] \neq E_{11}[48], E_{11}[30] \neq E_{11}[57], E_{11}[29] = E_{11}[56],$ $E_{11}[32] = E_{11}[55], E_{11}[31] = E_{11}[54], E_{11}[21] \neq E_{11}[48], E_{11}[16] \neq E_{11}[43].$
E_{12}	$E_{12}[11] = E_{12}[15], E_{12}[55] = E_{12}[14], E_{12}[21] \neq E_{12}[48].$
E_{14}	$E_{14}[11] = E_{14}[15], E_{14}[4] = E_{14}[27], E_{14}[62] = E_{14}[2], E_{14}[60] = E_{14}[23],$ $E_{14}[55] = E_{14}[14], E_{14}[47] = E_{14}[10], E_{14}[42] = E_{14}[1], E_{14}[37] = E_{14}[41],$ $E_{14}[34] = E_{14}[61], E_{14}[29] = E_{14}[52], E_{14}[24] = E_{14}[28], E_{14}[21] = E_{14}[48].$

start backtracking. In this way, the searching space can be significantly reduced compared with the naive method to directly exhaust all possible E_4 or W_8.

For each obtained valid (E_4, W_8), we then compute A_0 with the following equation:

$$A_4 = E_4 \boxminus A_0 \boxplus \Sigma_0(A_3) \boxplus \mathrm{MAJ}(A_3, A_2, A_1).$$

Then, we need to enumerate valid (E_3, W_7) satisfying their differential conditions and the following equation:

$$E_7 = A_3 \boxplus E_3 \boxplus \Sigma_1(E_6) \boxplus \mathrm{IF}(E_6, E_5, E_4) \boxplus K_7 \boxplus W_7.$$

This is indeed the same as enumerating (E_4, W_8), which can also be efficiently done with the depth-first search.

In our collision attack on 31-step SHA-512, we have used this method to efficiently enumerate all $(A_{-1}, A_0, E_3, E_4, W_7, W_8)$ for all the 4 starting points, and it takes less than 2 h on a single core. It is found that the size of $\mathtt{TAB}_2$ is $41826648064 \approx 2^{35.2}$ for our attack on SHA-512, which is much larger than the expected value $4 \times 2^{29} = 2^{31}$.

Consequently, the time complexity of our collision attacks on 31-step SHA-512 is dominated by the matching phase, and the best time complexity is

$$\frac{2^{64+65+0.9}}{2^{35.2}} = 2^{94.7}.$$

while the corresponding memory complexity of this attack is $2^{35.2}$. Compared with the previous attack published at EUROCRYPT 2024, the time and memory complexity have been improved by a factor of $2^{20.9}$ and $2^{42.1}$, respectively.

6 Conclusion

By exploiting the sparsity of the SHA-2 differential characteristics, we present novel memory-efficient collision attacks on 31-step SHA-256 and SHA-512, which can significantly improve the time and memory complexity over state-of-the-art. In particular, after more than a decade, we can eventually improve the practical collision attack on SHA-256 from 28 steps to 31 steps. We hope this is not the end of the (practical) collision attacks on step-reduced SHA-256, and we expect that more advanced attacks can be found in the near future with the open-source MILP/SAT/SMT-based automatic tools to search for signed differential characteristics.

Acknowledgement. We would like to thank the anonymous reviewers for their insightful comments. Yingxin Li and Gaoli Wang are supported by the National Key Research and Development Program of China (2022YFB2701900), the National Natural Science Foundation of China (Nos. 62472172, 62072181, 62202017), Shanghai Trusted Industry Internet Software Collaborative Innovation Center. Fukang Liu is supported by JSPS KAKENHI Grant Numbers JP22K21282, JP24K20733.

References

1. A. Bar-On, O. Dunkelman, N. Keller, E. Ronen, and A. Shamir. Improved key recovery attacks on reduced-round AES with practical data and memory complexities. In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 185–212. Springer, 2018.
2. C. D. Cannière and C. Rechberger. Finding SHA-1 characteristics: General results and applications. In X. Lai and K. Chen, editors, *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, volume 4284 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006.
3. I. Damgård. A design principle for hash functions. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.
4. C. Dobraunig, M. Eichlseder, and F. Mendel. Analysis of SHA-512/224 and SHA-512/256. In T. Iwata and J. H. Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 612–630. Springer, 2015.
5. F. Draft. Public comments on the draft federal information processing standard (fips) draft fips 180-2, secure hash standard (shs).

6. O. Dunkelman, N. Keller, E. Ronen, and A. Shamir. The retracing boomerang attack. In A. Canteaut and Y. Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 280–309. Springer, 2020.
7. M. Eichlseder, F. Mendel, and M. Schläffer. Branching heuristics in differential collision search with applications to SHA-512. In C. Cid and C. Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 473–488. Springer, 2014.
8. L. Grassi, C. Rechberger, and S. Rønjom. A new structural-differential property of 5-round AES. In J. Coron and J. B. Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 289–317, 2017.
9. S. Indesteege, F. Mendel, B. Preneel, and C. Rechberger. Collisions and other non-random properties for step-reduced SHA-256. In R. M. Avanzi, L. Keliher, and F. Sica, editors, *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, volume 5381 of *Lecture Notes in Computer Science*, pages 276–293. Springer, 2008.
10. D. Khovratovich, C. Rechberger, and A. Savelieva. Bicliques for preimages: Attacks on Skein-512 and the SHA-2 family. In A. Canteaut, editor, *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*, pages 244–263. Springer, 2012.
11. Y. Li, F. Liu, and G. Wang. New records in collision attacks on SHA-2. In M. Joye and G. Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 158–186. Springer, 2024.
12. F. Liu, W. Meier, S. Sarkar, G. Wang, R. Ito, and T. Isobe. New cryptanalysis of ZUC-256 initialization using modular differences. *IACR Trans. Symmetric Cryptol.*, 2022(3):152–190, 2022.
13. F. Liu, G. Wang, S. Sarkar, R. Anand, W. Meier, Y. Li, and T. Isobe. Analysis of RIPEMD-160: new collision attacks and finding characteristics with MILP. In C. Hazay and M. Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 189–219. Springer, 2023.
14. F. Mendel, T. Nad, and M. Schläffer. Finding SHA-2 characteristics: Searching through a minefield of contradictions. In D. H. Lee and X. Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 288–307. Springer, 2011.

15. F. Mendel, T. Nad, and M. Schläffer. Improving local collisions: New attacks on reduced SHA-256. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 262–278. Springer, 2013.
16. R. C. Merkle. One way hash functions and DES. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.
17. X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2005.
18. X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full SHA-1. In V. Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.
19. X. Wang and H. Yu. How to break MD5 and other hash functions. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.
20. X. Wang, H. Yu, and Y. L. Yin. Efficient collision search attacks on SHA-0. In V. Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.

Key Collisions on AES and Its Applications

Kodai Taiyama[1], Kosei Sakamoto[2], Ryoma Ito[3], Kazuma Taka[1], and Takanori Isobe[1,3](✉)

[1] University of Hyogo, Kobe, Japan
ad23x032@guh.u-hyogo.ac.jp, takanori.isobe@ai.u-hyogo.ac.jp
[2] Mitsubishi Electric Corporation, Kamakura, Japan
[3] NICT, Koganei, Japan
itorym@nict.go.jp

Abstract. In this paper, we explore a new type of key collisions called target-plaintext key collisions of AES, which emerge as an open problem in the key committing security and are directly converted into single-block collision attacks on Davies-Meyer (DM) hashing mode. For this key collision, a ciphertext collision is uniquely observed when a specific plaintext is encrypted under two distinct keys. We introduce an efficient automatic search tool that leverages bit-wise behaviors of differential characteristics and dependencies among operations and internal variables. As a result, we demonstrate single-block collision attacks on 2/5/6-round AES-128/192/256-DM and semi-free-start collision attacks on 5/7/9-round AES-128/192/256-DM, respectively. Furthermore, by exploiting a specific class of free-start collisions with our tool, we present two-block collision attacks on 3/9-round AES-128/256-DM, respectively.

Keywords: AES · Davies-Meyer mode · collision · rebound attacks

1 Introduction

1.1 Background

In block ciphers, a key collision is defined as two distinct keys that produce identical subkeys through the key scheduling function. When such colliding keys are used, any plaintext can be encrypted into the same ciphertext. Such key collisions are known for several ciphers [2,6,18,21,24]. Recently, the importance of a new variant of key collisions has been demonstrated in the domain of the key committing security. Albertini et al. [1] revealed that standard AE (Authenticated Encryption) schemes such as AES-GCM and ChaCha20-Poly1305 lack this type of security and introduce a simple countermeasure, referred to as the padding fix. This method involves prepending an ℓ-bit string of 0's, denoted as X, to the message M for each encryption, resulting in $Enc(K, N, A, X||M)$, and check for the presence of X at the start of the message after decryption; decryption fails if X is not present. This countermeasure leads to the following open problem [1].

> "In particular, the padding fix with AES-GCM assumes an ideal cipher, and therefore raises the following interesting problem: Is it possible to find two keys k_1 and k_2 such that $AES_{k_1}(0) = AES_{k_2}(0)$ in less than 2^{64} trials. If the key size is larger than the block size, then such a pair of keys must exist. While there has been some work on the chosen-key setting [13] or using AES in a hashing mode [25], we are not aware of any results on this specific problem."

Compared to existing key collisions [2,6,18,21,24], a ciphertext collision is uniquely observed when a specific plaintext is encrypted under two distinct keys. In this paper, we refer to this as a target-plaintext key collision. Such collisions can be further categorized into two variants based on whether the target plaintext is predetermined: fixed-target-plaintext key collision and free-target-plaintext key collision. These key collisions can be directly converted into single-block collisions or semi-free-start collisions on the Davies-Meyer (DM) hashing mode with AES.

Despite its significance, to the best of our knowledge, this type of attack has not yet been investigated for AES over the past 20 years. In fact, there are no results on collision and semi-free-start collision attacks on DM mode with AES over the significant number of rounds. On the other hand, there are numerous results of free-start collisions on DM hashing with AES [4,17,22], as well as collisions on Matyas-Meyer-Oseas (MMO) and Miyaguchi-Preneel (MP) modes [10–12].

1.2 Difficulties for Finding Target-Plaintext Key Collisions

Here, we discuss the technical obstacles for finding target-plaintext key collisions and (semi-free) collision attacks on DM mode with AES by existing approaches.

- First of all, as far as we know, no differential characteristics of AES where a difference is inserted into only key, not into plaintext, leading to a ciphertext collision, have been demonstrated in the literature. Compared to the search for best related-key differential characteristics for AES [5,9,13,14], this search requires strict conditions such that key differences should be canceled out by themselves without the help of plaintext differences. Consequently, the weights of target differential characteristics become quite high, resulting in time-consuming tasks.
- Even if differential characteristics for key collisions are identified, efficiently mounting rebound attacks [23] in DM remains non-trivial. This difficulty arises because the core techniques of rebound attacks, e.g. super S-box [15,16,20], non-full-active super S-box [11,12,26], and super inbound [10] techniques, face limitations in controlling plaintext values. These techniques rely on exploiting the degrees of freedom (DoF) in the plaintext of an underlying block cipher, which is a message domain that the adversary can manipulate in MMO and MP, while in DM, this is a chaining value domain.

1.3 Our Contribution

In this paper, we introduce an efficient automatic search tool designed to find target-plaintext key collisions of AES, and then apply it to AES-128/192/256 and its DM hashing mode.

New Tool for Finding Key Collision on AES. To address the limitations of existing approaches, we first utilize *bit-wise* differential characteristics, which lead to a ciphertext collision resulting solely from key differences, by SAT method [29] with state-of-the-art SAT solver and encoding methods. In contrast, existing collision attacks on AES utilize truncated differential characteristics [4,10,11,11,12,17,22]. Bit-wise differential characteristics enable us to exploit the bit-level relationship between the DoF of the inbound phase and the differential probability of the outbound phase in rebound attacks.

Specifically, we convert bit-wise differential characteristics into a graphical expression to leverage the DoF information of each vertex, dependencies between each vertex, and grouping based on these relationships. We then use a *depth-first search* in graph theory to generate a DoF tree, illustrating the optimal strategy for selecting inbound vertices, which are vertices categorized in the inbound phase, and determining the sequence for identifying outbound vertices, which are vertices categorized in outbound phase, from a given inbound vertex.

This DoF tree allows us to mount new rebound-type attacks, which *hierarchically* perform inbound and outbound phases across several groups categorized by vertex dependencies. As a result, our tool can efficiently mount rebound attacks for very complex differential characteristics, including those of the key scheduling function under the strict conditions of key collisions.

Application Results. We present fixed-target-plaintext key collisions on 2/5/6-round AES-128/192/256 and free-target-plaintext key collisions on 5/7/9-round AES-128/192/256. These are directly converted into single-block collision attacks on 2/5/6-round AES-128/192/256-DM and semi-free-start collision attacks on 5/7/9-round AES-128/192/256-DM, respectively. Table 1 summarizes our application results. To show the validity of our attacks, we provide an example of the fixed-target-plaintext key collision on AES-256 or semi-free-start collisions on 9-round AES-256-DM. Furthermore, by exploiting a specific class of free-start collisions, we present two-block collision attacks on 3/9-round AES-128/256-DM, respectively. As far as we know, these are the first results of collisions and semi-free start collisions of AES-DM over a significant number of rounds.

2 Preliminaries

2.1 Description of AES

AES [8] is a block cipher that supports a block size of 128 bits. It has three variants called AES-128, AES-192, and AES-256, depending on the combination of a key size of K_{len} bits and the number of rounds N_r. More specifically,

Table 1. Summary of our application results.

Target	Attack	Round	Time	Memory	Ref.
AES-128-DM	Collision	2	2^{49}	Negligible	Sect. 5.3
	Collision*	3	2^{60}	2^{52}	Sect. 6.4
	Semi-free-start	5	2^{57}	Negligible	Sect. 5.3
	Free-start	5	2^{56}	2^{32}	[22]
	Free-start	6	2^{32}	2^{16}	[17]
AES-192-DM	Collision	5	2^{61}	Negligible	Sect. 5.3
	Semi-free-start	7	2^{62}	Negligible	Sect. 5.3
AES-256-DM	Collision	6	2^{61}	Negligible	Sect. 5.1
	Collision*	9	2^{58}	2^{55}	Sect. 6.3
	Semi-free-start	9	2^{30}	Negligible	Sect. 5.2
	q pseudo-collision†	14 (full)	$q \cdot 2^{67}$	Negligible	[4]

* It is a two-block collision. † The q pseudo-collision attack is synonymous with the q free-start collision attack. Kim et al. [19] claimed that this attack [4] is insufficient for a free-start collision attack on AES-256-DM, as it is inferior to the birthday attack in terms of the generic notion of collision attacks.

$(K_{len}, N_r) = (128, 10)$, $(192, 12)$, and $(256, 14)$ for AES-128, AES-192, and AES-256, respectively. The internal state can be viewed as a 4×4 array of bytes.

Round Function. The round function consists of the following four operations:

- SubBytes (SB) is a parallel execution of 8-bit S-boxes. Due to page limitation, we do not give the specific table of the S-box.
- ShiftRows (SR) is a row-wise shuffle operation. Specifically, the i-th row of the state is cyclically shifted by i-bytes to the left.
- MixColumns (MC) is a column-wise 4×4 matrix multiplication over the finite field $\mathbb{F}_2^8$ with the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$.
- AddRoundKey (AK) is the application of the round key. The 128-bit round key is XORed to the internal state.

The round function of AES is denoted by $f = \mathsf{AK} \circ \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB}$; the i-th round internal state before the SB, SR, MC, and AK operations are denoted by x_i, y_i, z_i, and w_i, respectively; and the i-th round key is denoted by k_i, as depicted in Fig. 1. After an initial state w_0 is initialized with the initial round key k_0 by the AK operation, the internal state is updated by iterating the round function N_r times. Note that the MC operation is omitted for the final round.

Key Schedules. The key schedule algorithm takes the master key K and performs the key expansion function to generate the round keys k_i for $0 \leq i \leq N_r$. The

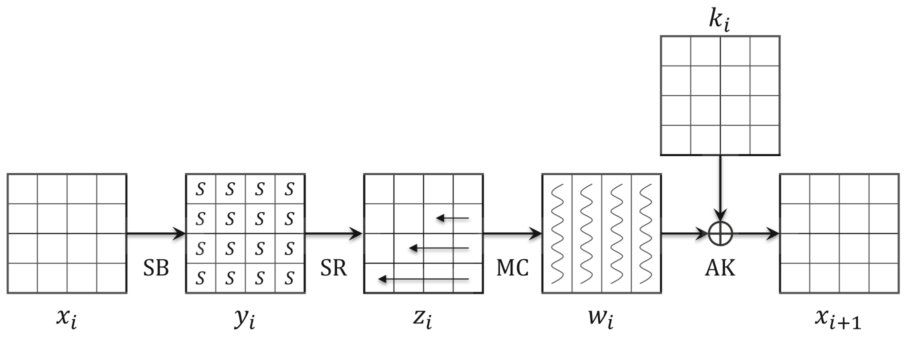


Fig. 1. The round function of AES.

resulting round keys consist of a linear array of 4-byte words, denoted by v_j with the range of $0 \leq j < 4 \cdot (N_r + 1)$; namely, $k_i[j \bmod 4] = v_{4i+j}$.

The key expansion function consists of two steps. The first step is to initialize v_j for $0 \leq j < N_k$ with K, where $N_k = 4$, 6, and 8 for AES-128, AES-192, and AES-256, respectively. The second step is to compute the remaining round keys, i.e., v_i for $N_k \leq j < 4 \cdot (N_r + 1)$, as the following procedure (see Fig. 2):

$$v_j = \begin{cases} v_{j-N_k} \oplus \mathsf{SW}(\mathsf{RW}(v_{j-1})) \oplus \mathsf{RC}(i/N_k) & \text{if } j \equiv 0 \bmod N_k, \\ v_{j-N_k} \oplus \mathsf{SW}(v_{j-1}) & \text{if } N_k = 8 \text{ and } j \equiv 4 \bmod N_k, \\ v_{j-N_k} \oplus v_{i-1} & \text{otherwise,} \end{cases}$$

where SW is the SubWord function that takes a 4-byte input word and applies the SB operation to each of four bytes to produce a 4-byte output word, RW is the RotWord function that takes a 4-byte input word $[u_0, u_1, u_2, u_3]$, performs the cyclic shift operation, and returns a 4-byte output word $[u_1, u_2, u_3, u_0]$, and $\mathsf{RC}(i/N_k)$ is the round constant.

2.2 Rebound Attack

The rebound attack, proposed by Mendel et al. at FSE 2009 [23], is a generic tool for cryptanalysis of hash functions, especially on AES-like hashing. The basic idea of the attack is to obtain a specific differential characteristic in the underlying primitive (e.g., a block cipher or a cryptographic permutation) of the target hash function. More specifically, the rebound attack consists of an inbound phase and an outbound phase by decomposing the target primitive E into three parts so that $E = E_{fw} \circ E_{in} \circ E_{bw}$, as depicted in Fig. 3.

– **The inbound phase** aims to find a differential characteristic with a lower probability in E_{in} of the target primitive. To achieve this, an attacker must carefully control an input/output differential pair of the non-linear layers in E_{in} and determine an input/output differential pair in E_{in} that maximizes

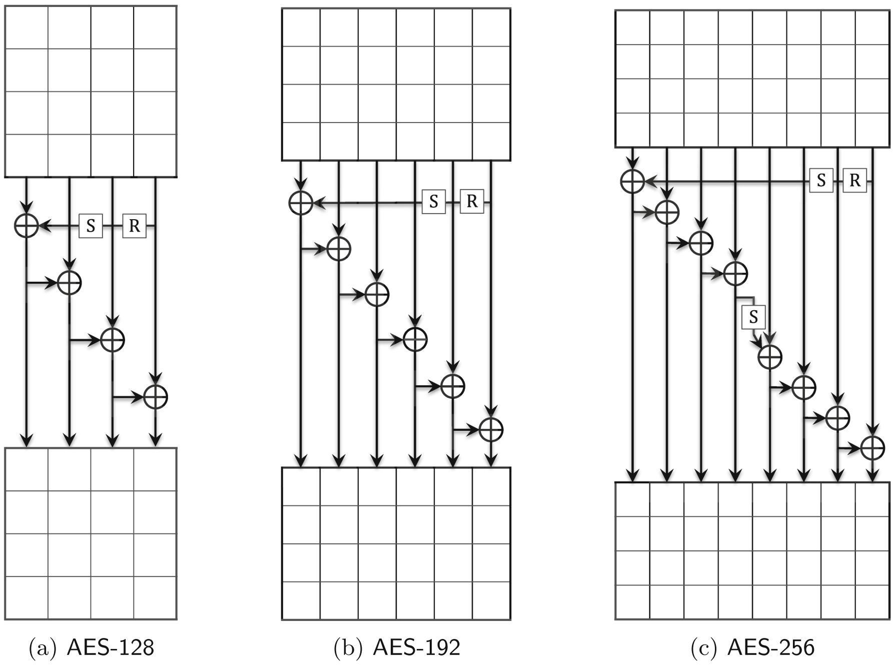


Fig. 2. Key schedules of AES-128, AES-192, and AES-256. The SubWord and RotWord functions are denoted by S and R, respectively. Note that the round constant operation is omitted.

the differential probability in the following outbound phase. Searching for such a pair enables us to obtain many available solutions, which are the starting points for the outbound phase. It becomes the degree of freedom in the inbound phase.

- **The outbound phase** aims to obtain a valid differential characteristic in both forward and backward direction through E_{fw} and E_{bw} to find a desired collision. If the obtained differential characteristic has a sufficient probability of violating the security of collision attacks, the rebound attack can be considered successful. Otherwise, the attacker repeats the inbound phase to obtain more starting points for the outbound phase.

2.3 Collision Attacks and Its Variant

Given a hash function H, a collision is to identify message pair (m, m') satisfies $H(IV, m) = H(IV, m')$, where the initial vector IV is a fixed initial value. Let v be the chaining value that is equal to the output of the previous block. A semi-free-start collision is to find a pair (v, m) and (v, m'), such that $H(v, m) = H(v, m')$, where $(v \neq IV)$. A free-start collision is to find a pair (v, m_{i-1}) and (v', m'_{i-1}), so that $H(v', m) = H(v', m')$, where $(v \neq v')$. When the hash function

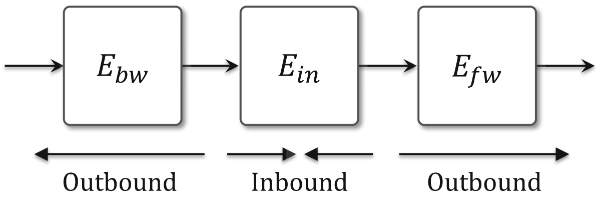


Fig. 3. A schematic view of the rebound attack.

H is built by iterating the compression function (CF) with the Merkle-Damgård construction, we can similarly define the semi-free-start and free-start collision attacks on the compression function.

3 Key Collision

In block ciphers, a key collision is defined as two distinct keys that produce identical subkeys through the key scheduling function. When such colliding keys are used, any plaintext can be encrypted into the same ciphertext. The existence of such keys is known for a few ciphers. For instance, Robshaw [24] has shown that the CRYPTREC candidate, the block cipher CIPHERUNICORN-A, has colliding keys. Kelsey et al. [18] have found colliding keys for the Tiny Encryption Algorithm (TEA) block cipher. Furthermore, Aumasson et al. [2] have discovered that the ISDB Scrambling Algorithm, the cipher MULTI2, allows such keys as well. Biryukov and Nikolic [6] have found colliding keys of SC2000-256 by exploiting the weakness of the key scheduling function. For stream ciphers, Matsui [21] has investigated the behavior of colliding key pairs for the stream cipher RC4, in which, two distinct keys generate the same key stream.

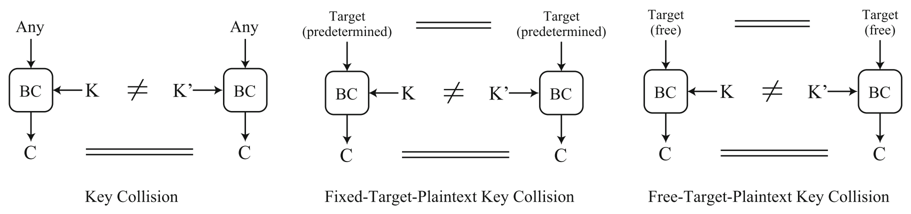
3.1 New Variants of Key Collision

In this paper, we introduce new variants of key collisions, termed *target-plaintext key collision*, defined as follows.

Definition 1 (Target-Plaintext Key Collision). *It is two distinct keys that generate the same ciphertext for a single target plaintext.*

Compared to existing key collisions, particularly the subkey collision in the key scheduling function, a ciphertext collision occurs exclusively with a specific plaintext under two distinct keys. Identifying such a collision can be classified into two different problems depending on whether a single target plaintext is predetermined or not, illustrated in Fig. 4.

Problem 1 (Fixed-Target-Plaintext Key Collision). *Given a single target plaintext, find a key pair that generates the same ciphertext.*



Any
BC
K
≠
K'
BC
C
C
Key Collision
Target (predetermined)
Fixed-Target-Plaintext Key Collision
Target (free)
Free-Target-Plaintext Key Collision

Fig. 4. Variants of key collisions.

Problem 2 (Free-Target-Plaintext Key Collision). *Find a key pair and a corresponding single plaintext that generates the same ciphertext.*

In Problem 1, given a predetermined target plaintext, the adversary must identify two distinct keys that yield the same ciphertext. In contrast, Problem 2 allows the adversary to choose the target plaintext and find colliding key pairs freely. Clearly, Problem 1 is more challenging than Problem 2.

The time complexity for solving these problems by generic attack (assuming an underlying block cipher is an ideal cipher) depends on the size of the ciphertext. Specifically, for an n-bit ciphertext, such pairs can be found within a time complexity of $2^{n/2}$, owing to the birthday paradox.

3.2 Applications of Target-Plaintext Key Collisions

We explore the implications and significance of target-plaintext key collisions in theoretical and practical domains with several applications.

Collision Attack on DM Hashing Mode. The DM hashing mode is used to construct a cryptographic hash function from a block cipher. In this mode, as shown in Fig. 5, the key in the block cipher is treated as the message input for the hash function, and the plaintext is the initial vector or chaining value.

A fixed-target-plaintext key collision, where two keys produce the same ciphertext for a predetermined plaintext, corresponds to a single-block collision in DM mode. This implies that two different messages (keys) result in the same hash output (ciphertext) for a given initial state (plaintext). Similarly, a free-target-plaintext key collision, where the adversary can choose the plaintext and find two keys that produce the same ciphertext, correlates with a semi-free-start collision in DM mode. In this scenario, the attacker has the freedom to select the initial state (plaintext) and find two different messages (keys) that lead to the same hash output (ciphertext). This flexibility makes the semi-free-start collision less constrained and potentially attacks on more rounds than a single-block collision.

Open Problem of Padding Fix for Key Committing Security. Recently, the key committing security has garnered significant attention. In this line of research, Albertini et al. [1] revealed that standard AE schemes such as AES-GCM and ChaCha20-Poly1305 lack this type of security. In their paper, the authors

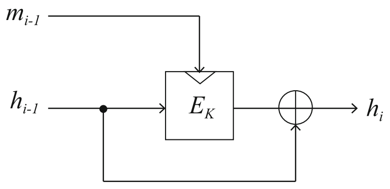


Fig. 5. A schematic view of the Davies-Meyer (DM) hashing mode.

introduce a simple countermeasure, referred to as the padding fix. This method involves prepending an ℓ-bit string of 0's, denoted as X, to the message M for each encryption, resulting in $Enc(K, N, A, X||M)$, and check for the presence of X at the start of the message after decryption; decryption fails if X is not present. This countermeasure leads to the following open problem [1].

> "In particular, the padding fix with AES-GCM assumes an ideal cipher, and therefore raises the following interesting problem: Is it possible to find two keys k_1 and k_2 such that $AES_{k_1}(0) = AES_{k_2}(0)$ in less than 2^{64} trials. If the key size is larger than the block size, then such a pair of keys must exist. While there has been some work on the chosen-key setting [13] or using AES in a hashing mode [25], we are not aware of any results on this specific problem."

This issue is equivalent to the task of identifying target-plaintext key collisions in AES. To the best of our knowledge, this type of attack has not yet been investigated for AES over the past 20 years.

Embedded Device Keys by Malicious Factory. Regarding other applications of target-plaintext key collisions, consider the scenario of a malicious factory setting in which a device-specific key is embedded at an untrustworthy factory. This situation is akin to the context of a hardware trojan [3], with Original Equipment Manufacturing (OEM) serving as an illustrative example.

In such cases, there exists a significant risk when the fixed key is utilized for generating session keys via key derivation functions based on block ciphers for each device [7]. If the factory puts in specific key pairs meant for key collisions in two distinct devices, it may result in identical session keys for certain inputs. Consequently, an adversary possessing one device might be able to decrypt messages intended for another.

Moreover, the adversary could feasibly impersonate another device in systems where devices authenticate their identities by responding to a challenge computed using a block cipher with a device-fixed key. This is because their responses, which are generated using colliding key pairs, would be identical.

4 Automatic Tools for Key Collision on AES

In this section, we propose a new automatic tool designed to efficiently identify key collisions. This tool is comprised of six steps, and we describe them step

Algorithm 1: The Proposed Rebound-type Attack

```
   Data: $b, rounds, LT$
   Result: $T_{comp}$
1  $cnf \leftarrow$ GenCNF($b, rounds$);
2  $path \leftarrow$ SAT($cnf$) ;                                // Step1
3  $G \leftarrow$ GenGraph($path$) ;                              // Step2
4  $starts \leftarrow$ GetStarts($rounds$) ;                      // Step3
5  forall $S$ in $starts$ do
6      $DoF_{tree} \leftarrow$ GenTree($S, G$) ;            // Step4 and Step5
7      $T_{comp} \leftarrow$ CalcTime($DoF_{tree}$) ;             // Step6
```

by step. The overview of our attack procedure is shown in Algorithm 1. Our rebound-type attack consists of the inbound and outbound phases as well as the standard rebound attack, but they have complex structures. Therefore, we call the internal states classified to the inbound phase (resp. outbound phase) as the inbound vertex (resp. outbound vertex).

Step 1: Searching for Differential Characteristics for Key Collisions. We employ the SAT-based automatic search method proposed by Sun et al. [29] to find good differential characteristics applied to our rebound-type attack. We only give an overview of this method and our environment for evaluation. For more information about the modeling method, please refer to [29].

In the SAT-based approach, we express a differential propagation in a primitive and its weight caused by non-linear operations into CNF (Conjunctive Normal Form). For linear operations, such as an XOR and a permutation, their CNFs are shown in the previous works [27–29]. In contrast, for a non-linear operation, such as an S-box, we can derive their CNF expression by some algorithms used to simplify the Boolean function. Then, we employ Espresso logic minimizer[1] to derive CNF of an S-box. In addition to linear and non-linear operations, it is necessary to model the output such that the output difference is zero in order to discover a differential characteristic that leads to a collision. After converting differential propagation and its weight into CNF expression, we set the target weight k by Boolean cardinality constraints expressed in CNF and give the created CNF to a SAT solver. If an SAT solver returns "SAT", we can find differential characteristics with a weight of $\leq k$. Otherwise, we increase k and repeat this procedure until a SAT solver returns "SAT". In this work, we utilize ParKissat-RS[2] and totalizer as a SAT solver and Boolean cardinality constraints, respectively. GenCNF() in Algorithm 1 accepts the number of rounds, *rounds*, that we attempt to find the key collision as the input and outputs a CNF, *cnf*, to find a differential characteristic. Then, SAT() accepts the

[1] https://ptolemy.berkeley.edu/projects/embedded/pubs/downloads/espresso/index.htm.

[2] https://github.com/shaowei-cai-group/ParKissat-RS.

CNF generated by GenCNF() and outputs the found differential characteristic, *path*, with its probability. In the standard situation where we attempt to identify optimal differential characteristics, we solve SAT models with a target weight k in ascending order. In contrast, we do not need to find optimal differential characteristics but good differential characteristics for our rebound-type attack.

In our attack, the probability of an underlying differential characteristic is constrained by the maximum DoF in each attack for successful attacks. A fixed-target-plaintext key collision can utilize the DoF of the key, while the free-target-plaintext key collision can utilize the DoF of both the key and plaintext. Thus, fixed-target-plaintext key collision requires a differential characteristic with probability of more than $2^{-128}/2^{-192}/2^{-256}$ on AES-128/192/256 while it is $2^{-256}/2^{-320}/2^{-384}$ on AES-128/192/256 for the free-target-plaintext collision, respectively.

Step 2: Converting Differential Characteristics into Graphical Expression. Since our rebound-type attack consists of the complex structure of the inbound and outbound vertices, we need to efficiently calculate the available DoF for fulfilling the differential characteristics found in Step 1. To this end, we first convert the differential characteristics into the graphical expression, including a set of vertices and edges, and utilize *depth-first search* to calculate the available DoF. GenGraph() in Algorithm 1 accepts the differential characteristic found in Step 1 as the input and generates the corresponding graph.

Let $G = (V, E)$ be the graph of the differential characteristic where V and E denote a set of vertices and edges, respectively. V and E are determined as following rules:

Degree of the Graph. We first determine how the degree of bit-wise we convert differential characteristics into a set of vertices and edges. It significantly influences the efficiency of the later steps in our tool; the smaller the degree will be, the more complex constructing the attack will be. In this paper, we generate the graph in 32-bit wise for AES. Therefore, we treat a 32-bit word as a single unit on AES.

A set of vertices *V*. V is a set of vertices v, representing a unit of internal states that we can independently determine the values. Each vertex holds information about its weight and the available DoF.

A set of edges *E*. E is a set of edges e, representing vertices to which a particular vertex is connected. For example, if the vertex v_α is connected to the vertices v_β, and v_γ, we hold the edge $e_{v_\alpha}[v_\beta, v_\gamma]$ in E. We can interact with the available DoF in each connected vertex via the edge.

After creating the graph of the differential characteristic, we classify each SB and SW layer depending on in which v they belong. It can allow us to add information about the total probability in each vertex, helping us determine which vertices will be set to inbound or outbound vertices. It should be mentioned that the shape of the graph is determined by a specification of a primitive and the number of analysed rounds while available DoF in each vertex corresponds to a differential characteristic.

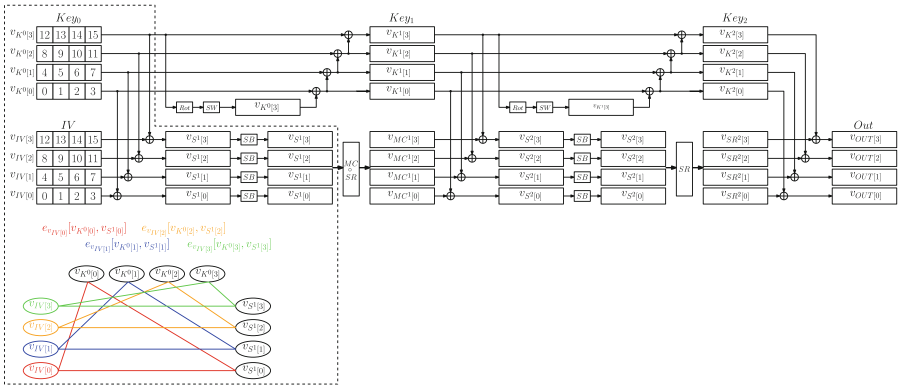


Fig. 6. Example of the graphical expression of round functions of AES.

Example. For a better understanding, we take Fig. 6 as an example, which shows the round function and key scheduling function of the 2-round AES. First, as shown in Fig. 6, we convert the internal state into a set of vertices in 32-bit wise as follows:

$$V = \{v_{IV[i]}, v_{s^1[i]}, v_{MC^1[i]}, v_{s^2[i]}, v_{SR^2[i]}, v_{OUT[i]}, v_{K^0[i]}, v_{K^1[i]}, v_{K^2[i]}\},$$

where $i \in \{0, 1, 2, 3\}$, each of which can independently determine their values.

From a set of vertices, we can obtain a set of edges corresponding to the vertices $v_{IV[i]}$ and $v_{K^0[i]}$ where $i \in \{0, 1, 2, 3\}$ as follows:

$$E = \{e_{v_{IV[i]}}[v_{K^0[i]}, v_{S^1[i]}], e_{v_{K^0[0]}}[v_{IV[0]}, v_{S^1[0]}, v_{K^0[3]}, v_{K^1[0]}], e_{v_{K^0[1]}}[v_{IV[1]}, v_{S^1[1]}, v_{K^1[0]}, v_{K^1[1]}],$$
$$e_{v_{K^0[2]}}[v_{IV[2]}, v_{S^1[2]}, v_{K^1[1]}, v_{K^1[2]}], e_{v_{K^0[3]}}[v_{IV[3]}, v_{S^1[3]}, v_{K^0[0]}, v_{K^1[2]}, v_{K^1[3]}]\},$$

where $i \in \{0, 1, 2, 3\}$, each of which connects vertices. We can obtain the remaining edges in the same manner. For simplicity, we depict edges in a part of the 2-round AES in Fig. 6.

Step 3: Determining the Starting Points. In the inbound phase of rebound attacks, we can determine the values following the differential characteristics by leveraging the DoF in the internal state. Specifically, by properly choosing values of internal state, we ensure that the probability of differential transition through an SB layer is one. This operation is executed for multiple vertices, depending on the available DoF. We call such a set of vertices as the starting points and vertices in the starting points as *inbound vertices*. The total number of inbound vertices is determined by the maximum available DoF in the target primitive and setting. For example, to find the fixed-target-plaintext key collision of the r-round AES-128, we can choose four vertices including SB or SW layers as the inbound vertices because the maximum available DoF is 2^{128}. In that case, since each round includes a total of 5 vertices, including SB and SW, the number of combinations for inbound vertices is $\binom{5r}{4}$, which will be too expensive to evaluate all combinations as r becomes large. Therefore, we restrict the number of rounds that include inbound vertices to the minimum in our evaluation.

We conduct the depth-first search from the inbound vertices to efficiently obtain the value pairs following the differential characteristics in the later steps. Hereafter, we call a vertex not belonging to the starting points as an *outbound vertex*. Once we determine the starting point, the *outbound vertices*, which consists of all the vertices not belonging to the starting point, is also determined simultaneously. GetStarts() in Algorithm 1 accepts the number of rounds that we attempt to find the key collisions as the input and outputs all combinations of starting points.

Step 4: Calculating the Degrees of Freedom in the Starting Points. After determining the starting point, we calculate DoF derived from the given individual inbound vertices. This calculation is equivalent to determining the number of valid values within the starting point that fulfills the corresponding differential characteristics.

Available Degrees of Freedom. Suppose that the total size of inbound vertices in the given starting point is b bits, and the probability of its differential propagation is 2^{-P_1}, we can obtain the DoF of 2^{b-P_1}. It should be emphasized that we can independently calculate the DoF derived from each inbound vertex. Therefore, the time complexity in a starting point will not be 2^{b-P_1} but can be much smaller than it. Inbound() in Algorithm 2 accepts an starting point as the input and outputs this starting point with its DoF if the starting point is valid. Note that any choice of starting points does not always lead to the valid collision attacks, meaning that some choices of starting point cannot bring the colliding pairs due to their structures. In that case, Inbound() outputs `NULL`.

Example. Suppose that we choose the vertex $v_{S^1[0]}$ in Fig. 6 as one of inbound vertices and its probability is 2^{-14}, the available DoF derived from $v_{S^1[0]}$ is estimated $2^{18}(= 2^{32-14})$. Note that we must derive the DoF from all those states if there are multiple states over the non-linear operation in an inbound vertex.

Step 5: Finding Value Pairs Fulfilling the Outbound Vertices. After determining the starting point, we proceed to derive the value pairs fulfilling the entire differential characteristics for key collisions. To derive such pairs efficiently, we categorize inbound and outbound vertices using the *depth-first search.* During this depth-first search, we classify them based on their ability to calculate the values independently using the available DoF. This categorization allows us to derive the value pairs independently in each category, thereby minimizing the time complexity. For a better understanding, we give a simple example of how to categorize the inbound and outbound vertices.

```
Algorithm 2: GenTree
    Data: $S, G$
    Result: $DoF_{tree}$
1   if InBound($S, G$) = NULL then // Step4
2       break;
3   else
4       $DoF_{tree}$ ← Grouping($S, G$) // Step5
```

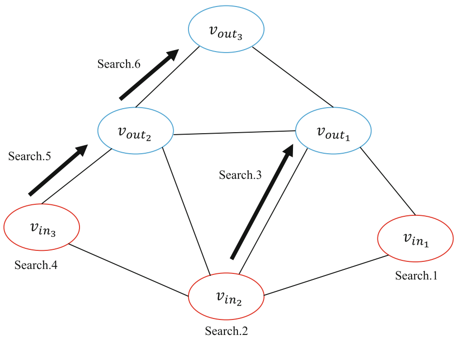


Fig. 7. The illustration of the depth-first search to categorize outbound vertices.

Categorizing Vertices by Depth-First Search. Figure 7 illustrates the overview of the depth-first search with a simple example. In this example, we have a set of vertices $V = \{v_{in_1}, v_{in_2}, v_{in_3}, v_{out_1}, v_{out_2}, v_{out_3}\}$ where $(v_{in_1}, v_{in_2}, v_{in_3})$ and $(v_{out_1}, v_{out_2}, v_{out_3})$ are inbound and outbound vertices, respectively. Besides, according to Fig. 7, we also have a set of edges

$$E = \{e_{v_{in_1}}[v_{in_2}, v_{out}1], e_{v_{in_2}}[v_{in_1}, v_{in_3}, v_{out_1}, v_{out_2}], e_{v_{in_3}}[v_{in_2}, v_{in_3}]$$
$$e_{v_{out_1}}[v_{in_1}, v_{in_2}, v_{out_2}, v_{out_3}], e_{v_{out_2}}[v_{in_2}, v_{in_3}, v_{out_1}, v_{out_3}], e_{v_{out_3}}[v_{out_1}, v_{out_2}]\}$$

Then, we attempt to categorize outbound vertices based on the inbound vertices.

This search starts from all inbound vertices in the inbound node. The procedure of our depth-first search is as follows:

Search 1. Start the depth-first search from v_{in_1}. According to $e_{v_{in_1}}[v_{in_2}, v_{out}1]$, we know that v_{in_1} is connected to v_{in_2} and v_{out_1}, but value pairs of both vertices have not been determined yet. Therefore, only the value pairs of in_1 can be determined in the first invocation of the depth-first search, and the corresponding vertices from v_{in_1} are only v_{in_1}.

Search 2. Start the depth-first search from v_{in_2}. According to $e_{v_{in_2}}[v_{in_1}, v_{in_3}, v_{out_1}, v_{out_2}]$, we know that v_{in_2} is connected to v_{in_1} and the value pairs of v_{in_1} have been already determined in Search 1. In other words, we have already conducted the inbound phase for v_{in_1} (or the outbound phase for the

Table 2. Categorizing outbound vertices. The outbound vertices are highlighted in red.

Step	Start points	Corresponding inbound vertices	Groups
1	v_{in_1}	$\{v_{in_1}\}$	$g_{\{v_{in_1}\}}$
2	v_{in_2}	$\{v_{in_2}\}$	$g_{\{v_{in_2}\}}$
3	v_{out_1}	$\{v_{in_1}, v_{in_2}\}$	$g_{\{v_{in_1}, v_{in_2}\}}$
4	v_{in_3}	$\{v_{in_3}\}$	$g_{\{v_{in_3}\}}$
5	v_{out_2}	$\{v_{in_1}, v_{in_2}, v_{in_3}\}$	$g_{\{v_{in_1}, v_{in_2}, v_{in_3}\}}$
6	v_{out_3}	$\{v_{in_1}, v_{in_2}, v_{in_3}\}$	$g_{\{v_{in_1}, v_{in_2}, v_{in_3}\}}$

outbound vertices). However, the corresponding vertex of v_{in_2} is only v_{in_2} because we can determine the value pairs of v_{in_2} independently.

Search 3. After Search 2, we know that value pairs of v_{out_1} can be determined because the value pairs of v_{in_1} and v_{in_2} have been already determined. Besides, according to $e_{v_{out_1}}[v_{in_1}, v_{in_2}, v_{out_2}, v_{out_3}]$, v_{out_1} is also connected to v_{out_2} and v_{out_3}, but we cannot determine their value pairs yet. Therefore, the corresponding vertices of v_{out_1} are v_{in_1} and v_{in_2}.

Search 4. Start the depth-first search from v_{in_3}. For the same reason as Searches 1 and 2, the corresponding vertex of v_{in_3} is only v_{in_3}.

Search 5. After Search 4, we know that v_{out_2} can be determined. The value pairs of v_{in_3} can be determined independently, but the value pairs of v_{in_2} have already been determined depending on the value pairs of v_{in_1} to obtain the value pairs of v_{out_1}. Therefore, the value pairs of v_{out_2} must correspond to the value pairs of v_{in_1}, and the corresponding vertices of v_{out_2} are v_{in_1}, v_{in_2}, and v_{in_3}.

Search 6. We determine the value pairs of the remaining vertex v_{out_3}. According to $e_{v_{out_3}}[v_{out_1}, v_{out_2}]$, v_{out_3} is connected to v_{out_1} and v_{out_2}, and their corresponding vertices contain v_{in_1}, v_{in_2}, and v_{in_3}. Therefore, the corresponding vertices of v_{out_3} are also v_{in_1}, v_{in_2}, and v_{in_3}.

Then, we obtain Table 2 that shows which inbound vertices each vertex corresponds to. After categorizing them, we generate a tree construction of vertices, called a *DoF tree*, which allows us to calculate the value pairs of each vertex with a minimum time complexity. Figure 8 shows the DoF tree based on Table 2. In Fig. 8, each surrounded vertex by the black square can be calculated from their value pairs independently. For $g_{\{v_{in_1}, v_{in_2}\}}$, we can use $DoF_{v_{in_1}}$ and $DoF_{v_{in_2}}$ to find the value pairs of $g_{\{v_{in_1}, v_{in_2}, v_{in_3}\}}$ while we can also calculate the value pairs of $g_{\{v_{in_1}, v_{in_2}, v_{in_3}\}}$ using $DoF_{v_{in_3}}$ at the same time. If $\text{Prob.}(v_{out_2}, v_{out_3})^{-1} > DoF_{v_{in_3}}$, we use $DoF_{v_{in_1}}$ and $DoF_{v_{in_2}}$ via $g_{\{v_{in_1}, v_{in_2}\}}$. Hence, in that case, we can partly calculate the value pairs of $g_{\{v_{in_1}, v_{in_2}, v_{in_3}\}}$ independently. The detailed attack complexity will be calculated in Step 6. GROUPING() in Algorithm 2 accepts the starting point and the graph expression of the differential characteristic as the input and outputs the DoF tree.

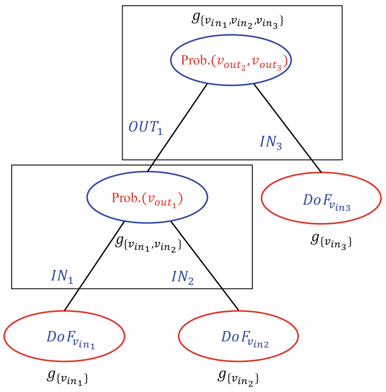


Fig. 8. Overview of the DoF tree. Prob.(v_{out_1}) and Prob.(v_{out_2}, v_{out_3}) denote the probability of the outbound groups $g_{\{v_{in_1},v_{in_2}\}}$ and $g_{\{v_{in_1},v_{in_2},v_{in_3}\}}$, respectively. $(DoF_{v_{in_1}}, DoF_{v_{in_2}}, DoF_{v_{in_3}})$ denote the available DoF of $(v_{in_1}, v_{in_2}, v_{in_3})$, respectively. $(IN_1, IN_2, IN_3, OUT_1)$ denote the DoF used to find the value pairs of the connected outbound groups.

Step 6: Estimating Attack Complexity. We check whether the total available DoF is enough to find the value pairs fulfilling the differential characteristic and its time complexity. We use MILP (*Mixed-Integer Linear Programming*) to check those and minimize the time complexity. During an MILP modeling, we assign DoF derived from each outbound group, highlighted red color in Table 2, as the linear constraints. For a better understanding, we give all constraints as the product of the probability and DoF, but we can express them by the linear inequalities by the weight and give them to an MILP solver. Let DoF_{OUT_i}, DoF_{C_j}, and Prob.(OUT_i) be DoF derived from outbound group OUT_i, the DoF used to calculate the value pairs in each connected group c_j, and the probability of outbound group OUT_i, respectively. The DoF derived from the outbound group OUT_i is calculated as follows:

$$DoF_{OUT_i} = \left(\prod_{j=1}^{n} DoF_{C_j}\right) \cdot \text{Prob.}(OUT_i), \tag{1}$$

where n denotes the number of connected groups to OUT_i. We assign Eq. (1) for all outbound groups as the linear constraints in an MILP model. For inbound groups, the black colored groups in Table 2, DoF used to calculate the value pairs of the connected outbound groups must be smaller than DoF derived from this inbound group. Thus, we assign such constraints as follows:

$$IN_k \leq DoF_{IN_k}, \tag{2}$$

where IN_k and DoF_{IN_k} denote the DoF used to calculate the value pairs of the connected outbound groups and DoF derived from this inbound group IN_k, respectively. We assign Eq. (2) for all inbound groups as the linear constraints in an MILP model.

Besides, for each outbound group, we need to ensure that the time complexity is smaller than the birthday bound as follows:

$$T^i_{max} \geq \prod_{j=1}^{m} IN_j \cdot \prod_{k=1}^{n} OUT_k \tag{3}$$

where T^i_{max}, m, and n denote the objective variables, the number of outbound groups, and the number of inbound groups connected to the target outbound group, respectively. We assign Eq. (3) for all outbound groups as the linear constraints in an MILP model.

Then, we need to set the objective function, which minimizes the time complexity. In the field of symmetric-key cryptography, we often assign one objective function, such as the number of active S-boxes and the total weight in primitives, and minimize it, equivalent to solving the minimization problem. In contrast, the time complexity of our attack is dominated by the maximum T^i_{max}, but there is no way to know which T^i_{max} will be maximum in the standard MILP model. Therefore, we solve the MIN-MAX problem, a class of problems where the objective is to minimize the maximum value of a set of variables and functions, instead of the minimization problem. Hence, we assign a set of all T^i_{max} as the objective function as follows:

$$(T^1_{max}, T^2_{max}, \ldots, T^m_{max}). \tag{4}$$

Then, we minimize the maximum variables in Eq. (4) by an MILP solver. In this work, we use `SageMath`[3] as an MILP solver. Our attack is successful if the maximum value in Eq. (4) will be smaller than the birthday bound. Otherwise, we conduct the same procedure for another DoF tree. CALCTIME() in Algorithm 1 accepts the DoF tree as the input and outputs the time complexity of this attack.

5 Key Collisions on AES-128/192/256

In this section, we show fixed-target-plaintext key collisions on 2/5/6-round AES-128/192/256 and free-target-plaintext key collisions on 5/7/9-round AES-128/192/256, which are found by our automatic tool provided in Sect. 4.

Notations. We use the following notations for our attacks. For $i \in \{1, \ldots, 12\}$, let in_i be the inbound vertices; let IN_i be the assigned degrees of freedom in the attack; let IN_{M_i} be the available degrees of freedom from a given differential characteristic; and let $P(in_i)$ be a differential probability of in_i in the corresponding SB operation.

Moreover, the internal states of AES are treated here as a column-wise array of 4-byte words, with columns numbered from the left. For example, $x_i[0]$ and

[3] https://www.sagemath.org.

$x_i[3]$ are represented as 4-byte words in the leftmost and rightmost columns in the i-th round internal state before the SB operation, respectively. In the same manner, the i-th round internal state before the AK operation is represented as $w_i[3]$ and $w_i[0]$, respectively, and the i-th round key is denoted by k_i. Besides, all inbound vertices are written in the same color since it is clear that their values are obtained independently.

5.1 Fixed-Target-Plaintext Key Collision on 6-Round AES-256

Figure 9 illustrates an underlying differential characteristic for a fixed-target-plaintext key collision for 6-round AES-256 with a probability of 2^{-179}. Assuming that the 1st and 2nd rounds in the data processing part are an inbound phase (i.e., $\{in_1, \ldots, in_8\} = \{x_1[3], \ldots, x_1[0], x_2[3], \ldots, x_2[0]\}$), and the remaining part, including the key scheduling part, is an outbound phase, the probability of inbound and outbound phases are 2^{-118} and 2^{-61}, respectively. After applying our tool, we can obtain the DoF tree as shown in Fig. 10 for collision attacks. By using this tree, we can construct a fixed-target-plaintext key collision on 6-round AES-256.

Degree of Freedom in the Inbound Phase. In our attacks, we exploit the degrees of freedom of each inbound vertex as follows: $IN_1 = 2^{17}$, $IN_2 = 2^{17}$, $IN_3 = 2^{17}$, $IN_4 = 2^{0}$, $IN_5 = 2^{11}$, $IN_6 = 2^{11}$, $IN_7 = 2^{11}$, and $IN_8 = 2^{12}$.

By using our automatic tool, we have obtained the differential probability of each inbound vertex as follows: $P(in_1) = 2^{-7}$, $P(in_2) = 2^{-0}$, $P(in_3) = 2^{-0}$, $P(in_4) = 2^{-28}$, $P(in_5) = 2^{-21}$, $P(in_6) = 2^{-21}$, $P(in_7) = 2^{-21}$, and $P(in_8) = 2^{-20}$. Based on these results, the maximum degrees of freedom in each vertex are estimated as $IN_{M_1} = 2^{25(=32-7)}$, $IN_{M_2} = 2^{32(=32-0)}$, $IN_{M_3} = 2^{32(=32-0)}$, $IN_{M_4} = 2^{4(=32-28)}$, $IN_{M_5} = 2^{11(=32-21)}$, $IN_{M_6} = 2^{11(=32-21)}$, $IN_{M_7} = 2^{11(=32-21)}$, and $IN_{M_8} = 2^{12(=32-20)}$. Thus, the degrees of freedom of each inbound vertex, which is required for the attack, are sufficiently available.

Attack Procedures

1. Start with $2^{51(=17+17+17+0)}$ sets of $\{in_1, \ldots, in_4\} = \{x_1[3], \ldots, x_1[0]\}$ and the fixed initial value sets of $\{w_0[3], \ldots, w_0[0]\}$ (red part in Fig. 11); then, obtain 2^{51} sets of $\{w_1[0], \ldots, w_1[3], k_0[0], \ldots, k_0[3]\}$ ((blue, (purple, (green, and (orange parts in Fig. 11).
2. Prepare 2^{11} values of $in_5 = x_2[3]$ ((red part in Fig. 11); then, obtain $2^{62(=51+11)}$ sets of $\{k_1[3], k_2[0], \ldots, k_2[3], k_3[0]\}$ ((turquoise part in Fig. 11). As the differential probability of the corresponding two SW functions in the key schedule part is $2^{-35(=-28-7)}$, there exist $2^{27(=62-35)}$ values that fulfill the differential characteristic ((turquoise part in Fig. 11).
3. Prepare $2^{22(11+11)}$ sets of $\{in_6, in_7\} = \{x_2[2], x_2[1]\}$ ((red part in Fig. 11); then, obtain $2^{49(=27+22)}$ sets of $\{k_1[2], k_1[1]\}$ ((yellow and (brown parts in Fig. 11).

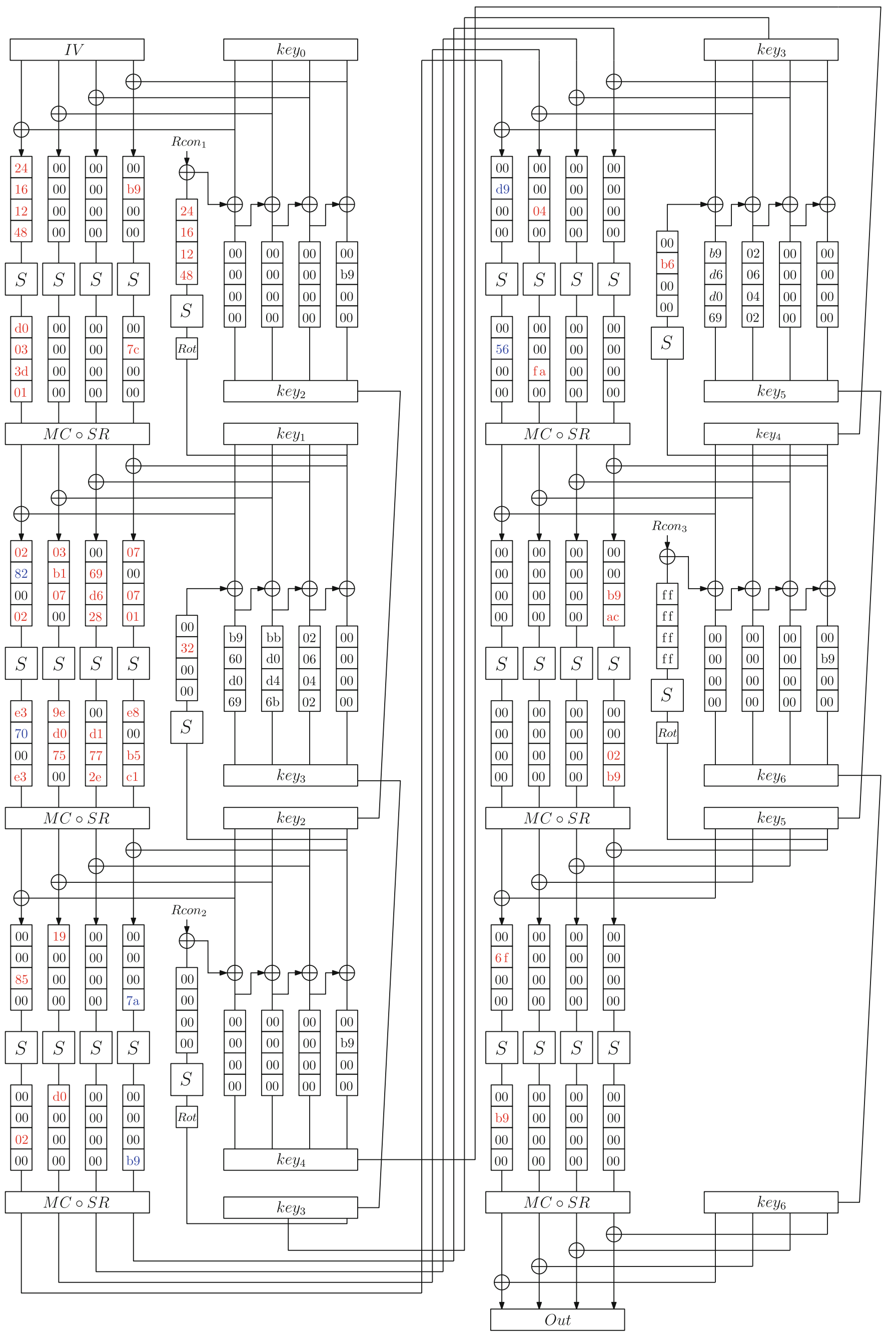


Fig. 9. Differential characteristic for a fixed-target-plaintext collision attack on 6-round AES-256.

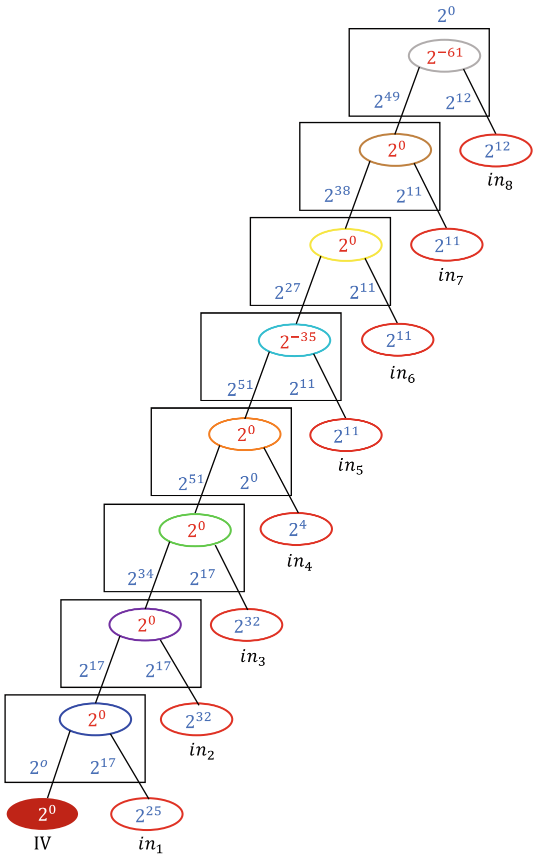


Fig. 10. DoF tree for a fixed-target-plaintext collision attack on 6-round AES-256.

4. Prepare 2^{12} values of $in_8 = x_2[0]$ ((red part in Fig. 11); then, obtain $2^{61(=49+12)}$ values of the remaining data processing and key scheduling parts ((gray part in Fig. 11). As the differential probability of the remaining parts is 2^{-61}, there exists $1 = 2^{0(=61-61)}$ value that fulfills the differential characteristic ((gray part in Fig. 11).

Attack Complexity. Following the above attack procedures, the attack complexity for Step 2 appears to be dominant, which requires approximately 2^{62} computations of the 1-round AES-256 key schedule. However, the attack complexity for Step 4 requires approximately 2^{61} computations of the partial (at least 4-round) AES-256 encryption; thus, from the perspective of a fair complexity estimation, it can be considered that the attack complexity for Step 4 is dominant. Therefore, total complexity is bounded by 2^{61} computations of 6-round AES-256.

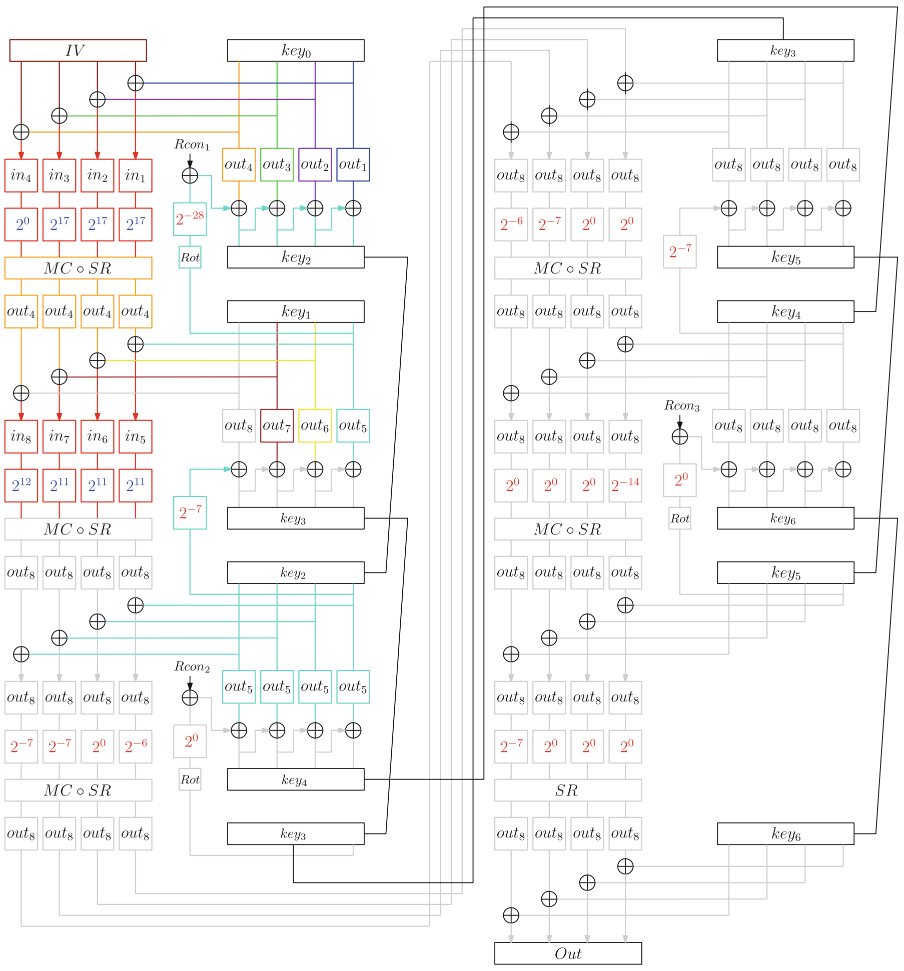


Fig. 11. Fixed-target-plaintext collision attack on 6-round AES-256. (Color figure online)

As in the actual execution of the attack, it is not necessary to store a complete set of values at each outbound vertices. For example, in Step 3, rather than storing $2^{49(=27+22)}$ sets of $\{k_1[2], k_1[1]\}$, we can proceed to evaluate the next outbound vertices for a single value of the previous outbound vertices. Thus, our attacks in this section can be done with negligible memory. This is confirmed by our experiment for the semi-free start collision attack on 9-round AES-256.

5.2 Free-Target-Plaintext Key Collision on 9-Round AES-256

Figure 12 illustrates an underlying differential characteristic for a free-target-plaintext key collision for 9-round AES-256 with a probability of 2^{-193}. Assuming that the 7th, 8th, and 9th rounds in the data processing part are an inbound phase (i.e., $\{in_1, \ldots, in_{12}\} = \{x_7[3], \ldots, x_7[0], x_8[3], \ldots, x_8[0], x_9[3], \ldots, x_9[0]\}$), and the remaining part, including the key scheduling part, is an outbound phase, the probability of inbound and outbound phases are 2^{-102} and 2^{-91}, respectively. After applying our tool, we can obtain the DoF tree as shown in Fig. 13 for collision attacks. By using this tree, we can construct a free-target-plaintext key collision on 9-round AES-256.

Degree of Freedom in the Inbound Phase. In our attacks, we exploit the degrees of freedom of each inbound vertex as follows: $IN_1 = 2^1$, $IN_2 = 2^0$, $IN_3 = 2^0$, $IN_4 = 2^0$, $IN_5 = 2^{26}$, $IN_6 = 2^0$, $IN_7 = 2^0$, $IN_8 = 2^9$, $IN_9 = 2^{19}$, $IN_{10} = 2^6$, $IN_{11} = 2^{11}$ and $IN_{12} = 2^{19}$.

By using our automatic tool, we have obtained the differential probability of each inbound vertex as follows: $P(in_1) = 2^{-6}$, $P(in_2) = 2^{-6}$, $P(in_3) = 2^{-6}$, $P(in_4) = 2^{-6}$, $P(in_5) = 2^{-6}$, $P(in_6) = 2^{-7}$, $P(in_7) = 2^{-6}$, $P(in_8) = 2^{-7}$, $P(in_9) = 2^{-13}$, $P(in_{10}) = 2^{-13}$, $P(in_{11}) = 2^{-13}$, and $P(in_{12}) = 2^{-13}$. Based on these results, the maximum degrees of freedom in each vertex are estimated as $IN_{M_1} = 2^{26(=32-6)}$, $IN_{M_2} = 2^{26(=32-6)}$, $IN_{M_3} = 2^{26(=32-6)}$, $IN_{M_4} = 2^{26(=32-6)}$, $IN_{M_5} = 2^{26(=32-6)}$, $IN_{M_6} = 2^{25(=32-7)}$, $IN_{M_7} = 2^{26(=32-6)}$, $IN_{M_8} = 2^{25(=32-7)}$, $IN_{M_9} = 2^{19(=32-13)}$, $IN_{M_{10}} = 2^{19(=32-13)}$, $IN_{M_{11}} = 2^{19(=32-13)}$ and $IN_{M_{12}} = 2^{19(=32-13)}$. Thus, the degrees of freedom of each inbound vertex, which is required for the attack, are sufficiently available.

Attack Procedures

1. Start with $2^{1(=1+0+0+0)}$ sets of $\{in_1, \ldots, in_4\} = \{x_7[0], \ldots, x_7[3]\}$ (red part in Fig. 14); then, obtain 2^1 sets of $\{w_7[3], \ldots, w_7[0]\}$ (blue part in Fig. 14).
2. Prepare 2^{26} values of $in_5 = x_8[3]$ (red part in Fig. 14); then, obtain $2^{27(=26+1)}$ values of $k_7[3]$ (purple part in Fig. 14). As the differential probability of the corresponding SW function in the key schedule part is 2^{-27}, there exist $2^{0(=27-27)}$ values that fulfill the differential characteristic (purple part in Fig. 14).
3. Prepare 2^0 values of $in_6 = x_8[2]$ (red part in Fig. 14); then, obtain $2^{0(=0+0)}$ values of $k_7[2]$ (green part in Fig. 14). As the differential probability of the corresponding key schedule part is 2^0, there exist $2^{(=0-0)}$ values that fulfill the differential characteristic (green part in Fig. 14).
4. Prepare 2^0 values of $in_7 = x_8[1]$ (red part in Fig. 14); then, obtain $2^{0(=0+0)}$ values of $k_7[1]$ (orange part in Fig. 14). As the differential probability of the corresponding key schedule part is 2^0, there exist $2^{0(=0-0)}$ values that fulfill the differential characteristic (orange part in Fig. 14).
5. Prepare 2^9 values of $in_8 = x_8[0]$ (red part in Fig. 14); then, obtain $2^{9(=9+0)}$ values of $k_7[0]$ (turquoise part in Fig. 14). As the differential probability of the corresponding key schedule part is 2^0, there exists $1 = 2^{9(=9-0)}$ value that fulfills the differential characteristic (turquoise part in Fig. 14).
6. Prepare 2^{19} values of $in_9 = x_9[3]$ (red part in Fig. 14); then, obtain $2^{28(=19+9)}$ values of $k_8[3]$ (yellow part in Fig. 14). As the differential probability of the

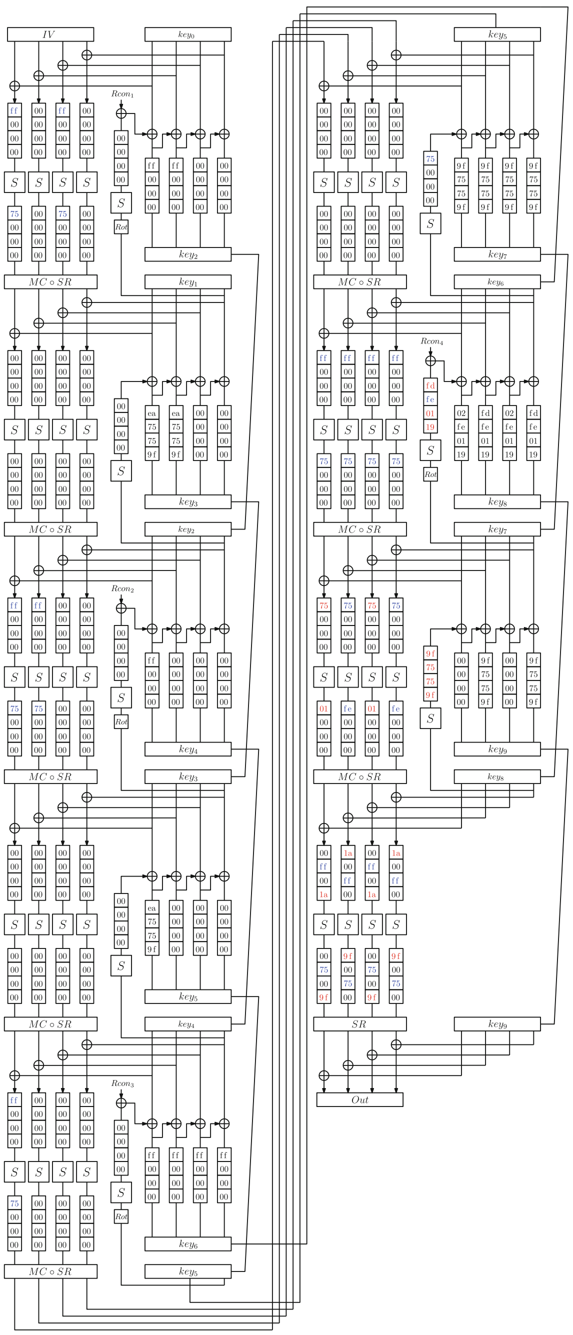


Fig. 12. Differential characteristic for a free-target-plaintext collision attack on 9-round AES-256.

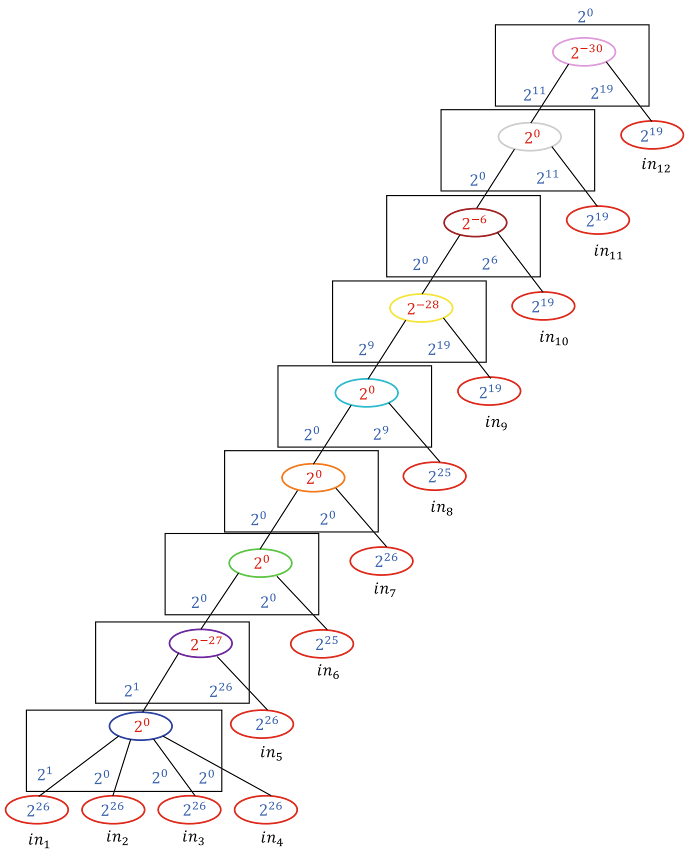


Fig. 13. DoF tree for a free-target-plaintext collision attack on 9-round AES-256.

corresponding key schedule part is 2^{-28}, there exist $2^{0(=28-28)}$ values that fulfill the differential characteristic (yellow part in Fig. 14).

7. Prepare 2^6 values of $in_{10} = x_9[2]$ (red part in Fig. 14); then, obtain $2^{6(=0+6)}$ values of $k_8[2]$ (brown part in Fig. 14). As the differential probability of the corresponding key schedule part is 2^{-6}, there exist $2^{0(=6-6)}$ values that fulfill the differential characteristic (brown part in Fig. 14).
8. Prepare 2^{11} values of $in_{11} = x_9[1]$ (red part in Fig. 14); then, obtain $2^{11(=0+11)}$ values of $k_8[1]$ (gray part in Fig. 14). As the differential probability of the corresponding key schedule part is 2^0, there exist $2^{11(=11-0)}$ values that fulfill the differential characteristic (gray part in Fig. 14).
9. Prepare 2^{19} values of $in_{12} = x_9[0]$ (red part in Fig. 14); then, obtain $2^{30(=11+19)}$ values of $k_8[0]$ (violet part in Fig. 14). As the differential probability of the remaining parts is 2^{-30}, there exists $1 = 2^{0(=30-30)}$ value that fulfills the differential characteristic (violet part in Fig. 14).

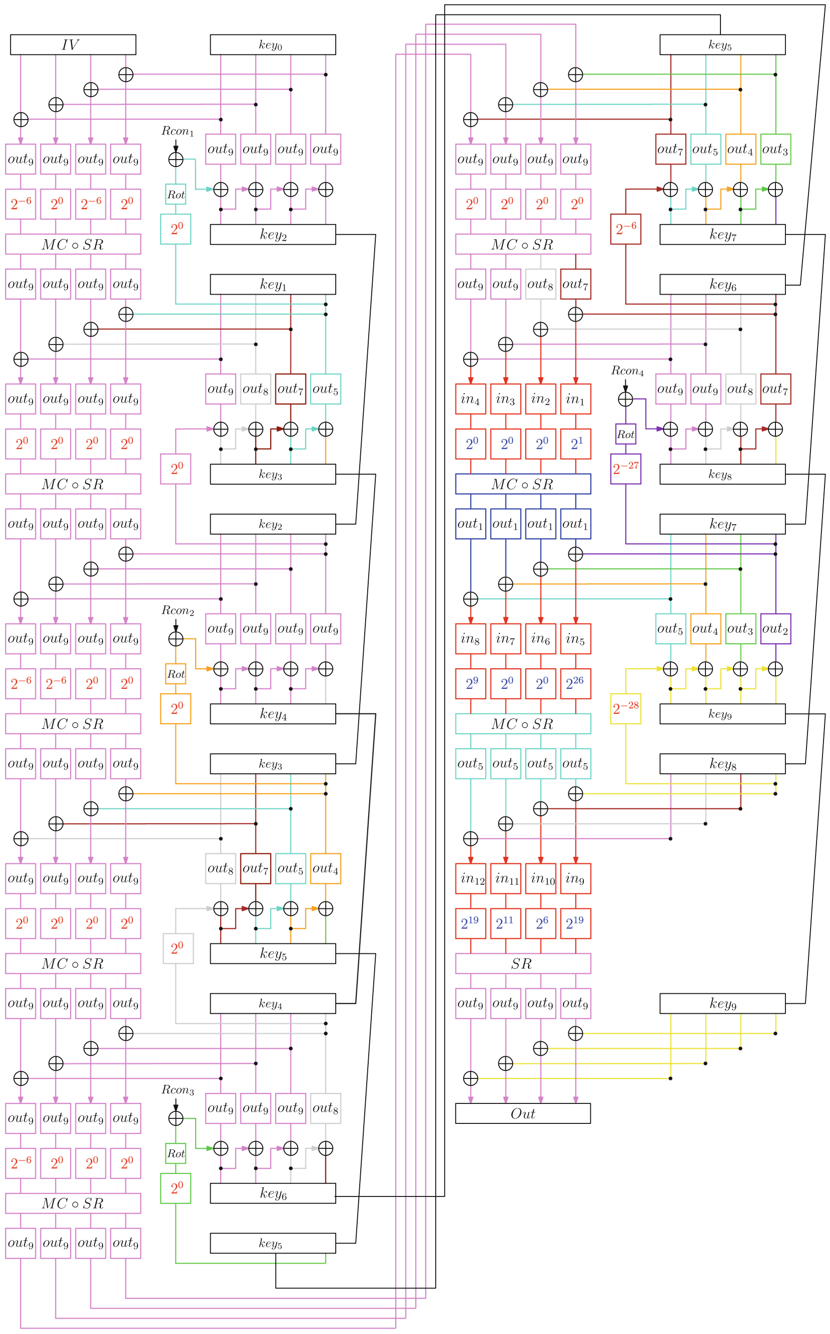


Fig. 14. Free-target-plaintext collision attack on 9-round AES-256. (Color figure online)

Table 3. An example input and output values of a free-target-plaintext key collision for 9-round AES-256. It is directly converted into a semi-free start collision attack on 9-round AES-256-DM by adding the feedword operation.

i	$Plaintext_i$	Key_i	$Ciphertext_i$
1	83 66 63 dc	ca 45 20 ea 26 11 ac 9c	7f ea d8 40
	b1 bc 61 82	30 3c c2 06 7c 39 55 e2	c0 59 30 d5
	30 38 ab f7	7e 2f d9 46 84 1f b2 3e	11 29 07 d0
	14 c3 d4 6a	96 2a 82 ef 21 00 57 6c	39 08 5a 65
2	83 66 63 dc	35 45 20 ea 26 11 ac 9c	7f ea d8 40
	b1 bc 61 82	cf 3c c2 06 7c 39 55 e2	c0 59 30 d5
	30 38 ab f7	94 5a ac d9 84 1f b2 3e	11 29 07 d0
	14 c3 d4 6a	7c 5f f7 70 21 00 57 6c	39 08 5a 65

Attack Complexity. Following the above attack procedures, it is obvious that the attack complexity for Step 9 is dominant, which requires approximately 2^{30} computations of the partial AES-256 encryption. Therefore, total complexity is bounded by 2^{30} computations of 9-round AES-256.

Experimental Verification. We experimentally verify the validity of the proposed attack, especially of the free-target-plaintext collision attack on 9-round AES-256. Our experiment was executed on AMD Ryzen Threadripper™PRO 5995WX @2.7GHz (64C/128T) with 512GB RAM; then, it was completed in 12 h. As a result, we have practically found the value at which such a key collision occurs. Table 3 shows an example case of a free-target-plaintext key collision for 9-round AES-256, and the values with differences in the keys are shown in red.

5.3 Key Collisions on AES-128/192

Due to the page limitation, we only show the results and omit the detailed procedures of finding fixed-target-plaintext key collision on 2/5-round AES-128/192 and free-target-plaintext key collisions on 5/7-round AES-128/192 in this proceeding version. The results are summarized in Table 1[4]. It should be mentioned that these collisions on AES-128/192 can be found by the same procedure for the fixed/free-target-plaintext key collision attacks on AES-256, described in Sects. 5.1 and 5.2.

[4] The results are shown as (semi-free-start) collision attacks on AES-128/192-DM. As described in Sect. 6.1, fixed-target-plaintext key collisions and free-target-plaintext key collisions on AES-128/192 are naturally converted into one-block collision and semi-free collision attacks on AES-128/192-DM.

6 Application to AES-DM

In this section, we present single-block (semi-free-start) collision attacks on AES-DM, which are based on the fixed/free-target-plaintext key collision attacks on AES provided in Sect. 5. Moreover, we show two-block collision attacks on AES-128-DM and AES-256-DM, also found by our automatic tool in Sect. 4.

Notations. Unlike the notations in Sect. 5, the internal states of AES are treated here as a byte-wise array. Then, we denote the i-th round of the state array at the m-th row from the left and the n-th column from the top by $x_i[4m+n]$ for $m, n \in \{0,1,2,3\}$. For example, $x_i[6]$ is represented as the i-th round state array at the 1st row from the left and the 2nd column from the top.

6.1 Single-Block (Semi-free-Start) Collision Attacks on AES-DM

As discussed in Sect. 3.2, fixed-target-plaintext key collisions and free-target-plaintext key collisions on AES-128/192/256 are naturally converted into one-block collision and semi-free collision attacks on AES-DM, respectively. Thus, we can construct collision attacks on 2/5/6 round AES-DM-128/192/256 and semi-free-start collision attacks on 5/7/9 round AES-DM-128/192/256, respectively.

6.2 How to Find Two-Block Collision Attack on AES-DM

We show that a class of single-block free-start collision attacks on AES-DM is converted into 2-block collision attacks. Specifically, in the second block, we prepare a specific class of free-start collision attacks in which an input chaining values could have any difference Δh_i, but its value h_i is predetermined, while in the setting of the standard free-start collision, both of value and difference of input chaining values are freely chosen by the adversary. We call this type of collision *free-differential-start collision*, defined as follows.

Definition 2 (Free-Differential-Start Collision). *Given a compression function CF, it finds a pair (v, m) and (v', m'), so that $CF(v, m) = CF(v', m')$, where v is a fixed value and $(v \neq v')$.*

Basic Idea. Suppose that a free-differential-start collision in the second block can be found with a time complexity of $T_2 < 2^{64}$. Furthermore, we assume that such collisions can be obtained, along with N different input differences of $\Delta h_1^{(1)}, \Delta h_1^{(2)}, \ldots, \Delta h_1^{(N)}$ with corresponding message differences of $\Delta m_1^{(1)}, \Delta m_1^{(2)}, \ldots, \Delta m_1^{(N)}$, and each of them can be found with the same complexity of $T_2 < 2^{64}$. It is expressed as follows:

$$\Delta CF(\Delta h_1^{(1)}, \Delta m_1^{(1)}) = \Delta CF(\Delta h_1^{(2)}, \Delta m_1^{(2)}) = \cdots = \Delta CF(\Delta h_1^{(N)}, \Delta m_1^{(N)}) = 0.$$

We first compute the first block by randomly choosing a pair of input messages m_0. Suppose we obtain a pair of h_1 having a difference that is equal to

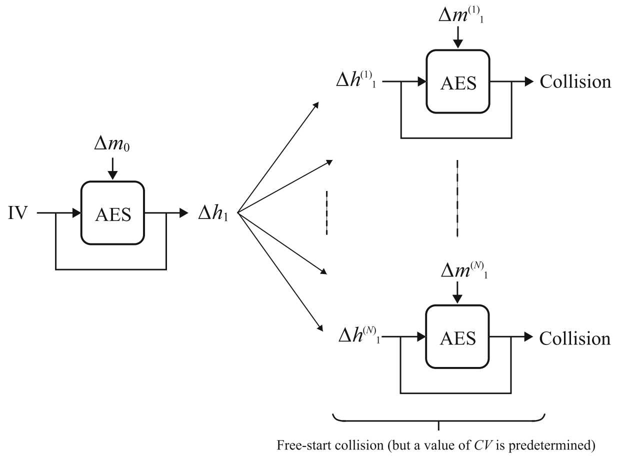


Fig. 15. Overview of the proposed two-block collision attack.

one of N patterns of $\Delta h_1^{(1)}, \Delta h_1^{(2)}, \ldots, \Delta h_1^{(N)}$. In that case, we mount a free-differential-start collision attack on the second block where a pair of input h_1 is fixed to the value identified in the search in the first block, and Δh_1 corresponds to one of N candidates. Consequently, this approach enables us to connect between the first and second blocks, thereby finding a collision in a two-block AES-DM as shown in Fig. 15.

Attack Complexity. The probability that Δh_1 coincides with one of N specified patterns of $\Delta h_1^{(1)}, \Delta h_1^{(2)}, \ldots, \Delta h_1^{(N)}$ is estimated to be $N/2^{128}$. Therefore, upon collecting $2^{128}/N$ pairs of h_1, we can find such a pair with high probability. Due to the birthday paradox, $2^{128}/N$ pairs can be prepared from $\sqrt{2^{128}/N} = 2^{64}/N^{\frac{1}{2}}$ values of h_i. Thus, the total complexity is estimated as

$$2^{64}/N^{\frac{1}{2}} \text{ one block comp.} + T_2 \text{ one block comp.}$$

For example, if $N = 4$ and $T_2 = 2^{62}$, the time complexity is estimated as

$$2^{63} \text{ one block comp.} + 2^{62} \text{ one block comp.} < 2^{63} \text{ two block comp.}$$

The memory requirements of $2^{64}/N^{\frac{1}{2}}$ in the first block.

6.3 Two-Block Collision Attacks on 9-Round AES-256-DM

Using our automatic tool, we can develop a 9-round free-differential-start collision in the second block with a time complexity of 2^{58} as shown in Fig. 16. To convert it into a two-block collision attack, we need to construct multiple attacks with N distinct input differences Δh_i and the same time complexity.

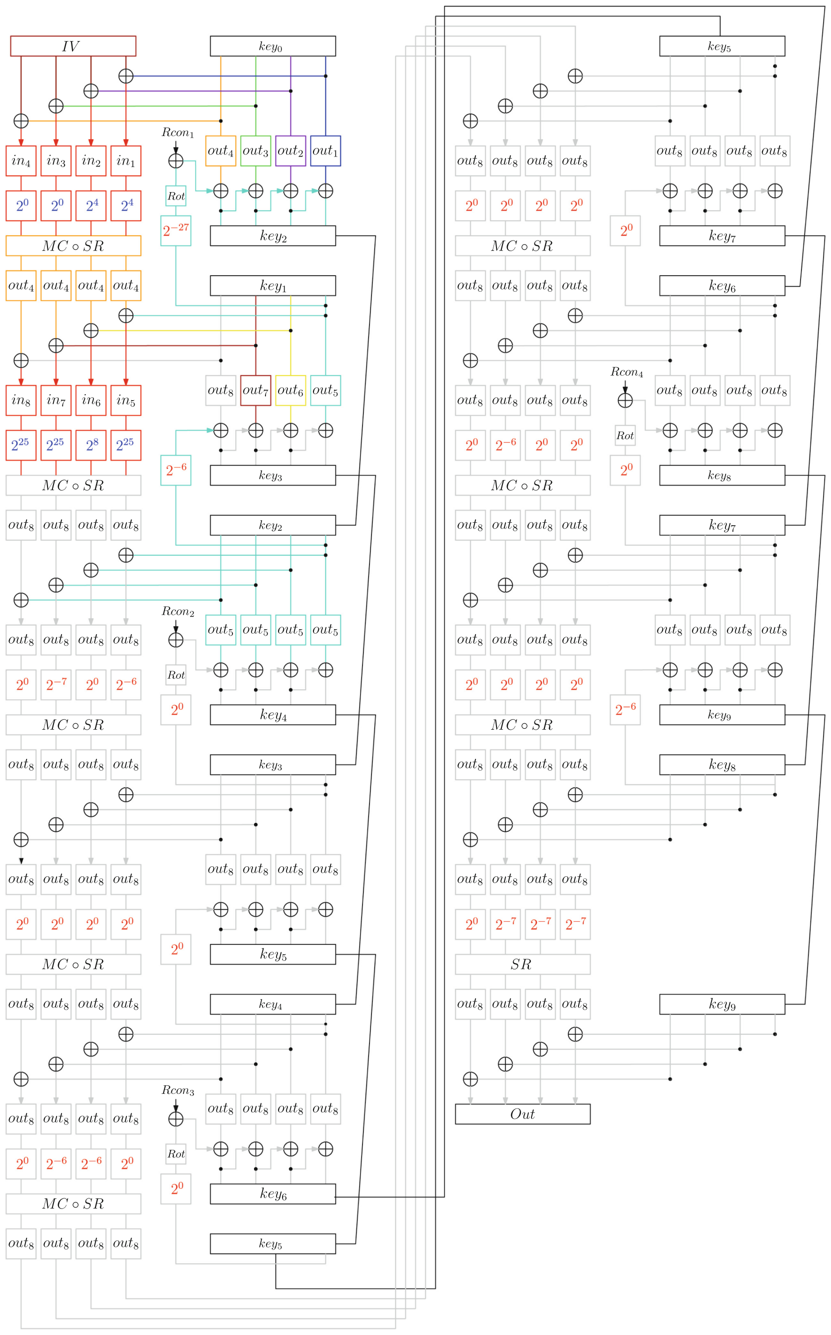


Fig. 16. Free-differential-start collision on 9-round AES-256-DM.

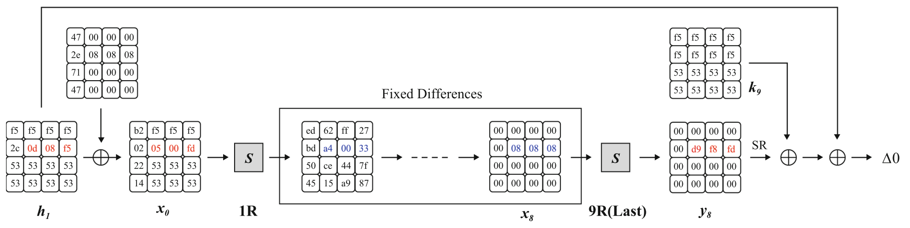


Fig. 17. Differential characteristic for the free-differential-start collision on 9-round AES-256-DM.

How to Find N Distinct Inputs. We can easily obtain such attacks by exploiting the differential characteristics in Fig. 17 and 18. It is well known that given a fixed input and output difference of Δx and Δy, the probability of $(\Delta y = Sbox(\Delta x))$ is approximately 1/2 where $Sbox()$ is an operation of S-box of AES [23].

This property indicates that there are about 128 distinct differences of $\Delta x_0[7]$, which result in $\Delta y_0[7] = 0\text{x}33$ through S-box. It means that $\Delta h_1[7](= \Delta x_0[7] \oplus \Delta k_0[7])$ also has 128 possible values, which lead to differential characteristics of y_0.

$\Delta h_1[7]$ is forwarded to the output, and $\Delta y_8[7]$ is computed as $\Delta y_8[7] = \Delta k_9[7] \oplus \Delta h_1[7]$. Once $\Delta h_1[7]$ is chosen out of 128 candidates, the corresponding $\Delta y_8[7]$ is determined. The probability that $(\Delta y_8[7] = Sbox(\Delta x_8[7] = 0\text{xfd}))$ is 1/2. Thus, there exists $128/2 = 64$ possible $\Delta h_1[7]$, which follow the characteristics for collision with time complexity of 2^{58}. As $\Delta h_1[5]$ and $\Delta h_1[6]$ also have 64 possible candidates, respectively, with the same reason, in total, there are $N = (64)^3 = 2^{18}$ distinct inputs with the same time complexity.

Attack Complexity. For $N = 2^{18}$ and $T_2 = 2^{58}$, the time complexity for 2-block collision attack is estimated as

$$2^{55} \text{ one block comp.} + 2^{58} \text{ one block comp.} < 2^{58} \text{ two block comp.}$$

The memory requirements of 2^{55} in the first block.

6.4 Two-Block Collision Attacks on 3-Round AES-128-DM

Due to the page limitation, we only show the result and omit the detailed attack procedure of two-block collision attacks on the 3-round AES-128-DM. The result is shown in Table 1. The detailed attack procedure is based on the same approach as that of two-block collision attacks on the 9-round AES-256-DM, described in Sect. 6.2

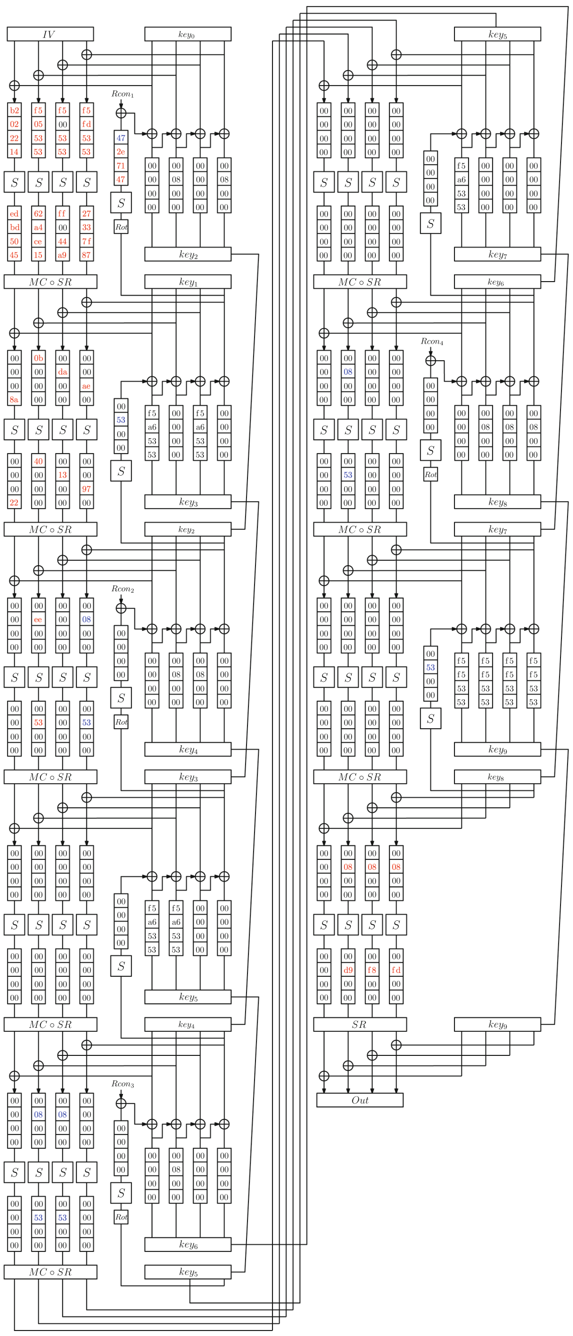


Fig. 18. The detailed differential characteristic for a free-differential-start collision on 9-round AES-256-DM.

7 Conclusion

In this paper, we investigated the new type of key collisions called target-plaintext key collisions of AES, which arise as an open problem in the key committing security and are directly converted into collision attacks on Davies-Meyer (DM) hashing mode. This key collision is such that a ciphertext collision is uniquely observed when a specific plaintext is encrypted under two distinct keys. We introduced an efficient automatic search tool designed to find target-plaintext key collisions. As a result, we demonstrated collision attacks on 2/5/6-round AES-128/192/256-DM and semi-free-start collision attacks on 5/7/9-round AES-128/192/256-DM, respectively. Furthermore, by exploiting a specific class of free-start collisions, we present collision attacks on 3/9-round AES-128/256-DM, respectively.

For further directions, it would be interesting to optimize how to determine the best choice of inbound vertices. Also, efficiently identifying invalid starting points would be promising to improve the efficiency of our method.

Acknowledgements. This result is obtained from the commissioned research (JPJ012368C05801) by the National Institute of Information and Communications Technology (NICT), Japan. This work was also supported by JSPS KAKENHI Grant Number JP24H00696.

References

1. Albertini, A., Duong, T., Gueron, S., Kölbl, S., Luykx, A., Schmieg, S.: How to abuse and fix authenticated encryption without key commitment. In: Butler, K.R.B., Thomas, K. (eds.) 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022. pp. 3291–3308. USENIX Association (2022), https://www.usenix.org/conference/usenixsecurity22/presentation/albertini
2. Aumasson, J., Jr., J.N., Sepehrdad, P.: Cryptanalysis of the ISDB scrambling algorithm (MULTI2). In: Dunkelman, O. (ed.) Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5665, pp. 296–307. Springer (2009). https://doi.org/10.1007/978-3-642-03317-9_18, https://doi.org/10.1007/978-3-642-03317-9_18
3. Becker, G.T., Regazzoni, F., Paar, C., Burleson, W.P.: Stealthy dopant-level hardware trojans: extended version. J. Cryptogr. Eng. **4**(1), 19–31 (2014). https://doi.org/10.1007/S13389-013-0068-0, https://doi.org/10.1007/s13389-013-0068-0
4. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and related-key attack on the full AES-256. In: CRYPTO. Lecture Notes in Computer Science, vol. 5677, pp. 231–249. Springer (2009)
5. Biryukov, A., Nikolic, I.: Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to aes, camellia, khazad and others. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 6110, pp. 322–344. Springer (2010)

6. Biryukov, A., Nikolic, I.: Colliding keys for SC2000-256. In: Joux, A., Youssef, A.M. (eds.) Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8781, pp. 77–91. Springer (2014). https://doi.org/10.1007/978-3-319-13051-4_5, https://doi.org/10.1007/978-3-319-13051-4_5
7. Chen, L.: Recommendation for key derivation using pseudorandom functions. NIST SP 800-108r1 (2022)
8. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002)
9. Derbez, P., Euler, M., Fouque, P., Nguyen, P.H.: Revisiting related-key boomerang attacks on AES using computer-aided tool. In: ASIACRYPT (3). Lecture Notes in Computer Science, vol. 13793, pp. 68–88. Springer (2022)
10. Dong, X., Guo, J., Li, S., Pham, P.: Triangulating rebound attack on aes-like hashing. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 13507, pp. 94–124. Springer (2022)
11. Dong, X., Sun, S., Shi, D., Gao, F., Wang, X., Hu, L.: Quantum collision attacks on aes-like hashing with low quantum random access memories. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 12492, pp. 727–757. Springer (2020)
12. Dong, X., Zhang, Z., Sun, S., Wei, C., Wang, X., Hu, L.: Automatic classical and quantum rebound attacks on aes-like hashing by exploiting related-key differentials. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 13090, pp. 241–271. Springer (2021)
13. Fouque, P., Jean, J., Peyrin, T.: Structural evaluation of AES and chosen-key distinguisher of 9-round AES-128. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 8042, pp. 183–203. Springer (2013)
14. Gérault, D., Minier, M., Solnon, C.: Constraint programming models for chosen key differential cryptanalysis. In: CP. Lecture Notes in Computer Science, vol. 9892, pp. 584–601. Springer (2016)
15. Gilbert, H., Peyrin, T.: Super-sbox cryptanalysis: Improved attacks for aes-like permutations. In: FSE. Lecture Notes in Computer Science, vol. 6147, pp. 365–383. Springer (2010)
16. Hosoyamada, A., Sasaki, Y.: Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 12106, pp. 249–279. Springer (2020)
17. Jean, J., Naya-Plasencia, M., Peyrin, T.: Multiple limited-birthday distinguishers and applications. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 8282, pp. 533–550. Springer (2013)
18. Kelsey, J., Schneier, B., Wagner, D.A.: Key-schedule cryptanalysis of idea, g-des, gost, safer, and triple-des. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 237–251. Springer (1996). https://doi.org/10.1007/3-540-68697-5_19, https://doi.org/10.1007/3-540-68697-5_19
19. Kim, H., Park, M., Cho, J., Kim, J., Kim, J.: Weaknesses of some lightweight blockciphers suitable for iot systems and their applications in hash modes. Peer-to-Peer Netw. Appl. **13**(2), 489–513 (2020)
20. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: Results on the full whirlpool compression function. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 5912, pp. 126–143. Springer (2009)

21. Matsui, M.: Key collisions of the RC4 stream cipher. In: Dunkelman, O. (ed.) Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5665, pp. 38–50. Springer (2009). https://doi.org/10.1007/978-3-642-03317-9_3, https://doi.org/10.1007/978-3-642-03317-9_3
22. Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved cryptanalysis of the reduced grøstl compression function, ECHO permutation and AES block cipher. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 5867, pp. 16–35. Springer (2009)
23. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: Cryptanalysis of reduced whirlpool and grøstl. In: FSE. Lecture Notes in Computer Science, vol. 5665, pp. 260–276. Springer (2009)
24. Robshaw, M.: A cryptographic review of cipherunicorn-a. CRYPTRECT Technical report (2001)
25. Sasaki, Y.: Meet-in-the-middle preimage attacks on AES hashing modes and an application to whirlpool. In: Joux, A. (ed.) Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6733, pp. 378–396. Springer (2011). https://doi.org/10.1007/978-3-642-21702-9_22, https://doi.org/10.1007/978-3-642-21702-9_22
26. Sasaki, Y., Li, Y., Wang, L., Sakiyama, K., Ohta, K.: Non-full-active super-sbox analysis: Applications to ECHO and grøstl. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 6477, pp. 38–55. Springer (2010)
27. Sun, L., Wang, M.: Sok: Modeling for large s-boxes oriented to differential probabilities and linear correlations. IACR Trans. Symmetric Cryptol. **2023**(1), 111–151 (2023)
28. Sun, L., Wang, W., Wang, M.: More Accurate Differential Properties of LED64 and Midori64. IACR Trans. Symmetric Cryptol. **2018**(3), 93–123 (2018)
29. Sun, L., Wang, W., Wang, M.: Accelerating the Search of Differential and Linear Characteristics with the SAT Method. IACR Cryptol. ePrint Arch. p. 213 (2021)

The Boomerang Chain Distinguishers: New Record for 6-Round AES

Xueping Yan, Lin Tan(✉), Hong Xu, and Wenfeng Qi

Information Engineering University, Zhengzhou, China
yanxueping163@163.com,tanlin100@163.com

Abstract. AES is the most used block cipher, and its round-reduced variants are popular underlying components to design cryptographic schemes. How to effectively distinguish round-reduced AES from random permutations has always been a hot research topic. Currently, the longest rounds of AES can be distinguished is 6 rounds, where the best result is the 6-round exchange distinguisher with the data complexity 2^{84}. In this paper, we extend the classical boomerang distinguisher which uses only one boomerang property to use two or more related boomerangs and the technique of 'friend pairs' to enhance the distinguishing effect. We propose the frameworks of the re-boomerang and boomerang chain distinguishers and apply them to 6-round AES. The re-boomerang distinguisher uses two related boomerangs sequentially, which have the same upper truncated differential trail in the forward direction. A plaintext pair is called a right pair if it follows this truncated differential trail. By the first boomerang, a target set of plaintext pairs containing one right pair can be obtained. Then for each pair in the target set, construct its 'friend pairs' as the input of the second boomerang to distinguish the cipher. Due to the dependence of the two boomerangs, all 'friend pairs' of the right pair are right pairs, so the probability of the second boomerang is increased. To further improve the complexity, we insert a new boomerang in the middle of the re-boomerang and repeat it to reduce the target set. Combining the strategies of using more data in each boomerang and repeating the distinguishing process several times, we give a boomerang chain distinguisher on 6-round AES with success probability 60% and complexity $2^{76.57}$, reduced by a factor of 172 compared with the previous best result. This is a new record for the secret-key distinguisher on 6-round AES.

Keywords: AES · Distinguisher · Boomerang · Boomerang Chain

1 Introduction

The Advanced Encryption Standard (AES) [16] is the most widely used block cipher and its security has been studied worldwide in the last twenty years. There is no known attack on full AES faster than exhaustive search. The security evaluation on round-reduced AES is also an important problem. Many ciphers use

https://doi.org/10.1007/978-981-96-0941-3_10

round-reduced AES as their core components, such as Hound [22] and WEM [13] which use 5-round AES, and TNT-AES [1] which uses 6-round AES. Many cryptanalysis techniques have been applied on round-reduced AES, such as integral [21], impossible differential [29], zero-correlation linear [27,34], subspace trail [25], mixture differential [3,24,26], meet-in-the-middle [18], yoyo [32], exchange [5] and boomerang [6,19,31].

The distinguishing attack and key recovery attack are two different aspects on the security evaluation of a cipher. The secret-key distinguisher can be used to evaluate the randomness of the cipher for any key, and has the potential to be developed to a key recovery attack. Studying the distinguishers for round-reduced AES is an important field and has attracted much attention of scholars. Before 2016, distinguishers on AES cover at most 4 rounds, including integral [21], impossible differential [8,38] and zero-correlation linear [11]. In [23,27,34], researchers exploited the properties of MixColumns matrix of AES to present key-dependent 5-round distinguishers. In EUROCRYPT 2017, Grassi et al. [26] discovered a new structural non-random property and proposed the first secret-key distinguisher on 5-round AES with complexity 2^{32}, called multiple-of-8 distinguisher. In FSE 2019, Boura et al. [12] gave a general proof of multiple-of-8 property for 5-round AES-like ciphers. The best 5-round distinguisher with complexity 2^{30} was given by exchange attacks in [5].

The first secret-key distinguisher for 6-round AES was proposed by Rønjom et al. [32], which used yoyo tricks in the adaptively chosen plaintexts and ciphertexts setting. In [30], the 6-round yoyo distinguisher is shown to be ineffective with the proposed data complexity. In ASIACRYPT 2019, Bardeh et al. [5] proposed the exchange attacks, and gave a 6-round distinguisher on AES with data and time complexities $2^{88.2}$. In [4], Bardeh studied the exchange attacks in the adaptively chosen ciphertexts setting and gave the best distinguisher for 6-round AES with data complexity of 2^{84} and time complexity of 2^{83}. In [2], Bao et al. showed a truncated differential distinguisher on 6-round AES with data complexity $2^{89.4}$ and time complexity $2^{96.5}$ memory accesses. In EUROCRYPT 2023, Bariant et al. [6] gave a truncated boomerang distinguisher on 6-round AES in the adaptively chosen ciphertexts setting with data and time complexities of 2^{87}.

Motivation. Based on previous work, we have the following consideration.

-Compared with the distinguishers on 5-round AES [5,26,30] having the data complexity less than 2^{32}, the previous distinguishers on 6-round AES [2,4–6] have much high complexities. The best result is the exchange distinguisher [4] with the data complexity 2^{84}. How to shorten this gap or improve the data complexity of the secret-key distinguisher on 6-round AES? It is a challenging problem in the academic research.

-By the observation of current techniques of distinguishers on 6-round AES including yoyo tricks [32], exchange [4,5] and boomerang [6], it is shown that the attacker is provided a wider space in the adaptively chosen plaintexts and ciphertexts setting. How to fully utilize the advantage of this setting to develop new cryptanalysis techniques?

-Boomerang and its variants have shown the power of cryptanalysis for round-reduced AES and AES-like ciphers [6, 19, 31]. The classical boomerang distinguisher usually uses only one boomerang property. Whether we can use two or more boomerangs to enhance the distinguishing effect, especially combined with the techniques of 'friend pairs' and exchange due to the structural property of AES.

Contributions. Fully utilizing the advantage of the adaptively chosen plaintexts and ciphertexts attack setting, we extend the classical boomerang to propose the re-boomerang and boomerang chain distinguishers, combined with the techniques of 'friend pairs' [7] and exchange attack [4, 5]. Application to 6-round AES, we present the best secret-key distinguisher with the success probability 60% and complexity of $2^{76.57}$, which is a new record.

-Re-boomerang Distinguisher. We extend the classical boomerang distinguisher which uses only one boomerang property to combine two related boomerangs to enhance the distinguishing effect. The two boomerangs are constructed by combining truncated boomerangs with exchange technique, where the probability of the lower trail in E_1 is 1. The two boomerangs have the same upper truncated differential trail for E_0 in the forward direction, of which the probability is $\overrightarrow{p} = 2^{-22}$. A plaintext pair is called a right pair if it follows this truncated differential trail. By the first boomerang, we can obtain a target set L of 2^{12} plaintext pairs containing one right pair on average. Then for each pair in L, construct its 'friend pairs' as the input of the second boomerang to distinguish the cipher. Due to the dependence of the two boomerangs, all 'friend pairs' of the right pair are right pairs, so the probability of the second boomerang is increased by a factor of $\overrightarrow{p}^{-1} = 2^{22}$. Compared with the size of L, we still have a gain of 2^{10}. The re-boomerang distinguisher on 6-round AES has the data and time complexities of $2^{82.33}$.

-Boomerang Chain Distinguisher. To improve the complexity, we insert a new boomerang in the middle of the re-boomerang and repeat it to reduce the size of the target set L, which forms the boomerang chain distinguisher. The wrong pairs in L are filtered gradually, then the input data of the last boomerang is reduced and the total attack complexity is improved. To increase the success probability of the boomerang chain distinguisher, we consider the two strategies of using more data in each boomerang and repeating the distinguishing process several times, and their combination. For the optimal strategy, we get a boomerang chain distinguisher on 6-round AES with success probability 60% and complexity of $2^{76.57}$, reduced by a factor of 172 compared with the previous best result [4].

Comparison with Previous Work. Compared with the truncated boomerang distinguisher for 6-round AES in [6], we generate new ciphertext pairs by exchanging inverse diagonals of original ciphertext pairs instead of adding a fixed truncated difference. The properties of the lower part E_1 in this paper and [6] are different. The probability of E_1 in this paper is $q = 1$, while $q = 2^{-22}$ in [6]. Compared with 6-round exchange attacks in [4], our first boomerang uses a larger returned truncated difference set, which increases the boomerang probability. Based on the first boomerang and technique of 'friend pairs' we propose the re-boomerang and boomerang chain distinguishers for 6-round AES. Our lowest data complexity is $2^{76.57}$, compared with the 6-round exchange attacks [4] of data complexity 2^{84}. Compared with the retracing boomerang in [19], we increase the boomerang probability by a factor of $\overrightarrow{p}^{-1}$ in E_0, using the techniques of related boomerangs and 'friend pairs'. The retracing boomerang requires some relations of ciphertexts in E_1 such that the probability is increased by a factor of q^{-1}.

The current secret-key distinguishers for 5 and 6 rounds of AES are shown in Table 1. Data complexity is measured in chosen plaintexts (CP), adaptively chosen ciphertexts (ACC) or adaptively chosen plaintexts and ciphertexts (ACPC). Time complexity is measured in equivalent number of AES encryptions (E) or memory accesses (M).

Table 1. Secret-key distinguishers for 5 and 6 rounds of AES

Property	Rounds	Data	Time	Success Probability	Ref.
Multiple-of-8	5	2^{32} CP	$2^{35.6}$ M	100%	[26]
Exchange Attack	5	2^{30} CP	2^{30} E	63%	[5]
yoyo	5	$2^{29.95}$ ACPC	$2^{29.95}$ M	55%	[30]
yoyo	5	$2^{30.65}$ ACPC	$2^{30.65}$ M	81%	[30]
Truncated Differential	6	$2^{89.4}$ CP	$2^{96.5}$ M	95%	[2]
Exchange Attack	6	$2^{88.2}$ CP	$2^{88.2}$ E	73%	[5]
Truncated Boomerang	6	2^{87} ACC	2^{87} E	84%	[6]
Exchange Attack	6	2^{84} ACC	2^{83} E	63%	[4]
Re-boomerang	6	$2^{82.33}$ ACPC	$2^{82.33}$ E	64%	Sect. 3
Triple Boomerangs	6	$2^{77.82}$ ACPC	$2^{77.82}$ E	66%	Subsect. 4.1
Boomerang Chain	6	$2^{76.57}$ ACPC	$2^{76.57}$ E	60%	Subsect. 4.2

Organization. This paper is organized as follows. In Sect. 2 we briefly describe AES and the previous related work. In Sect. 3 we introduce the re-boomerang distinguisher for 6-round AES. In Sect. 4 we improve the re-boomerang distinguisher and propose the boomerang chain distinguisher. Conclusion is given in Sect. 5. The source codes of the experiments in this paper are available online[1].

2 Preliminaries

2.1 Brief Description of AES

AES [16] is a Substitution-Permutation Network cipher which has 128-bit block and 128-bit, 192-bit or 256-bit keys. The 128-bit state of AES can be described

[1] https://github.com/XuepingYan/The-Boomerang-Chain-Distinguishers.

as a 4×4 matrix over the finite field $\mathbb{F}_{2^8}$. The round transformation of AES consists of the following four operations.

-*SubBytes* (SB) : applying the same 8-bit S-box on each byte of the state.

-*ShiftRows* (SR) : cyclic shift of the i-th row by i bytes to the left for $i = 0, 1, 2, 3$.

-*MixColumns* (MC) : multiplication of each column by an MDS matrix over $\mathbb{F}_{2^8}$.

-*AddRoundKey* (AK) : XORing the state with a 128-bit round key.

Before the first round, an additional AK is used and in the last round the MC is omitted. Depending on the master key length of AES, N_r rounds are applied to the state: $N_r = 10$ for AES-128, $N_r = 12$ for AES-192 and $N_r = 14$ for AES-256, respectively.

In this paper, denote by $Col(i)$ the i-th column of the state, denote by $SR(Col(i))$ the result of applying SR operation on the i-th column, and denote by $SR^{-1}(Col(i))$ the result of applying SR^{-1} operation on the i-th column, $0 \le i \le 3$. In other words, $SR^{-1}(Col(i))$ denotes the i-th diagonal of the state, and $SR(Col(i))$ denotes the i-th inverse diagonal of the state, $0 \le i \le 3$. If a pair (X_1, X_2) of states satisfies that $SR^{-1}(Col(i))$ of $X_1 + X_2$ is nonzero, we call (X_1, X_2) is active in the i-th diagonal. If $SR(Col(i))$ of $X_1 + X_2$ is nonzero, we call (X_1, X_2) is active in the i-th inverse diagonal, $0 \le i \le 3$.

2.2 Differentials and Truncated Differentials

Differential cryptanalysis [9] is a powerful cryptanalysis approach for block ciphers. A differential of the cipher E is defined by an input difference $\alpha \in \{0,1\}^n$ and an output difference $\beta \in \{0,1\}^n$, denoted by $\alpha \xrightarrow{E} \beta$. The probability of $\alpha \xrightarrow{E} \beta$ is defined as

$$Pr(\alpha \xrightarrow{E} \beta) = \frac{1}{2^n} \# \left\{ P \in \{0,1\}^n \mid E(P) + E(P + \alpha) = \beta \right\}.$$

Since E is a permutation, $Pr(\alpha \xrightarrow{E} \beta) = Pr(\beta \xrightarrow{E^{-1}} \alpha)$.

A truncated differential [28] is defined by a set of input differences $\mathcal{D}_{in}$ and a set of output differences $\mathcal{D}_{out}$, denoted by $\mathcal{D}_{in} \xrightarrow{E} \mathcal{D}_{out}$. The probability of $\mathcal{D}_{in} \xrightarrow{E} \mathcal{D}_{out}$ is defined as

$$Pr(\mathcal{D}_{in} \xrightarrow{E} \mathcal{D}_{out}) = \frac{\sum\limits_{\alpha \in \mathcal{D}_{in}, \beta \in \mathcal{D}_{out}} Pr(\alpha \xrightarrow{E} \beta)}{|\mathcal{D}_{in}|}.$$

In general, $Pr(\mathcal{D}_{in} \xrightarrow{E} \mathcal{D}_{out})$ and $Pr(\mathcal{D}_{out} \xrightarrow{E^{-1}} \mathcal{D}_{in})$ are different, and related as follows:

$$\frac{Pr(\mathcal{D}_{in} \xrightarrow{E} \mathcal{D}_{out})}{|\mathcal{D}_{out}|} = \frac{Pr(\mathcal{D}_{out} \xrightarrow{E^{-1}} \mathcal{D}_{in})}{|\mathcal{D}_{in}|}.$$

2.3 Boomerang Attacks

In 1999, Wagner [35] introduced the boomerang attack which makes use of two differential trails to construct a boomerang trail spanning over a large number of rounds. Then [10] showed a minor change on the boomerang attack. As is shown in the left of Fig. 1, the cipher E is decomposed as two parts: $E = E_1 \circ E_0$. For the upper part E_0, there exists a differential trail $\alpha \xrightarrow{E_0} \beta$ with probability $\overrightarrow{p}$ in the forward direction and a differential trail $\beta \xrightarrow{E_0^{-1}} \alpha^*$ with probability $\overleftarrow{p}$ in the backward direction. For the lower part E_1, there exists a differential trail $\gamma \xrightarrow{E_1} \delta$ with probability q in the forward direction. The boomerang process is as follows:

1. Choose plaintext pairs (P_1, P_2) such that $P_1 + P_2 = \alpha$, and ask for the corresponding ciphertext pairs (C_1, C_2).
2. Compute $C_3 = C_1 + \delta$ and $C_4 = C_2 + \delta$, and ask for the decryption of (C_3, C_4) to obtain (P_3, P_4).
3. Count the number of pairs (P_3, P_4) such that $P_3 + P_4 = \alpha^*$.

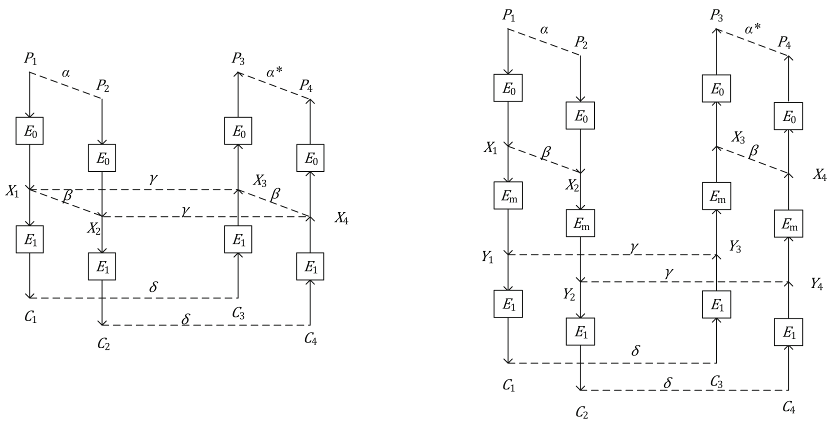


Fig. 1. The boomerang attack (left) and the sandwich attack (right)

Denote that $X_i = E_0(P_i), i = 1, 2, 3, 4$. Since $P_1 + P_2 = \alpha$, we have

$$X_1 + X_2 = \beta \tag{1}$$

with probability $\overrightarrow{p}$. The differential trail $\delta \xrightarrow{E_1^{-1}} \gamma$ holds with probability q, thus $C_1 + C_3 = C_2 + C_4 = \delta$ implies

$$X_1 + X_3 = X_2 + X_4 = \gamma \tag{2}$$

with probability q^2. If Eqs. (1) and (2) hold, we have

$$X_3 + X_4 = X_1 + X_2 = \beta$$

with probability 1, and then $P_3 + P_4 = \alpha^*$ holds with probability $\overleftarrow{p}$. Assuming that all these events are independent, the probability of the boomerang trail is $P_B = \overrightarrow{p}\overleftarrow{p}q^2$. For a random permutation, denote by P_R the probability of $P_3 + P_4 = \alpha^*$. If $P_B > P_R$, we take P_B^{-1} plaintext pairs (P_1, P_2) such that $P_1 + P_2 = \alpha$ to distinguish E from a random permutation. If the number of (P_3, P_4) such that $P_3 + P_4 = \alpha^*$ is greater than 0, the distinguishing result is "E", otherwise it is "a random permutation". The distinguisher succeeds if there is one plaintext pair following the boomerang trail, of which the probability is $1 - (1 - P_B)^{P_B^{-1}} \approx 1 - e^{-1} \approx 63\%$.

Sandwich Attack. To further study the dependence and the connectivity of upper and lower differentials in the boomerang attack, Dunkelman et al. [20] proposed the sandwich attack. As is shown in the right of Fig. 1, the cipher E is split into three parts $E = E_1 \circ E_m \circ E_0$, and the connection probability r of E_m is introduced:

$$r = \left(E_m^{-1}\left(E_m(X_1) + \gamma\right) + E_m^{-1}\left(E_m\left(X_1 + \beta\right) + \gamma\right) = \beta\right).$$

Then the probability of the boomerang trail is $P_B = \overrightarrow{p}\overleftarrow{p}q^2 r$. The connection probability r can be estimated theoretically and experimentally. In [14], Cid et al. used the Boomerang Connectivity Table (BCT) to analyze the case where E_m is a single S-Box layer. In [17,33,36], researchers studied the case where E_m is composed of several rounds. In [37], Yang et al. introduced the Double Boomerang Connectivity Table (DBCT) and showed that the relation between neighboring rounds cannot be ignored.

Retracing Boomerang Attack. In EUROCRYPT 2020, Dunkelman et al. [19] changed the way of generating ciphertext pairs in boomerang attacks and presented the retracing boomerang attack. Instead of adding a fixed difference δ to (C_1, C_2) to obtain (C_3, C_4), they constructed (C_3, C_4) depended on (C_1, C_2) such that if (C_1, C_2) follows the differential trail for E_1, (C_3, C_4) follows it too. This reduces the boomerang probability by a factor of q, from $\overrightarrow{p}\overleftarrow{p}q^2 r$ to $\overrightarrow{p}\overleftarrow{p}qr$.

Truncated Boomerang Attack. In EUROCRYPT 2023, Bariant et al. [6] replaced all differential trails in boomerang attacks by truncated differential trails to propose truncated boomerang attacks. Truncated boomerang attacks use structures on both plaintext and ciphertext sides, which can reduce the complexity effectively. Applied to 6-round AES, the truncated boomerang distinguisher has data and time complexities 2^{87}.

2.4 Exchange Attacks

In ASIACRYPT 2019, Bardeh et al. [5] used the 4-round exchange-difference relation of AES given in [32] to propose exchange attacks and showed the best 5- and 6-round secret-key chosen-plaintext distinguishers for AES. The 4-round exchange-difference relation shows that when exchanging one or more diagonals of a plaintext pair to obtain another plaintext pair, two corresponding ciphertext pairs after 4-round AES encryption have the same zero-difference inverse diagonals. Bardeh et al. analyzed the probability that exchanging one or more diagonals between a plaintext pair will result in the exchange of diagonals after one-round AES encryption. Combined 4-round exchange-difference relation with the probability analysis, Bardeh et al. presented a chosen-plaintext exchange distinguisher on 5-round AES with complexity 2^{30}. The 6-round exchange attack for AES is a straight-forward extension of 5-round attack by adding one round at the end and has complexity $2^{88.2}$.

In decryption direction 4-round exchange-difference relation holds as well by applying an appropriate exchange operation considering the corresponding linear layer. In [4], Bardeh transformed the chosen-plaintext exchange attacks into attacks in the adaptively chosen ciphertexts setting by performing the appropriate exchange operation on the ciphertext sides. Besides, Bardeh added the limitation on ciphertext differences. The 5-round exchange attack in the adaptively chosen ciphers setting has data complexity of 2^{39} and time complexity of $2^{35.5}$, and the 6-round attack has data complexity of 2^{84} and time complexity of 2^{83}.

3 The Re-boomerang Distinguisher

In this section, we extend the classical boomerang distinguisher which uses only one boomerang property to combine two related boomerangs to enhance the distinguishing effect, which is called the re-boomerang distinguisher. By the first boomerang, a small target set of plaintext pairs containing one right pair can be obtained. Then for each pair in the target set, construct its 'friend pairs' and input to the second boomerang to distinguish the cipher from a random permutation. A framework of the re-boomerang distinguisher is given in Subsect. 3.1. Two exchanged boomerang trails of 6-round AES are introduced in Subsect. 3.2, then the re-boomerang distinguisher for 6-round AES with the complexity of $2^{82.33}$ is given in Subsect. 3.3.

3.1 Framework of the Re-boomerang Distinguisher

Assume the cipher E is decomposed as $E = E_1 \circ E_m \circ E_0$, and there exist two truncated boomerang trails B_1 and B_2, with probabilities of P_{B_1} and P_{B_2} respectively. For the upper part E_0, B_1 and B_2 have the same truncated differential trail in the forward direction $\mathcal{D}_{in} \xrightarrow{E_0} \mathcal{D}_{out}$, of which the probability is denoted by $\overrightarrow{p}$. A plaintext pair (P_1, P_2) is called a **right pair** if it satisfies $P_1 + P_2 \in \mathcal{D}_{in}$ and $E_0(P_1) + E_0(P_2) \in \mathcal{D}_{out}$. Denote by P_{R_1} and P_{R_2} the probabilities that a random pair satisfies the boomerang properties of B_1 and B_2, respectively. If $P_{B_2} > P_{R_2}$, we can use B_2 to distinguish the cipher E from a random permutation with $P_{B_2}^{-1}$ plaintext pairs. We will show that by the related boomerang B_1 and the technique of 'friend pairs', P_{B_2} can be increased by a factor of $\overrightarrow{p}^{-1}$. Then plaintext pairs needed to distinguish the cipher E will be reduced to $l \cdot \overrightarrow{p} \cdot P_{B_2}^{-1}$, where l is the size of the target set containing one right pair by the boomerang B_1. We need $l < \overrightarrow{p}^{-1}$ in order to ensure $l \cdot \overrightarrow{p} \cdot P_{B_2}^{-1} < P_{B_2}^{-1}$.

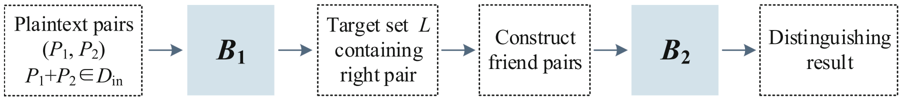


Fig. 2. Framework of the re-boomerang distinguisher

The framework of the re-boomerang distinguisher is shown in Fig. 2, and involves the following two steps.

Step 1: Use the boomerang B_1 to obtain a target set L of size $l < \overrightarrow{p}^{-1}$, containing one right pair on average.

Choose $P_{B_1}^{-1}$ plaintext pairs (P_1, P_2) such that $P_1 + P_2 \in \mathcal{D}_{in}$. For each pair perform the first boomerang B_1 to obtain the returned plaintext pairs (P_3, P_4). If (P_3, P_4) satisfies the boomerang property of B_1, then save the corresponding (P_1, P_2) to the target set L. There will exist one right pair saved in L on average. Since the probability of a random pair satisfying the boomerang property of B_1 is P_{R_1}, the expected number of plaintext pairs in L is $l = P_{B_1}^{-1} \cdot (P_{B_1} + P_{R_1}) = 1 + P_{B_1}^{-1} P_{R_1}$.

Step 2: For each plaintext pair in L, construct its 'friend pairs' and input to the boomerang B_2 to distinguish the cipher.

For each plaintext pair (P_1, P_2) in the target set L, construct its $\overrightarrow{p} P_{B_2}^{-1}$ 'friend pairs' (P_1', P_2') in the following way. For the i-th byte, if $P_{1,i} + P_{2,i} \neq 0$, then take $P_{1,i}' = P_{1,i}$ and $P_{2,i}' = P_{2,i}$; if $P_{1,i} + P_{2,i} = 0$, then take $P_{1,i}' = P_{2,i}'$ for any value except $P_{1,i}$. Input all $l \cdot \overrightarrow{p} P_{B_2}^{-1}$ pairs to the boomerang B_2. If there is a pair satisfying the boomerang property of B_2, then it is the cipher E. Otherwise, it is a random permutation. Note that in general there is an upper bound for the number of active bytes of $P_1 + P_2$ so that the required $\overrightarrow{p} P_{B_2}^{-1}$ friend pairs can be constructed. For example, if a pair of AES plaintexts (P_1, P_2) is active in m bytes, then the number of its all friend pairs is $(2^8 - 1)^{16-m}$. To make $(2^8 - 1)^{16-m} \geq \overrightarrow{p} P_{B_2}^{-1}$, we have $m \leq 16 - 8^{-1} log_2 \overrightarrow{p} P_{B_2}^{-1}$. 'Friend pairs' (P_1', P_2') have the same truncated differential trail with (P_1, P_2) in the forward direction of E_0. If (P_1, P_2) is a right pair, then all its 'friend pairs' (P_1', P_2') are right pairs. The probability of right pairs satisfying B_2 is increased from P_{B_2} to $\overrightarrow{p}^{-1} P_{B_2}$. Since L contains one right pair on average, there will be $\overrightarrow{p} P_{B_2}^{-1}$ right pairs obtained by the construction of 'friend pairs'. Thus there exists one pair (P_1', P_2') satisfying the boomerang property of B_2 on average. For a random permutation, the number of pairs satisfying the boomerang property of B_2 is $l \cdot \overrightarrow{p} P_{B_2}^{-1} P_{R_2} < 1$.

Complexity. The data and time complexities of Step 1 are $4P_{B_1}^{-1}$, which may be reduced by using plaintext structure. The data and time complexities of Step 2 are $4 \cdot l \cdot \overrightarrow{p} P_{B_2}^{-1} = 4 \cdot \left(1 + P_{B_1}^{-1} P_{R_1}\right) \cdot \overrightarrow{p} P_{B_2}^{-1}$.

3.2 Exchanged Boomerangs for 6-Round AES

We combine the truncated boomerang [6] with exchange technique [4,5] to construct new boomerangs, and present two boomerang trails of 6-round AES which will be used in the re-boomerang distinguisher. Decompose 6-round AES as three parts $E_1 \circ E_m \circ E_0$, where

$$E_0 = SR \circ SB \circ AK \circ MC \circ SR \circ SB \circ AK \circ MC \circ SR \circ SB \circ AK$$

is the upper 2.5 rounds before MC of the third round,

$$E_m = AK \circ MC \circ SR \circ SB \circ AK \circ MC$$

is the middle 1.5 rounds, and

$$E_1 = AK \circ SR \circ SB \circ AK \circ MC \circ SR \circ SB$$

is the final 2 rounds, as shown in Fig. 3. Let (P_1, P_2) be a pair of plaintexts and (C_1, C_2) the corresponding ciphertexts encrypted by 6-round AES. Denote by (C_3^j, C_4^j), $1 \leq j \leq 4$, ciphertext pairs generated from (C_1, C_2), and (P_3^j, P_4^j) their corresponding plaintext pairs. Denote by X the intermediate state after E_0, and Y the intermediate state after E_m.

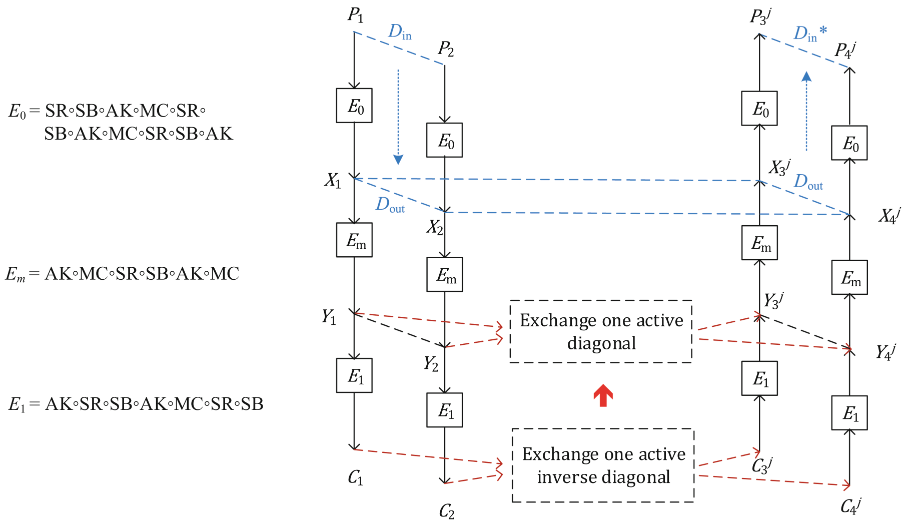

E_0 = SR∘SB∘AK∘MC∘SR∘SB∘AK∘MC∘SR∘SB∘AK
E_m = AK∘MC∘SR∘SB∘AK∘MC
E_1 = AK∘SR∘SB∘AK∘MC∘SR∘SB
P_1
P_2
D_{in}
P_3^j
P_4^j
$D_{in}*$
E_0
X_1
X_2
X_3^j
X_4^j
D_{out}
E_m
Y_1
Y_2
Y_3^j
Y_4^j
E_1
C_1
C_2
C_3^j
C_4^j
Exchange one active diagonal
Exchange one active inverse diagonal

Fig. 3. Exchanged boomerang for 6-round AES

Truncated Boomerang for E_0. In the forward direction, let $\mathcal{D}_{in}$ be the input truncated difference, and $\mathcal{D}_{out}$ the output truncated difference of E_0. Denote by $\overrightarrow{p}$ the probability of $\mathcal{D}_{in} \xrightarrow{E_0} \mathcal{D}_{out}$. In the backward direction we consider a different truncated difference $\mathcal{D}_{in}^*$ for returned plaintext pairs and denote by $\overleftarrow{p}$ the probability of $\mathcal{D}_{out} \xrightarrow{E_0^{-1}} \mathcal{D}_{in}^*$.

Exchange Ciphertexts in E_1. We exchange the inverse diagonal of ciphertext pairs so that the probability $q = 1$ for E_1. Choose ciphertext pair (C_1, C_2) such that $C_1 + C_2$ has t inactive inverse diagonals, $t = 0$ or 1, where the probability is $\binom{4}{t}2^{-32t}$. Then exchange one active inverse diagonal of (C_1, C_2) to obtain $4 - t$ ciphertext pairs (C_3^j, C_4^j), $j \in \{1, 2, ..., 4 - t\}$. Note that E_1 can be seen as four super S-boxes applied in parallel. If (C_3^j, C_4^j) is obtained by exchanging $SR(Col(i))$ of (C_1, C_2), then (Y_3^j, Y_4^j) can be regarded as exchanging $SR^{-1}(Col(i))$ of (Y_1, Y_2), $0 \le i \le 3$, and $Y_3^j + Y_4^j$ have the same t inactive diagonals as $Y_1 + Y_2$, $j \in \{1, 2, ..., 4 - t\}$.

Connection Probability for E_m. The connection probability r is the probability that there exists $j \in \{1, 2, ..., 4 - t\}$ such that $X_3^j + X_4^j \in \mathcal{D}_{out}$. Theorem 1 gives the lower bound of r, and its proof is shown in Appendix.

Theorem 1. *Let E_m and $\mathcal{D}_{out}$ be defined as above, (X_1, X_2) an input pair of E_m such that $X_1 + X_2 \in \mathcal{D}_{out}$, and (Y_1, Y_2) the corresponding output pair such that $Y_1 + Y_2$ is inactive in t diagonals, $t = 0$ or 1. Let (Y_3^j, Y_4^j) be the pairs by exchanging one active diagonal of (Y_1, Y_2), and (X_3^j, X_4^j) the corresponding output pairs after E_m^{-1}, $j \in \{1, 2, ..., 4 - t\}$. Then the probability r that there exists $j \in \{1, 2, ..., 4 - t\}$ such that $X_3^j + X_4^j \in \mathcal{D}_{out}$ satisfies*

$$r \ge (4 - t) \cdot \sum_{d=1}^{3} \binom{4}{d} \cdot (2^{-8})^{4+(2-t)\cdot d}.$$

The process of the exchanged boomerang is as follows:

1. Choose plaintext pairs (P_1, P_2) such that $P_1 + P_2 \in \mathcal{D}_{in}$, and ask for the corresponding ciphertext pairs (C_1, C_2).
2. Filter the ciphertext pairs (C_1, C_2) such that $C_1 + C_2$ is inactive in t inverse diagonals, and then exchange one active inverse diagonal of (C_1, C_2) to obtain $4 - t$ ciphertext pairs (C_3^j, C_4^j), $0 \le t \le 1, j \in \{1, 2, ..., 4 - t\}$.
3. Ask for the decryption of (C_3^j, C_4^j) to obtain returned plaintext pairs (P_3^j, P_4^j), and check whether there exists $j \in \{1, 2, ..., 4 - t\}$ such that $P_3^j + P_4^j \in \mathcal{D}_{in}^*$.

The probability of the exchanged boomerang is $P_B = \overrightarrow{p}\overleftarrow{p}r \cdot \binom{4}{t}2^{-32t}$, where $\overrightarrow{p}$ is the probability of $\mathcal{D}_{in} \xrightarrow{E_0} \mathcal{D}_{out}$, $\overleftarrow{p}$ is the probability of $\mathcal{D}_{out} \xrightarrow{E_0^{-1}} \mathcal{D}_{in}^*$, r is the connection probability of E_m, and $\binom{4}{t}2^{-32t}$ is the probability that $C_1 + C_2$ is inactive in t inverse diagonals, $0 \le t \le 1$. The following are two exchanged boomerang trails of 6-round AES, which will be used in the re-boomerang distinguisher.

The First Boomerang Trail B_1. For the upper part E_0, the input truncated difference $\mathcal{D}_{in}$ is active only in $SR^{-1}(Col(0))$, the output truncated difference $\mathcal{D}_{out}$ is active in only one inverse diagonal, and the truncated difference for returned plaintext pairs $\mathcal{D}_{in}^*$ is active in only two diagonals, as shown in Fig. 4. The probability of $\mathcal{D}_{in} \xrightarrow{E_0} \mathcal{D}_{out}$ is $\overrightarrow{p} = 4 \times 2^{-24} = 2^{-22}$, since it holds if and only if the difference before MC of the first round is active in only one byte. The probability of $\mathcal{D}_{out} \xrightarrow{E_0^{-1}} \mathcal{D}_{in}^*$ is $\overleftarrow{p} = 6 \times 2^{-16} = 2^{-13.42}$, since it holds if and only if the difference before MC^{-1} of the second round is active in only two bytes. For the lower part E_1, the trail in the backward direction is shown in Fig. 5. We filter ciphertext pairs (C_1, C_2) such that $C_1 + C_2$ have $t = 1$ inactive inverse diagonal, of which the probability is $\binom{4}{t}2^{-32t} = 4 \times 2^{-32} = 2^{-30}$, and then exchange one of the other three inverse diagonals of (C_1, C_2) to obtain ciphertext pairs (C_3^j, C_4^j), $j \in \{1, 2, 3\}$. For the middle part E_m, we have $r \geq (4-t) \cdot \sum_{d=1}^{3} \binom{4}{d} \cdot (2^{-8})^{4+(2-t)\cdot d} \approx 2^{-36.42}$ from Theorem 1. The probability of the first boomerang trail B_1 is $P_{B_1} = \overrightarrow{p}\,\overleftarrow{p}\,r \cdot \binom{4}{t}2^{-32t} \approx 2^{-101.84}$.

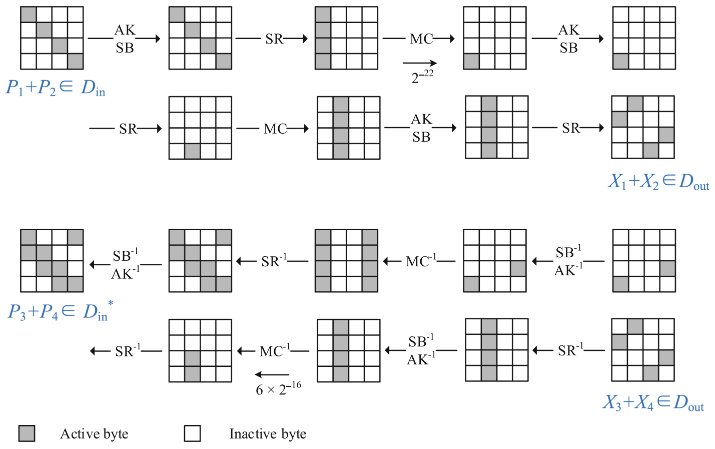


Fig. 4. The truncated boomerang for E_0 in the first boomerang

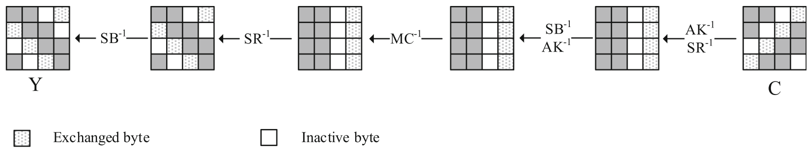


Fig. 5. The trail for E_1 in the first boomerang

The Second Boomerang Trail B_2. For the upper part E_0, the truncated differences $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$ are the same as them in the first boomerang trail B_1, and the probability of $\mathcal{D}_{in} \xrightarrow{E_0} \mathcal{D}_{out}$ is $\overrightarrow{p} = 2^{-22}$. The truncated difference for returned pairs $\mathcal{D}^*_{in}$ is active in only one diagonal. The probability of $\mathcal{D}_{out} \xrightarrow{E_0^{-1}} \mathcal{D}^*_{in}$ is $\overleftarrow{p} = 4 \times 2^{-24} = 2^{-22}$, since it holds if and only if the difference before MC^{-1} of the second round is active in only one byte. For the lower part E_1, we take $t = 0$. For each (C_1, C_2), exchange one active inverse diagonal to obtain ciphertext pairs (C_3^j, C_4^j), $j \in \{1, 2, 3, 4\}$. For the middle part E_m, we have $r \geq (4-t) \cdot \sum_{d=1}^{3} \binom{4}{d} \cdot (2^{-8})^{4+(2-t)\cdot d} \approx 2^{-44}$ from Theorem 1. The probability of the second boomerang trail B_2 is $P_{B_2} = \overrightarrow{p}\overleftarrow{p}r \approx 2^{-88}$.

Experimental Verification. We mounted an experiment to verify the connection probability r in the second boomerang trail B_2. Randomly take 3×2^{45} pairs of states (X_1, X_2) such that $X_1 + X_2$ are active in $SR(Col(0))$. Calculate $Y_1 = E_m(X_1)$ and $Y_2 = E_m(X_2)$ and exchange one active diagonal of (Y_1, Y_2) to obtain (Y_3^j, Y_4^j), $j \in \{1, 2, 3, 4\}$. Calculate $X_3^j = E_m^{-1}(Y_3^j)$, $X_4^j = E_m^{-1}(Y_4^j)$ and check whether $X_3^j + X_4^j = X_1 + X_2$ holds. There exist 6 pairs satisfying the condition, so the experimental probability of r is 2^{-44}, which matches the theoretical estimation. The program is written in C++ including the Intel AES-NI instruction set, which is partitioned to 24 subprograms and was performed on PC about 36 days. The CPU is Intel(R) Core(TM) i7-9700@3.00 GHz and the RAM is 32 GB.

3.3 The Re-boomerang Distinguisher for 6-Round AES

We show the detailed process of re-boomerang distinguisher for 6-round AES in this subsection. The distinguisher has the data complexity and time complexity of $2^{82.33}$ and success probability 64%.

For the first boomerang B_1, we need $(P_{B_1})^{-1} \approx 2^{101.84}$ plaintext pairs to obtain a target set L containing one right pair. Choosing $2^{38.84}$ plaintext structures in which the four bytes in $SR^{-1}(Col(0))$ take all possible values and the rest bytes are any constants, we can obtain $2^{38.84} \times 2^{32} \times 2^{31} = 2^{101.84}$ plaintext pairs. Note that B_1 requires that $C_1 + C_2$ is inactive in one inverse diagonal and there exists a returned pair (P_3, P_4) active in only two diagonals. The probability for a random pair satisfying the requirement is $P_{R_1} = 2^{-30} \times 3 \times 6 \times 2^{-64} \approx 2^{-89.84}$. So the number of plaintext pairs (P_1, P_2) in L is $l = 1 + P_{B_1}^{-1} P_{R_1} \approx 1 + 2^{12}$. For each pair (P_1, P_2) in L, we construct its $\overrightarrow{p} P_{B_2}^{-1} = 2^{-22} \times 2^{88} = 2^{66}$ 'friend pairs' (P_1', P_2'). Then the number of pairs inputted to B_2 is $l \cdot 2^{66} \approx 2^{78}$. For 6-round AES, there exists one pair satisfying B_2. The probability of a random pair satisfying the boomerang property of B_2 is $P_{R_2} = 4 \times 4 \times 2^{-96} = 2^{-92}$. So for a random permutation, there are $2^{78} \times 2^{-92} = 2^{-14} < 1$ pairs satisfying B_2. The re-boomerang distinguisher for 6-round AES is as follows, and the pseudocode is given in Algorithm 1.

1. Choose $2^{38.84}$ plaintext structures of size 2^{32} in which the four bytes in $SR^{-1}(Col(0))$ take all possible values and the rest bytes are any constants, and ask for the corresponding ciphertexts.

2. For each structure, insert 2^{32} ciphertexts into a hash table indexed by $SR(Col(i))$, and extract all ciphertext pairs (C_1, C_2) such that $(C_1 + C_2)_{SR(Col(i))} = 0$, $i = 0, 1, 2, 3$.
3. For each $j \in \{0, 1, 2, 3\} \setminus i$, exchange the j-th inverse diagonal of (C_1, C_2) to obtain (C_3, C_4), and ask for the decryption of (C_3, C_4) to obtain (P_3, P_4). If there exists one (P_3, P_4) such that $P_3 + P_4$ is active in only two diagonals, we store the corresponding plaintext pairs (P_1, P_2) to the set L.
4. For each (P_1, P_2) in L, construct 2^{66} 'friend pairs' (P'_1, P'_2) such that $P'_{1,SR^{-1}(Col(0))} = P_{1,SR^{-1}(Col(0))}$, $P'_{2,SR^{-1}(Col(0))} = P_{2,SR^{-1}(Col(0))}$, and in the other bytes P'_1 and P'_2 take any equal values except the value of P_1. Ask for the encryption of (P'_1, P'_2) to obtain (C'_1, C'_2).
5. Filter (C'_1, C'_2) such that $C'_1 + C'_2$ are active in four inverse diagonals. For each (C'_1, C'_2) and each $j \in \{0, 1, 2, 3\}$, we exchange the j-th inverse diagonal of (C'_1, C'_2) to obtain (C'_3, C'_4), and decrypt (C'_3, C'_4) to obtain (P'_3, P'_4). If there exists one pair (P'_3, P'_4) such that $P'_3 + P'_4$ is active in only one diagonal, the distinguishing result is "6-round AES", otherwise it is "a random permutation".

Complexity Analysis. In step 1, we need $2^{38.84} \times 2^{32} = 2^{70.84}$ chosen plaintexts. In step 2, the time complexity is $2^{38.84} \times 2^{32} \times 4 + 2^{38.84} \times 2^{63} \times 2^{-30} \approx 2^{73.42}$ memory accesses, and the number of (C_1, C_2) which are inactive in one inverse diagonal is $2^{38.84} \times 2^{63} \times 2^{-30} = 2^{71.84}$. In step 3 we need $2^{71.84} \times 3 \times 2 \approx 2^{74.42}$ adaptively chosen ciphertexts. In step 4, we need $(1 + 2^{12}) \times 2^{66} \times 2 \approx 2^{79}$ adaptively chosen plaintexts. In step 5, filtering (C'_1, C'_2) needs $(1+2^{12}) \times 2^{66} \times 4 \approx 2^{80}$ comparisons. The probability that a pair (C'_1, C'_2) is active in four inverse diagonals is $(1-2^{-32})^4$, which is close to 1, then we need $(1+2^{12}) \times 2^{66} \times 4 \times 2 \approx 2^{81}$ adaptively chosen ciphertexts. Therefore, the data complexity of a distinguishing process is $2^{70.84}$ CP $+2^{74.42}$ ACC $+2^{79}$ ACP $+2^{81}$ ACC $\approx 2^{81.33}$ ACPC, and the time complexity is $2^{81.33}$.

The distinguisher succeeds if two boomerangs B_1 and B_2 are both satisfied. The probability of B_1 being satisfied is

$$ps_1 = 1 - (1 - P_{B_1})^{P_{B_1}^{-1}} \approx 1 - e^{-1},$$

and the probability of B_2 being satisfied is

$$ps_2 = 1 - (1 - \overrightarrow{p}^{-1} P_{B_2})^{\overrightarrow{p} P_{B_2}^{-1}} \approx 1 - e^{-1}.$$

Then the success probability of a re-boomerang distinguisher is $ps_1 \cdot ps_2 \approx (1 - e^{-1})^2 \approx 40\%$. To improve the success probability, we can repeat this process twice. Once there exists one output result of "6-round AES", the final distinguishing result is "6-round AES". In this way the success probability of the distinguisher can be improved to $P_s = 1 - (1 - ps_1 \cdot ps_2)^2 \approx 64\%$. Then the total data and time complexities of the distinguisher are both $2^{81.33} \times 2 = 2^{82.33}$. The type-II error probability (the probability to wrongfully accept a random permutation as 6-round AES) is the probability that there exists one pair (P'_1, P'_2)

Algorithm 1: The re-boomerang distinguisher for 6-round AES

```
/* the re-boomerang process is repeated 2 times */
for $1 \le t \le 2$ do
    /* the first boomerang begins */
    Ask for the encryption of $2^{38.84}$ plaintext structures in which the four bytes
    in $SR^{-1}(Col(0))$ take all possible values and the rest bytes are any
    constants;
    for each plaintext structure do
        for each $i \in \{0, 1, 2, 3\}$ do
            Insert $2^{32}$ ciphertexts $C$ into a hash table indexed by $C_{SR(Col(i))}$;
            for each ciphertext pair $(C_1, C_2)$ such that $(C_1 + C_2)_{SR(Col(i))} = 0$
            do
                Exchange the $j$-th inverse diagonal of $(C_1, C_2)$ to obtain 3 pairs
                $(C_3, C_4)$, $j \in \{0, 1, 2, 3\} \setminus i$;
                Decrypt 3 pairs $(C_3, C_4)$ to obtain 3 pairs $(P_3, P_4)$;
                if there exists one $(P_3, P_4)$ such that $P_3 + P_4$ is active in only
                two diagonals then
                    Save the corresponding $(P_1, P_2)$ to the set $L$.
                end
            end
        end
    end
    /* the second boomerang begins */
    for each plaintext pair $(P_1, P_2)$ in $L$ do
        for $1 \le k \le 2^{66}$ do
            Construct one 'friend pair' $(P_1', P_2')$ of $(P_1, P_2)$;
            Ask for the encryption of $(P_1', P_2')$ to obtain $(C_1', C_2')$;
            Exchange one active inverse diagonal of $(C_1', C_2')$ to obtain 4 pairs
            $(C_3', C_4')$;
            Decrypt 4 pairs $(C_3', C_4')$ to obtain 4 pairs $(P_3', P_4')$;
            if there exists one $(P_3', P_4')$ such that $P_3' + P_4'$ is active in only one
            diagonal then
                Return: This is 6-round AES.
            end
        end
    end
end
Return: This is a random permutation.
```

satisfying the boomerang property of B_2, which is

$$1 - (1 - 2^{-92})^{2^{78} \times 2} \approx 1 - e^{-2^{-13}} \approx 2^{-13}.$$

4 The Boomerang Chain Distinguishers

In this section, We improve the re-boomerang distinguisher by inserting a new boomerang trail B_m of 6-round AES in the middle of B_1 and B_2, and repeating it to form the boomerang chain distinguisher. After obtaining the target set L of plaintext pairs by B_1, we use B_m to filter the wrong pairs in L gradually. Then the input data of B_2 is reduced and the complexity is improved. The triple boomerangs distinguisher, which is the simplest boomerang chain distinguisher, is presented in Subsect. 4.1. The general boomerang chain distinguisher is introduced in Subsect. 4.2. We give a 6-round distinguisher with success probability 60% and complexity $2^{76.57}$. This is a new record in distinguishers on 6-round AES.

4.1 The Triple Boomerangs Distinguisher

The Middle Boomerang Trail B_m. For the upper part E_0, the truncated boomerang trail of B_m is the same as it in B_1. For the middle part E_m and lower part E_1, the trail of B_m is the same as it in B_2. The probability of B_m is $P_{B_m} = \overrightarrow{p}\overleftarrow{p}r \approx 2^{-22} \times 2^{-13.42} \times 2^{-44} = 2^{-79.42}$.

We insert B_m between B_1 and B_2 to obtain the triple boomerangs distinguisher for 6-round AES, as shown in Fig. 6. The pseudocode is given in Algorithm 2.

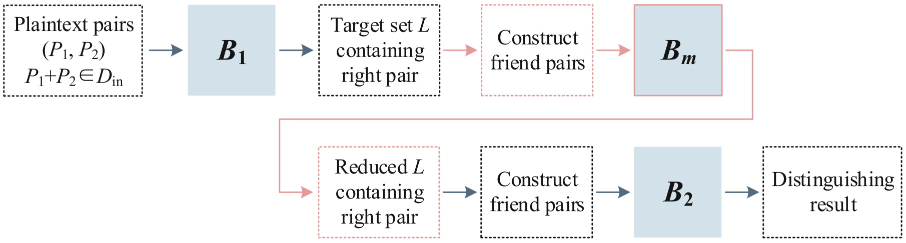


Fig. 6. Framework of the triple boomerangs distinguisher

For each plaintext pair (P_1, P_2) in L, construct its $\overrightarrow{p}P_{B_m}^{-1} = 2^{57.42}$ 'friend pairs' (P_1', P_2') as the input of B_m and perform the middle boomerang to obtain returned plaintext pairs (P_3', P_4'). For one plaintext pair (P_1', P_2') there are 4 returned pairs (P_3', P_4'). If there is no (P_3', P_4') which is active in only two diagonals, we delete the corresponding plaintext pair (P_1, P_2) from the set L. When (P_1, P_2) is a right pair, its 'friend pairs' (P_1', P_2') are all right pairs, and among these 'friend pairs' the probability of the middle boomerang trail B_m is increased by a factor of $\overrightarrow{p} = 2^{-22}$, from $P_{B_m} = 2^{-79.42}$ to $\overrightarrow{p}^{-1}P_{B_m} = 2^{-57.42}$. Then for the right pair (P_1, P_2), there exists one 'friend pairs' following B_m on average and the right pair will be kept in L. The probability of a random pair satisfying the boomerang property of B_m is $P_{R_m} = 4 \times 6 \times 2^{-64} \approx 2^{-59.42}$. When (P_1, P_2) is a wrong pair, the expected number of (P_3', P_4') active in only two diagonals is $2^{57.42} \times 2^{-59.42} = 2^{-2} < 1$, then some wrong pairs will be filtered. A wrong pair (P_1, P_2) will be kept in L with probability

$1-(1-2^{-59.42})^{2^{57.42}} \approx 1-e^{-2^{-2}} \approx 2^{-2.18}$, then after the middle boomerang B_m the size of L is $l = 1+2^{12} \times 2^{-2.18} = 1+2^{9.82}$.

To further improve the filtering probability of B_m, we increase the number of 'friend pairs' (P_1', P_2') constructed in the middle boomerang. For each plaintext pair (P_1, P_2) in L, construct $n \cdot \left(\overrightarrow{p} P_{B_m}^{-1}\right) = 2^{57.42}n$ 'friend pairs' (P_1', P_2') and perform the middle boomerang to obtain returned plaintext pairs (P_3', P_4'), $n \geq 1$. If the number of pairs (P_3', P_4') active in only two diagonals is less than n, we delete the corresponding plaintext pair (P_1, P_2) from the target set L. For the right pair (P_1, P_2), the probability that a returned pair (P_3', P_4') is active in two diagonals is $2^{-57.42}+2^{-59.42} = 2^{-57.10}$. Since $2^{57.42}n \cdot 2^{-57.10} \approx 1.25n > n$, then the right pair will be kept. For a wrong pair (P_1, P_2), the expected number of returned pairs (P_3', P_4') active in two diagonals is $2^{57.42}n \cdot 2^{-59.42} = 0.25n < n$. A wrong pair (P_1, P_2) will be kept in L with probability

$$p_f(n) = 1 - \sum_{k=0}^{n-1} \binom{2^{57.42}n}{k} \cdot (2^{-59.42})^k \cdot (1-2^{-59.42})^{2^{57.42}n-k}. \tag{3}$$

Then after filtering the size of L is $1+2^{12}p_f(n)$.

The middle boomerang process is as follows, which can be inserted between step 3 and step 4 of the re-boomerang process in Subsect. 3.3 to obtain the triple boomerangs distinguisher.

1. For each plaintext pair (P_1, P_2) in L, construct its $2^{57.42}n$ 'friend pairs' (P_1', P_2'), and ask for the encryption of (P_1', P_2') to obtain (C_1', C_2').
2. For each $j \in \{0,1,2,3\}$, exchange the j-th inverse diagonal of (C_1', C_2') to obtain (C_3', C_4'), and ask for the decryption of (C_3', C_4') to obtain (P_3', P_4'). If the number of (P_3', P_4') active in only two diagonals is less than n, we delete the corresponding plaintext pair (P_1, P_2) from L.

Complexity Analysis. We recall that the data complexity of the first boomerang is $D_1 = 2^{71.84}+2^{74.42} \approx 2^{74.64}$. In the middle boomerang process, the data complexity is $D_2 = (1+2^{12}) \cdot 2^{57.42}n \cdot 2 \cdot (1+4) \approx 2^{72.74}n$. After the middle boomerang process, the number of plaintext pairs in L is $1+2^{12}p_f(n)$. Then the complexity of the second boomerang is $D_3 = \left(1+2^{12}p_f(n)\right) \cdot 2^{66} \cdot 2 \cdot (1+4) \approx \left(1+2^{12}p_f(n)\right) \cdot 2^{69.32}$. The time and data complexities of the triple boomerangs distinguishing process is $T = D = D_1 + D_2 + D_3$. The triple boomerangs process can distinguish 6-round AES successfully if the boomerangs B_1, B_m and B_2 are all satisfied. B_1 and B_2 are satisfied with probabilities $ps_1 \approx 1-e^{-1}$ and $ps_2 \approx 1-e^{-1}$ respectively. The probability that B_m is satisfied with probability

$$ps_m(n) = 1 - \sum_{k=0}^{n-1} \binom{2^{57.42}n}{k} \cdot (2^{-57.10})^k \cdot (1-2^{-57.10})^{2^{57.42}n-k}. \tag{4}$$

The success probability of the triple boomerangs process is $ps_1 \cdot ps_m(n) \cdot ps_2 \approx (1-e^{-1})^2 \cdot ps_m(n) \leq (1-e^{-1})^2 \approx 40\%$. To improve the success probability, we repeat the triple boomerangs process several times in a distinguisher. Once there exists one output of “6-round AES”, the final distinguishing result is “6-round AES”. Denote by w the number of times the triple boomerangs process is repeated in a distinguisher. The time and data complexities of the distinguisher are $T = D = w \cdot (D_1 + D_2 + D_3)$. The success probability of the distinguisher is

$$P_s = 1 - \left[1 - \left(1 - e^{-1}\right)^2 \cdot ps_m(n)\right]^w,$$

where $ps_m(n)$ is given by Eqs. (4). When the triple boomerangs process is repeated w times, the number of input plaintext pairs (P'_1, P'_2) in the second boomerang is $w \cdot \left(1 + 2^{12} p_f(n)\right) \cdot 2^{66}$ totally. The type-II error probability of the triple boomerang distinguisher is the probability that there exists one pair (P'_1, P'_2) satisfying the boomerang property of B_2, which is

$$1 - (1 - 2^{-92})^{w \cdot \left(1+2^{12} p_f(n)\right) \cdot 2^{66}} < 1 - \left(1 - 2^{-92}\right)^{2^{78} w} \approx 1 - e^{-2^{-14} w}.$$

In general, we have $w < 6$, then the type-II error probability can be ignored.

For $n = 1, 2, ..., 16$, we calculate the values of w such that the corresponding success probability $P_s > 60\%$, and the data and time complexities by PC program. The results are shown in Table 2. When $n = 6$, we repeat the triple boomerangs process 3 times to obtain the distinguisher with data and time complexities of $2^{77.82}$, and the success probability $P_s = 66\%$.

Table 2. The parameter w, complexities and success probability of the triple boomerangs distinguisher

n	w	$D = T$	Success Probability	n	w	$D = T$	Success Probability
1	3	$2^{80.81}$	63%	2	3	$2^{79.65}$	63%
3	3	$2^{78.79}$	64%	4	3	$2^{78.21}$	65%
5	3	$2^{77.91}$	65%	6	3	$2^{77.82}$	66%
7	3	$2^{77.83}$	67%	8	3	$2^{77.90}$	67%
9	3	$2^{78.00}$	68%	10	3	$2^{78.10}$	68%
11	3	$2^{78.20}$	69%	12	3	$2^{78.28}$	69%
13	3	$2^{78.37}$	70%	14	3	$2^{78.46}$	70%
15	3	$2^{78.54}$	71%	16	3	$2^{78.61}$	71%

4.2 The General Boomerang Chain Distinguisher

We extend the triple boomerangs distinguisher to the general boomerang chain distinguisher by the following improvements.

- The middle boomerang trail B_m is repeated several times in the middle so that the size of the target set L can be further reduced and the complexity can be improved.

Algorithm 2: The triple boomerangs distinguisher for 6-round AES

```
Determine the parameter $n$;
Determine the parameter $w$, $w \geq 1$;
/* the triple boomerangs process is repeated $w$ times */
for $1 \leq t \leq w$ do
    /* the first boomerang begins */
    Ask for the encryption of $2^{38.84}$ plaintext structures in which the four bytes in
    $SR^{-1}(Col(0))$ take all possible values and the rest bytes are any constants;
    for each plaintext structure do
        for each $i \in \{0,1,2,3\}$ do
            Insert $2^{32}$ ciphertexts $C$ into a hash table indexed by $C_{SR(Col(i))}$;
            for each ciphertext pair $(C_1, C_2)$ such that $(C_1 + C_2)_{SR(Col(i))} = 0$ do
                Exchange the $j$-th inverse diagonal of $(C_1, C_2)$ to obtain 3 pairs
                $(C_3, C_4)$, $j \in \{0,1,2,3\} \setminus i$;
                Decrypt 3 pairs $(C_3, C_4)$ to obtain 3 pairs $(P_3, P_4)$;
                if there exists one $(P_3, P_4)$ such that $P_3 + P_4$ is active in only two
                diagonals then
                    Save the corresponding $(P_1, P_2)$ to the set $L$.
                end
            end
        end
    end
    /* the middle boomerang begins */
    for each plaintext pair $(P_1, P_2)$ in $L$ do
        for $1 \leq k \leq 2^{57.42}n$ do
            Construct one 'friend pair' $(P_1', P_2')$ of $(P_1, P_2)$;
            Ask for the encryption of $(P_1', P_2')$ to obtain $(C_1', C_2')$;
            Exchange one active inverse diagonal of $(C_1', C_2')$ to obtain 4 pairs $(C_3',$
            $C_4')$;
            Decrypt 4 pairs $(C_3', C_4')$ to obtain 4 pairs $(P_3', P_4')$;
        end
        if the total number of $(P_3', P_4')$ active in only two diagonals is less than $n$ then
            Delete the corresponding plaintext pair $(P_1, P_2)$ from the set $L$.
        end
    end
    /* the second boomerang begins */
    for each plaintext pair $(P_1, P_2)$ in $L$ do
        for $1 \leq k \leq 2^{66}$ do
            Construct one 'friend pair' $(P_1', P_2')$ of $(P_1, P_2)$;
            Ask for the encryption of $(P_1', P_2')$ to obtain $(C_1', C_2')$;
            Exchange one active inverse diagonal of $(C_1', C_2')$ to obtain 4 pairs $(C_3',$
            $C_4')$;
            Decrypt 4 pairs $(C_3', C_4')$ to obtain 4 pairs $(P_3', P_4')$;
            if there exists one $(P_3', P_4')$ such that $P_3' + P_4'$ is active in only one
            diagonal then
                Return: This is 6-round AES.
            end
        end
    end
end
Return: This is a random permutation.
```

- Besides middle boomerangs, we increase the data size in the first and second boomerangs B_1 and B_2 to increase the success probability of the distinguisher.

The framework of the general boomerang chain distinguisher is shown in Fig. 7. Denote by s the number of times B_m is repeated, then a boomerang chain consists of $s+2$ boomerang trails, starting from B_1, repeating B_m s times, and ending with B_2. The boomerang chain distinguisher contains the following steps and the pseudocode is given in Algorithm 3.

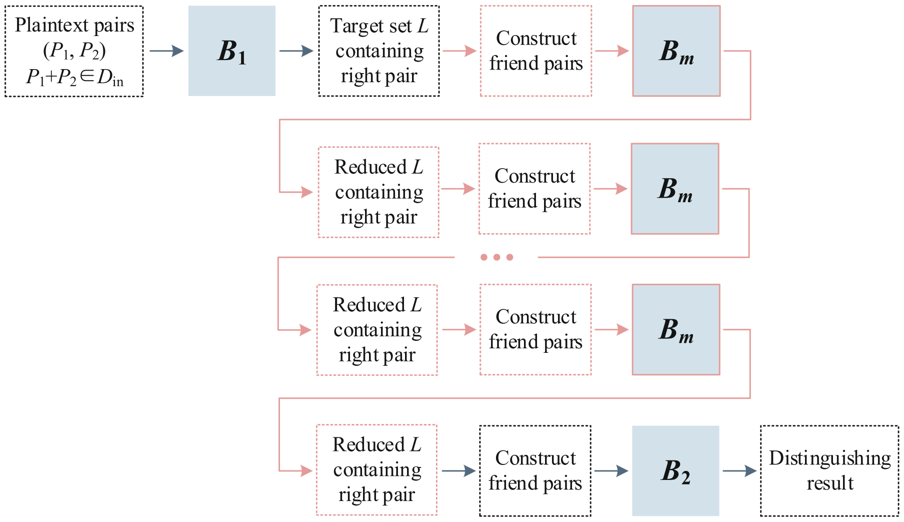

Plaintext pairs (P_1, P_2) $P_1+P_2 \in D_{in}$
B_1
Target set L containing right pair
Construct friend pairs
B_m
Reduced L containing right pair
Construct friend pairs
B_m
Reduced L containing right pair
Construct friend pairs
B_m
Reduced L containing right pair
Construct friend pairs
B_2
Distinguishing result

Fig. 7. Framework of the general boomerang chain distinguisher

1. Choose $2^{38.84}n_1$ plaintext structures to form $2^{101.84}n_1$ plaintext pairs and perform the first boomerang B_1. Then we obtain the target set L of size $(1+2^{12})n_1$. The probability ps_1 that there exists one plaintext pair following B_1 is increased from $1-e^{-1}$ to $1-(1-P_{B_1})^{P_{B_1}^{-1}n_1} \approx 1-e^{-n_1}$. The data complexity of this step is $D_1 = 2^{71.84}n_1 + 2^{74.42}n_1 \approx 2^{74.64}n_1$.
2. For each plaintext pair in L construct its $2^{57.42}n_2$ 'friend pairs' and perform the middle boomerang B_m. Then the size of L is reduced to $\left(1+2^{12}p_f(n_2)\right)n_1$, where $p_f(n)$ is given by Eqs. (3). The right pair will be kept in L with probability $ps_m(n_2)$, where $ps_m(n)$ is given by Eqs. (4). The data complexity of this step is $D_2 = (1+2^{12})\ n_1 \cdot 2^{57.42}n_2 \cdot 2 \cdot (1+4) \approx 2^{60.74}n_1n_2 \cdot \left(1+2^{12}\right)$.
3. When $3 \le i \le s+1$, for each plaintext pair in L construct its $2^{57.42}n_i$ 'friend pairs' and perform the middle boomerang B_m. Then the size of L is reduced to $\left(1+2^{12}p_f(n_2)p_f(n_3)\cdots p_f(n_i)\right)n_1$. The right pair will be kept in L with probability $ps_m(n_i)$. The data complexity of this step is $D_i = (1+2^{12}p_f(n_2)p_f(n_3)\ \cdots\ p_f(n_{i-1}))\ n_1 \cdot 2^{57.42}n_i \cdot 2 \cdot (1+4) \approx 2^{60.74}n_1n_i \cdot \left(1+2^{12}p_f(n_2)p_f(n_3)\cdots p_f(n_{i-1})\right)$.

4. For each plaintext pair in L, construct $2^{66}n_{s+2}$ 'friend pairs' and perform the second boomerang B_2 to distinguish 6-round AES. The probability ps_2 that B_2 is satisfied is increased from $1 - e^{-1}$ to $1 - (1 - \overrightarrow{p}^{-1}P_{B_2})^{\overrightarrow{p}P_{B_2}^{-1}n_{s+2}} \approx 1 - e^{-n_{s+2}}$. The data complexity of this step is $D_{s+2} = (1 + 2^{12}p_f(n_2)p_f(n_3) \cdots p_f(n_{s+1}))n_1 \cdot 2^{66}n_{s+2} \cdot 2 \cdot (1 + 4) \approx 2^{69.32}n_1 n_{s+2} \cdot \left(1 + 2^{12}p_f(n_2)p_f(n_3) \cdots p_f(n_{s+1})\right)$.

The data and time complexities of the boomerang chain distinguishing process are $T = D = \sum_{i=1}^{s+2} D_i$, and the success probability is $ps_1 \cdot ps_m(n_2) \cdot ps_m(n_3) \cdots ps_m(n_{s+1}) \cdot ps_2$. Besides using more data in each boomerang, we also consider repeating the boomerang chain process several times to increase the success probability of attacks. Denote that the boomerang chain process is repeated w times to form a distinguisher. The time and data complexities of the distinguisher are $T = D = w \cdot \sum_{i=1}^{s+2} D_i$. The success probability of the distinguisher is

$$P_s = 1 - \left[1 - ps_1 \cdot ps_m(n_2) \cdot ps_m(n_3) \cdots ps_m(n_{s+1}) \cdot ps_2\right]^w,$$

where $ps_m(n)$ is given by Eqs. (4). We can obtain that the type-II error probability can be ignored from a similar analysis with Subsect. 4.1.

When $s = 1, 2, 3, 4$ and $n_i = 1, 2, ..., 128$, $1 \leq i \leq s+2$, we calculate the times w the boomerang chain process is repeated such that the success probability $P_s \geq 60\%$, and the data and time complexities by PC program. Table 3 shows the lowest complexities of the distinguishers as well as the corresponding parameters $n_1, n_2, ..., n_{s+2}$, w and the success probability P_s for $s = 1, 2, 3, 4$. When $s = 2$, $n_1 = 1$, $n_2 = 2$, $n_3 = 13$ and $n_4 = 5$, the boomerang chain process is repeated $w = 2$ times to form the distinguisher, the data and time complexities of the distinguisher are $2^{76.57}$, and the corresponding success probability $P_s = 60\%$.

Table 3. The parameters $n_1, n_2, ..., n_{s+2}$, w, complexities and success probability of the boomerang chain distinguisher

s	$n_1, n_2, ..., n_{s+2}$	w	$D = T$	Success Probability
1	1,7,2	2	$2^{77.36}$	66%
2	1,2,13,5	2	$2^{76.57}$	60%
3	1,2,15,110,4	2	$2^{76.59}$	60%
4	1,2,14,116,122,5	2	$2^{76.60}$	60%

Experimental Simulation on Small-Scale AES. To verify our boomerang chain technique, we mounted an experiment to the small-scale variant of AES with 64-bit block proposed in [15]. For 6-round small-scale AES, the probabilities of B_1, B_m and B_2 are $P_{B_1} = 2^{-47.42}$, $P_{B_m} = 2^{-37.42}$ and $P_{B_2} = 2^{-42}$ respectively, and the probability of $\mathcal{D}_{in} \rightarrow \mathcal{D}_{out}$ is $\overrightarrow{p} = 2^{-10}$. We construct a 6-round boomerang chain distinguisher with the parameters in the second row of Table 3. The boomerang chain is composed of 4 boomerangs, that is $B_1 \to B_m \to B_m \to B_2$. We take $2^{14.83}$ plaintext structures in which the four nibbles in $SR^{-1}(Col(0))$ take all possible values and the rest nibbles are any constants. In the 2-nd, 3-rd and 4-th boomerangs, for each pair in the target set L, $2^{26.42}$, $2^{29.12}$ and $2^{32.32}$ 'friend pairs' are constructed respectively. The complexities of four boomerangs are $2^{34.53}$, $2^{33.82}$, $2^{34.02}$ and $2^{35.64}$ respectively. The distinguisher is performed twice, then the overall data and time complexities are both $2^{37.69}$. We implement 500 experiments for random keys and plaintext structures. There are 346 results returning "6-round small-scale AES". The experimental success probability on small-scale AES is about 69%. The program is written in C++, which is partitioned to 25 subprograms and was performed on PC about 10 days.

Key Recovery Attack. Our techniques can be directly used in the key recovery attacks for 6-round AES with the lowest complexity of $2^{78.15}$, higher than the previous attacks. It is shown that the advantage of our techniques lies in distinguishing attacks not key recovery attacks. The key recovery process is as follows.

1. Execute the distinguishing attack and obtain a quartet (P_1', P_2', P_3', P_4') which satisfies B_2, where (P_1', P_2') is active in the 0-th diagonal $SR^{-1}(Col(0))$ and (P_3', P_4') is active in the i-th diagonal $SR^{-1}(Col(i))$, $0 \leq i \leq 3$.
2. Guess and filter K_0 according to the fact that the differences of (P_1', P_2') and (P_3', P_4') after MC of the first round are active in one byte. When $i = 0$, the key guesses of $SR^{-1}(Col(0))$ of K_0 is reduced from 2^{32} to $2^{32} \times 2^{-22} \times 2^{-22} = 2^{-12}$ and we obtain the right value of $SR^{-1}(Col(0))$ of K_0. When $1 \leq i \leq 3$, the key guesses of $SR^{-1}(Col(0, i))$ of K_0 are both reduced from 2^{32} to $2^{32} \times 2^{-22} = 2^{10}$.
3. Repeat the above two steps three times so that each diagonal of K_0 is guessed and filtered.
4. We obtain no more than $(2^{10})^4 = 2^{40}$ possible values of K_0 and check them by trial encryption.

The Boomerang Chain Distinguisher Without B_2. Without B_2, we can also design the 6-round boomerang chain distinguisher only using B_1 and B_m. But the lowest complexity of the distinguisher without B_2 is $2^{77.86}$, which is higher than the complexity of the distinguisher using B_2. Because the power of B_m to distinguish AES from a random permutation is smaller than B_2, we must increase the input data size of B_m to decrease the type-II error probability of judging the random permutation as 6-round AES. Here gives a brief introduction of the boomerang chain distinguisher with only B_1 and B_m. In such a distinguisher, B_1 is used to generate the target set L, and B_m is repeated several times to delete wrong pairs in L. By choosing appropriate times of repetition of B_m and appropriate data size in each boomerang, all wrong pairs in L will be deleted by B_m. If there is still a plaintext pair left in L, that pair is considered as a right pair and the distinguishing result is "6-round AES", otherwise it is "a random permutation". Following the parameters above, when $s = 1, 2, 3, 4$

Algorithm 3: The boomerang chain distinguisher for 6-round AES

```
Determine the parameter $s$, $s \geq 1$;
Determine the parameters $n_1, n_2, \cdots, n_{s+2}$;
Determine the parameter $w$, $w \geq 1$;
/* the boomerang chain process is repeated $w$ times */
for $1 \leq t \leq w$ do
    /* the first boomerang begins */
    Ask for the encryption of $2^{38.84} n_1$ plaintext structures in which the four bytes in
    $SR^{-1}(Col(0))$ take all possible values and the rest bytes are any constants;
    for each plaintext structure do
        for each $i \in \{0, 1, 2, 3\}$ do
            Insert $2^{32}$ ciphertexts $C$ into a hash table indexed by $C_{SR(Col(i))}$;
            for each ciphertext pair $(C_1, C_2)$ such that $(C_1 + C_2)_{SR(Col(i))} = 0$ do
                Exchange the $j$-th inverse diagonal of $(C_1, C_2)$ to obtain 3 pairs $(C_3, C_4)$,
                $j \in \{0, 1, 2, 3\} \setminus i$;
                Decrypt 3 pairs $(C_3, C_4)$ to obtain 3 pairs $(P_3, P_4)$;
                if there exists one $(P_3, P_4)$ such that $P_3 + P_4$ is active in only two
                diagonals then
                    Save the corresponding $(P_1, P_2)$ to the set $L$.
                end
            end
        end
    end
    /* the middle boomerangs begin */
    for $1 \leq i \leq s$ do
        for each plaintext pair $(P_1, P_2)$ in $L$ do
            for $1 \leq k \leq 2^{57.42} n_{i+1}$ do
                Construct one 'friend pair' $(P_1', P_2')$ of $(P_1, P_2)$;
                Ask for the encryption of $(P_1', P_2')$ to obtain $(C_1', C_2')$;
                Exchange one active inverse diagonal of $(C_1', C_2')$ to obtain 4 pairs $(C_3',$
                $C_4')$;
                Decrypt 4 pairs $(C_3', C_4')$ to obtain 4 pairs $(P_3', P_4')$;
            end
            if the total number of $(P_3', P_4')$ active in only two diagonals is less than $n_{i+1}$
            then
                Delete the corresponding plaintext pair $(P_1, P_2)$ from $L$.
            end
        end
    end
    /* the second boomerang begins */
    for each plaintext pair $(P_1, P_2)$ in $L$ do
        for $1 \leq k \leq 2^{66} n_{s+2}$ do
            Construct one 'friend pair' $(P_1', P_2')$ of $(P_1, P_2)$;
            Ask for the encryption of $(P_1', P_2')$ to obtain $(C_1', C_2')$;
            Exchange one active inverse diagonal of $(C_1', C_2')$ to obtain 4 pairs $(C_3', C_4')$;
            Decrypt 4 pairs $(C_3', C_4')$ to obtain 4 pairs $(P_3', P_4')$;
            if there exists one $(P_3', P_4')$ such that $P_3' + P_4'$ is active in only one diagonal
            then
                Return: This is 6-round AES.
            end
        end
    end
end
Return: This is a random permutation.
```

and $n_i = 1, 2, ..., 128$, $1 \leq i \leq s+1$, we calculate the times w the boomerang chain process without B_2 is repeated such that the success probability $P_s \geq 60\%$ and the type-II error probability is lower than 5%, and the data and time complexities by PC program. For $s = 1$, there exist no parameters satisfying the requirement. For $s = 2, 3, 4$, Table 4 shows the lowest complexities of the distinguisher as well as the corresponding parameters $n_1, n_2, ..., n_{s+1}$, w, the success probability P_s and the type-II error probability.

Table 4. The parameters $n_1, n_2, ..., n_{s+1}$, w, complexities, success probability and the type-II error probability of the boomerang chain distinguisher without B_2

s	$n_1, n_2, ..., n_{s+1}$	w	$D = T$	Success Probability	Type-II Error Probability
2	1,14,1	2	$2^{77.86}$	61%	4%
3	1,14,1,79	2	$2^{77.87}$	60%	4%
4	1,14,1,97,105	2	$2^{77.87}$	60%	4%

5 Conclusion

In this paper, we make use of the advantage of adaptively chosen plaintexts and ciphertexts setting and propose the re-boomerang and boomerang chain distinguishers for 6-round AES. We extend the classical boomerang distinguisher to combine two or more related boomerangs with the technique of 'friend pairs' to enhance the distinguishing effect. The re-boomerang distinguisher for 6-round AES has the data and time complexities $2^{82.33}$ and success probability 64%. The boomerang chain distinguisher has the data and time complexities $2^{76.57}$ and success probability 60%. Compared with the previous best result, the data complexity is reduced by a factor of 172, which is a new record for 6-round distinguisher on AES. The boomerang chain distinguisher is easily generalized to other AES-like block ciphers. How to combine the boomerang chain with other cryptanalysis techniques to further improve 6-round distinguishers for AES is worth studying in the future. It is also an open problem that whether 7-round AES can be distinguished with the boomerang chain technique.

Appendix: Proof of Theorem 1

The proof is similar with that of Theorem 5 in [5]. Note that $E_m = AK \circ MC \circ SR \circ SB \circ AK \circ MC$, (X_1, X_2) is an input pair such that $X_1 + X_2 \in \mathcal{D}_{out}$, and (Y_1, Y_2) are the corresponding output pair such that $Y_1 + Y_2$ is inactive in t diagonals, $t = 0$ or 1. (Y_3^j, Y_4^j) are the pairs by exchanging one active diagonal of (Y_1, Y_2), and (X_3^j, X_4^j) are the corresponding pairs after E_m^{-1}, $j \in \{1, 2, ..., 4-t\}$. It needs to be proved that the probability r that there exists $j \in \{1, 2, ..., 4-t\}$ such that $X_3^j + X_4^j \in \mathcal{D}_{out}$ satisfies $r \geq (4-t) \cdot \sum_{d=1}^{3} \binom{4}{d} \cdot (2^{-8})^{4+(2-t)\cdot d}$.

Change the order of SB and SR in E_m, as shown in Fig. 8. Denote by Z the intermediate state after the first MC of E_m, and denote by W the intermediate state before $AK \circ MC \circ SB$ of E_m. We consider a special case here. Assume there exist $j \in \{1, 2, ..., 4-t\}$ and $J \subset \{0, 1, 2, 3\}$ such that (Y_3^j, Y_4^j) can be obtained by exchanging $Col(J)$ of (Y_1, Y_2). Since the operation $AK \circ MC \circ SB$ is applied on each column independently, then (W_3^j, W_4^j) can be seen the result of exchanging $Col(J)$ of (W_1, W_2). After the operation $AK^{-1} \circ SR^{-1}$, (Z_3^j, Z_4^j) is the result of exchanging $SR^{-1}(Col(J))$ of (Z_1, Z_2), which implies that $Z_3^j + Z_4^j = Z_1 + Z_2$. Since MC^{-1} is linear, we have $X_3^j + X_4^j = X_1 + X_2$, then $X_3^j + X_4^j \in \mathcal{D}_{out}$. Therefore, the lower bound of the probability r is the probability that there exist $j \in \{1, 2, ..., 4-t\}$ and $J \subset \{0, 1, 2, 3\}$ such that (Y_3^j, Y_4^j) can be obtained by exchanging $Col(J)$ of (Y_1, Y_2).

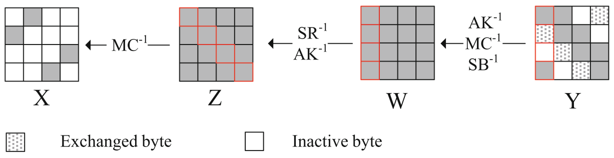


Fig. 8. The backward trail in E_m

In the following we prove that for each $j \in \{1, 2, ..., 4-t\}$ and $J \subset \{0, 1, 2, 3\}$, the probability that (Y_3^j, Y_4^j) can be obtained by exchanging $Col(J)$ of (Y_1, Y_2) is $\left(2^{-8}\right)^{4+(2-t)|J|}$. Denote that the i-th diagonal $SR^{-1}(Col(i))$ of (Y_1, Y_2) is exchanged to obtain (Y_3^j, Y_4^j), $i \in \{0, 1, 2, 3\}$. Denote by S the set of all bytes in $\left(SR^{-1}\left(Col(i)\right) \cup Col(J)\right) \setminus \left(SR^{-1}\left(Col(i)\right) \cap Col(J)\right)$ and denote by $|S|$ the number of bytes in S. Since $\left|SR^{-1}\left(Col(i)\right) \cap Col(J)\right| = |J|$, then $|SR^{-1}\left(Col(i)\right) \cup Col(J)| = 4\left(1 + |J|\right) - |J| = 4 + 3|J|$, and $|S| = 4 + 2|J|$. When exchanging $SR^{-1}\left(Col(i)\right)$ results in the exchange of $Col(J)$, $4 + 2|J|$ bytes in S of $Y_1 + Y_2$ need to be all inactive. Since the difference $Y_1 + Y_2$ is inactive in t diagonals, then $t \cdot |J|$ bytes have been inactive among $4 + 2|J|$ bytes in S. Therefore, the required probability is $\left(2^{-8}\right)^{4+(2-t)|J|}$.

Taking all possible $j \in \{1, 2, ..., 4-t\}$ and $J \subset \{0, 1, 2, 3\}$, we obtain the probability $r \geq (4-t) \cdot \sum_{d=1}^{3} \binom{4}{d} \cdot (2^{-8})^{4+(2-t)\cdot d}$.

References

1. Bao, Z., Guo, C., Guo, J., Song, L.: TNT: How to tweak a block cipher. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020, Part II. Lecture Notes in Computer Science, vol. 12106, pp. 641–673. Springer, Heidelberg, Germany, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45724-2_22

2. Bao, Z., Guo, J., List, E.: Extended truncated-differential distinguishers on round-reduced AES. IACR Transactions on Symmetric Cryptology **2020**(3), 197–261 (2020). https://doi.org/10.13154/tosc.v2020.i3.197-261
3. Bar-On, A., Dunkelman, O., Keller, N., Ronen, E., Shamir, A.: Improved key recovery attacks on reduced-round AES with practical data and memory complexities. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, Part II. Lecture Notes in Computer Science, vol. 10992, pp. 185–212. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2018). https://doi.org/10.1007/978-3-319-96881-0_7
4. Bardeh, N.G.: A key-independent distinguisher for 6-round AES in an adaptive setting. Cryptology ePrint Archive, Report 2019/945 (2019), https://eprint.iacr.org/2019/945
5. Bardeh, N.G., Rønjom, S.: The exchange attack: How to distinguish six rounds of AES with $2^{88.2}$ chosen plaintexts. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019, Part III. Lecture Notes in Computer Science, vol. 11923, pp. 347–370. Springer, Heidelberg, Germany, Kobe, Japan (Dec 8–12, 2019). https://doi.org/10.1007/978-3-030-34618-8_12
6. Bariant, A., Leurent, G.: Truncated boomerang attacks and application to AES-based ciphers. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 3–35. Springer, Heidelberg, Germany, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30634-1_1
7. Beierle, C., Leander, G., Todo, Y.: Improved differential-linear attacks with applications to ARX ciphers. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 329–358. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56877-1_12
8. Biham, E., Keller, N.: Cryptanalysis of reduced variants of Rijndael (2000), in third AES Conference
9. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A.J., Vanstone, S.A. (eds.) Advances in Cryptology – CRYPTO'90. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 11–15, 1991). https://doi.org/10.1007/3-540-38424-3_1
10. Biryukov, A.: The boomerang attack on 5 and 6-round reduced AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) Advanced Encryption Standard – AES. pp. 11–15. Springer Berlin Heidelberg, Berlin, Heidelberg (2005). https://doi.org/10.1007/11506447_2
11. Bogdanov, A., Rijmen, V.: Linear hulls with correlation zero and linear cryptanalysis of block ciphers. Designs, Codes and Cryptography **70**, 369–383 (Mar 2014). https://doi.org/10.1007/s10623-012-9697-z
12. Boura, C., Canteaut, A., Coggia, D.: A general proof framework for recent AES distinguishers. IACR Transactions on Symmetric Cryptology **2019**(1), 170–191 (2019). https://doi.org/10.13154/tosc.v2019.i1.170-191
13. Cho, J., Choi, K.Y., Dinur, I., Dunkelman, O., Keller, N., Moon, D., Veidberg, A.: WEM: A new family of white-box block ciphers based on the Even-Mansour construction. In: Handschuh, H. (ed.) Topics in Cryptology – CT-RSA 2017. Lecture Notes in Computer Science, vol. 10159, pp. 293–308. Springer, Heidelberg, Germany, San Francisco, CA, USA (Feb 14–17, 2017).https://doi.org/10.1007/978-3-319-52153-4_17

14. Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang connectivity table: A new cryptanalysis tool. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018, Part II. Lecture Notes in Computer Science, vol. 10821, pp. 683–714. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78375-8_22
15. Cid, C., Murphy, S., Robshaw, M.J.B.: Small scale variants of the AES. In: Gilbert, H., Handschuh, H. (eds.) Fast Software Encryption – FSE 2005. Lecture Notes in Computer Science, vol. 3557, pp. 145–162. Springer, Heidelberg, Germany, Paris, France (Feb 21–23, 2005). https://doi.org/10.1007/11502760_10
16. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer, Heidelberg, Germany (2002). https://doi.org/10.1007/978-3-662-04722-4
17. Delaune, S., Derbez, P., Vavrille, M.: Catching the fastest boomerangs application to SKINNY. IACR Transactions on Symmetric Cryptology **2020**(4), 104–129 (2020). https://doi.org/10.46586/tosc.v2020.i4.104-129
18. Derbez, P., Fouque, P.A., Jean, J.: Improved key recovery attacks on reduced-round AES in the single-key setting. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. Lecture Notes in Computer Science, vol. 7881, pp. 371–387. Springer, Heidelberg, Germany, Athens, Greece (May 26–30, 2013). https://doi.org/10.1007/978-3-642-38348-9_23
19. Dunkelman, O., Keller, N., Ronen, E., Shamir, A.: The retracing boomerang attack. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 280–309. Springer, Heidelberg, Germany, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45721-1_11
20. Dunkelman, O., Keller, N., Shamir, A.: A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony. Journal of Cryptology **27**(4), 824–849 (Oct 2014). https://doi.org/10.1007/s00145-013-9154-9
21. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved cryptanalysis of Rijndael. In: Schneier, B. (ed.) Fast Software Encryption – FSE 2000. Lecture Notes in Computer Science, vol. 1978, pp. 213–230. Springer, Heidelberg, Germany, New York, NY, USA (Apr 10–12, 2001). https://doi.org/10.1007/3-540-44706-7_15
22. Fouque, P.A., Karpman, P., Kirchner, P., Minaud, B.: Efficient and provable white-box primitives. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 159–188. Springer, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016). https://doi.org/10.1007/978-3-662-53887-6_6
23. Grassi, L.: MixColumns properties and attacks on (round-reduced) AES with a single secret S-box. In: Smart, N.P. (ed.) Topics in Cryptology – CT-RSA 2018. Lecture Notes in Computer Science, vol. 10808, pp. 243–263. Springer, Heidelberg, Germany, San Francisco, CA, USA (Apr 16–20, 2018). https://doi.org/10.1007/978-3-319-76953-0_13
24. Grassi, L.: Mixture differential cryptanalysis: a new approach to distinguishers and attacks on round-reduced AES. IACR Transactions on Symmetric Cryptology **2018**(2), 133–160 (2018). https://doi.org/10.13154/tosc.v2018.i2.133-160
25. Grassi, L., Rechberger, C., Rønjom, S.: Subspace trail cryptanalysis and its applications to AES. IACR Transactions on Symmetric Cryptology **2016**(2), 192–225 (2016). https://doi.org/10.13154/tosc.v2016.i2.192-225, https://tosc.iacr.org/index.php/ToSC/article/view/571

26. Grassi, L., Rechberger, C., Rønjom, S.: A new structural-differential property of 5-round AES. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017, Part II. Lecture Notes in Computer Science, vol. 10211, pp. 289–317. Springer, Heidelberg, Germany, Paris, France (Apr 30 – May 4, 2017). https://doi.org/10.1007/978-3-319-56614-6_10
27. Hu, K., Cui, T., Gao, C., Wang, M.: Towards key-dependent integral and impossible differential distinguishers on 5-round AES. In: Cid, C., Jacobson Jr:, M.J. (eds.) SAC 2018: 25th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 11349, pp. 139–162. Springer, Heidelberg, Germany, Calgary, AB, Canada (Aug 15–17, 2019). https://doi.org/10.1007/978-3-030-10970-7_7
28. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) Fast Software Encryption – FSE'94. Lecture Notes in Computer Science, vol. 1008, pp. 196–211. Springer, Heidelberg, Germany, Leuven, Belgium (Dec 14–16, 1995). https://doi.org/10.1007/3-540-60590-8_16
29. Leurent, G., Pernot, C.: New representations of the AES key schedule. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021, Part I. Lecture Notes in Computer Science, vol. 12696, pp. 54–84. Springer, Heidelberg, Germany, Zagreb, Croatia (Oct 17–21, 2021). https://doi.org/10.1007/978-3-030-77870-5_3
30. Mondal, S.K., Rahman, M., Sarkar, S., Adhikari, A.: Revisiting yoyo tricks on AES. IACR Transactions on Symmetric Cryptology **2023**(4), 28–57 (2023). https://doi.org/10.46586/tosc.v2023.i4.28-57
31. Rahman, M., Saha, D., Paul, G.: Boomeyong: Embedding yoyo within boomerang and its applications to key recovery attacks on AES and Pholkos. IACR Transactions on Symmetric Cryptology **2021**(3), 137–169 (2021).https://doi.org/10.46586/tosc.v2021.i3.137-169
32. Rønjom, S., Bardeh, N.G., Helleseth, T.: Yoyo tricks with AES. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017, Part I. Lecture Notes in Computer Science, vol. 10624, pp. 217–243. Springer, Heidelberg, Germany, Hong Kong, China (Dec 3–7, 2017).https://doi.org/10.1007/978-3-319-70694-8_8
33. Song, L., Qin, X., Hu, L.: Boomerang connectivity table revisited. IACR Transactions on Symmetric Cryptology **2019**(1), 118–141 (2019). https://doi.org/10.13154/tosc.v2019.i1.118-141
34. Sun, B., Liu, M., Guo, J., Qu, L., Rijmen, V.: New insights on AES-like SPN ciphers. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 605–624. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016). https://doi.org/10.1007/978-3-662-53018-4_22
35. Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) Fast Software Encryption – FSE'99. Lecture Notes in Computer Science, vol. 1636, pp. 156–170. Springer, Heidelberg, Germany, Rome, Italy (Mar 24–26, 1999). https://doi.org/10.1007/3-540-48519-8_12
36. Wang, H., Peyrin, T.: Boomerang switch in multiple rounds. IACR Transactions on Symmetric Cryptology **2019**(1), 142–169 (2019). https://doi.org/10.13154/tosc.v2019.i1.142-169

37. Yang, Q., Song, L., Sun, S., Shi, D., Hu, L.: New properties of the double boomerang connectivity table. IACR Transactions on Symmetric Cryptology **2022**(4), 208–242 (2022). https://doi.org/10.46586/tosc.v2022.i4.208-242
38. Zhang, W., Wu, W., Feng, D.: New results on impossible differential cryptanalysis of reduced AES. In: Nam, K.H., Rhee, G. (eds.) ICISC 07: 10th International Conference on Information Security and Cryptology. Lecture Notes in Computer Science, vol. 4817, pp. 239–250. Springer, Heidelberg, Germany, Seoul, Korea (Nov 29–30, 2007)

Multiple-Tweak Differential Attack Against SCARF

Christina Boura[1](✉), Shahram Rasoolzadeh[2], Dhiman Saha[3], and Yosuke Todo[4](✉)

[1] IRIF, Université Paris Cité, Paris, France
christina.boura@irif.fr
[2] Ruhr University Bochum, Bochum, Germany
shahram.rasoolzadeh@rub.de
[3] de.ci.phe.red Lab, Department of Computer Science and Engineering, Indian Institute of Technology Bhilai, Bhilai 491001, India
dhiman@iitbhilai.ac.in
[4] NTT Social Informatics Laboratories, Tokyo, Japan
yosuke.todo@ntt.com

Abstract. In this paper, we present the first third-party cryptanalysis of SCARF, a tweakable low-latency block cipher designed to thwart contention-based cache attacks through cache randomization. We focus on multiple-tweak differential attacks, exploiting biases across multiple tweaks. We establish a theoretical framework explaining biases for any number of rounds and verify this framework experimentally. Then, we use these properties to develop a key recovery attack on 7-round SCARF with a time complexity of 2^{76}, achieving a 98.9% success rate in recovering the 240-bit secret key. Additionally, we introduce a distinguishing attack on the full 8-round SCARF in a multi-key setting, with a complexity of $c \times 2^{67.55}$, demonstrating that SCARF does not provide 80-bit security under these conditions. We also explore whether our approach could be extended to the single-key model and discuss the implications of different S-box choices on the attack success.

Keywords: SCARF · cache randomization · low latency · differential cryptanalysis · multiple-tweak differential attack

1 Introduction

Due to the significant disparity in performance between the CPU and main memory, a strategy used in modern CPUs is to store frequently accessed data within fast and compact memory modules, situated physically close to the cores. This form of memory, referred to as cache, has been the target of several devastating attacks in recent years. Among these attacks, those known as contention-based, exploit the internal architecture of caches and are particularly hard to prevent. A common countermeasure against them consists in randomizing the address-to-cache-index mapping by some randomization function. Such a function needs to have extremely low latency while also ensuring security.

With this objective in mind, Canale et al. designed SCARF in 2023 [11], the first dedicated cache randomization cipher that achieves low latency and is cryptographically secure in the cache attacker model. Given its particular application scenario, SCARF differs significantly from traditional ciphers like the AES [1], as well as from previous low-latency designs such as PRINCE [9] or MANTIS [3]. Notably, SCARF is an 8-round tweakable block cipher that operates on blocks of only 10 bits, whereas traditional block ciphers utilize much larger blocks. Additionally, it employs a large 240-bit key, which is integrated into the state within a few rounds through a nonlinear key schedule and a unique function, i.e., the G function.

A particularly interesting aspect of SCARF, making its security analysis a very challenging task, is its specific security model. Due to the way SCARF is meant to be employed, an attacker is able to choose the plaintexts (index part of the address) and tweaks (tag part of the address) to be encrypted, but cannot observe the ciphertexts computed. The only information an attacker can obtain is whether two ciphertexts collide. This led the designers to state a first security requirement for SCARF: If an attacker queries an oracle with message-tweak inputs $(P_1, T_1), (P_2, T_2)$ and gets as a response 1 if the ciphertext of P_1 encrypted with the tweak T_1 equals the ciphertext of P_2 encrypted with the tweak T_2 and 0 otherwise, he cannot tell if the ciphertexts were produced with SCARF parametrized with some secret key or by a tweakable random permutation with the same input/tweak/output lengths to SCARF with at most 2^{40} queries and at most 2^{80} running time.

The issue with this first security requirement, is that it does not allow cryptanalysts to use common and well-understood security arguments and techniques to assess the security of the cipher. For this reason, the designers provided a second security requirement by observing that whenever two plaintexts P_1 and P_2 and two tweaks T_1 and T_2 lead to the same ciphertext and hence satisfy $E_{T_1}(P_1) = E_{T_2}(P_2)$ then the attacker can learn the evaluation of the function $E^{-1}_{T_2} \circ E_{T_1}(P_1) = P_2$ in case of a collision and can learn that $E^{-1}_{T_2} \circ E_{T_1}(P_1) \neq P_2$ otherwise. Therefore, the second security requirement states that an attacker that asks for the computation of $C = E^{-1}_{T_2} \circ E_{T_1}(P)$ for a plaintext P and a pair of tweaks T_1, T_2 and that is limited to 2^{40} queries and 2^{80} running time, cannot tell if the computation was done with SCARF or with a tweakable random permutation with the same input/tweak/output lengths to SCARF.

The designers of SCARF provided an extended security analysis of E_T and $\tilde{E}_{T_1,T_2} = E^{-1}_{T_2} \circ E_{T_1}$ against different cryptanalysis techniques such as differential [6] or linear [16] cryptanalysis. The small block size of SCARF, particularly facilitates the analysis against those classical attacks, especially statistical ones, as one can easily study the distribution of the different statistical properties by simply computing the relevant tables such as the Difference Distribution Table (DDT), the Linear Approximation Table (LAT) or the Boomerang Connectivity Table (BCT) [12]. This is not possible for traditional ciphers with larger blocks. However, the designers' initial analysis did not identify any distinguishing attack or key-recovery attack on the full cipher. Among the different techniques against reduced-round versions of SCARF, the ones that were identified by the designers to be the most promising ones were the multiple-tweak attacks.

The objective of these attacks is to exploit the fact that the block length of SCARF is smaller than the tweak length and significantly smaller than the security level. This allows for the potential observation of a bias across multiple tweaks, indicating a property that occurs with a probability of $\frac{1}{2^n-1}+\varepsilon$, where n represents the block size, even if ε is much smaller than $\frac{1}{2^n-1}$. The designers of SCARF analyzed such biases occurring in the case of differential properties. For example, they observed the existence of multiple-tweak differentials for $(5+5)$-round SCARF with biases of approximately 2^{-30}, and they also predicted the existence of multiple-tweak differentials for $(6+6)$-round SCARF with biases of about 2^{-40}. However, these biases were only determined experimentally or through analogical reasoning, and no theoretical analysis was provided. Additionally, the designers did not propose any key recovery procedure based on these distinguishing properties.

Our Contributions. In this work, we provide the first third-party cryptanalysis of SCARF by focusing on multiple-tweak differential attacks for $E_{T_2}^{-1} \circ E_{T_1}$. We first provide a theoretical framework to explain the biases for any number of rounds. Our framework is based on an efficient method to compute the expected differential probability (EDP). We also show that the experimentally observed biases perfectly match the theoretical analysis for up to 5+5 rounds. Using these properties as a distinguisher, we present a key recovery attack on 7-round SCARF with a time complexity of 2^{76}. This attack allows the recovery of the 240-bit secret key with a success probability of 98.9%. Next, we provide a distinguishing attack for the full 8-round SCARF in the multi-key setting. Our distinguishing attack has a complexity of $c \times 2^{67.55}$ for some constant c and demonstrates that SCARF does not provide 80-bit security in this setting. Whether our approach leads to a distinguishing attack in the single-key model is an interesting open question. Answering this requires a deep understanding of the definition of bit security [17,23,24], which we discuss extensively later in this work. Finally, we examine the impact of the choice of S-box on this attack and compare the success of the attack when the actual S-box is replaced by other relevant permutations.

The rest of the article is organized as follows. In Sect. 2, the specifications of SCARF and its security requirements are provided together with a brief introduction to differential cryptanalysis. Section 3 presents in detail the idea of multiple-tweak differential attacks and discusses different related works. Section 4 is dedicated to analyzing the bias theoretically, while Sect. 5 presents our key recovery on 7-round SCARF. Section 6 discusses multi-key distinguishing attacks on full SCARF, and finally, Sect. 7 provides an analysis of the impact of the S-box choice on this attack.

2 Preliminaries

We start this section by first describing SCARF and then discussing its security requirements as stated by the designers. At the end, we briefly discuss how to compute the differential probability of a differential transition over several rounds by recalling notably the notion of excepted differential probability (EDP).

2.1 SCARF

SCARF (Secure CAche Randomization Function) is a tweakable block cipher designed by Canale, Güneysu, Leander, Thoma, Todo and Ueno [11]. SCARF was designed to be the first dedicated cache randomization cipher to threaten a particular class of cache attacks. It processes blocks of 10 bits and uses a 48-bit tweak and a 240-bit secret key. Its design, composed of a tweakey schedule and a data encryption part, can be visualised in Fig. 1.

Data Encryption Part. This part takes as input a plaintext block $P \in \mathbb{F}_2^{10}$ and produces a ciphertext block $C \in \mathbb{F}_2^{10}$ by first iterating 7 times a round function R_1 and then applying once a different round function R_2, as seen in Fig. 1. Each round function is parametrized by a 30-bit subkey k generated by the tweakey schedule.

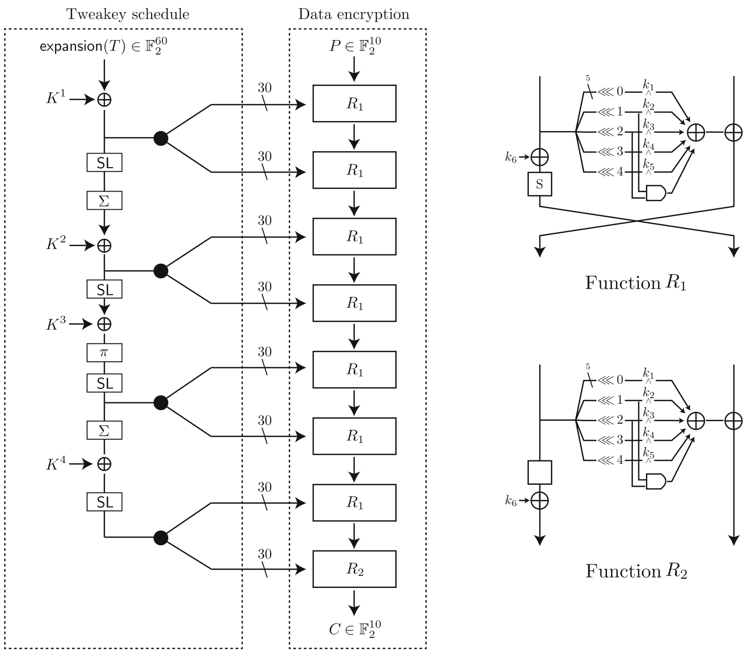


Fig. 1. The structure of SCARF together with its two round functions R_1 and R_2.

Round Functions. R_1 and R_2. Both round functions follow a Feistel-like structure and take as input a 10-bit value x and a 30-bit subkey k. The subkey k can be seen as a concatenation of six 5-bit values, i.e., $k = k_6||k_5||k_4||k_3||k_2||k_1$. The value x is also divided into two 5-bit halves, i.e. $x = x_L||x_R$. We further denote by τ_i the left rotation of x by i positions, i.e., $\tau_i(x) = x \lll i$. The function R_1 updates x by applying the following steps:

$$\begin{aligned} y &= G(x_L, k_1, k_2, k_3, k_4, k_5) \oplus x_R, \\ x_R &= S(x_L \oplus k_6), \\ x_L &= y, \end{aligned}$$

where G is

$$G(x, k_1, k_2, k_3, k_4, k_5) = \left[\bigoplus_{i=0}^{4} (\tau_i(x) \wedge k_{i+1})\right] \oplus (\tau_1(x) \wedge \tau_2(x))$$

and S is a 5-bit S-box defined as

$$S(x) = \Big((\tau_0(x) \vee \tau_1(x)) \wedge (\overline{\tau_3(x)} \vee \overline{\tau_4(x)})\Big) \oplus \Big((\tau_0(x) \vee \tau_2(x)) \wedge (\overline{\tau_2(x)} \vee \tau_3(x))\Big).$$

The S-box S was chosen with low-latency criteria in mind by following the framework of Rasoolzadeh [22]. It has an algebraic degree 4, a differential uniformity of 4 and a linearity of 12.

The final round function R_2 is very similar to R_1. Indeed, the only differences in R_2 is that the order of applying the S-box and the key addition with k_6 is swapped with respect to R_1 and the swap of the branches is omitted:

$$\begin{aligned} x_R &= G(x_L, k_1, k_2, k_3, k_4, k_5) \oplus x_R, \\ x_L &= S(x_L) \oplus k_6. \end{aligned}$$

Both round functions are depicted in Fig. 1.

Tweakey Schedule. The tweakey schedule takes as input a 48-bit tweak T and a 240-bit secret key $K = K^1||K^2||K^3||K^4$, where $K^i \in \mathbb{F}_2^{60}$, $i = 1, 2, 3, 4$ and produces four tweakey values T^1, T^2, T^3, T^4 of 60 bits each by applying the following algorithm:

$$\begin{aligned} T^1 &= \text{expansion}(T) \oplus K^1, \\ T^2 &= \Sigma(\text{SL}(T^1)) \oplus K^2, \\ T^3 &= \text{SL}(\pi(\text{SL}(T^2) \oplus K^3)), \\ T^4 &= \text{SL}(\Sigma(T^3) \oplus K^4). \end{aligned}$$

The role of the expansion function is to extend the 48-bit tweak to a 60-bit value as follows:

$$\begin{aligned} \text{expansion}(T) = 0 \,||\, & T[48] \,||\, T[47] \,||\, T[46] \,||\, T[45] \,|| \\ 0 \,||\, & T[44] \,||\, T[43] \,||\, T[42] \,||\, T[41] \,||\cdots|| \\ 0 \,||\, & T[4] \,||\, T[3] \,||\, T[2] \,||\, T[1]. \end{aligned}$$

The function Σ is a linear function defined as

$$\Sigma(x) = x \oplus \tau_6(x) \oplus \tau_{12}(x) \oplus \tau_{19}(x) \oplus \tau_{29}(x) \oplus \tau_{43}(x) \oplus \tau_{51}(x).$$

The function SL is just the application six times in parallel of the S-box S defined above. Finally, π is a bit-permutation, mapping x_i to $x_{P[i]}$, where P is given by

$$\begin{aligned} P = {} & 1, 6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56, \\ & 2, 7, 12, 17, 22, 27, 32, 37, 42, 47, 52, 57, \\ & 3, 8, 13, 18, 23, 28, 33, 38, 43, 48, 53, 58, \\ & 4, 9, 14, 19, 24, 29, 34, 39, 44, 49, 54, 59, \\ & 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60. \end{aligned}$$

Once the four tweakeys T^1, T^2, T^3 and T^4 have been computed, each of them is split into two equal parts and each part forms a 30-bit subkey rk_i, for the i-th round $(i = 1, \ldots, 8)$:

$$rk_2 \,||\, rk_1 = T_1, \qquad rk_4 \,||\, rk_3 = T_2, \qquad rk_6 \,||\, rk_5 = T_3, \qquad rk_8 \,||\, rk_7 = T_4.$$

2.2 Security Claims

Because of the specific application scenario SCARF was designed for, the attacker model is very different from the models cryptanalysts have traditionally dealt with for more classical ciphers and applications. In particular, while an attacker can choose the message and the tweak, he can never observe the ciphertext produced. The only thing an attacker can observe is whether a cache hit or a cache miss occurred, which corresponds to detecting ciphertext collisions.

The first security requirement stated by the designers for SCARF, given below, specifies that if an attacker can choose plaintexts and tweaks and can observe whether a collision in the ciphertexts occurs, he should not be able to distinguish between SCARF and a tweakable random permutation of the same dimensions. This is under the condition that the attacker is allowed at most 2^{40} queries and his computation time does not exceed 2^{80}.

Security Requirement 1 [11]**.** Let O_{real} be the oracle in the real world that takes addresses (x_1, T_1) and (x_2, T_2), and returns 1 if $E_{T_1}(x_1) = E_{T_2}(x_2)$ and 0 otherwise, where E is SCARF. Let O_{ideal} be the oracle in the ideal world that takes addresses (x_1, T_1) and (x_2, T_2), and returns 1 if $\Pi_{T_1}(x_1) = \Pi_{T_2}(x_2)$ and 0 otherwise, where Π is a tweakable random permutation with the same input/tweak/output lengths to SCARF. An adversary is allowed to make at most 2^{40} queries. Then, the adversary running in time at most 2^{80} cannot distinguish the real from the ideal world.

The above security requirement is quite uncommon compared to the security requirements for more traditional ciphers. Furthermore, it is not clear how cryptanalysts can use well-known methods and tools to assess the security of SCARF under this requirement. For this reason, the designers transformed the problem of collision detection into a problem that is easier to study from a cryptanalytic point of view. Indeed, if the attacker detects that for two plaintexts P_1 and P_2 and for two tweaks T_1 and T_2, the corresponding ciphertexts $C_1 = E_{T_1}(P_1)$ and $C_2 = E_{T_2}(P_2)$ collide, then this equivalently means that P_2 is the decryption of $C_1 = C_2$ under T_2, that is $P_2 = E_{T_2}^{-1} \circ E_{T_1}(P_1)$.

So we can now suppose that an attacker can query

$$\tilde{E}_{T_1,T_2}(P) := E_{T_2}^{-1} \circ E_{T_1}(P) = C,$$

for a chosen plaintext P and for two chosen tweaks T_1, T_2. The second security requirement stated by the designers is that an attacker that can do at most 2^{40} queries and whose computation time is bounded by 2^{80} cannot distinguish whether he is interacting with SCARF or with a tweakable random permutation with the same length parameters as SCARF.

Security Requirement 2 [11]**.** Let $\tilde{O}_{real}$ be the oracle in the real world that takes a plaintext P and a pair of tweaks T_1, T_2 as input and returns C such that $C = E_{T_2}^{-1} \circ E_{T_1}(P)$, where E is SCARF. Let $\tilde{O}_{ideal}$ be the oracle in the ideal world that takes a plaintext P and a pair of tweaks T_1, T_2 as input and returns C such that $C = \Pi_{T_2}^{-1} \circ \Pi_{T_1}(P)$, where Π is a tweakable random permutation with the same input/tweak/output lengths to SCARF. An adversary is allowed to make at most 2^{40} queries. Then, the adversary running in time at most 2^{80} cannot distinguish the real from the ideal world.

Our analysis tackles mainly this second security requirement but we will eventually discuss the impact of our distinguishers and attacks on the first security requirement too.

2.3 Differential Cryptanalysis and Expected Differential Probability

Differential cryptanalysis exploits the existence of high-probability differentials for a reduced-round version of the target cipher. Let E_k be a n-bit block cipher parametrized by a secret key k. We will write $\alpha \xrightarrow{E_k} \beta$ to denote that an input difference α propagates to an output difference β through E_k. The couple (α, β) is called a *differential* for E_k.

We now define the notions of differential probability and expected differential probability (EDP) for E_k.

Definition 1 (Differential Probability). *Let E_k be a n-bit block cipher and let $\alpha, \beta \in \mathbb{F}_2^n$. The (fixed-key)* differential probability *of the differential (α, β) over E_k is defined as*

$$\mathrm{DP}[a \xrightarrow{E_k} b] := \frac{|\{x \in \mathbb{F}_2^n : E_k(x) \oplus E_k(x \oplus \alpha) = \beta\}|}{2^n}.$$

As the key is unknown to the attacker, what he is aiming to compute is what is called the expected differential probability (EDP) which is defined as the average of the probabilities, taken over all keys, of a differential (α, β).

Definition 2 (Expected Differential Probability). *Let E be a n-bit block cipher and let $\alpha, \beta \in \mathbb{F}_2^n$. The* expected differential probability *of the differential (α, β) over E is defined as*

$$\mathrm{EDP}[\alpha \xrightarrow{E} \beta] := 2^{-\kappa} \times \sum_{k \in \mathbb{F}_2^\kappa} \mathrm{DP}[a \xrightarrow{E_k} b],$$

which is the average over all keys $k \in \mathbb{F}_2^\kappa$, where it is assumed that the keys are uniformly distributed.

If the block size is sufficiently small, it is possible to compute the EDP over a single round of the cipher, for all possible input differences α and all possible output differences β and store these values in a $2^n \times 2^n$ matrix. Then, by considering each round key to be independent, one can compute the probability of any transition over several rounds, by simply computing the powers of this EDP matrix [15].

3 Multiple-Tweak Differential Attack

The designers of SCARF identified the multiple-tweak differential cryptanalysis as being one of the most powerful attack strategies against this design. The idea is to observe for some well-chosen differences α and β, a bias ε such that

$$\Pr_{x, T_1, T_2}\left(\tilde{E}_{T_1,T_2}(x) \oplus \tilde{E}_{T_1,T_2}(x \oplus \alpha) = \beta\right) = p + \varepsilon,$$

where $p = \frac{1}{2^n - 1}$, e.g., $p = \frac{1}{1023}$ for SCARF. When we use M pairs, assuming a binomial distribution, the average of the empirical number of pairs satisfying the differential is $M \times (p + \varepsilon)$, and the variance is $M \times (p + \varepsilon)\,(1 - (p + \varepsilon))$. The main focus of the multiple-tweak differential attack is the case where $\varepsilon \ll p$. Then, the variance is approximated by $M \times 2^{-n}$. To distinguish from the ideal case, i.e., $\varepsilon = 0$, we need to use at least $\varepsilon^{-2} \times 2^{-n}$ pairs. Moreover, we approximate the binomial distribution to the normal distribution to estimate the distinguishing advantage, similar to what is done in linear cryptanalysis. Namely, we use the following distributions.

$$\begin{cases} \mathcal{N}(M\,(p+\varepsilon)\,,\ M \times 2^{-n}), & \text{real} \\ \mathcal{N}(M \times p,\ M \times 2^{-n}), & \text{ideal.} \end{cases}$$

The designers of SCARF developed an efficient algorithm that constructs $\left(\frac{N_T}{2}\right)^2 \times 2^9$ pairs with a query/time complexity of $N_T \times 2^{10}$ and 2^{20} memory only. They used $N_T = 2^{23}$, i.e., $M = 2^{55}$ pairs, and experimentally observed the bias up to 5+5 rounds, where this notation means that the cipher $\tilde{E}_{T_1,T_2}$ is restricted to 5 rounds for E_{T_1} and 5 rounds for $E_{T_2}^{-1}$. In particular, for $\alpha = \beta = (\texttt{0x001}, \texttt{0x001})$, the observed bias for each round $(r + r)$, $r = 2, 3, 4, 5$ was:

Round	Bias ε
2+2	$2^{-9.6792}$
3+3	$2^{-14.6761}$
4+4	$2^{-24.8467}$
5+5	$2^{-29.8025}$

The designers did not provide a theoretical explanation for the observed bias. They expected the bias to be close to 2^{-40} in 6+6 rounds, but their experiments could not permit to observe a differential bias due to the extremely high computational complexity. One of the goals in this paper is to provide a theoretical explanation for the experimentally observed differential bias for any number of rounds and in particular to estimate the bias for more than 6+6 rounds.

3.1 LLR Statistic

A multiple-tweak multiple-differential attack is a natural extension of a multiple-tweak differential attack. The idea is to exploit multiple differences instead of a single one. Nowadays, we have substantial knowledge about this type of extension in the context of linear cryptanalysis [2] or differential cryptanalysis [8].

The Log-Likelihood Ratio (LLR) is a classical approach to hypothesis testing. It is used to compare the goodness of fit between two competing hypotheses. Specifically, it compares the likelihood of the data under one hypothesis (usually the null hypothesis) to the likelihood of the data under an alternative hypothesis.

We assume that each differential bias is statistically independent for each pair of α and β. Let V be a set of pairs of input and output differences. Then, the LLR statistic in the case of a multiple-tweak multiple-differential attack can be computed as:

$$LLR = \sum_{(\alpha,\beta)\in V} N(\alpha,\beta) \times \log \frac{p + \varepsilon_{\alpha,\beta}}{p} \approx \sum_{(\alpha,\beta)\in V} N(\alpha,\beta) \times \frac{\varepsilon_{\alpha,\beta}}{p},$$

where $N(a,b)$ denotes the number of pairs satisfying the differential transition from α to β.

Definition 3 (KL Divergence). *Let q and q' be two probability distribution vectors over V. The* Kullback-Leibler divergence *between q and q' is defined as*

$$D(q\|q') := \sum_{v\in V} q_v \times \log\left(\frac{q_v}{q'_v}\right)$$

We also define the following metrics

$$D_2(q\|q') := \sum_{v\in V} q_v \times \log^2\left(\frac{q_v}{q'_v}\right) \quad \text{and} \quad \Delta D(q\|q') := D_2(q\|q') - D(q\|q')^2$$

In our case, $q = p + \varepsilon$ and $q' = p$. The notion of KL divergence can be used to prove that the LLR statistic asymptotically tends towards a normal distribution. The mean and variance exhibit the following properties.

Proposition 1 (Proposition 3 in [2]). *The distributions of LLR asymptotically tend toward a normal distribution as the number of samples M increases. If samples are obtained from the real (resp. ideal) distribution, the LLR statistic tends toward $\mathcal{N}(M\mu_0, M\sigma_0^2)$ (resp. $\mathcal{N}(M\mu_1, M\sigma_1^2)$), where*

$$\begin{aligned}
\mu_0 &= D(q\|q') = \sum_{(\alpha,\beta)\in V} (p+\varepsilon_{\alpha,\beta}) \times \log\frac{p+\varepsilon_{\alpha,\beta}}{p},\\
\mu_1 &= -D(q'\|q) = \sum_{(\alpha,\beta)\in V} p \times \log\frac{p+\varepsilon_{\alpha,\beta}}{p},\\
\sigma_0^2 &= \Delta D(q\|q') = \left(\sum_{(\alpha,\beta)\in V} (p+\varepsilon_{\alpha,\beta}) \times \log^2\frac{p+\varepsilon_{\alpha,\beta}}{p}\right) - \mu_0^2,\\
\sigma_1^2 &= \Delta D(q'\|q) = \left(\sum_{(\alpha,\beta)\in V} p \times \log^2\frac{p+\varepsilon_{\alpha,\beta}}{p}\right) - \mu_1^2.
\end{aligned}$$

In our case, $|\varepsilon_{\alpha,\beta}| \ll p$ and $\left(\sum \varepsilon_{\alpha,\beta}\right)^2 \ll p^{-1} \times \sum \varepsilon_{\alpha,\beta}^2$ hold. Then approximately, we have

$$\mu_0 - \mu_1 \approx \sigma_0^2 \approx \sigma_1^2 \approx \frac{1}{p} \times \sum_{(\alpha,\beta)\in V} \varepsilon_{\alpha,\beta}^2.$$

We sometimes refer to the quantity $\frac{1}{p} \times \sum_{(\alpha,\beta)\in V} \varepsilon_{\alpha,\beta}^2$ as *capacity*. Roughly, the number of required samples, M, must be at least the inverse of the capacity to be able to distinguish between two distributions using the LLR statistic.

3.2 Related Works

The concept of a multiple-tweak differential attack is not entirely novel. Patarin, in particular, discussed this in the context of Feistel ciphers using several permutations [19–21]. A similar attack was also discussed against format-preserving encryption [14].

Generic Attacks against the Luby-Rackoff Construction. Patarin's work provides a precise differential bias of the so-called Luby-Rackoff construction, which is a Feistel cipher instantiated with pseudo-random functions. However, SCARF is not a Feistel cipher and contains an S-box. Moreover, the G function and the S-box are fully specified, and cannot be regarded as pseudo-random functions. Therefore, we cannot directly use Patarin's approach in our work.

Attack against Feistel-Based Format-Preserving Encryption. Dunkelman et al. discussed the same attack in [14]. They provided a more straightforward method to estimate the differential bias. First, similar to traditional differential attacks, they focused on the following two-round iterative trail

$$(\delta, 0) \xrightarrow[prob.=2^{-n}]{\text{1 round}} (0, \delta) \xrightarrow[prob.=1]{\text{1 round}} (\delta, 0).$$

Assuming that pairs that do not satisfy this trail behave randomly, the probability such that $(0, \delta)$ transits to $(0, \delta)$ by $2r$ rounds is estimated as $2^{-2n} + 2^{-nr}$. A truncated differential allows for extension by additional two rounds as

$$(0, \delta) \xrightarrow[prob.=1]{\text{1 round}} (\delta, 0) \xrightarrow[prob.=2^{-n}]{2r \text{ rounds}} (\delta, 0) \xrightarrow[prob.=1]{\text{1 round}} (*, \delta),$$

and they estimated this probability to be $2^{-n} + 2^{-nr}$. In conclusion, while their estimation requires assumptions, the result is almost identical to Patarin's result.

Since the estimation focusing on the trail provides a good estimation for Feistel ciphers, we tried to estimate the bias of SCARF by using this method. Unfortunately, such an estimation leads to completely inaccurate results.

4 Efficient Estimation of the Differential Bias of SCARF

Because of the heavy tweakey-schedule, it is natural to assume that each round function of SCARF is independent. Then, the expected differential probability (EDP) can be computed as the power of the so-called EDP matrix [4,15] (see Sect. 2.3). For n-bit block ciphers, one constructs an $2^n \times 2^n$ matrix, and the EDP is evaluated as a power of this matrix. This computation can be done with complexity 2^{3n}.

Usually, this method is not computationally feasible for common block ciphers with typical block sizes of 64, 128, or 256 bits. However, the block size of SCARF is 10 bits, so the above computation requires only 2^{30} operations and can thus be performed in practice. Nevertheless, we propose a more efficient algorithm for this computation, as 2^{30} operations, while feasible, remain inefficient.

We focus on differential transitions from $\alpha = (\alpha^L, \alpha^R)$ to $\beta = (\beta^L, \beta^R)$ on $\tilde{E}$. We write

$$\alpha \xrightarrow{R+R'} \beta$$

where $\tilde{E}$ is the enc-then-dec structure whose encryption and decryption are composed of R and R' rounds, respectively. Moreover,

$$\text{EDP}[\alpha \xrightarrow{R+R'} \beta]$$

denotes the expected differential probability of the differential transition.

When the round numbers R and R' are explicit from the context, we will simply write $p_{\alpha,\beta}$ to denote the expected differential probability of the transition from α to β. Let $\varepsilon_{\alpha,\beta}$ be the differential bias, i.e., $p_{\alpha,\beta} = \frac{1}{1023} + \varepsilon_{\alpha,\beta}$.

To evaluate the EDP efficiently, we fully exploit the structure of SCARF, particularly the property of the G function. As a result, we show that we can

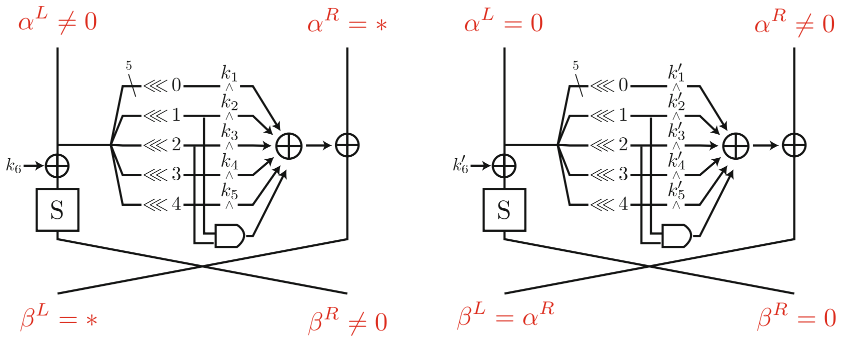


Fig. 2. Differential transitions for R_1

evaluate the EDP from all α to all β with a complexity of 2^{15}. This complexity improvement is important as, in Sect. 7, we discuss how the use of an alternative S-box could improve the security against the proposed attacks. To find the S-box that would provide the higher resistance, we need to estimate the EDP for many S-box candidates. This is the reason why the efficiency of this computation is important, as we need to repeat it several times.

4.1 Some Unique Properties of SCARF

Before analyzing SCARF in detail, we first describe the differential properties of the G function. By averaging on the keys, the differential probability of the G function is

$$\mathrm{EDP}[\alpha \xrightarrow{G} \beta] = \begin{cases} 2^{-5}, & \text{if } \alpha \neq 0, \beta = *, \\ 1, & \text{if } \alpha = 0, \beta = 0, \\ 0, & \text{if } \alpha = 0, \beta \neq 0, \end{cases}$$

where $*$ denotes an arbitrary difference (including the zero difference). Therefore, the differential probability for R_1 from $\alpha = (\alpha^L, \alpha^R)$ to $\beta = (\beta^L, \beta^R)$ is

$$\mathrm{EDP}[\alpha \xrightarrow{R_1} \beta] = \begin{cases} P_S[\alpha^L, \beta^R] \times 2^{-5}, & \text{if } \alpha^L \neq 0, \\ 1, & \text{if } \alpha^L = 0 \text{ and } (\beta_L, \beta_R) = (\alpha_R, 0), \\ 0, & \text{if } \alpha^L = 0 \text{ and } (\beta_L, \beta_R) \neq (\alpha_R, 0), \end{cases}$$

where $P_S[\alpha^L, \beta^R]$ denotes the differential probability for the S-box.

Figure 2 shows each differential transition. Interestingly, when $\alpha^L \neq 0$, the differential probability is independent of α^R and β^L.

More importantly, when considering the composition of R_1 with some arbitrary permutation F, the EDP of $F \circ R_1$ does not depend on α^R in the case $\alpha^L \neq 0$. This is demonstrated in the following lemma.

Lemma 1. *Let F denote an arbitrary permutation from $\mathbb{F}_2^{10}$ to $\mathbb{F}_2^{10}$. Then, the EDP of the differential transition from $\alpha = (\alpha^L, \alpha^R)$ to $\beta = (\beta^L, \beta^R)$ for $F \circ R_1$ does not depend on α^R in the case $\alpha^L \neq 0$.*

Proof. Suppose $\alpha^L \neq 0$. Then,

$$\begin{aligned}\mathrm{EDP}[\alpha \xrightarrow{F \circ R_1} \beta] &= \sum_{\gamma} \mathrm{EDP}[\alpha \xrightarrow{R_1} \gamma] \times \mathrm{EDP}[\gamma \xrightarrow{F} \beta] \\ &= \sum_{\gamma} 2^{-5} \times P_S[\alpha^L, \gamma^R] \times \mathrm{EDP}[\gamma \xrightarrow{F} \beta].\end{aligned}$$

We see indeed from the above formula that α^R is not involved in the evaluation of the EDP of $\alpha \xrightarrow{F \circ R_1} \beta$. □

In Lemma 1, R_1 is applied before F. However, a similar property holds if R_1^{-1} is applied after F. Then, the EDP of the differential transition from $\alpha = (\alpha^L, \alpha^R)$ to $\beta = (\beta^L, \beta^R)$ for $R_1^{-1} \circ F$ does not depend on β^R in the case $\beta^L \neq 0$.

4.2 Analysis for 1+1 Rounds

Although it is straightforward, we begin our discussion with this case. For any non-zero α and β, it holds that

$$\mathrm{EDP}[\alpha \xrightarrow{1+1} \beta] = \begin{cases} 1, & \alpha = (0, \alpha^R), \beta = (0, \beta^R), \alpha^R = \beta^R, \\ 0, & \alpha = (0, *), \beta = (\beta^L, *), \beta^L \neq 0, \\ 0, & \alpha = (\alpha^L, *), \beta = (0, *), \alpha^L \neq 0, \\ P_{\tilde{S}}[\alpha^L, \beta^L] \times 2^{-5}, & \alpha = (\alpha^L, *), \beta = (\beta^L, *), \alpha^L \neq 0, \beta^L \neq 0, \end{cases}$$

where $*$ denotes an arbitrary difference (including the zero difference). The first three cases are trivial.

For the last equation, since $k_{8,6} \oplus k'_{8,6}$ is a variable depending on the tweak value in the multiple-tweak differential attack, the EDP from α^L to β^L is

$$P_{\tilde{S}}(\alpha^L, \beta^L) = \frac{\#\{(x, k) \in \mathbb{F}_2^5 \times \mathbb{F}_2^5 \mid S^{-1}(S(x) \oplus k) \oplus S^{-1}(S(x \oplus \alpha^L) \oplus k) = \beta^L\}}{2^5 \times 2^5}.$$

Table 7 in Appendix A shows the number of solutions of the equation

$$\#\{(x, k) \in \mathbb{F}_2^5 \times \mathbb{F}_2^5 \mid S^{-1}(S(x) \oplus k) \oplus S^{-1}(S(x \oplus \alpha^L) \oplus k) = \beta^L\}$$

for all possible differences α^L and β^L. This table can be seen as a special form of DDT for $\tilde{S} = S^{-1} \circ S(\cdot \oplus k)$ that takes the values of the key into account.

The probability that R'_7 (see Fig. 3) has zero difference, i.e., $\Delta R'_7 = 0$, is

$$\sum_{\gamma \in \mathbb{F}_2^5} \mathrm{EDP}[\alpha^L \xrightarrow{G} \gamma] \times \mathrm{EDP}[\beta^L \xrightarrow{G} \gamma].$$

Due to the property of the G function, by averaging on the keys, the expected differential probability of any non-zero input difference to any output difference (including the zero difference) is 2^{-5}. Therefore, the probability is $2^{5-5-5} = 2^{-5}$.

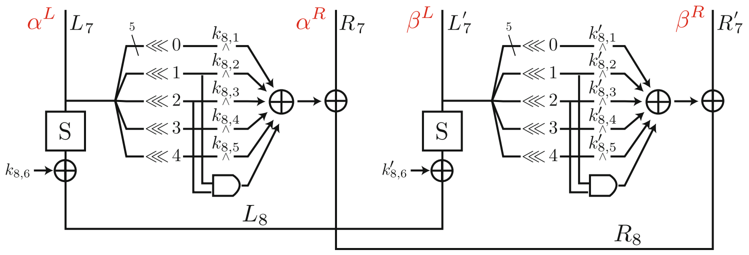


Fig. 3. Differential transitions from α to β through 1+1 rounds.

4.3 Analysis for $R + R'$ Rounds

Recall Lemma 1. The EDP from α to β through $F \circ R_1$ is determined independently of α^R when $\alpha^L \neq 0$. Similarly, the EDP from α to β for $R_1^{-1} \circ F$ is determined independently of β^R when $\beta^L \neq 0$. Taking these properties into account, we obtain the following proposition, since SCARF is represented as $R_1^{-1} \circ F \circ R_1$.

Proposition 2. *When* $\alpha^L \neq 0$*, the* $\mathrm{EDP}[(\alpha^L, \alpha^R) \xrightarrow{R+R'} (\beta^L, \beta^R)]$ *does not depend on* α^R*. Similarly, when* $\beta^L \neq 0$*, the* $\mathrm{EDP}[(\alpha^L, \alpha^R) \xrightarrow{R+R'} (\beta^L, \beta^R)]$ *does not depend on* β^R*.*

The proof of Proposition 2 is trivial due to Lemma 1. Proposition 2 implies that instead of computing all EDPs from $\alpha \to \beta$, considering the following four cases is enough:

$$\mathrm{EDP}[(0, \alpha^R) \xrightarrow{R+R'} (0, \beta^R)], \qquad \mathrm{EDP}[(0, \alpha^R) \xrightarrow{R+R'} (\beta^L, *)],$$
$$\mathrm{EDP}[(\alpha^L, *) \xrightarrow{R+R'} (0, \beta^R)], \qquad \mathrm{EDP}[(\alpha^L, *) \xrightarrow{R+R'} (\beta^L, *)].$$

Here, $*$ denotes any arbitrary difference, and Proposition 2 shows that these EDPs are determined independently of the value of $*$. While the size of the full EDP matrix is 1023×1023, considering only four 31×31 matrices is enough to compute each EDP.

Note that $\mathrm{EDP}[\alpha \xrightarrow{1+1} \beta]$ also satisfies the same property. Namely,

$$\mathrm{EDP}[(0, \alpha^R) \xrightarrow{1+1} (0, \beta^R)] = 1, \quad \text{if} \quad \alpha^R = \beta^R$$
$$\mathrm{EDP}[(0, \alpha^R) \xrightarrow{1+1} (\beta^L, *)] = 0$$
$$\mathrm{EDP}[(\alpha^L, *) \xrightarrow{1+1} (0, \beta^R)] = 0$$
$$\mathrm{EDP}[(\alpha^L, *) \xrightarrow{1+1} (\beta^L, *)] = P_{\tilde{S}}[\alpha^L, \beta^L] \times 2^{-5}.$$

Given four EDP matrices for $R + R'$ rounds, it is easy to compute four EDP matrices for $R + (R' + 1)$ rounds or $(R + 1) + R'$ rounds. Specifically, to compute four EDP matrices for $(R + 1) + R'$ rounds, we have

$$\mathrm{EDP}[(0, \alpha^R) \xrightarrow{R+(R'+1)} (0, \beta^R)] = \mathrm{EDP}[(0, \alpha^R) \xrightarrow{R+R'} (\beta^R, 0)],$$

Table 1. Summary of differential bias for 2+2 rounds. Each element of this table corresponds to $-\log_2(|\varepsilon|)$ for a differential bias ε. The elements colored in light-blue have a negative bias, i.e., $\varepsilon < 0$.

	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
01	9.68	12.99	11.41	13.01	13.01	20.00	20.00	13.01	11.41	12.01	13.01	20.00	11.99	12.01	12.99	12.99	20.00	12.01	12.99	13.01	12.99	11.99	20.00	20.00	12.01	20.00	12.99	13.01	12.01	11.99	20.00
02	12.99	9.68	20.00	12.99	12.01	11.41	12.99	13.01	13.01	13.01	12.99	20.00	11.99	20.00	20.00	13.01	20.00	11.41	12.01	12.01	20.00	13.01	12.99	20.00	13.01	11.99	12.01	12.01	11.99	12.99	20.00
03	11.41	20.00	9.68	20.00	12.99	12.01	11.99	20.00	12.99	11.99	20.00	11.42	20.00	12.01	20.00	20.00	12.01	20.00	13.01	11.99	20.00	20.00	11.42	11.42	20.00	20.00	20.00	12.99	11.99	12.99	11.41
04	13.01	12.99	20.00	9.68	11.41	20.00	12.01	12.99	12.01	12.01	20.00	11.41	13.01	12.99	12.99	13.01	20.00	13.01	13.01	13.01	11.99	12.99	12.01	20.00	12.01	11.99	11.99	20.00	12.99	20.00	20.00
05	13.01	12.01	12.99	11.41	9.68	20.00	11.99	13.01	20.00	12.99	12.99	12.99	12.01	13.01	11.99	12.01	11.99	12.99	20.00	20.00	11.00	12.99	13.01	11.99	11.99	20.00	20.00	13.01	12.99	12.01	13.01
06	20.00	11.41	12.01	20.00	20.00	9.68	13.01	20.00	11.99	12.99	20.00	12.01	20.00	11.99	11.42	20.00	11.42	12.99	20.00	11.99	20.00	20.00	20.00	11.42	12.99	20.00	11.99	12.01	12.99	20.00	11.41
07	20.00	12.99	11.99	12.01	11.99	13.01	9.42	13.01	13.01	20.00	12.99	20.00	12.99	12.01	11.99	12.01	12.01	13.01	12.01	11.99	11.41	20.00	20.00	12.99	11.99	11.41	13.01	11.99	13.01	11.99	11.00
08	13.01	13.01	20.00	12.99	13.01	20.00	13.01	9.68	13.01	11.41	11.99	20.00	12.99	12.01	12.01	12.99	20.00	12.01	12.01	12.01	11.99	20.00	11.99	11.41	20.00	13.01	12.99	12.99	20.00	12.99	20.00
09	11.41	13.01	12.99	12.01	20.00	11.99	13.01	13.01	9.68	20.00	12.01	11.99	11.00	11.99	12.99	12.01	20.00	12.99	13.01	12.99	12.99	20.00	12.01	12.99	11.99	12.99	11.99	20.00	13.01	20.00	13.01
0A	12.01	13.01	11.99	12.01	12.99	12.99	20.00	11.41	20.00	9.68	11.00	20.00	12.99	11.99	13.01	13.01	11.99	20.00	11.99	12.99	20.00	12.99	20.00	12.99	13.01	12.01	12.99	13.01	12.01	11.99	13.01
0B	13.01	12.99	20.00	20.00	12.99	20.00	12.99	11.99	12.01	11.00	9.42	20.00	11.99	11.41	13.01	11.99	20.00	12.99	20.00	20.00	12.01	12.01	13.01	20.00	12.99	11.99	12.01	11.41	11.00	11.99	11.99
0C	20.00	20.00	11.42	11.41	12.99	12.01	20.00	20.00	11.99	20.00	20.00	9.68	20.00	13.01	20.00	20.00	11.42	11.99	12.99	12.99	20.00	20.00	11.99	12.01	12.01	20.00	12.99	11.99	20.00	11.42	11.41
0D	11.99	11.99	20.00	13.01	12.01	20.00	12.99	12.99	11.00	12.99	11.99	20.00	9.42	20.00	12.01	20.00	20.00	20.00	11.41	12.99	11.99	12.01	11.00	20.00	11.41	12.01	11.99	12.99	13.01	13.01	11.99
0E	12.01	20.00	12.01	12.99	13.01	11.99	12.01	12.01	11.99	11.99	11.41	13.01	20.00	9.42	20.00	13.01	12.99	13.01	11.99	20.00	11.41	12.99	13.01	20.00	11.99	12.99	13.01	12.01	11.99	11.99	11.00
0F	12.99	20.00	20.00	12.99	11.99	11.42	11.99	12.01	12.99	13.01	13.01	20.00	12.01	20.00	9.68	11.99	12.99	12.01	11.99	20.00	11.99	13.01	12.99	11.99	13.01	11.00	20.00	13.01	20.00	12.99	11.99
10	12.99	13.01	20.00	13.01	12.01	20.00	12.01	12.99	12.01	13.01	11.99	20.00	20.00	13.01	11.99	9.68	11.41	13.01	20.00	11.41	13.01	11.99	12.99	20.00	12.99	12.99	20.00	12.01	12.99	12.01	20.00
11	20.00	20.00	12.01	20.00	11.99	11.42	12.01	20.00	20.00	11.99	20.00	11.42	20.00	12.99	12.99	11.41	9.68	12.99	11.99	12.99	20.00	20.00	20.00	12.01	13.01	20.00	11.42	20.00	20.00	11.99	11.41
12	12.01	11.41	20.00	13.01	12.99	12.99	13.01	12.01	12.99	20.00	12.99	11.99	20.00	13.01	12.01	13.01	12.99	9.68	11.99	20.00	12.99	12.01	11.99	11.99	20.00	11.00	13.01	11.99	20.00	12.99	13.01
13	12.99	12.01	13.01	13.01	20.00	20.00	12.01	12.01	13.01	11.99	20.00	12.99	11.41	11.99	11.99	20.00	11.99	11.99	9.42	13.01	12.99	12.99	11.99	12.01	12.01	11.41	20.00	11.99	13.01	13.01	11.00
14	13.01	12.01	11.99	13.01	20.00	11.99	11.99	12.01	12.99	12.99	20.00	12.99	12.99	20.00	20.00	11.41	12.99	20.00	13.01	9.68	12.01	11.00	12.99	20.00	13.01	12.99	12.01	11.99	11.99	13.01	13.01
15	12.99	20.00	20.00	11.99	11.00	20.00	11.41	11.99	12.99	20.00	12.01	20.00	11.99	11.41	11.99	13.01	20.00	12.99	12.99	12.01	9.42	11.99	13.01	20.00	20.00	12.01	13.01	12.99	12.01	11.00	11.99
16	11.99	13.01	20.00	12.99	12.99	20.00	20.00	20.00	20.00	12.99	12.01	20.00	12.01	12.99	13.01	11.99	20.00	12.01	12.99	11.00	11.99	9.42	12.01	20.00	11.41	11.99	11.00	11.41	11.99	13.01	11.99
17	20.00	12.99	11.42	12.01	13.01	20.00	20.00	11.99	12.01	20.00	13.01	11.99	11.00	13.01	12.99	12.99	20.00	11.99	11.99	12.99	13.01	12.01	9.68	12.99	11.99	11.99	12.99	13.01	20.00	20.00	11.99
18	20.00	20.00	11.42	20.00	11.99	11.42	12.99	11.41	12.99	12.99	20.00	12.01	20.00	20.00	11.99	20.00	12.01	11.99	12.01	20.00	20.00	20.00	12.99	9.68	11.99	20.00	20.00	13.01	11.42	20.00	11.41
19	12.01	13.01	20.00	12.01	11.99	12.99	11.99	20.00	11.99	13.01	12.99	12.01	11.41	11.99	13.01	12.99	13.01	20.00	12.01	13.01	20.00	11.41	11.99	11.99	9.42	12.99	11.99	12.01	20.00	13.01	11.00
1A	20.00	11.99	20.00	11.99	20.00	20.00	11.41	13.01	12.99	12.01	11.99	20.00	12.01	12.99	11.00	12.99	20.00	11.00	11.41	12.99	12.01	11.99	11.99	20.00	12.99	9.42	13.01	20.00	13.01	12.01	11.99
1B	12.99	12.01	20.00	11.99	20.00	11.99	13.01	12.99	11.99	12.99	12.01	12.99	11.99	13.01	20.00	20.00	11.42	13.01	20.00	12.01	13.01	11.00	12.99	20.00	11.99	13.01	9.68	11.99	12.99	20.00	11.99
1C	13.01	12.01	12.99	20.00	13.01	12.01	11.99	12.99	20.00	13.01	11.41	11.99	12.99	12.01	13.01	12.01	20.00	11.99	11.99	11.99	12.99	11.41	13.01	13.01	12.01	20.00	11.99	9.42	11.99	20.00	11.00
1D	12.01	11.99	11.99	12.99	12.99	12.99	13.01	20.00	13.01	12.01	11.00	20.00	13.01	11.99	20.00	12.99	20.00	20.00	13.01	11.99	12.01	11.99	20.00	11.42	20.00	13.01	12.99	11.99	9.68	12.99	11.99
1E	11.99	12.99	12.99	20.00	12.01	20.00	11.99	12.99	20.00	11.99	11.99	11.42	13.01	11.99	12.99	12.01	11.99	12.99	13.01	13.01	11.00	13.01	20.00	20.00	13.01	12.01	20.00	20.00	12.99	9.68	11.99
1F	20.00	20.00	11.41	20.00	13.01	11.41	11.00	20.00	13.01	13.01	11.99	11.41	11.99	11.00	11.99	20.00	11.41	13.01	11.00	13.01	11.99	11.99	11.99	11.41	11.00	11.99	11.99	11.00	11.99	11.99	8.83

$$
\begin{aligned}
\mathrm{EDP}[(0,\alpha^R) \xrightarrow{R+(R'+1)} (\beta^L,*)] &= \sum_{\gamma^R \neq 0} 2^{-5} \times P_S[\beta^L,\gamma^R] \times \mathrm{EDP}[(0,\alpha^R) \xrightarrow{R+R'} (0,\gamma^R)] \\
&\quad + \sum_{\gamma^L \neq 0} 2^{-5} \times \mathrm{EDP}[(0,\alpha^R) \xrightarrow{R+R'} (\gamma^L,*)], \\
\mathrm{EDP}[(\alpha^L,*) \xrightarrow{R+(R'+1)} (0,\beta^R)] &= \mathrm{EDP}[(\alpha^L,*) \xrightarrow{R+R'} (\beta^R,0)], \\
\mathrm{EDP}[(\alpha^L,*) \xrightarrow{R+(R'+1)} (\beta^L,*)] &= \sum_{\gamma^R \neq 0} 2^{-5} \times P_S[\beta^L,\gamma^R] \times \mathrm{EDP}[(\alpha^L,*) \xrightarrow{R+R'} (0,\gamma^R)] \\
&\quad + \sum_{\gamma^L \neq 0} 2^{-5} \times \mathrm{EDP}[(\alpha^L,*) \xrightarrow{R+R'} (\gamma^L,*)].
\end{aligned}
$$

In a similar way, we can compute the $\mathrm{EDP}[\alpha \xrightarrow{(R+1)+R'} \beta]$. Given α, β, and four EDP matrices for $R+R'$ rounds, the complexity to update four EDP matrices as above is approximately 4×2^{15}.

For an arbitrary number of rounds R, we can obtain the EDP for $R+R$ rounds very efficiently. On the other hand, we notice that $\mathrm{EDP}[(0,\alpha^R) \xrightarrow{R+R} (0,\beta^R)]$ is significantly more biased than the others. This is not surprising, as their EDPs are equal to $\mathrm{EDP}[(\alpha^R,0) \xrightarrow{(R-1)+(R-1)} (\beta^R,0)]$.

Tables 1 and 2 summarize differential biases computed by EDPs for 2+2 and 3+3 rounds, respectively. Each row corresponds to a value of α^R and each column

Table 2. Summary of differential bias for 3+3 rounds. Each element of this table corresponds to $-\log_2(|\varepsilon|)$ for a differential bias ε. The elements colored in light-blue have a negative bias, i.e., $\varepsilon < 0$.

	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
01	14.71	17.68	16.30	18.42	18.42	20.00	20.00	18.42	16.30	17.19	18.42	20.00	16.83	17.19	17.68	17.68	20.00	17.19	17.68	18.42	17.68	16.83	20.00	20.00	17.19	20.00	17.68	18.42	17.19	16.83	20.00
02	17.68	14.71	20.00	17.68	17.19	16.30	17.68	18.42	18.42	18.42	17.68	20.00	16.83	20.00	20.00	18.42	20.00	16.30	17.19	17.19	20.00	18.42	17.68	20.00	18.42	16.83	17.19	17.19	16.83	17.68	20.00
03	16.30	20.00	14.71	20.00	17.68	17.19	16.83	20.00	17.68	16.83	20.00	16.54	20.00	17.19	20.00	20.00	17.19	20.00	18.42	16.83	20.00	20.00	16.54	16.54	20.00	20.00	20.00	17.68	16.83	17.68	16.30
04	18.42	17.68	20.00	14.71	16.30	20.00	17.19	17.68	17.19	17.19	20.00	16.30	18.42	17.68	17.68	18.42	20.00	18.42	18.42	18.42	16.83	17.68	17.19	20.00	17.19	16.83	16.83	20.00	17.68	20.00	20.00
05	18.42	17.19	17.68	16.30	14.71	20.00	16.83	18.42	20.00	17.68	17.68	17.68	17.19	18.42	16.83	17.19	16.83	17.68	20.00	20.00	16.09	17.68	18.42	16.83	16.83	20.00	20.00	18.42	17.68	17.19	18.42
06	20.00	16.30	17.19	20.00	20.00	14.71	18.42	20.00	16.83	17.68	20.00	17.19	20.00	16.83	16.54	20.00	16.54	17.68	20.00	16.83	20.00	20.00	20.00	16.54	17.68	20.00	16.83	17.19	17.68	20.00	16.30
07	20.00	17.68	16.83	17.19	16.83	18.42	14.45	18.42	18.42	20.00	17.68	20.00	17.68	17.19	16.83	17.19	17.19	18.42	17.19	16.83	16.30	20.00	20.00	17.68	16.83	16.30	18.42	16.83	18.42	16.83	16.09
08	18.42	18.42	20.00	17.68	18.42	20.00	18.42	14.71	18.42	16.30	16.83	20.00	17.68	17.19	17.19	17.68	20.00	17.19	17.19	17.19	16.83	20.00	16.83	16.30	20.00	18.42	17.68	17.68	20.00	17.68	20.00
09	16.30	18.42	17.68	17.19	20.00	16.83	18.42	18.42	14.71	20.00	17.19	16.83	16.09	16.83	17.68	17.19	20.00	17.68	18.42	17.68	17.68	20.00	17.19	17.68	16.83	17.68	16.83	20.00	18.42	20.00	18.42
0A	17.19	18.42	16.83	17.19	17.68	17.68	20.00	16.30	20.00	14.71	16.09	20.00	17.68	16.83	18.42	18.42	16.83	20.00	16.83	17.68	20.00	17.68	20.00	17.68	18.42	17.19	17.68	18.42	17.19	16.83	18.42
0B	18.42	17.68	20.00	20.00	17.68	20.00	17.68	16.83	17.19	16.09	14.45	20.00	16.83	16.30	18.42	16.83	20.00	17.68	20.00	20.00	17.19	17.19	18.42	20.00	17.68	16.83	17.19	16.30	16.09	16.83	16.83
0C	20.00	20.00	16.54	16.30	17.68	17.19	20.00	20.00	16.83	20.00	20.00	14.71	20.00	18.42	20.00	20.00	16.54	16.83	17.68	17.68	20.00	20.00	16.83	17.19	17.19	20.00	17.68	16.83	20.00	16.54	16.30
0D	16.83	16.83	20.00	18.42	17.19	20.00	17.68	17.68	16.09	17.68	16.83	20.00	14.45	20.00	17.19	20.00	20.00	20.00	16.30	17.68	16.83	17.19	16.09	20.00	16.30	17.19	16.83	17.68	18.42	18.42	16.83
0E	17.19	20.00	17.19	17.68	18.42	16.83	17.19	17.19	16.83	16.83	16.30	18.42	20.00	14.45	20.00	18.42	17.68	18.42	16.83	20.00	16.30	17.68	18.42	20.00	16.83	17.68	18.42	17.19	16.83	16.83	16.09
0F	17.68	20.00	20.00	17.68	16.83	16.54	16.83	17.19	17.68	18.42	18.42	20.00	17.19	20.00	14.71	16.83	17.68	17.19	16.83	20.00	16.83	18.42	17.68	16.83	18.42	16.09	20.00	18.42	20.00	17.68	16.83
10	17.68	18.42	20.00	18.42	17.19	20.00	17.19	17.68	17.19	18.42	16.83	20.00	20.00	18.42	16.83	14.71	16.30	18.42	20.00	16.30	18.42	16.83	17.68	20.00	17.68	17.68	20.00	17.19	17.68	17.19	20.00
11	20.00	20.00	17.19	20.00	16.83	16.54	17.19	20.00	20.00	16.83	20.00	16.54	20.00	17.68	17.68	16.30	14.71	17.68	16.83	17.68	20.00	20.00	20.00	17.19	18.42	20.00	16.54	20.00	20.00	16.83	16.30
12	17.19	16.30	20.00	18.42	17.68	17.68	18.42	17.19	17.68	20.00	17.68	16.83	20.00	18.42	17.19	18.42	17.68	14.71	16.83	20.00	17.68	17.19	16.83	16.83	20.00	16.09	18.42	16.83	20.00	17.68	18.42
13	17.68	17.19	18.42	18.42	20.00	20.00	17.19	17.19	18.42	16.83	20.00	17.68	16.30	16.83	16.83	20.00	16.83	16.83	14.45	18.42	17.68	17.68	16.83	17.19	17.19	16.30	20.00	16.83	18.42	18.42	16.09
14	18.42	17.19	16.83	18.42	20.00	16.83	16.83	17.19	17.68	17.68	20.00	17.68	17.68	20.00	20.00	16.30	17.68	20.00	18.42	14.71	17.19	16.09	17.68	20.00	18.42	17.68	17.19	16.83	16.83	18.42	18.42
15	17.68	20.00	20.00	16.83	16.09	20.00	16.30	16.83	17.68	20.00	17.19	20.00	16.83	16.30	16.83	18.42	20.00	17.68	17.68	17.19	14.45	16.83	18.42	20.00	20.00	17.19	18.42	17.68	17.19	16.09	16.83
16	16.83	18.42	20.00	17.68	17.68	20.00	20.00	20.00	20.00	17.68	17.19	20.00	17.19	17.68	18.42	16.83	20.00	17.19	17.68	16.09	16.83	14.45	17.19	20.00	16.30	16.83	16.09	16.30	16.83	18.42	16.83
17	20.00	17.68	16.54	17.19	18.42	20.00	20.00	16.83	17.19	20.00	18.42	16.83	16.09	18.42	17.68	17.68	20.00	16.83	16.83	17.68	18.42	17.19	14.71	17.68	16.83	16.83	17.68	18.42	20.00	20.00	16.83
18	20.00	20.00	16.54	20.00	16.83	16.54	17.68	16.30	17.68	17.68	20.00	17.19	20.00	20.00	16.83	20.00	17.19	16.83	17.19	20.00	20.00	20.00	17.68	14.71	16.83	20.00	20.00	18.42	16.54	20.00	16.30
19	17.19	18.42	20.00	17.19	16.83	17.68	16.83	20.00	16.83	18.42	17.68	17.19	16.30	16.83	18.42	17.68	18.42	20.00	17.19	18.42	20.00	16.30	16.83	16.83	14.45	17.68	16.83	17.19	20.00	18.42	16.09
1A	20.00	16.83	20.00	16.83	20.00	20.00	16.30	18.42	17.68	17.19	16.83	20.00	17.19	17.68	16.09	17.68	20.00	16.09	16.30	17.68	17.19	16.83	16.83	20.00	17.68	14.45	18.42	20.00	18.42	17.19	16.83
1B	17.68	17.19	20.00	16.83	20.00	16.83	18.42	17.68	16.83	17.68	17.19	17.68	16.83	18.42	20.00	20.00	16.54	18.42	20.00	17.19	18.42	16.09	17.68	20.00	16.83	18.42	14.71	16.83	17.68	20.00	16.83
1C	18.42	17.19	17.68	20.00	18.42	17.19	16.83	17.68	20.00	18.42	16.30	16.83	17.68	17.19	18.42	17.19	20.00	16.83	16.83	16.83	17.68	16.30	18.42	18.42	17.19	20.00	16.83	14.45	16.83	20.00	16.09
1D	17.19	16.83	16.83	17.68	17.68	17.68	18.42	20.00	18.42	17.19	16.09	20.00	18.42	16.83	20.00	17.68	20.00	20.00	18.42	16.83	17.19	16.83	20.00	16.54	20.00	18.42	17.68	16.83	14.71	17.68	16.83
1E	16.83	17.68	17.68	20.00	17.19	20.00	16.83	17.68	20.00	16.83	16.83	16.54	18.42	16.83	17.68	17.19	16.83	17.68	18.42	18.42	16.09	18.42	20.00	20.00	18.42	17.19	20.00	20.00	17.68	14.71	16.83
1F	20.00	20.00	16.30	20.00	18.42	16.30	16.09	20.00	18.42	18.42	16.83	16.30	16.83	16.09	16.83	20.00	16.30	18.42	16.09	18.42	16.83	16.83	16.83	16.30	16.09	16.83	16.83	16.09	16.83	16.83	13.85

Table 3. Theoretical capacity of differential biases for each number of rounds.

capacity	2+2	3+3	4+4	5+5	6+6	7+7	8+8
$-\log_2(C)$	3.37	13.39	32.20	42.20	60.89	70.89	88.95
$-\log_2(C_{all})$	2.38	13.38	31.20	42.19	59.89	70.88	87.95

to a value of β^R. The values of α^L and β^L are fixed to 0. Differential biases for other number of rounds are summarized in the full version.

Table 3 summarizes the capacity of the differential biases for different numbers of rounds. Note that, the capacities C and C_{all} are estimated as

$$C = 1023 \times \sum_{\alpha^R \neq 0} \sum_{\beta^R \neq 0} \varepsilon^2_{(0,\alpha^R),(0,\beta^R)}, \tag{1}$$

$$C_{all} = 1023 \times \sum_{\alpha \neq 0} \sum_{\beta \neq 0} \varepsilon^2_{\alpha,\beta}. \tag{2}$$

Of course, C_{all} has a higher capacity, but it is computed on almost twice as many biases. Because of the cost increase for computing the LLR statistics exploiting 1023×1023 input and output differences, we only focus on the cases for which $\alpha^L = \beta^L = 0$.

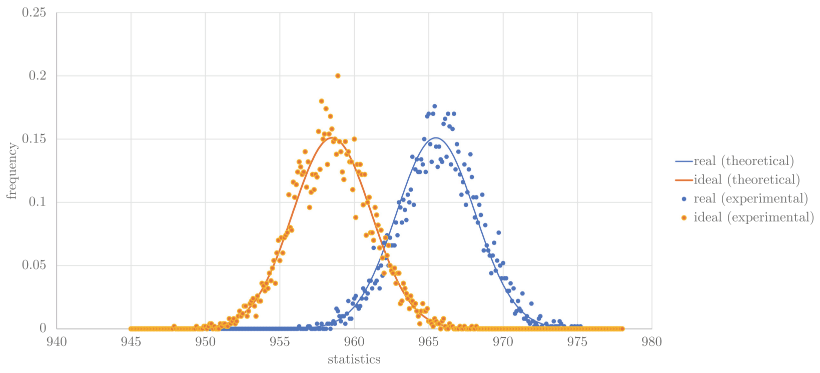


Fig. 4. Comparison between the theoretical estimations and the experimental results of LLR statistics for 4+4 rounds and the ideal distribution.

Experimental Verification

2+2 Rounds. To verify the Markov assumption in practice, we experimentally computed each differential bias for 2+2 rounds by using 2^{15} tweaks T_i and 2^{15} tweaks T_j. In total, our experiments use $2^{15} \times 2^{15} \times 2^9 = 2^{39}$ pairs. Table 1 summarizes the differential biases computed by each EDP. Some differential biases are as low as $-2^{-20.00}$, but 2^{39} pairs are still enough to confirm such a low bias experimentally. Here, we just focus on some concrete cases of interest.

– When $(\alpha^R, \beta^R) = (\texttt{0x01}, \texttt{0x01})$, the theoretical bias is estimated as $+2^{-9.68}$. Experimentally, we also obtain the same bias.
– When $(\alpha^R, \beta^R) = (\texttt{0x02}, \texttt{0x1F})$, the theoretical bias is estimated as $-2^{-20.00}$. The same bias is also obtained experimentally for those differences.

3+3 Rounds. We next verify experimentally the differential bias for 3+3 rounds, and Table 2 summarizes these results. To experimentally verify each differential bias, we used 2^{17} tweaks T_i and 2^{17} tweaks T_j. In total, our experiments use $2^{17} \times 2^{17} \times 2^9 = 2^{43}$ pairs. Here, we depict some concrete cases of interest.

– When $(\delta_i, \delta_j) = (\texttt{0x01}, \texttt{0x01})$, the theoretical bias is estimated as $+2^{-14.71}$. Experimentally, we obtain that this bias equals $2^{-17.68}$.
– When $(\delta_i, \delta_j) = (\texttt{0x02}, \texttt{0x1E})$, the theoretical bias is estimated as $-2^{-17.68}$. Experimentally, we obtain that this bias equals $-2^{-17.66}$.

4+4 Rounds and LLR Statistics. Besides the Markov assumption, we need an independent assumption for the LLR statistics. To verify the statistically independent assumption, we experimentally compared the LLR statistics for 4+4 rounds and the ideal case. The experiments used $2^{13} \times 2^{13} \times 2^9 = 2^{35}$ pairs and were repeated 5000 times. According to Proposition 1, each distribution tends toward the following distributions

$$Real \sim \mathcal{N}(965.43, 6.98) \qquad Ideal \sim \mathcal{N}(958.45, 6.98). \tag{3}$$

Figure 4 compares the theoretical estimation and experimental frequencies, which justifies the correctness of our theoretical estimation.

5+5 Rounds and LLR Statistics. Similar to the experiment for 4+4 rounds, we experimentally compared the LLR statistics for 5+5 rounds and the ideal case. The experiments used $2^{18} \times 2^{18} \times 2^{9} = 2^{45}$ pairs and were repeated 5000 times. When we use 2^{45} pairs, each distribution tends towards the same distribution as Eq. (3). Figure 5 compares the theoretical estimation and experimental frequencies.

5 Key-Recovery Attack on 7-Round SCARF

In this section, we propose a key-recovery attack on 7-round SCARF.

First of all, we define a proper "reduced-round SCARF" version. To reduce SCARF to 7 rounds, we decided to remove the first round function, R_1. Of course, another choice would have been to remove the last round, R_2, but R_2 was designed to prevent that the last S-box is always canceled. Such a reduced-round version would change the security property of SCARF significantly. Therefore, we believe that removing the first round is more meaningful for discussing the security margin of the full cipher. The tweakey schedule is also nonlinear, and it has a block-cipher-like structure. In our 7-round SCARF, we decided to take the most conservative choice and to use the same tweakey schedule as the original one. Therefore, the secret key size is still $60 \times 4 = 240$ bits even after removing the first round.

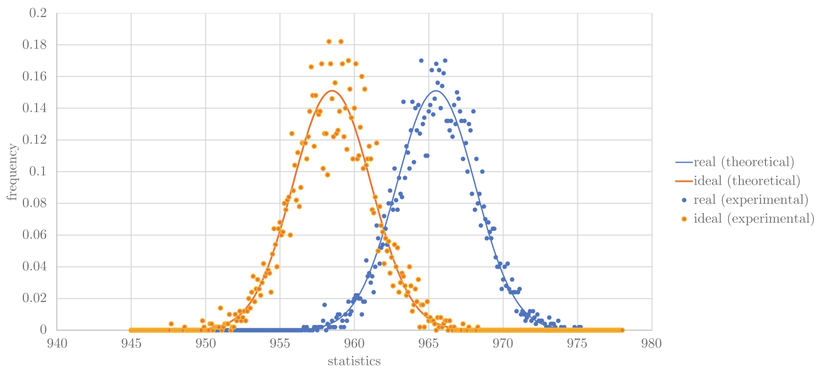


Fig. 5. Comparison between the theoretical estimations and the experimental results of LLR statistics for 5+5 rounds and the ideal distribution.

5.1 Attack Procedure on Security Requirement 2

Step 1: Partial Key Recovery Using the (6+6)-Round Multiple Differential Distinguisher. We first collect data by using the enc-then-dec oracle, $\tilde{E}$.

Algorithm 1. Algorithm to recover the top 30 bits of K^1

```
Input: $2^{30}$ tweaks $T_i$, a tweak $T'$, differential probability $p_{\delta_{in},\delta_{out}}$, a threshold $\theta$.
Output: The top 30 bits of $K^1$.
  Prepare two two-dimensional arrays $A[][]$ and $B[][]$, of size $31 \times 2^{20}$.
  for $K^1 \in \mathbb{F}_2^{30} \| 0^{30}$ do
      for $i = 0$ to $2^{29} - 1$ do
          Derive $rk_2$ from $K^1$ and $T_i$.
          for all $\delta \in \mathbb{F}_2^5 \setminus \{0\}$ do
              for all $x_1 \in \mathbb{F}_2^{10}$ do
                  $x_0 = R_1^{-1}(x_1, rk_2)$.
                  $x_0' = R_1^{-1}(x_1 \oplus 0^5 \| \delta, rk_2)$.
                  if $\tilde{E}_{T_i,T'}(x_0) < \tilde{E}_{T_i,T'}(x_0')$ then
                      $y = \tilde{E}_{T_i,T'}(x_0) \| \tilde{E}_{T_i,T'}(x_0')$
                      $A[\delta][y] = A[\delta][y] + 1$
      Repeat the same procedure as above for $i = 2^{29}$ to $2^{30} - 1$ and get $B[\delta][y]$.
      $s = 0$.
      for all $\delta_{in} \in \mathbb{F}_2^5 \setminus \{0\}$ do
          for all $\delta_{out} \in \mathbb{F}_2^5 \setminus \{0\}$ do
              $n = 0$.
              for all $y \in \mathbb{F}_2^{20}$ do
                  $n = n + A[\delta_{in}][y] \times B[\delta_{out}][y]$
              $s = s + n \times \log(\frac{p_{\delta_{in},\delta_{out}}}{1/1023})$
      if $s > \theta$ then
          Return $K^1$
```

We use 2^{30} tweaks T_i, where the bottom 30 bits of T_i are active, i.e., $T_i := 0^{18} \| i$. In the security requirement 2, the oracle accepts a plaintext and a pair of tweaks. Therefore, we choose another fixed tweak, T', different from any of the T_i, e.g., $T' = 1^{48}$. We query the full code book for the 2^{30} T_i and a fixed tweak T' and store $\tilde{E}_{T_i,T'}(x)$ for all $x \in \mathbb{F}_2^{10}$. From these plaintext-ciphertext pairs, we have $\tilde{E}_{T_i,T_j} = \tilde{E}_{T_j,T'}^{-1} \circ \tilde{E}_{T_i,T'}$ and can construct $M = 2^{29+29+9} = 2^{67}$ pairs[1].

The initial goal is to recover the top 30 bits of K^1 by using the (6+6)-round distinguisher. The highest differential bias is $2^{-38.19}$, which is not always sufficient to filter key candidates with a reasonable success probability. Therefore, we use multiple differentials and the LLR statistic. According to Proposition 1, we compute μ_0, μ_1, σ_0^2, and σ_1^2. Then, $(\mu_0 - \mu_1) \approx \sigma_0^2 \approx \sigma_1^2 \approx 2^{-60.8887}$. When we use 2^{67} pairs, each distribution tends towards the following distribution

$$\text{Real} \sim \mathcal{N}(34.56, 69.13) \qquad \text{Ideal} \sim \mathcal{N}(-34.56, 69.13),$$

where, from each average, we subtract $(\mu_0 + \mu_1)/2$ in the sake of readability, i.e., Real $\sim \mathcal{N}(M \times \left(\mu_0 - \frac{\mu_0+\mu_1}{2}\right), M\sigma_0^2)$ and Ideal $\sim \mathcal{N}(M \times \left(\mu_1 - \frac{\mu_0+\mu_1}{2}\right), M\sigma_1^2)$

[1] It is also possible to construct $\binom{2^{30}}{2} 2^9 \approx 2^{68}$ pairs by considering all combinations of tweaks. However, if we do this, we have a critical problem with the independence of every pair. Indeed, when we observe the differential property by $E_{T_2}^{-1} \circ E_{T_1}$ and $E_{T_3}^{-1} \circ E_{T_1}$, we cannot include $E_{T_3}^{-1} \circ E_{T_2}$ as a statistically independent sample.

here. We can construct a 30-bit filter with a success probability of 98.9 %. In other words, we can uniquely recover the key candidates for the last 30 bits of K^1 with a high probability.

Algorithm 1 shows the detailed attack procedure. The algorithm requires $2 \times 31 \times 2^{20}$ memory and $2 \times 2^{30} \times 2^{29} \times 31 \times 2^{10} = 2^{74.95}$ time.

Step 2: Partial Key Recovery Using (5S+5S)-Round Differential Distinguisher. We next recover the bottom 30 bits of K^1 and the 5 bits of K^2. Note that we already know the top 30 bits of K^1 from Step 1.

Instead of the (6+6)-round distinguisher, we introduce a (5S+5S)-round distinguisher, where the S-box layer is added to the (5+5)-round distinguisher. By applying the S-box transition probability from the EDPs of 5+5 rounds, we can estimate these biases. As a result, for any δ with Hamming weight 1, we have

$$\mathrm{Prob}[(\delta, 0) \xrightarrow{5S+5S} (\delta, 0)] = \frac{1}{1023} + 2^{-34.17}. \tag{4}$$

When the Hamming weight of δ is 1, the G function in the 3rd round involves only 5 bits of the subkey, and they are computed by guessing the 30-bit K^1 and the 5-bit K^2 when the tweak is active. Using 2^{67} pairs is enough to construct a 35-bit filter with a success probability of almost one. The time complexity is $2^{35} \times 2^{40} = 2^{75}$.

We experimentally verified the differential bias of Eq. (4) by using $2^{26} \times 2^{26} \times 2^9 = 2^{61}$ pairs and were repeated 10 times. As a result, we observed $2^{-34.29}$ of differential bias experimentally.

Step 3: Full Key Recovery. A single δ in Step 2 is enough to recover the entire K^1. Therefore, we can compute the first $\Sigma \circ \mathsf{SL}$ layer in the tweakey schedule for arbitrary tweaks. This implies that the first two rounds, including the tweakey schedule, are peeled off. While we do not explicitly show the procedure to recover K^2, K^3, and K^4, we can recover these keys by auxiliary procedures because it should be easier than recovering K^1.

In summary, with a complexity of about 2^{76}, we can recover the 240-bit key with a success probability of 98.9 %.

5.2 The Case of the Security Requirement 1

When we consider the security requirement 1 instead of the security requirement 2, the number of available pairs is further limited. As discussed in [11], we need approximately 2^{18} queries to collect the full-code book data. Therefore, 2^{40} queries allow us to collect the full code book for 2^{22} different tweaks. As a result, we only have $2^{21+21+9} = 2^{51}$ pairs. The distinguishing advantage of the attacker is not enough in this case to filter the wrong keys.

6 Multi-key Distinguishing Attacks on Full SCARF

The highest differential bias for full-round SCARF is $2^{-52.34}$. To detect this bias, $2^{52.34\times 2}/1024 = 2^{94.68}$ pairs are needed. As the security requirements for SCARF restrict the number of queries, it is unlikely to collect such a large number of pairs. If we instead use the LLR statistic, the capacity is $2^{-88.95}$. This means that the number of needed pairs is approximately $2^{88.95}$, and it is still unlikely to collect them due to the restriction on the number of queries.

Nevertheless, in this section, we discuss the advantage of the distinguishing algorithm against the full-round SCARF. This discussion is motivated by the state-of-the-art theory regarding the definition of *bit security* [17,23,24], which defines *bit security* based on the adversary's time and advantage.

6.1 Time and Advantage of the Distinguishing Algorithm

When we compute the LLR statistic on the full-round SCARF, we do not need to guess the key. Therefore, a simplification of Algorithm 1 allows us to compute the LLR statistic with a time complexity of $31 \times 2^{40} \approx 2^{44.95}$. The capacity of the differential bias for the full-round SCARF is $2^{-88.95}$. When we compute the LLR statistic using 2^{67} pairs, after adjusting the average by subtracting from it $(\mu_0 + \mu_1)/2$, each distribution has

$$\text{Real} \sim \mathcal{N}(2^{-22.95}, 2^{-21.95}) \qquad \text{Ideal} \sim \mathcal{N}(-2^{-22.95}, 2^{-21.95}).$$

It is unlikely to distinguish these two worlds in practice because $\sigma_0 \sim \sigma_1 \gg \mu_0 - \mu_1$. However, we do have a non-negligible distinguishing advantage. When we estimate the probability that the statistic is higher than $-2^{-22.95}$, it is 0.5 in the ideal case but 0.5001982 in the real case. This implies that the (traditional) distinguishing advantage is

$$0.5001982 - 0.5 = 0.0001982 \approx 2^{-12.30}.$$

Micciancio and Walter defined the notion of *bit security* from the adversary's time, $T(A)$, and advantage, adv^A [17].

Definition 4 (Bit Security of a Primitive [17]**).** Let $T(A)$ be the time complexity of the algorithm A, that is linear under repetition. For any primitive, its bit security is defined as $\min_A \log \frac{T(A)}{adv^A}$.

Here, adv^A is defined by the Shannon entropy and mutual information. In our attack scenario, $adv^A = (2 \times (0.5 + 2^{-12.30}) - 1)^2 = 2^{-22.60}$. Therefore, the bit security of SCARF is defined as

$$44.95 + 22.60 = 67.55.$$

It is significantly less than 80. Therefore, we can conclude that SCARF does not provide an 80-bit security in the context of [17].

Watanabe and Yasunaga also defined a notion of *bit security* as the computational cost of winning the game [23]. Later, they showed that the two definitions are equivalent [24].

6.2 Multi-key Distinguisher

On the other hand, it is not easy to conclude that the above observation implies a security claim break. The definition of [17] does not provide an explicit attack algorithm that wins a distinguishing game with high probability by the computational cost of the bit security. The definition in [23] provides an explicit attack algorithm, but their distinguishing game accesses multiple encryption oracles while re-keying the oracle. Finally, the adversary wins the distinguishing game by combining all knowledge gained from the oracles with multiple keys. Such an attack setting can be classified as a *multi-key model* rather than a *single-key model*. Indeed, we have the following concrete attack procedure to distinguish the full SCARF in the multi-key setting.

1. The attacker accesses an oracle and evaluates the LLR statistic. If it is higher than μ_1, the score is increased. Otherwise, the score remains unchanged.
2. The attacker re-keys the oracle and repeats Step 1, $c \times 2^{22.60}$ times, for some constant c.
3. Check the score. If it is significantly higher than $c \times 2^{21.60}$, the attacker returns 1. Otherwise, he returns 0.

The procedure above distinguishes SCARF from the ideal with a complexity of $c \times 2^{44.95+22.60} = c \times 2^{67.55}$. It is clear that SCARF does not provide a 80-bit security in the multi-key setting[2].

On the contrary, to the best of our knowledge, we do not have an explicit algorithm to win the distinguishing game with this bit security notion in a single-key model. This is different from a key-recovery attack with a low success probability. In a key-recovery attack with a success probability p, we might repeat the attack procedure p^{-1} times while re-keying. However, unlike the multi-key attack, we do not need to combine all knowledge from the re-keyed oracles.

The Case of the Security Requirement 1. When we consider the Security Requirement 1, the number of available pairs is reduced to 2^{51}. Then, the traditional distinguishing advantage is

$$0.500000774 - 0.5 = 0.000000774 \approx 2^{-20.30}.$$

The adversary needs at least 2^{40} times as the query complexity. The time to compute the LLR statistics is negligible because it is $2 \times 2^{21} \times 31 \times 2^{10} \approx 2^{36.95}$. Therefore, the bit security is $40+19.30\times 2 = 78.6$. Even for Security Requirement 1, SCARF does not provide an 80-bit security in the multi-key setting.

[2] We also have another procedure, where instead of computing the LLR statistics every time, we collect pairs while re-keying. We have 2^{67} pairs for each key. Therefore, roughly, $2^{88.95-67} = 2^{21.95}$ re-keys are enough to collect $2^{88.95}$ pairs. As a result, both procedures successfully distinguish the full SCARF with almost the same complexity on the multi-key setting.

6.3 Discussion and Open Questions

The existing definition of bit security and our multi-key distinguishing attack prompt several discussions and open questions in both practice and theory.

First, should we distinguish between the single-key and multi-key models? Currently, the existing definition of bit security [17,23,24] does not make this distinction. However, these two attack models are regarded as different in practice [18]. We presented one such example. Bridging this gap between theory and practice is an interesting open question.

From a practical perspective, the question is whether we have an explicit distinguishing attack in the single-key model with even a slight advantage. If such an attack is found, it would imply that the security requirements of SCARF would be broken. It may be beneficial to redefine bit security by distinguishing between the single-key and multi-key models. This redefinition would be useful for primitives whose claimed security level exceeds the block length and where the number of queries is limited.

Finally, does our conclusion that SCARF does not provide 80-bit security in the multi-key setting pose a practical problem for its use case? SCARF is used to counter (contention-based) cache attacks. With each exhaustion of the limited queries, the key is re-keyed, creating an environment where a multi-key distinguishing attack could be executed in practice. However, the attacker's true goal is to efficiently construct a victim set to mount the cache attack. Whether the existence of a multi-key distinguisher aids in constructing these victim sets remains an open question.

7 Impact of the S-Box Choice on the Differential Bias

The differential properties of the S-box play an important role in the multiple-tweak differential attack and influence the magnitude of the differential bias. The goal of this section is to provide a deeper understanding of the role that the S-box plays in the attack and to discuss the impact of choosing a different S-box would have on the bias.

7.1 Impact of the S-Box on the Bias for 2+2 and 3+3 Rounds

As shown in Sect. 4, the differential bias for 1+1 rounds depends on the properties of the function $S^{-1} \circ S(\cdot \oplus k)$. More precisely, for a couple of differences (α, β), the bias depends on the number of solutions of the equation :

$$S^{-1}(S(x) \oplus k) \oplus S^{-1}(S(x \oplus \alpha) \oplus k) = \beta, \tag{5}$$

for all couples $(x, k) \in \mathbb{F}_2^5 \times \mathbb{F}_2^5$. This number of solutions for all possible differences α and β is given in Table 7 of Appendix A.

We provide now an analysis of the number of solutions of Eq. (5) in the case $\alpha = \beta$. Then, we analyze the more general case $\alpha \neq \beta$.

Case $\alpha = \beta$. We show that in the case where $\alpha = \beta$ the elements of Table 7, are directly related to a classical table, called the Boomerang Connectivity Table (BCT) [12], used to estimate the power of boomerang attacks at the S-box level.

For an n-bit invertible S-box S, the BCT of S is a $2^n \times 2^n$ table, denoted by B_S and defined as follows:

$$B_S(\alpha, \beta) = \#\{x \in \mathbb{F}_2^n : S^{-1}(S(x) \oplus \beta) \oplus S^{-1}(S(x \oplus \alpha) \oplus \beta) = \alpha\}.$$

Let's see how the number of solutions of Eq. (5) is related to the BCT of S in the case where $\alpha = \beta$:

$$\begin{aligned}&\#\{(x,k) \in \mathbb{F}_2^5 \times \mathbb{F}_2^5 : S^{-1}(S(x) \oplus k) \oplus S^{-1}(S(x \oplus \alpha) \oplus k) = \alpha\}\\ =&\sum_{k=0}^{2^5-1} \#\{x \in \mathbb{F}_2^5 : S^{-1}(S(x) \oplus k) \oplus S^{-1}(S(x \oplus \alpha) \oplus k) = \alpha\} = \sum_{k=0}^{2^5-1} B_S(\alpha, k).\end{aligned}$$

The above computation shows that for a given α the number of solutions of Eq. (5) is just the sum of all the elements of the BCT of S on the row α.

Case $\alpha \neq \beta$. We define for any $k \in \mathbb{F}_2^5$, the permutation S_k that maps x to $S^{-1}(S(x) + k)$. Then,

$$\begin{aligned}&\#\{(x,k) \in \mathbb{F}_2^5 \times \mathbb{F}_2^5 : S^{-1}(S(x) \oplus k) \oplus S^{-1}(S(x \oplus \alpha) \oplus k) = \beta\}\\ =&\#\{(x,k) \in \mathbb{F}_2^5 \times \mathbb{F}_2^5 : S_k(x) \oplus S_k(x \oplus \alpha) = \beta\}\\ =&\sum_{k=0}^{2^5-1} \mathrm{DDT}_{S_k}(\alpha, \beta).\end{aligned}$$

We see from the above equation, that in the case $\alpha \neq \beta$ we obtain a quite different interpretation for the elements in Table 7. Indeed, the elements of this table that are not on the diagonal can be interpreted as the sum, for all the 2^5 different functions k of the S_k's Difference Distribution Table (DDT) coefficient at row α and column β.

We can observe from Table 7 that the coefficients on the diagonal (corresponding to the case $\alpha = \beta$) have much higher values than the other coefficients. This could be potentially explained by the fact that in general, BCTs have higher coefficients than DDTs. However, obtaining a more precise explanation is hard, as the correlation between the DDTs of the different functions S_k and their relation to the BCT of S is unclear.

We investigate now the role that plays the S-box in the biases for 2+2 and 3+3 rounds. For this, we explicitly provide the EDPs for this number of rounds by using the formulas given in Sect. 4.3.

For any non-zero α and β from $\mathbb{F}_2^5$, we have for 2+2 rounds:

$$\mathrm{EDP}[(0, \alpha) \xrightarrow{2+2} (0, \beta)] = 2^{-5} \times P_{\tilde{S}}(\alpha, \beta)$$

The formulas for 3+3 rounds are as follows:

$$\mathrm{EDP}[(0, \alpha) \xrightarrow{3+3} (0, \beta)] = 2^{-10} \times (1 - 2^{-5}) + 2^{-10} \times P_{\tilde{S}}(\alpha, \beta),$$

It can be seen from the above formulas that the differential properties of the function $S^{-1} \circ S(\cdot \oplus k)$ influence the bias for 2+2 and 3+3 rounds. More precisely, as in the analysis of 1+1 rounds, the BCT of S plays a role in the case where $\alpha = \beta$. The higher the boomerang uniformity (the maximal non-trivial value in a BCT), the higher the bias is expected to be in the differential transitions $(0, \alpha) \xrightarrow{R+R} (0, \beta)$, for $R = 1, 2, 3$.

Other S-Boxes. It is interesting to see at this point, what the bias would have been if a different S-box than the original one of SCARF had been used instead. It is important to notice that contrary to the more classical 4-bit or 8-bit S-boxes, very few 5-bit S-boxes are used in the literature. However, in odd dimension, and contrary to the 4-bit and 8-bit cases, there exist S-boxes that ensure optimal resistance to differential cryptanalysis. Such S-boxes are called Almost Perfect Nonlinear (APN). These are S-boxes that have only 0 and 2 in their Difference Distribution Table (DDT). What is also known about these S-boxes is that their BCT is also optimal which means it also has only the elements 0 and 2 inside (see for example [12] or [10]). Further, we know that the sum of all elements in the BCT of an n-bit APN S-box equals 2^{n+1}. This explains why in the case of a 5-bit APN S-box, in the case $\alpha = \beta$, the number of solutions of Eq. (5) is always equal to $2^{5+1} = 64$.

As can be seen from Table 7, the values on the diagonal for the SCARF S-box are higher than 64, which means that if an APN S-box had been used instead of the original SCARF S-box, this would have led to much lower biases for 2+2 and 3+3 rounds in the case of $\alpha = \beta$. On the other hand, there are other popular choices for a 5-bit S-box that would have produced for some values a much higher bias than the one observed for SCARF. This is notably the case of the 5-bit S-box used in `Ascon` [13], which is affine equivalent to the χ-mapping used in `Keccak` [5]. The `Ascon` S-box has a differential uniformity of 8 (maximal non-trivial coefficient in the DDT) and a boomerang uniformity of 16 (maximal non-trivial coefficient in the BCT). In comparison, the same numbers for the SCARF S-box are 4 and 6 respectively. Table 4 provides the theoretical bias obtained by the corresponding EDPs for all differences $\alpha = \beta = (0, \delta)$ for the original S-box of SCARF, the S-box of `Ascon` and a 5-bit APN S-box.

7.2 Analysis for a Higher Number of Rounds

For a higher number of rounds, we expect that the difference distribution table (DDT) of the S-box and specifically its interaction with the properties of $S^{-1} \circ S(\cdot \oplus k)$ plays an important role in the estimation of the bias. This can be seen from the formulas given in Sect. 4.3 where the probabilities of differential transitions over S clearly appear and are multiplied with the EDP for the transition over a smaller number of rounds, where we showed the function $S^{-1} \circ S(\cdot \oplus k)$ to play a role.

Understanding better the interaction of the differential properties of those two functions and their influence on the bias is an interesting open question.

We computed the theoretical bias for up to 8+8 rounds for the original SCARF S-box, the `Ascon` S-box and the 5-bit APN used inside the cipher

Table 4. Bias (in $\log_2$ representation) for $3+3$ rounds of SCARF with an APN S-box, with the ASCON S-box and the original S-box. All biases appear with the $(+)$ sign.

δ	APN	ASCON	SCARF	δ	APN	ASCON	SCARF
0x1	-15.046	-13.430	-14.715	0x11	-15.046	-12.199	-14.715
0x2	-15.046	-13.430	-14.715	0x12	-15.046	-15.046	-14.715
0x3	-15.046	-13.430	-14.715	0x13	-15.046	-12.199	-14.445
0x4	-15.046	-12.199	-14.715	0x14	-15.046	-13.430	-14.715
0x5	-15.046	-13.430	-14.715	0x15	-15.046	-13.430	-14.445
0x6	-15.046	-15.046	-14.715	0x16	-15.046	-15.046	-14.445
0x7	-15.046	-13.430	-14.445	0x17	-15.046	-13.430	-14.715
0x8	-15.046	-13.430	-14.715	0x18	-15.046	-15.046	-14.715
0x9	-15.046	-15.046	-14.715	0x19	-15.046	-13.430	-14.445
0xa	-15.046	-15.046	-14.715	0x1a	-15.046	-15.046	-14.445
0xb	-15.046	-15.046	-14.445	0x1b	-15.046	-15.046	-14.715
0xc	-15.046	-12.199	-14.715	0x1c	-15.046	-13.430	-14.445
0xd	-15.046	-15.046	-14.445	0x1d	-15.046	-13.430	-14.715
0xe	-15.046	-13.430	-14.445	0x1e	-15.046	-15.046	-14.715
0xf	-15.046	-13.430	-14.715	0x1f	-15.046	-13.430	-13.850
0x10	-15.046	-12.199	-14.715				

Table 5. Theoretical capacity for each number of rounds when we replace the S-box by S_{alt}, where C and C_{all} are computed as in Eq. (1) and Eq. ((2), respectively.

capacity	2+2	3+3	4+4	5+5	6+6	7+7	8+8
$\log_2(C)$	-3.29	-13.30	-32.43	-42.43	-63.29	-73.29	-93.49
$\log_2(C_{all})$	-2.30	-13.30	-31.43	-42.43	-62.29	-73.29	-92.49

Fides [7]. We observed that for these computations for many difference values, the bias was higher for Ascon than for SCARF and the APN S-box. This can be explained by the fact that the differential spectrum of the S-box of Ascon (the set of all the elements in the DDT) has much higher values than the differential spectrum of the SCARF S-box and of course of any APN S-box.

7.3 Searching for Alternative S-Boxes for SCARF

An interesting question is whether we can replace the original SCARF S-box with a different 5-bit S-box such that the bias after 8+8 rounds is lower than for the original SCARF. Of course, the new S-box has to follow the same design criteria as the original one. More precisely, it must ensure the low latency of the global design, i.e., the maximum gate depth using 2-bit NAND, 2-bit NOR and INV gates should be 4. It must also have the same cryptographic properties as

Table 6. Experimental comparison for 4+4 rounds with four S-boxes. We obtained these results using 2^{35} pairs with 5000 repetitions. Note that to compute μ_1, we did experiments using a random 10-bit tweakable block cipher.

	capacity	theoretical estimations		experimental results	
		$\mu_1 - \mu_0$	σ_1^2	$\mu_1 - \mu_0$	σ_1^2
$S_{orig.}$	$2^{-32.20}$	6.98	6.98	7.18	7.09
$S_{\mathtt{Fides}}$	$2^{-35.09}$	0.94	0.94	1.00	0.96
$S_{\mathtt{Ascon}}$	$2^{-25.89}$	533.75	533.82	553.12	682.20
$S_{alt.}$	$2^{-32.43}$	5.95	5.95	5.85	5.92

the original S-box, which means having a differential uniformity of at most 4, a linearity of at most 12 and a maximal algebraic degree, i.e., 4.

Using the tool of [22], given the above criteria and a restriction on the coordinate functions to be extended bit-permutation equivalent of each other, we found 1016 S-box representatives up to equivalence. Note that in the SCARF original S-box, all the coordinates are the same function and the inputs for each coordinate is just a rotation of the inputs for the first coordinate. However, in this search, by fixing all the coordinates to use the same function, we allow the inputs to be any permutation of the inputs for the first coordinate along with a constant addition.

Applying bit-permutations, one at the input and one at the output of an S-box, does not change the aforementioned criteria, but it can affect the capacity. However, using two bit-permutations, P_0 and P_1, on either side of the S-box will result in the same capacity as using the combined bit-permutation $P_0 \circ P_1$ on just one side of the S-box. Therefore, we explored $1016 \times 5!$ S-boxes and estimated the capacity using our tool as described in Sect. 4. Among them, the one with table representation

$$\begin{aligned} S_{alt} = [&\mathtt{00}, \mathtt{01}, \mathtt{03}, \mathtt{0D}, \mathtt{06}, \mathtt{13}, \mathtt{16}, \mathtt{0F}, \mathtt{19}, \mathtt{10}, \mathtt{0B}, \mathtt{17}, \mathtt{09}, \mathtt{1D}, \mathtt{1A}, \mathtt{1C}, \\ &\mathtt{1E}, \mathtt{0C}, \mathtt{15}, \mathtt{04}, \mathtt{08}, \mathtt{1B}, \mathtt{11}, \mathtt{0A}, \mathtt{1F}, \mathtt{14}, \mathtt{12}, \mathtt{02}, \mathtt{05}, \mathtt{07}, \mathtt{18}, \mathtt{0E}] \end{aligned}$$

achieves the best security against the multiple-tweak differential attack in 8+8 rounds. Table 5 summarizes C and C_{all} when we replace the S-box with S_{alt}. Interestingly, S_{alt} is worse than the original S-box up to 3+3 rounds but improves the security from 4+4 rounds. Finally, the theoretical capacity reaches $2^{-93.49}$ for 8+8 rounds, which is better than $2^{-88.9514}$ for the original S-box.

We finally revisit the bit security when we replace the S-box. On the security requirement 2, we can collect $2^{29} \times 2^{29} \times 2^{9} = 2^{67}$ pairs. Then, the distinguishing advantage is about $2^{-14.57}$. Therefore, the bit security is $44.95 + 13.57 \times 2 = 72.09$. Although it is improved, the bit security is still lower than 80. In other words, the multi-key distinguishing attack still works even if we replace the S-box with S_{alt}. The security requirement 1 limits the number of pairs that can be collected by 2^{51}. Then, the distinguishing advantage is 22.57. Thus, we have $40 + 21.57 \times 2 = 83.14$, which is larger than 80.

7.4 Experiments

We provide an experimental verification for our discussion. We compared differential capacities for 4+4 rounds among four S-boxes, $S_{orig.}$, $S_{alt.}$, $S_{\texttt{Ascon}}$, and $S_{\texttt{Fides}}$, where $S_{\texttt{Ascon}}$ and $S_{\texttt{Fides}}$ are

$$\begin{aligned} S_{\texttt{Ascon}} =& [\texttt{04},\texttt{0b},\texttt{1f},\texttt{14},\texttt{1a},\texttt{15},\texttt{09},\texttt{02},\texttt{1b},\texttt{05},\texttt{08},\texttt{12},\texttt{1d},\texttt{03},\texttt{06},\texttt{1c},\\ & \texttt{1e},\texttt{13},\texttt{07},\texttt{0e},\texttt{00},\texttt{0d},\texttt{11},\texttt{18},\texttt{10},\texttt{0c},\texttt{01},\texttt{19},\texttt{16},\texttt{0a},\texttt{0f},\texttt{17}],\\ S_{\texttt{Fides}} =& [\texttt{01},\texttt{00},\texttt{19},\texttt{1a},\texttt{11},\texttt{1d},\texttt{15},\texttt{1b},\texttt{14},\texttt{05},\texttt{04},\texttt{17},\texttt{0e},\texttt{12},\texttt{02},\texttt{1c},\\ & \texttt{0f},\texttt{08},\texttt{06},\texttt{03},\texttt{0d},\texttt{07},\texttt{18},\texttt{10},\texttt{1e},\texttt{09},\texttt{1f},\texttt{0a},\texttt{16},\texttt{0c},\texttt{0b},\texttt{13}]. \end{aligned}$$

For all S-boxes, our experimental results almost match the theoretical estimations based on the EDP. As expected, $S_{\texttt{Fides}}$ is the most secure, $S_{\texttt{Ascon}}$ is weak, and S_{alt} is superior to the original S-box, $S_{orig.}$ (Table 6).

8 Conclusion

In this work, we provided the first third-party cryptanalysis of the tweakable block cipher SCARF, by means of multiple-tweak differential cryptanalysis. We first provided a theoretical framework to compute the bias of the differential transitions for any number of rounds and confirmed the theory by experimental verification for up to 5 rounds. We then mounted a key recovery attack on 7 out of 8 rounds of the cipher. In parallel, we showed distinguishing attacks on full 8-round SCARF in the multi-key setting that demonstrate that in this particular scenario the cipher does not offer the claimed 80-bit security. An interesting open question is what could be said in the single-key model, and to answer this question the actual notion of bit security should probably be re-defined. Finally, the practical impact of SCARF's vulnerabilities in real-world applications, such as its effectiveness against cache attacks under realistic conditions, needs further exploration to understand whether there is a true risk posed by multi-key distinguishers in constructing victim sets for cache attacks. Last, we analyzed the role that the differential properties of the S-box play in this attack. We showed in particular that it is possible to replace the original S-box of SCARF with a different one that follows the same design criteria as the S-box of SCARF but that offers a higher resistance against multiple-tweak differential attacks. However, we also showed that the replacement of the S-box with the best possible variant would still not prevent the distinguisher on 8+8 rounds in the multi-key setting.

Acknowledgements. Christina Boura was partially supported by the French Agence Nationale de la Recherche through the SWAP project under Contract ANR-21-CE39-0012 and through the France 2030 program under grant agreement No. ANR-22-PECY-0010. Besides, Shahram Rasoolzadeh is funded by the ERC project 101097056 (SYMTRUST) and Dhiman Saha was supported by the Science and Engineering Research Board (SERB), Department of Science and Technology, Government of India, under MATRICS scheme File No. MTR/2023/001161. Finally, the authors would like to thank the Dagstuhl Seminar 24041 on Symmetric Cryptography that gave the opportunity to the authors to initiate this collaboration.

A Special DDT for $S^{-1} \circ S$

Table 7 provides the number of solutions of the equation $\#\{(x, k) \in \mathbb{F}_2^5 \times \mathbb{F}_2^5 \mid S^{-1}(S(x) \oplus k) \oplus S^{-1}(S(x \oplus \alpha) \oplus k) = \beta\}$. It can be seen as a special form of DDT for the function $S^{-1} \circ S$, where the key is taken into account.

Table 7. $\#\{(x, k) \in \mathbb{F}_2^5 \times \mathbb{F}_2^5 \mid S^{-1}(S(x) \oplus k) \oplus S^{-1}(S(x \oplus \alpha) \oplus k) = \beta\}$

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
00	2^{10}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	0	72	28	20	36	36	32	32	36	20	40	36	32	24	40	28	28	32	40	28	36	28	24	32	32	40	32	28	36	40	24	32
02	0	28	72	32	28	40	20	28	36	36	36	28	32	24	32	32	36	32	20	40	40	32	36	28	32	36	24	40	40	24	28	32
03	0	20	32	72	32	28	40	24	32	28	24	32	44	32	40	32	32	40	32	36	24	32	32	44	44	32	32	32	28	24	28	20
04	0	36	28	32	72	20	32	40	28	40	40	32	20	36	28	28	36	32	36	36	36	24	28	40	32	40	24	24	32	28	32	32
05	0	36	40	28	20	72	32	24	36	32	28	28	28	40	36	24	40	24	28	32	32	48	28	36	24	24	32	32	36	28	40	36
06	0	32	20	40	32	32	72	36	32	24	28	32	40	32	24	44	32	44	28	32	24	32	32	32	44	28	32	24	40	28	32	20
07	0	32	28	24	40	24	36	80	36	36	32	28	32	28	40	24	40	40	36	40	24	20	32	32	28	24	20	36	24	36	24	48
08	0	36	36	32	28	36	32	36	72	36	20	24	32	28	40	40	28	32	40	40	40	24	32	24	20	32	36	28	28	32	28	32
09	0	20	36	28	40	32	24	36	36	72	32	40	24	48	24	28	40	32	28	36	28	28	32	40	28	24	28	24	32	36	32	36
0A	0	40	36	24	40	28	28	32	20	32	72	48	32	28	24	36	36	24	32	24	28	32	28	32	28	36	40	28	36	40	24	36
0B	0	36	28	32	32	28	32	28	24	40	48	80	32	24	20	36	24	32	28	32	32	40	40	36	32	28	24	40	20	48	24	24
0C	0	32	32	44	20	28	40	32	32	24	32	32	72	32	36	32	32	44	24	28	28	32	32	24	40	40	32	28	24	32	44	20
0D	0	24	24	32	36	40	32	28	28	48	28	24	32	80	32	40	32	32	32	20	28	24	40	48	32	20	40	24	28	36	36	24
0E	0	40	32	40	28	36	24	40	40	24	24	20	36	32	80	32	36	28	36	24	32	20	28	36	32	24	28	36	40	24	24	48
0F	0	28	32	32	28	24	44	24	40	28	36	36	32	40	32	72	24	28	40	24	32	24	36	28	24	36	48	32	36	32	28	24
10	0	28	36	32	36	40	32	40	28	40	36	24	32	32	36	24	72	20	36	32	20	36	24	28	32	28	28	32	40	28	40	32
11	0	32	32	40	32	24	44	40	32	32	24	32	44	32	28	28	20	72	28	24	28	32	32	32	40	36	32	44	32	32	24	20
12	0	40	20	32	36	28	28	36	40	28	32	28	24	32	36	40	36	28	72	24	32	28	40	24	24	32	48	36	24	32	28	36
13	0	28	40	36	36	32	32	40	40	36	24	32	28	20	24	24	32	24	24	80	36	28	28	24	40	40	20	32	24	36	36	48
14	0	36	40	24	36	32	24	24	40	28	28	32	28	28	32	32	20	28	32	36	72	40	48	28	32	36	28	40	24	24	36	36
15	0	28	32	32	24	48	32	20	24	28	32	40	32	24	20	24	36	32	28	28	40	80	24	36	32	32	40	36	28	40	48	24
16	0	24	36	32	28	28	32	32	32	32	28	40	32	40	28	36	24	32	40	28	48	24	80	40	32	20	24	48	20	24	36	24
17	0	32	28	44	40	36	32	32	24	40	32	36	24	48	36	28	28	32	24	24	28	36	40	72	28	24	24	28	36	32	32	24
18	0	32	32	44	32	24	44	28	20	28	28	32	40	32	32	24	32	40	24	40	32	32	32	28	72	24	32	32	36	44	32	20
19	0	40	36	32	40	24	28	24	32	24	36	28	40	20	24	36	28	36	32	40	36	32	20	24	24	80	28	24	40	32	36	48
1A	0	32	24	32	24	32	32	20	36	28	40	24	32	40	28	48	28	32	48	20	28	40	24	24	32	28	80	36	32	36	40	24
1B	0	28	40	32	24	32	24	36	28	24	28	40	28	24	36	32	32	44	36	32	40	36	48	28	32	24	36	72	24	28	32	24
1C	0	36	40	28	32	36	40	24	28	32	36	20	24	28	40	36	40	32	24	24	24	28	20	36	36	40	32	24	80	24	32	48
1D	0	40	24	24	28	28	28	36	32	36	40	48	32	36	24	32	28	32	32	36	24	40	24	32	44	32	36	28	24	72	28	24
1E	0	24	28	28	32	40	32	24	28	32	24	24	44	36	24	28	40	24	28	36	36	48	36	32	32	36	40	32	32	28	72	24
1F	0	32	32	20	32	36	20	48	32	36	36	24	20	24	48	24	32	20	36	48	36	24	24	24	20	48	24	24	48	24	24	104

References

1. Advanced encryption standard (AES). National Institute of Standards and Technology. NIST FIPS PUB 197, U.S. Department of Commerce (2001)
2. Baignères, T., Junod, P., Vaudenay, S.: How far can we go beyond linear cryptanalysis? In: Lee, P.J. (ed.) ASIACRYPT 2004. Lecture Notes in Computer Science, vol. 3329, pp. 432–450. Springer (2004)
3. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer (2016)
4. Belkheyar, Y., Daemen, J., Dobraunig, C., Ghosh, S., Rasoolzadeh, S.: BipBip: A low-latency tweakable block cipher with small dimensions. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(1), 326–368 (2023)
5. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak reference. Submission to NIST (Round 3) (2011), https://keccak.team
6. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO '90. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer (1990). https://doi.org/10.1007/3-540-38424-3_1, https://doi.org/10.1007/3-540-38424-3_1
7. Bilgin, B., Bogdanov, A., Knezevic, M., Mendel, F., Wang, Q.: Fides: Lightweight authenticated cipher with side-channel resistance for constrained hardware. In: Bertoni, G., Coron, J. (eds.) CHES 2013. Lecture Notes in Computer Science, vol. 8086, pp. 142–158. Springer (2013)
8. Blondeau, C., Gérard, B., Nyberg, K.: Multiple differential cryptanalysis using LLR and χ 2 statistics. In: Visconti, I., Prisco, R.D. (eds.) SCN 2012. Lecture Notes in Computer Science, vol. 7485, pp. 343–360. Springer (2012)
9. 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., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. Lecture Notes in Computer Science, vol. 7658, pp. 208–225. Springer (2012)
10. Boura, C., Canteaut, A.: On the boomerang uniformity of cryptographic Sboxes. IACR Trans. Symmetric Cryptol. **2018**(3), 290–310 (2018)
11. Canale, F., Güneysu, T., Leander, G., Thoma, J.P., Todo, Y., Ueno, R.: SCARF - A low-latency block cipher for secure cache-randomization. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX 2023. pp. 1937–1954. USENIX Association (2023)
12. Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang connectivity table: A new cryptanalysis tool. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. Lecture Notes in Computer Science, vol. 10821, pp. 683–714. Springer (2018)
13. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2: Lightweight authenticated encryption and hashing. J. Cryptol. **34**(3), 33 (2021)
14. Dunkelman, O., Kumar, A., Lambooij, E., Sanadhya, S.K.: Cryptanalysis of Feistel-based format-preserving encryption. IACR Cryptol. ePrint Arch. p. 1311 (2020), https://eprint.iacr.org/2020/1311
15. Lai, X., Massey, J.L., Murphy, S.: Markov ciphers and differential cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT '91. Lecture Notes in Computer Science, vol. 547, pp. 17–38. Springer (1991)

16. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT '93. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer (1993). https://doi.org/10.1007/3-540-48285-7_33
17. Micciancio, D., Walter, M.: On the bit security of cryptographic primitives. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. Lecture Notes in Computer Science, vol. 10820, pp. 3–28. Springer (2018)
18. Mouha, N., Luykx, A.: Multi-key security: The Even-Mansour construction revisited. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 209–223. Springer (2015)
19. Patarin, J.: Generic attacks on Feistel schemes. In: Boyd, C. (ed.) ASIACRYPT 2001. Lecture Notes in Computer Science, vol. 2248, pp. 222–238. Springer (2001)
20. Patarin, J.: Security of random Feistel schemes with 5 or more rounds. In: Franklin, M.K. (ed.) CRYPTO 2004. Lecture Notes in Computer Science, vol. 3152, pp. 106–122. Springer (2004)
21. Patarin, J.: Generic attacks on Feistel schemes. IACR Cryptol. ePrint Arch. p. 36 (2008), http://eprint.iacr.org/2008/036
22. Rasoolzadeh, S.: Low-latency Boolean functions and bijective S-boxes. IACR Trans. Symmetric Cryptol. **2022**(3), 403–447 (2022)
23. Watanabe, S., Yasunaga, K.: Bit security as computational cost for winning games with high probability. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part III. Lecture Notes in Computer Science, vol. 13092, pp. 161–188. Springer (2021)
24. Watanabe, S., Yasunaga, K.: Unified view for notions of bit security. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VI. Lecture Notes in Computer Science, vol. 14443, pp. 361–389. Springer (2023)

Generic Differential Key Recovery Attacks and Beyond

Ling Song[1], Huimin Liu[1], Qianqian Yang[2,3](✉), Yincen Chen[1], Lei Hu[2,3], and Jian Weng[1]

[1] College of Cyber Security, Jinan University, Guangzhou, China
[2] Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{yangqianqian,hulei}@iie.ac.cn
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. At Asiacrypt 2022, a holistic key guessing strategy was proposed to yield the most efficient key recovery for the rectangle attack. Recently, at Crypto 2023, a new cryptanalysis technique–the differential meet-in-the-middle (MITM) attack–was introduced. Inspired by these two previous works, we present three generic key recovery attacks in this paper. First, we extend the holistic key guessing strategy from the rectangle to the differential attack, proposing the generic classical differential attack (GCDA). Next, we combine the holistic key guessing strategy with the differential MITM attack, resulting in the generalized differential MITM attack (GDMA). Finally, we apply the MITM technique to the rectangle attack, creating the generic rectangle MITM attack (GRMA). In terms of applications, we improve 12/13-round attacks on AES-256. For 12-round AES-256, by using the GDMA, we reduce the time complexity by a factor of 2^{62}; by employing the GCDA, we reduce both the time and memory complexities by factors of 2^{61} and 2^{56}, respectively. For 13-round AES-256, we present a new differential attack with data and time complexities of 2^{89} and 2^{240}, where the data complexity is 2^{37} times lower than previously published results. These are currently the best attacks on AES-256 using only two related keys. For KATAN-32, we increase the number of rounds covered by the differential attack from 115 to 151 in the single-key setting using the basic differential MITM attack (BDMA) and GDMA. Furthermore, we achieve the first 38-round rectangle attack on SKINNYe-64-256 v2 by using the GRMA.

Keywords: Differential cryptanalysis · Rectangle attack · Meet-in-the-middle · Key recovery · AES · KATAN · SKINNYe

1 Introduction

Differential cryptanalysis, which was introduced by Biham and Shamir [BS90, BS91], is one of the most powerful cryptanalytic approaches for assessing the security of block ciphers. The basic idea is to exploit non-random propagation of input difference to output difference, *i.e.*, high-probability differentials. The first step in mounting a differential attack is to find a high-probability differential covering a large number of rounds. This procedure has been extensively studied, and many approaches have been proposed [Mat94, MWGP11, MP13, SHW+14, SWW21]. Once an r-round high-probability differential of a certain cipher has been found, one could add some outer rounds and restrict the possible values of key bits in the outer rounds. Indeed, the right key of the outer rounds will reveal the non-randomness of the differential.

Since the introduction of differential cryptanalysis, many improvements have been proposed for the key recovery: structures of data [BS92], conditional differentials [KMN10], probabilistic neutral bits [AFK+08], the early abort technique [LKKD08], the probabilistic extensions [SYC+24] and so on. Among the techniques for key recovery, the key guessing strategy has received considerable attention. A common approach is to guess key bits corresponding to S-boxes in a default order as in [SWW21]. To improve complexity, the dynamic key-guessing technique [WWJZ18] and a delicate key-guessing technique [BCF+21] that takes advantage of the structure of the S-box have been introduced. In [BDD+24], an automated tool was developed for the first time to find the best key guessing order for block ciphers that use a bit-permutation as the linear layer. Recently, significant progress was made on the differential key recovery in [BDD+23] where the key is recovered in a meet-in-the-middle (MITM) manner, and the new attack is called the differential MITM attack. In classical differential attacks, pairs of data are constructed first, and for each pair, the attacker identifies the keys, under which the differential is respected. In the differential MITM attack, pairs of data are constructed together with the keys it suggests. The new attack has produced favorable results on block ciphers SKINNY-128-384 and AES-256. Later, this attack was extended to truncated differentials in [AKM+24].

The rectangle attack [BDK01], which is a variant of the differential attack, combines two differentials and utilizes the non-randomness of quartets. There have been a series of works on the key recovery of rectangle attacks [BDK01, BDK02, ZDM+20, DQSW21], each employing a different key guessing strategy. In [SZY+22, YSZ+24], the previous key recovery attacks are unified into a generic rectangle key recovery attack. Notably, the generic rectangle attack supports any key guessing strategy and can optimize complexity by selecting the most appropriate one.

Motivations. Even though the key guessing strategy received great attention in both differential attacks and rectangle attacks, the strategies differ. In the rectangle attack, it means guessing some key bits before any pairs or quartets are generated, which allows for filtering out some wrong pairs without even generating them. This is a natural approach. Under the guessed key some conditions of the distinguisher can be checked. In rectangle attacks, the number of conditions or filters is doubled as there are two pairs. Therefore, guessing some key bits initially is likely to be advantageous. The key guessing strategy lies in determining which part of the key bits to guess in advance. In contrast, in the differential attack, the attacker first generates the pairs that potentially satisfy the differential and then guesses some key bits. Here, the strategy focuses on the order in which the key bits are guessed.

From now on, let the key guessing strategy refer to the one from the rectangle attack. Although in the differential attack the number of filters under the guessed key bits is not doubled, it is interesting to investigate whether guessing some key bits in advance affects the time complexity of the differential attack. Second, the differential MITM attack employs a fixed key guessing strategy, can it be generalized to support any key guessing strategy? Further, can the MITM technique be integrated into the generic rectangle attack [SZY+22, YSZ+24]? These questions form the starting point of this paper.

Our Contributions. Guessing some key bits in advance before any pairs of data are generated is a technique that has been used in the rectangle attack. It plays a core role in optimizing the time complexity. In this paper, we introduce the key guessing strategy from the rectangle attack to the differential attack and also introduce the MITM key recovery to the rectangle attack, resulting in three generic key recovery attacks as follows.

GCDA A generic classical differential attack is proposed that first considers the key guessing strategy. Notably, the key guessing strategy can be any, as in the generic rectangle attack [SZY+22, YSZ+24]. Therefore, the GCDA encompasses the previous differential attack with no key bits guessed in advance.

GDMA A generalized differential meet-in-the-middle attack is proposed that extends the basic differential meet-in-the-middle attack (BDMA) [BDD+23] and allows a flexible key guessing strategy.

GRMA A generic rectangle meet-in-the-middle attack is proposed that incorporates the MITM technique into the rectangle key recovery.

To demonstrate the efficiency of these attacks, we revisit the attacks on AES-256, KATAN-32, and SKINNYe-64-256 v2 using previously published distinguishers. The following results are obtained and comparisons of the results with previous ones are summarized in Table 1.

- Using the GCDA and GDMA, the time complexity of the 12-round attack on AES-256 in the related-key setting can be optimized to 2^{144}. Notably, using the BDMA, the time complexity cannot be reduced below 2^{206}. This improvement is primarily due to the flexible key guessing strategy. Additionally, we extend the attack to 13 rounds, achieving lower data and time complexities than previous works. Specifically, the data complexity is reduced by a factor of 2^{37}. These are the best attacks so far on AES-256 using only two related keys.
- We add various numbers of rounds to a 42-round differential of KATAN-32 and compare the three attacks, *i.e.*, BDMA, GCDA, and GDMA. We confirm some properties of the time complexity of these attacks: the GDMA encompasses the BDMA; GDMA outperforms GCDA under certain conditions (see Sect. 3.3). Using a 91-round differential from the literature, we improve the differential attack on KATAN-32 from 115 rounds to 151 rounds, marking the best differential attack on KATAN-32 to date.
- We apply the GRMA to SKINNYe-64-256 v2 and extend the rectangle attack by one round. This is the best attack on SKINNYe-64-256 v2 so far. Note that, using the same distinguisher, the generic rectangle attack in [SZY+22,

Table 1. Summary of the cryptanalytic results. RK: related-key. SK: single-key.

Cipher	Rounds	Data	Time	Memory	Setting	Type
AES-256	12	2^{89}	2^{214}	2^{89}	RK	BDMA [BDD+23]
			2^{206}	2^{184}	RK	BDMA [BDD+23]
			$\mathbf{2^{185}}$	$\mathbf{2^{89}}$	RK	GCDA (Sect. 4.1)
			$\mathbf{2^{144}}$	$\mathbf{2^{184}}$	RK	GDMA (Sect. 4.1)
			$\mathbf{2^{145}}$	$\mathbf{2^{128}}$	RK	GCDA (Sect. 4.1)
	13	2^{126}	2^{253}	2^{89}	RK	BDMA [BDF23]
		2^{126}	2^{250}	2^{231}	RK	BDMA [BDF23]
		$\mathbf{2^{89}}$	$\mathbf{2^{248}}$	$\mathbf{2^{89}}$	RK	GCDA (Sect. 4.1)
		$\mathbf{2^{89}}$	$\mathbf{2^{240}}$	$\mathbf{2^{144}}$	RK	GCDA (App. A.3 [SLY+24])
KATAN-32	115	2^{32}	$2^{79.98}$	–	SK	Differential [AL13]
	151		$2^{79.98}$	2^{38}	SK	BDMA (Sect. 4.2)
SKINNYe-64-256 v2	37	$2^{62.8}$	$2^{240.03}$	$2^{62.8}$	RK	Rectangle [QDW+22]
	38	$\mathbf{2^{65.4}}$	$\mathbf{2^{251.07}}$	$\mathbf{2^{254.8}}$	RK	GRMA (Sect. 4.3)

YSZ+24] cannot cover as many rounds. This confirms the advantage of the MITM technique in certain cases of the rectangle key recovery.

Our work demonstrates that guessing key bits in advance is an excellent complement to existing techniques for differential attacks. Furthermore, allowing flexible key guessing strategies enhances the power of the differential MITM attack. Finally, the MITM technique can be extended to the key recovery of the rectangle attack.

Organization. The rest of the paper is organized as follows. In Sect. 2, we recall the generic rectangle attack and the differential MITM attack. In Sect. 3, we introduce three generic key recovery attacks, *i.e.*, the generic classical differential attack (GCDA), the generalized differential MITM attack (GDMA), and the generic rectangle MITM attack (GRMA). In Sect. 4, we provide new cryptanalytic results on AES-256, KATAN-32, and SKINNYe-64-256 v2 using the newly proposed key recovery attacks. Finally, we conclude the paper in Sect. 5.

2 Preliminaries

2.1 Notations

E	The block cipher $E = E_f \circ E_m \circ E_b$ with a distinguisher over E_m
n	The block size
k	The key size
k_b (resp. k_f)	The subset of subkey bits that are employed in E_b (resp. E_f)

k_b' (resp. k_f')	The part of k_b (resp. k_f) guessed in advance
k_b^* (resp. k_f^*)	$k_b^* = k_b \setminus k_b'$ (resp. $k_f^* = k_f \setminus k_f'$)
$\lvert \cdot \rvert$	The size of an object
r_b (resp. r_f)	The dimension of the space spanned by all possible plaintext (resp. ciphertext) differences
r_b' (resp. r_f')	The number of conditions that can be verified under k_b' (resp. k_f')
r_b^* (resp. r_f^*)	$r_b^* = r_b - r_b'$ (resp. $r_f^* = r_f - r_f'$)
GCRA	The generic classical rectangle attack [SZY+22]
GCDA	The generic classical differential attack
BDMA	The basic differential MITM attack [BDD+23]
GDMA	The generalized differential MITM attack
GRMA	The generic rectangle MITM attack

2.2 The Generic Rectangle Key Recovery Attack

At Asiacrypt 2022, a generic rectangle key recovery attack was proposed by Song *et al.* [SZY+22]. It contains a generic key recovery algorithm and a strategy for finding the best attack. To make a distinction, we call the key recovery attacks that do not use the MITM technique classical attacks, and then the attack in [SZY+22] is called the generic classical rectangle attack (GCRA).

As shown in Fig. 1, given a block cipher E, we treat it as the composition of three sub-ciphers: $E = E_f \circ E_m \circ E_b$. Suppose the probability of the boomerang distinguisher over E_m is $Pr = 2^{-2p}$. When we extend the differential outwards with probability 1, Δx will propagate to the plaintext difference ΔP over E_b^{-1} and Δy will propagate to the ciphertext difference ΔC over E_f. Let all possible ΔP span a space of dimension r_b. Similarly, let all possible ΔC span a space with dimension r_f. Suppose that it requires subkey information k_b (resp. k_f) to verify the difference Δx (resp. Δy) for plaintext (resp. ciphertext) pairs.

In the GCRA, some key bits may be guessed in advance to sieve the data faster. Suppose a part of k_b and k_f, denoted by k_b', k_f', is guessed at first, $0 \le |k_b'| \le |k_b|, 0 \le |k_f'| \le |k_f|$. With the guessed subkey bits, an r_b'-bit condition on the top and an r_f'-bit condition on the bottom can be verified. Finally, let $r_b^* = r_b - r_b'$ and $r_f^* = r_f - r_f'$. The specific steps for the algorithm are as follows.

1. Phase of data collection. Collect and store y structures of 2^{r_b} plaintexts. The time complexity of this step is T_0.
2. Phase of extracting key candidates.
   (a) Subkeys guessed. For each data (P_1,C_1), partially encrypt P_1 and partially decrypt C_1 under the guessed subkey bits. There are $y^* = y \cdot 2^{r_b'}$ sub-structures of $2^{r_b*} = 2^{r_b - r_b'}$ plaintexts. The time complexity of this step is T_1.
   (b) Pairs constructed. Insert all the obtained (P_1^*, C_1^*) into a hash table by the inactive bits of P_1^* or C_1^* to construct a set of pairs $S = \{(P_1^*, C_1^*, P_2^*, C_2^*)\}$ or $S = \{(P_1^*, C_1^*, P_3^*, C_3^*)\}$. The time complexity is T_2.
   (c) Quartets generated. Insert S into a hash table by the inactive bits of C_1^* and C_2^* or P_1^* and P_3^*. Then, generate the quartet for each index.

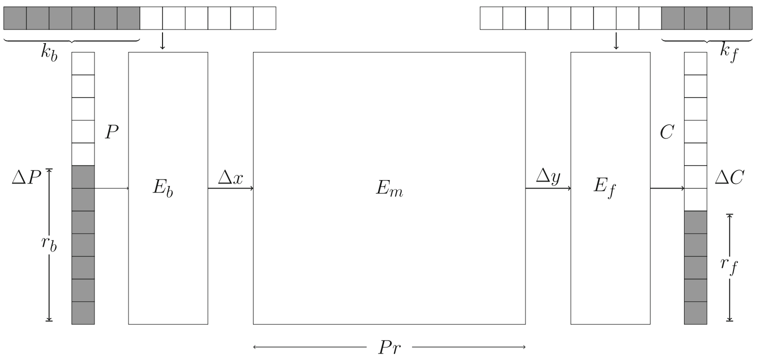


Fig. 1. A high-level description of the rectangle/differential MITM attack

(d) Quartets processed and key information extracted. Determine the key candidates involved in E_b and E_f and increase the corresponding counters. The time complexity of this step is T_3.

3. Phase of exhaustive search. Guess the remaining unknown key bits according to the key schedule algorithm and exhaustively search over them to recover the correct key. The time complexity of this step is T_4.

Complexities. The data complexity is $D = y \cdot 2^{r_b} = \sqrt{s}2^{n+1+p}$ where s is the expected number of right quartets and $y = \sqrt{s}2^{n/2-r_b+p}$. The memory complexity is $M = f_M(D, k'_b, k'_f) = D + \min\{D \cdot 2^{r^*_b-1}, D^2 \cdot 2^{r^*_f-n-1}\} + 2^{t+|k_b \cup k_f|-|k'_b \cup k'_f|}$ for storing the data, the pairs and the key counters, where $0 \le t \le |k'_b \cup k'_f|$. The time complexity $T = f_T(D, k'_b, k'_f)$ is composed of four parts. The time complexity of collecting data is $T_0 = D$, the time complexity of doing partial encryption and decryption under guessed key bits is

$$T_1 = 2^{|k'_b \cup k'_f|} \cdot D,$$

the time complexity of generating pairs is

$$T_2 = 2^{|k'_b \cup k'_f|} \cdot D \cdot \min\{2^{r^*_b-1}, D \cdot 2^{r^*_f-n-1}\},$$

the time complexity of generating and processing quartet candidates is

$$T_3 = 2^{|k_b \cup k_f|} \cdot D^2 \cdot 2^{-2n-2} \cdot \epsilon,$$

where $\epsilon \ge 1$ and its value depends on the concrete situation, and the time complexity of the exhaustive search is $T_4 = 2^{k-h}$, where $h \le t + |k_b \cup k_f| - |k'_b \cup k'_f|$.

It can be seen that the time complexities are affected by the key guessing strategy k'_b, k'_f. Given a distinguisher, different strategies for guessing key bits may lead to different time complexities. The `GCRA`, which supports any strategy, is supposed to find an optimal attack with the lowest time complexity using a holistic key guessing strategy.

2.3 The Basic Differential MITM Attack

The basic differential meet-in-the-middle (MITM) attack was first proposed by Boura *et al.* [BDD+23] at Crypto 2023, as depicted in Fig. 1. Suppose the probability of the differential distinguisher over E_m is $Pr = 2^{-p}$. The differential MITM attack can be divided into two phases.

1. MITM phase.
   Choose 2^p plaintexts. For each one:
   (a) Given the plaintext P, for each guess i for k_b, compute the associated $\widetilde{P}^i$ that ensures $E_b(P) \oplus E_b(\widetilde{P}^i) = \Delta x$. There are $2^{|k_b|}$ possible $(i, \widetilde{P}^i)$. Acquire the associated ciphertexts $\widehat{C}^i = E(\widetilde{P}^i)$ and store $(\widehat{C}^i, i)$ in a hash table H.
   (b) Given $C = E(P)$, for each guess j for k_f, we can compute the associated $\widetilde{C}^j$ that ensures $E_f^{-1}(C) \oplus E_f^{-1}(\widetilde{C}^j) = \Delta y$. There are $2^{|k_f|}$ possible $(j, \widetilde{C}^j)$.
   (c) Match $\widetilde{C}^j$ with $\widehat{C}^i$ by looking up the table H. Each collision of $(\widehat{C}^i, \widetilde{C}^j)$ suggests an associated key $k_b = i, k_f = j$, that we will consider as a candidate. The number of expected collisions for one plaintext P is $2^{|k_b|+|k_f|-|k_b \cap k_f|-n}$.
2. Exhaustive search phase.
   (a) Guess the remaining key bits (if there are) and test the guess with additional pairs.

Complexities. The time complexity of this attack can be estimated as

$$T = 2^p \times \left(2^{|k_b|} + 2^{|k_f|}\right) + 2^{|k_b \cup k_f|-n+p} + 2^{k-n+p}, \tag{1}$$

where the first term corresponds to the computations done in the upper part E_b and the lower part E_f, the second one to the number of expected key candidates for $k_b \cup k_f$ and the last one to the exhaustive search.

The data complexity of the attack can be roughly estimated as $D = \min\{2^n, 2^{p+\min(|k_b|,|k_f|)}\}$, which may be improved using data structures. The memory complexity is given by $M = 2^{\min(|k_b|,|k_f|)}$, but it can be improved to $2^{\min(|k_b|,|k_f|)-|k_b \cap k_f|}$ by guessing the common key material at the beginning. In particular, [BDD+23] claimed the attack can become much more efficient when the key size of the cipher is bigger than the state size.

3 New Generic Key Recovery Attacks

Inspired by the holistic key guessing strategy of the GCRA [SZY+22], we propose counterparts for the differential attack, *i.e.*, a generic classical differential attack (GCDA) and a generalized differential MITM attack (GDMA). Our core idea is to enhance key recovery attacks using the holistic key guessing strategy and the MITM technique. By selecting an appropriate strategy (including the type of key recovery attacks, key guessing strategy, etc.), the different terms of the time complexity can be more balanced, resulting in a lower overall time complexity. Further, upon the techniques used in GDMA, we combine the MITM technique with the rectangle attack and propose a generic rectangle MITM attack (GRMA) in return.

3.1 The Generic Classical Differential Attack

In [SZY+22], it was demonstrated for rectangle attacks that guessing some key bits in advance affects the time complexity and that any key guessing strategies should be allowed to find the best attack in terms of the time complexity. Since the rectangle attack is a variant of the differential attack, it is interesting to explore how these strategies can be applied back to the differential attack.

Suppose the differential $\Delta x \to \Delta y$ used in the attack has probability 2^{-p}. Then the data complexity for the attack is $D = 2^{p+1}$ if one right pair satisfying the differential is expected[1]. Other parameters for the key recovery are the same as introduced in Sect. 2.1 and Fig. 1. The attacker first guesses k'_b, k'_f, a part of the involved key k_b and k_f, where $0 \le |k'_b| \le |k_b|$, $0 \le |k'_f| \le |k_f|$. Suppose there are additional r'_b and r'_f filtering bits under the guess of k'_b and k'_f. Let $k^*_b = k_b \setminus k'_b$, $k^*_f = k_f \setminus k'_f$, $r^*_b = r_b - r'_b$, $r^*_f = r_f - r'_f$.

Like the rectangle key recovery algorithm in [SZY+22], a generic differential attack can be given in Algorithm 1. In this algorithm, it is assumed that $r_b - 1 \le p$, so multiple plaintext structures [BS92] are used. However, when $r_b - 1 > p$, a partial structure is enough, and this can be handled similarly. For conciseness, Algorithm 1 focuses on the former case.

Complexities. The time complexity of Algorithm 1 contains five parts:

- $T_0 = D$ for getting the data;
- $T_1 = 2^{|k'_b \cup k'_f|} \cdot D$ for partial encryption and decryption under the guessed key bits;
- $T_2 = 2^{|k'_b \cup k'_f|} \cdot D \cdot 2^{r_b - 1 + r_f - n - r'_b - r'_f}$ for getting the pairs that satisfy the specific filtering conditions;
- $T_3 = 2^{|k_b \cup k_f| + p - n} \cdot \epsilon = D \cdot 2^{|k_b \cup k_f| - n - 1} \cdot \epsilon$ for extracting all the $2^{|k_b \cup k_f| + p - n}$ key candidates, where ϵ depends on the concrete situation;
- $T_4 = 2^{k-h}$ for the exhaustive search, where $n - p \le h \le |k_b \cup k_f|$ when Line 10 uses the counting method while $h = n - p$ when enumerating all candidates is chosen.

As there will be $2^{|k_b \cup k_f| + p - n}$ key candidates on average, T_3 is at least $2^{|k_b \cup k_f| + p - n}$ and thus $\epsilon \ge 1$.

[1] If a bit more right pairs are needed, then D should be increased by a factor.

Algorithm 1: The generic classical differential attack (GCDA)

```
1  $S \leftarrow 2^{p-r_b+1}$ structures, each of $2^{r_b}$ messages.
2  for each possible $k'_b$ and $k'_f$, $0 \le |k'_b| \le |k_b|$, $0 \le |k'_f| \le |k_f|$, do
3      for structure $S[i], 0 \le i < |S|$ do
4          Do partial encryption and decryption for elements in $S[i]$ if $k'_b \cup k'_f \neq \emptyset$.
               // Additional $r'_b$, $r'_f$ filtering bits are obtained, respectively.
5          Store the data into a hash table indexed by the filtering bits.
6          Get $2^{2r_b-1+r_f-n-r'_b-r'_f}$ pairs having fixed differences on the filtering bits.
7          for each of such pairs do
8              Extract $2^{|k^*_b|-r^*_b}$ candidates for $k^*_b$, under which $\Delta x$ can be reached.
9              Extract $2^{|k^*_f|-r^*_f}$ candidates for $k^*_f$, under which $\Delta y$ can be reached.
10             Update the key counters or test directly.
```

The data should be stored. In addition, key counters consume memory if the counting method is used. Thus, the memory complexity is $M = \max\{2^{|k^*_b \cup k^*_f|}, D\}$ or $M = \min\{2^{r_b}, D\}$ depending on Line 10.

Remark 1. We check if the key guessing strategy makes a difference in the time complexity. First, the time complexity depends on the guessed key bits k'_b, k'_f. Additionally, we can compare two typical cases: guessing no key bits and guessing k'_b, k'_f in advance. From the data, $N = 2^{r_b+p-(n-r_f)}$ pairs can be constructed, satisfying the $n - r_f$ bits of the ciphertext difference. If the common guess-and-filter method is then used, it takes a time complexity of $N \cdot 2^{|k'_b \cup k'_f|}$ to get $N \cdot 2^{|k'_b \cup k'_f|} \cdot 2^{-r'_b-r'_f}$ pairs which satisfy $r'_b + r'_f$ additional bit conditions. Since $N \cdot 2^{|k'_b \cup k'_f|}$ is higher than T_1 when $r_b + r_f > n$, the key guessing strategy may matter in certain cases. In Sect. 4.1, we will see that using different key guessing strategies results in different time complexities for attacks on 12-round AES-256.

3.2 The Generalized Differential MITM Attack

In the original differential MITM attack [BDD+23], *i.e.*, the BDMA, a fixed key guessing strategy is used. Namely, the attacker separately guesses all the k_b and k_f in the MITM phase. Similar to the GCDA, it is beneficial to allow all possible key guessing strategies for the differential MITM attack.

Note that there are $n - r_f$ filtering bits from the fixed ciphertext difference, which are available at no extra cost. However, the BDMA cannot exploit these filtering bits until two sets are matched. When $n - r_f$ is large, whether these filtering bits can be exploited early or not makes a significant difference. The attacks on 12-round AES-256 in Sect. 4.1 are typical examples to confirm this.

Storing Pairs Instead of Single Messages. To exploit the $n - r_f$ filtering bits earlier, we propose to store pairs instead of single messages in the MITM stages, as these filtering bits can only be used for pairs. Then, it is more efficient to perform the MITM stages for all data together in a structure rather than for each single (P, C) one by one when pairs are considered.

The Detailed `GDMA`*.* In Algorithm 2, we propose our generalized differential MITM attack (`GDMA`), which allows any possible key guessing strategies and exploits the filtering bits of ciphertext difference early. The notations showed in Algorithm 2 are defined similarly. The attacker first guesses k'_b, k'_f, a part of the involved key k_b and k_f, respectively. Let $k'_\cap = k'_b \cap k'_f$. Suppose there are additional r'_b and r'_f filtering bits under the guessed key bits for the upper and lower parts. Let $k^*_b = k_b \setminus k'_b$, $k^*_f = k_f \setminus k'_f$, $r^*_b = r_b - r'_b$, $r^*_f = r_f - r'_f$. For conciseness, Algorithm 2 also focuses on the case $r_b - 1 \leq p$ where multiple data structures are used.

Algorithm 2: The generalized differential MITM attack (`GDMA`)

```
1  $S \leftarrow 2^{p-r_b+1}$ structures, each of $2^{r_b}$ messages.
2  for each possible $v_\cap$ for $k'_\cap$ do
3      for structure $S[i]$, $0 \leq i < |S|$ do
4          for each possible $v_b$ for $k'_b \setminus k'_\cap$ do
5              Do partial encryption for data in $S[i]$.
6              Get $2^{2r_b-1+r_f-n-r'_b}$ pairs $(P, \tilde{P})$ satisfying $n - r_f + r'_b$ filtering bits.
7              Store the corresponding $\left(C, \tilde{C}, v_b\right)$ in a table $H$.
8          for each possible $v_f$ for $k'_f \setminus k'_\cap$ do
9              Do partial decryption for data in $S[i]$.
10             Get $2^{2r_b-1+r_f-n-r'_f}$ pairs $\left(C, \hat{C}\right)$ satisfying $n - r_f + r'_f$ filtering bits.
11             for each $v_b \in H(C, \tilde{C})$ do
12                 Get $(P, \tilde{P})$ and $(v_\cap, v_b, v_f)$.
13                 Extract $2^{|k^*_b \cup k^*_f| - r^*_b - r^*_f}$ candidates for $k^*_b \cup k^*_f$ for each pair.
14                 Update the key counters or test directly.
```

Complexities. Without the pivot (P, C), both ciphertexts $\left(C, \tilde{C}\right)$ are stored in Line 7. Given two random pairs from the same structure, they will match with probability 2^{-2r_b+1} as there are 2^{2r_b-1} pairs in a structure. Therefore, in Line 12, there will be $D \cdot 2^{|k'_b \cup k'_f|} \cdot 2^{r_b-1+r_f-n-r'_b-r'_f}$ pairs[2], the same as T_2 of the `GCDA`. Like the `GCDA`, it may need to take further actions to extract the

[2] $2^{|k'_\cap|} \cdot 2^{p-r_b+1} \cdot 2^{|k'_b \cup k'_f| - |k'_\cap|} \cdot 2^{2r_b-1-r'_b} \cdot 2^{2r_b-1+r_f-n-r'_f} \cdot 2^{-2r_b+1} = D \cdot 2^{|k'_b \cup k'_f|} \cdot 2^{r_b-1+r_f-n-r'_b-r'_f}$.

remaining information of $k_{in} \cup k_{out}$ using the remaining filters. Similarly, the time complexity of the whole attack has five parts:

- $T_0 = D$ for getting the data;
- $T_1 = (2^{|k'_b|} + 2^{|k'_f|}) \cdot D$ for partial encryption and decryption under the guessed key bits;
- $T_2 = D \cdot 2^{|k'_b|} \cdot 2^{r_b - 1 + r_f - n - r'_b} + D \cdot 2^{|k'_f|} \cdot 2^{r_b - 1 + r_f - n - r'_f} + D \cdot 2^{|k'_b \cup k'_f|} \cdot 2^{r_b - 1 + r_f - n - r'_b - r'_f}$ for getting pairs that satisfy the specific filtering conditions;
- $T_3 = 2^{|k_b \cup k_f| + p - n} \cdot \epsilon = D \cdot 2^{|k_b \cup k_f| - n - 1} \cdot \epsilon$ for extracting all the $2^{|k_b \cup k_f| + p - n}$ key candidates, where ϵ depends on the concrete situation;
- $T_4 = 2^{k-h}$ for the exhaustive search, where $n - p \le h \le |k_b \cup k_f|$ when Line 14 uses the counting method while $h = n - p$ when enumerating all candidates is chosen.

The data complexity is $D = 2^{p+1}$, the same as the GCDA. The memory complexity comes from the storage of the data, the pairs, and key counters if needed. To save the memory for storing the counters of $k'_\cap$, we store the whole data and count candidates for $(k_b \cup k_f) \setminus k'_\cap$. However, we can also do it the other way around, *i.e.*, store one structure each time and count candidates for $k_b \cup k_f$, if it is more beneficial. Therefore, the memory complexity is $M = \min\left\{\max\left\{D, 2^{|k_b \cup k_f| - |k'_\cap|}\right\}, \max\left\{2^{r_b}, 2^{|k_b \cup k_f|}\right\}\right\}$ if the counting method is used or $M = \max\left\{2^{r_b}, 2^{2r_b - 1 + r_f - n - |k'_\cap|} \cdot \min\left\{2^{|k'_b| - r'_b}, 2^{|k'_f| - r'_f}\right\}\right\}$ if the enumeration method is used.

3.3 Comparison

GDMA *versus* BDMA. Let $k'_b = k_b$ and $k'_f = k_f$. Then the GDMA turns out to be almost identical to the BDMA: the time complexities are exactly the same, while the formulas for the memory and data complexities are different. Since $D = 2^{p+1}$ is already minimal, the data complexity of GDMA is not larger than that of BDMA. The memory complexity of GDMA depends on the key guessing strategy, so it is hard to compare the memory complexity of two attacks. Hence, Algorithm 2 can be seen as a generalization of BDMA if the time complexity is of the greatest concern.

GDMA *versus* GCDA. If the same key guessing strategy is used, then GDMA and GCDA share the same time complexity parts T_0, T_3 and T_4. Let us look into T_1 and T_2 which are rewritten in Table 2. Using the GDMA, the time complexity T_1 of partial encryption and decryption gets lower. For T_2, however, it depends but GDMA's T_2 is at least the one of GCDA. On the one hand, if the GDMA outperforms the GCDA, then T_1 must be dominant for the GCDA. It is the case for the differential attack on KATAN-32 in Sect. 4.2 when a 42-round differential is used.

On the other hand, if $r'_b \le |k'_b|$ and $r'_f \le |k'_b|$, GDMA will not be worse than GCDA. Usually, this is the case when a relatively large number of rounds are added around the distinguisher.

Table 2. Time Complexity Comparison of GCDA and GDMA

	GCDA	GDMA
T_1	$2^{\lvert k'_b \cup k'_f \rvert} \cdot D$	$(2^{\lvert k'_b \rvert} + 2^{\lvert k'_f \rvert}) \cdot D$
	-	$D \cdot 2^{\lvert k'_b \rvert} \cdot 2^{r_b-1+r_f-n-r'_b}$
T_2	-	$D \cdot 2^{\lvert k'_f \rvert} \cdot 2^{r_b-1+r_f-n-r'_f}$
	$D \cdot 2^{\lvert k'_b \cup k'_f \rvert} \cdot 2^{r_b-1+r_f-n-r'_b-r'_f}$	$D \cdot 2^{\lvert k'_b \cup k'_f \rvert} \cdot 2^{r_b-1+r_f-n-r'_b-r'_f}$

BDMA*,* GCDA *, and* GDMA. As there will be $2^{|k_b \cup k_f|+p-n}$ key candidates on average in any way, the following property can be obtained for differential attacks.

Property 1. When the overall time complexity reaches $2^{|k_b \cup k_f|+p-n}$, the differential key recovery attack cannot be further improved in terms of time complexity.

If the time complexity of a certain stage exceeds this term $2^{|k_b \cup k_f|+p-n}$, there are ways to balance.

- If the time complexity T_4 of the exhaustive search is high, the counting method can be used to select the most likely candidates to test. This reduces T_4 at the cost of increasing the data by a small factor, say 4.
- The holistic key guessing strategy can balance T_1 and T_2.
- If T_3 is large due to a large ϵ, precomputed tables may help to reduce ϵ.

In a nutshell, balancing the different components of the time complexity makes the attack more efficient.

3.4 The Generic Rectangle MITM Attack

From the comparison of the GCDA and the GDMA, it is known that when a significant number of rounds is added around the distinguisher, the MITM technique is likely to be beneficial. Then, a natural question arises: can the MITM technique enhance the rectangle attack so that more rounds can be attacked in certain cases? Next, we study the combination of the MITM technique with the rectangle attack.

Suppose the boomerang distinguisher has a probability of $P^2 = 2^{-2p}$ and y structures of plaintexts are needed. Note y structures can constitute $2 \cdot \binom{y2^{r_b-1}}{2}$[3] quartets that satisfy the input difference. Then $y = 2^{n/2-r_b+1+p}$ and the data complexity $D = y2^{r_b} = 2^{n/2+p+1}$. Other notations are similar to the ones in GDMA. The only difference is that two differentials are used in the rectangle attack. Can we do MITM for pairs of data as in the GDMA? Since pairs on the upper and lower parts of the rectangle attack are constructed in different directions, we cannot perform MITM on pairs but on quartets. The generalized rectangle MITM attack is given below with this taken into account.

[3] If both (P_1, P_2) and (P_3, P_4) satisfy the input difference of the distinguisher, then we can form two quartets: (P_1, P_2, P_3, P_4) and (P_1, P_2, P_4, P_3).

Algorithm 3: The generic rectangle MITM attack (GRMA)

```
1  $S \leftarrow y = 2^{n/2-r_b+1+p}$ structures, each of $2^{r_b}$ messages.
2  for each possible $v_\cap$ for $k'_\cap$ do
3      for each possible $v_b$ for $k'_b \setminus k'_\cap$ do
4          Do partial encryption for data in $S$.
5          Get $D \cdot 2^{r_b^*-1}$ pairs $(P, \tilde{P})$ satisfying $r'_b$ filtering bits.
6          Generate $D^2 \cdot 2^{2r_b^*-2}$ quartets $(C_1, C_2, C_3, C_4, v_b)$.
7          Store the quartets in a table $H$.
8      for each possible $v_f$ for $k'_f \setminus k'_\cap$ do
9          Do partial decryption for data in $S$.
10         Get $D^2 \cdot 2^{r_f^*-n-1}$ pairs $\left(C, \tilde{C}\right)$ satisfying $n - r_f + r'_f$ filtering bits.
11         Generate $D^4 \cdot 2^{2r_f^*-2n-2} \cdot y^{-2}$ quartets $(C_1, C_2, C_3, C_4, v_b)$.
12         for each $v_b \in H(C_1, C_2, C_3, C_4)$ do
13             Get $(P_1, P_2, P_3, P_4)$ and $(v_\cap, v_b, v_f)$.
14             Extract $2^{|k_b^* \cup k_f^*|-2r_b^*-2r_f^*}$ candidates for $k_b^* \cup k_f^*$ for each quartet.
15             Update the key counters or test directly.
```

In Line 13 of Algorithm 3, there are $2^{|k'_b \cup k'_f|} \cdot D^2 \cdot 2^{2r_b^*+2r_f^*-2n-2}$ matches[4]. In Line 14, further actions may be needed to get the other key bits. Similarly, the time complexity of the whole attack has five parts:

- $T_0 = D$ for getting the data;
- $T_1 = (2^{|k'_b|} + 2^{|k'_f|}) \cdot D$ for partial encryption and decryption under the guessed key bits;
- $T_2 = D^2 \cdot 2^{|k'_b|+2r_b^*-2} + D^4 \cdot 2^{|k'_f|+2r_f^*-2n-2} \cdot y^{-2}$ for getting quartets that satisfy the specific filtering conditions on one side, where y^{-2} is the probability of both pairs falling in the same structure;
- $T_3 = 2^{|k_b \cup k_f|} \cdot D^2 \cdot 2^{-2n-2} \cdot \epsilon$ for extracting all the $2^{|k_b \cup k_f|} \cdot D^2 \cdot 2^{-2n-2}$ key candidates, where $\epsilon \geq 1$ and its value depends on the concrete situation;
- $T_4 = 2^{k-h}$ for the exhaustive search, where $n - 2p \leq h \leq |k_b \cup k_f|$ when Line 15 uses the counting method while $h = n - 2p$ when enumerating all candidates is chosen.

The data, the quartets on one side, and the key counters should be stored, so the memory complexity is $M = \max\{D, \min\{D^2 \cdot 2^{|k'_b|+2r_b^*-2+2r_f-2n}, D^4 \cdot 2^{|k'_f|+2r_f^*-2n-2} \cdot y^{-2}\}, 2^{k_b^* \cup k_f^*}\}$ when the counting method is used and $M = \max\{D, \min\{D^2 \cdot 2^{|k'_b|+2r_b^*-2+2r_f-2n}, D^4 \cdot 2^{|k'_f|+2r_f^*-2n-2} \cdot y^{-2}\}\}$ when the enumeration method is used in Line 15. Like BDMA, the GRMA is usually more effective when the ratio k/n is large, which is the inherent limitation of the differential MITM attack itself.

[4] From the set of D plaintexts, there are $D^2 2^{2r_b-2}$ quartets, so a match happens with probability $D^{-2} 2^{2-2r_b}$ and $2^{|k'_b \cup k'_f|} \cdot D^2 2^{2r_b-2-2r'_b} \cdot D^4 2^{2r_f-2n-2r'_f-2} y^{-2} \cdot D^{-2} 2^{2-2r_b} = 2^{|k'_b \cup k'_f|} \cdot D^2 \cdot 2^{2r_b^*+2r_f^*-2n-2}$.

In Sect. 4.3, we will show with application to SKINNYe-64-256 v2 that when a large number of rounds are added to the distinguished, the GRMA can help to attack more rounds.

4 Applications

In this section, we analyze three block ciphers: AES-256, KATAN-32, and SKINNYe-64-256 using generic key recovery attacks proposed in Sect. 3. First, we provide improved attacks on 12/13-round AES-256 using the GCDA and GDMA in the related-key setting, which are the best attacks on AES-256 to date using only 2 related keys. Then, for KATAN-32, we use the GDMA and BDMA to extend the attack from 115 rounds to 151 rounds. These improved results demonstrate the advantage of our new key recovery attacks that enjoy the flexibility of key guessing strategies. Lastly, we apply the GRMA to SKINNYe-64-256 and extend the rectangle attack by 1 round, confirming the advantage of the GRMA in certain cases.

4.1 Application to AES-256

In this subsection, we give a brief description of AES-256 and recall the differential MITM attack on 12-round AES-256. We then propose improved attacks on 12-/13-round AES-256 using the GCDA and the GDMA.

Description of AES. The Advanced Encryption Standard (AES) [DR02] is a block cipher that encrypts 128-bit plaintext with the secret key of sizes 128, 192, or 256 bits. Its internal state can be represented by a 4×4 matrix whose elements are byte values in the finite field of $GF(2^8)$. As shown in Fig. 2, the round function consists of four basic transformations in the following order:

– SubBytes (SB) is a nonlinear substitution that applies the same S-box to each byte of the internal state.
– ShiftRows (SR) is a cyclic rotation of the i-th row by i bytes to the left, for $i = 0, 1, 2, 3$.
– MixColumns (MC) is a multiplication of each column with a Maximum Distance Separable (MDS) matrix over $GF(2^8)$.
– AddRoundKey (AK) is an exclusive-or with the round key.

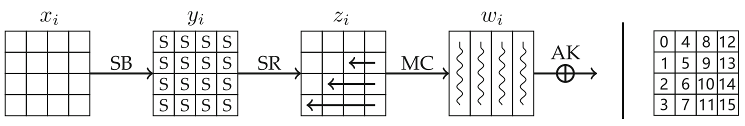


Fig. 2. AES round function and the ordering of bytes

At the beginning of the encryption, an additional whitening key addition is performed, and the last round does not contain MixColumns. AES-128, AES-192, and AES-256 share the same round function with a different number of rounds: 10, 12, and 14, respectively. AES-256 has a 256-bit key, which is twice as large as the internal state and derives round keys from the master key based on the key schedule illustrated in Fig. 3. Please refer to [DR02] for more details.

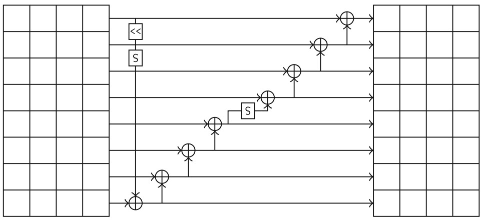


Fig. 3. Key schedule of AES-256

Distinguisher. Our differential attacks below are all based on the distinguisher proposed in [BDD+23], as shown in Fig. 4. The attacks require a pair of related keys. The attacker chooses two bytes a and b such that the differential transition $b \rightarrow a$ through the S-box happens with probability 2^{-6}. The attacker then injects the difference b on the first byte of the round key k_8 and $\texttt{MC}(a, 0, 0, 0)$ to the first column of the round key k_9. Figure 4 displays the attacks on AES-256, where $u_i, 0 \le i \le 14$ are unknown key differences. The differential used for the attacks starts from columns 0 and 3 of the state w_0 and columns 1 and 2 of z_1 and stops at state x_{11}. The differential holds with a probability of 0.25. If it does, the probability of the distinguisher is 2^{-86}. Moreover, the differential in the red dashed rectangle represents a 13-round attack on AES-256. The key difference propagation is depicted in Appendix A.1 in the full version of the paper.

Attacks on AES-256 Reduced to 12 Rounds. In [BDD+23], to show the power of the differential MITM attack, a 12-round attack on AES-256 was proposed. To verify the input difference of the differential, it involves 15 key bytes, namely, k_b contains $k_0[0, 2, 3, 4, 7, 8, 9, 13, 14], k_1[5, 10]$ and $k_3[12, 13, 14, 15]$. Similarly, to verify the output difference of the differential, it requires the information of 8 key bytes, *i.e.*, k_f consists of $k_{11}[12, 13, 14, 15]$ and $k_{12}[0, 4, 8, 12]$, from which an extra key byte $k_{10}[12]$ can also be derived. These key bytes are highlighted as red squares in Fig. 4. The remaining information of the master key contains 9 bytes. Applying the differential MITM attack presented in Sect. 2.3, one can have an attack of data, memory, and time[5] complexities $D = 2^{89}$, $M = 2^{89}$ and $T = 2^{214}$, as

[5] We contacted the authors of [BDD+23] and confirmed that they mistook the time complexity $2^p \cdot \max\{2^{120}, 2^{64}\} + 2^{p+56+72} = 2^{214}$ for $2^p \cdot \max\{2^{120}, 2^{64}, 2^{72}\} = 2^{206}$.

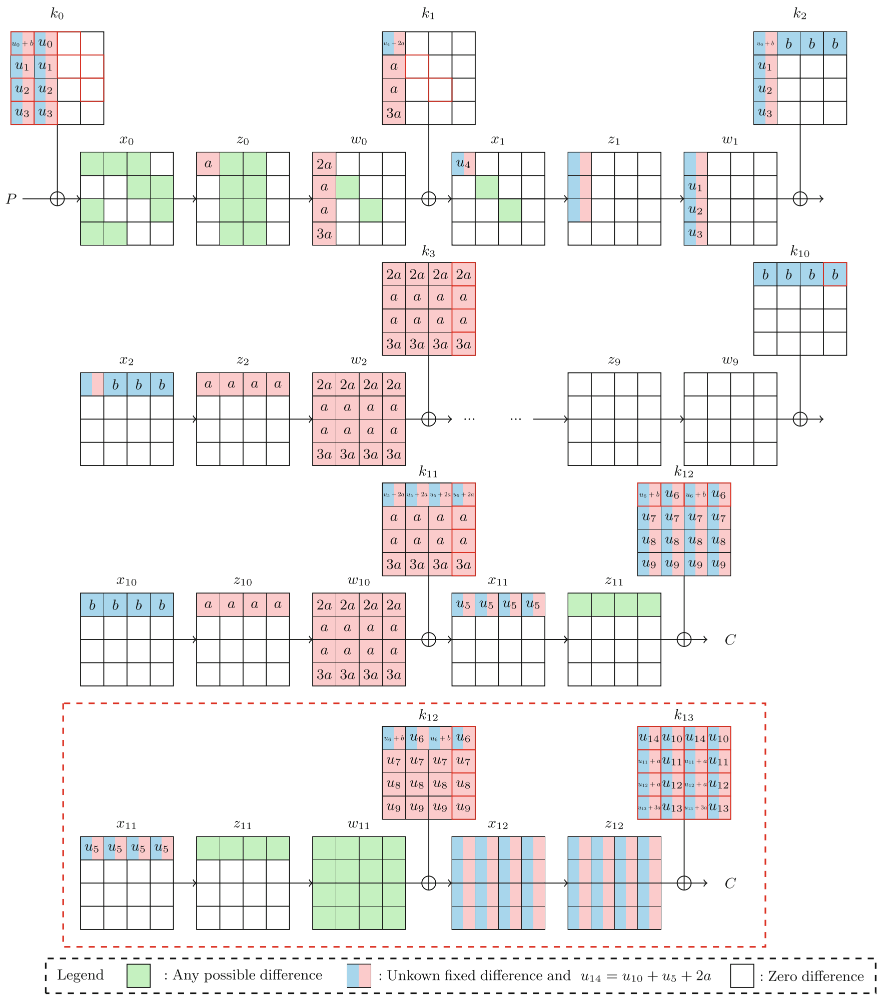


Fig. 4. Differential attacks on 12-round and 13-round AES-256 where a, b are chosen and known and the bottom part of the 13-round attack is shown in the dashed square.

$$
\begin{aligned}
T &= 2^p \times \left(2^{|k_{in}|} + 2^{|k_{out}|}\right) + 2^{|k_{in} \cup k_{out}|-n+p} + 2^{k-n+p} \\
&= 2^{86} \times \left(2^{120} + 2^{64}\right) + 2^{120+64-128+86} + 2^{256-128+86} \\
&= 2^{206} + 2^{150} + 2^{142} + 2^{214} \approx 2^{214}.
\end{aligned} \tag{2}
$$

From Eq. (2), it is evident that the time for the exhaustive search dominates the overall time complexity. We could turn to the counting method to reduce the time complexity of the exhaustive search. However, the time complexity cannot be reduced below 2^{206}, primarily due to the large k_b in the upper part of the

MITM phase. To further reduce the time complexity, it is worthwhile to explore different key guessing strategies.

The First GCDA *on 12-round* AES *-256.* We apply the GCDA to AES-256 and put forward the following attack using the same differential trail. The plaintext difference falls in a space of dimension 11×8 since $\Delta k_0[1] = \Delta k_0[5]$, *i.e.*, $r_b = 11 \times 8$. Since the bottom three bytes of the columns of Δk_{12} are the same, 9 bytes of the ciphertext difference are known, *i.e.*, $r_f = 7 \times 8$. We choose to guess 4 bytes of k_b and 3 bytes of k_f in advance. The attack starts with preparing a structure of 2^{88} plaintexts under both related keys, respectively.

1. Guess $k_3[12, 13, 14, 15]$,$k_{11}[13]$, and $k_{12}[8, 12]$:
   (a) Compute differences (u_0, u_1, u_2, u_3) from $k_3[12, 13, 14, 15]$ and $(2a, a, a, 3a)$; compute u_5 from b and $k_{10}[12] = k_{12}[8] \oplus k_{12}[12]$; compute u_6 from a and $k_{11}[13]$.
   (b) Initialize counters for all possible values of k_b^*, k_f^*, *i.e.*, $k_0[0, 2, 3, 4, 7, 8, 9, 13, 14]$, $k_1[5, 10]$ and $k_{11}[12, 14, 15]$, $k_{12}[0, 4]$.
   (c) Do partial encryption and decryption. There are 2 additional byte filters on both sides, *i.e.*, $r'_b = r'_f = 8 \times 2$.
   (d) Construct $2^{88 \times 2 - (2+2+9) \times 8} = 2^{72}$ pairs of data. Each pair has i) fixed differences on 7 x_0 bytes at positions $1, 5, 6, 10, 11, 12, 15$, ii) the last three rows of Δz_{11} satisfy the pattern of Δk_{12} and iii) $\Delta x_{11}[8, 12]$ are u_5.
   (e) For each pair of data, extract the other bytes of k_b by guessing $\Delta w_0[5, 10]$:
      - Compute the two middle columns of Δz_0. From Δx_0 and Δz_0, derive x_0 at the 9 active bytes as well as $k_0[0, 2, 3, 4, 7, 8, 9, 13, 14]$. Compute $w_0[5, 10]$.
      - Among the bytes of $\Delta z_1[0, 1, 2, 3]$ and $\Delta w_1[0, 1, 2, 3]$, four bytes are known. From them, compute $\Delta z_1[1, 2]$. From $\Delta x_1[5, 10]$ and $\Delta z_1[1, 2]$, recover $x_1[5, 10]$ and $k_1[5, 10] = w_0[5, 10] \oplus x_1[5, 10]$.
   (f) For each pair of data, extract the other bytes of k_f :
      - As $\Delta k_{12}[0] = u_6 + b$, $\Delta k_{12}[4] = u_6$, and $\Delta x_{11}[0] = \Delta x_{11}[4] = u_5$, compute $\Delta z_{11}[0, 4]$ and derive $z_{11}[0, 4]$, $k_{12}[0, 4]$.
      - Let $(u_7, u_8, u_9) = \Delta C[1, 2, 3]$. As $\Delta k_{11}[14, 15, 12] \xrightarrow{S} (u_7, u_8, u_9)$, derive $k_{11}[14, 15, 12]$.
   (g) Update the counters.
   (h) Select the key values with the top 2^{14} counters. Together with the guessed key bytes and all possible values for another 9 bytes outside $k_b \cup k_f$, test exhaustively to find the right master key.

The data complexity is still 2^{89}. The data and the counters should be stored, so the memory complexity is $\max\{2^{89}, 2^{16 \times 8}\} = 2^{128}$. The time complexity is

$$T = 2^{56} \times (2^{89} + 2^{72} + 2^{88}) + 2^{142} \approx 2^{145},$$

which is much smaller than the time complexity 2^{206} by the BDMA. The expected number of right pairs from the data is 4, so the right key ranks first with a high probability.

The Second GCDA *on 12-round* AES *-256.* However, the memory complexity of the above attack is higher than that of the BDMA. If we guess more key bytes in advance, the memory consumption for the counters decreases. Suppose $k_0[0], k_{12}[0,4]$ and $k_0[8,13]$ are also guessed. Then more filters $\Delta z_0[0] = a, 11 \cdot \Delta z_0[8] = 13 \cdot \Delta z_0[9]$, and $\Delta x_{11}[0,4]$ are u_5, *i.e.*, r'_b, r'_f now are both 4 bytes. Consequently, the time complexity becomes

$$T = 2^{96} \times (2^{89} + 2^{40} + 2^{48}) + 2^{182} \approx 2^{185},$$

and the memory complexity decreases to $\max\{2^{89}, 2^{11\times 8}\} = 2^{89}$. This attack has the same data and memory complexities as the BDMA but a lower time complexity.

The GDMA *on 12-round* AES *-256.* Using the GDMA, let $k'_b = k_3[12,13,14,15]$, $k'_b = (k_{11}[13],\ k_{12}[8,12])$, and we have $k'_\cap = 0$, $r'_b = 16$, $r'_f = 16$, $M = 2^{|k_b \cup k_f|} = 2^{184}$ and $T = D(2^{32} + 2^{32-1} + 2^{24} + 2^{24-1}) + 2^{128} + 2^{144} = 2^{144}$. Now the overall time complexity is as good as the time complexity of GCDA. If in the attack only single messages are stored instead of pairs, then the $n - r_f = 72$ bit filters cannot be used early and the time complexity is $T = D(2^{32} + 2^{32-1+72} + 2^{24} + 2^{24-1}) + 2^{128} + 2^{144} = 2^{192}$. Even though this attack is more efficient than the BDMA, it is less efficient than the GDMA which stores pairs.

Remark 2. According to Property 1, when the overall time complexity of the attack on 12-round AES-256 based on the differential trail in Fig. 4 reaches 2^{144}, there is no room for further improvement. That is, both the GCDA and the GDMA can lead to attacks with the optimal time complexity. However, using the BDMA, the time complexity cannot be lower than 2^{206}. This confirms that allowing flexible key guessing strategies is critical for finding the best attack.

Extending the Attack to 13 Rounds. We use the same differential in Fig. 4 and add one more round to the end. In this case, k_b remains the same as in the 12-round attacks, while k_f becomes large. k_f involves 28 round key bytes: $k_{10}[12], k_{11}[12,13,14,15], k_{12}[12,13,14,15]$, $\overline{k_{12}}[0,4,8]$ and k_{13} where $\overline{k_{12}} = \texttt{MC}^{-1}(k_{12})$. Now $k_b \cup k_f$ covers the whole key information. In addition, $r_b = 88, r_f = 128$.

The BDMA fails due to a large k_f in this case. Again, the flexible key guessing strategy shows its advantage with the following attack. Similar to the GCDA attack on 12-round AES, we also construct a structure of 2^{88} plaintexts under the two related keys, respectively.

1. Let k'_b be $k_3[14,15]$ and k'_f be $k_{10}[12]$, $k_{11}[12 \sim 15]$, $k_{12}[12 \sim 15]$ and $k_{13}[8 \sim 11]$. Guess k'_b, k'_f:
   (a) Initialize counters for all possible values for k^*_b and k^*_f.
   (b) Derive the difference for partial k_b and the whole k_f: From $\Delta k_3[14,15]$ and $k_3[14,15]$, compute u_1, u_2 so $\Delta k_0[1,2,5,6]$ is known; from $k_{10}[12]$ and b compute u_5 so $\Delta k_{11}[0,4,8,12]$ are known; from $k_{11}[12 \sim 15]$, compute $u_6 \sim u_9$ and then Δk_{12} is known; from $k_{12}[12 \sim 15]$, compute $u_{10} \sim u_{14}$ so now Δk_{13} is known. From $k_{13}[8 \sim 11]$ and $k_{12}[12 \sim 15]$ compute $k_{13}[12 \sim 15]$.

(c) Do partial encryption and decryption and then construct pairs of data satisfying 6 filtering bytes: $\Delta x_0[5,6] = 0$, *i.e.*, $r'_b = 16$, and $\Delta x_{12}[1] = \Delta x_{12}[2], \Delta x_{12}[6] = \Delta x_{12}[7], \Delta x_{12}[8] = \Delta x_{12}[11], \Delta x_{12}[12] = \Delta x_{12}[13]$ due to the MDS property of `MC`, *i.e.*, $r'_f = 32$. There will be $2^{88\times 2-6\times 8} = 2^{128}$ pairs of data.

(d) Derive other key bytes $k_{13}[0 \sim 7]$,$\overline{k_{12}}[0,4]$, $k_{12}[8 \sim 11]$, $k_3[12,13]$ and $k_0[0]$. Now, Δz_{11} and Δx_{12} are determined. Given the input and the output difference of the S-box, there is one solution on average. Therefore, for each pair, we can get one solution for $k_{13}[0 \sim 7]$, $\overline{k_{12}}[0,4,8,12]$ and $k_0[0]$. As $\overline{k_{12}}[12]$ must match with the guessed $k_{12}[12 \sim 15]$, this is a **1-byte filter** and the number of pairs becomes 2^{120}.
From the key schedule, $k_3[14] = S(k_{12}[10] \oplus k_{12}[14]) \oplus k_{11}[14]$, so we can get $k_{12}[10]$. Similarly, we can get $k_{12}[11]$. And $k_{12}[8] = k_{12}[12] \oplus k_{10}[12]$. As $\overline{k_{12}}[8]$ is also known, we can get $k_{12}[9]$. From the known bytes of k_{11}, k_{12}, we can compute $k_3[12,13]$.

(e) For each pair of data, guess $\Delta w_0[10]$ and recover the remaining key bytes of k_b, k_f.
   i. From $\Delta w_0[10]$, get the difference of the third column of z_0. For the S-boxes of that column, the input difference and the output difference are known, so we can get $k_0[8,13,2,7]$. Similarly, $k_1[10]$ can be known as $\Delta z_1[3]$ is known from u_1, u_2, u_3.
   According to the relation of key bytes, as shown in Appendix A.2 in the full version of the paper, from $k_0[0,2,7,8,13]$ and $k_1[10]$, we can get $k_{12}[0,2,5,6,7]$ and a redundant relation which acts as a **1-byte filter**.
   From $k_{12}[5,6,7]$ and $\overline{k_{12}}[4]$, compute $k_{12}[4]$. From $k_{12}[4,6]$, derive $k_0[4,14]$ which helps to compute $\Delta z_0[4,6]$. This is a **1-byte filter** due to the MDS property. Also, $\Delta z_0[5,7]$ and $\Delta w_0[5]$ can be known now, so we can derive $k_0[3,9]$ and $k_1[5]$. From these three bytes, we can derive $k_{12}[1,3]$ and one redundant relation which is also a **1-byte filter**. From $k_{12}[0 \sim 3]$ and $\overline{k_{12}}[0]$, we get the last **1-byte filter**.
   Now the whole k_{12} and k_{13} are determined.
   ii. Test the key candidates exhaustively to find the right master key.

Complexities. The data complexity is $D = 2^{89}$. The memory complexity is also 2^{89}. Since $|k'_b| = 16$, $|k'_f| = 104$, we have $T_1 = D \cdot 2^{120} = 2^{209}$, $T_2 = 2^{120+128} = 2^{248}$. In Step (e), from 2^{240} pairs of data, we guess one byte more and get the other 4 filtering bytes. Therefore, 2^{216} key candidates will be suggested finally in a time complexity of 2^{248}. Since $T_4 = 2^{216}$ is not dominant, the overall time complexity is 2^{248}.

13-Round Attack with an Improved Time Complexity. In the above attack, several filters are used in the last steps. If some of these filters can be used earlier, the number of pairs may not reach 2^{248} and hopefully the time complexity is lower. We then propose an adapted attack that guesses fewer key bytes in advance and utilizes two precomputed tables so that the number of pairs is at most 2^{240}.

This attack has the same data complexity. The time complexity is 2^{240} while the memory complexity is increased to 2^{120} due to the precomputed tables. The detailed attack is given in Appendix A.3 in the full version of the paper.

Comparison and Discussion. On an existing differential, we analyze AES-256 reduced to 12 and 13 rounds using the GCDA and GDMA in the related-key setting. The results are summarized as follows.

- On 12-round AES-256, both the GCDA and GDMA can achieve attacks with the optimal time complexity, which is 2^{62} times lower than that of BDMA.
- Using the GCDA, the attack can be extended to 13 rounds. In this attack, $r'_b \leq |k'_b|$ and $r'_f \leq |k'_b|$, so the GDMA is not worse (not better as well in this case) under the same key guessing strategy, as explained in Sect. 3.3.
- Using the same differential, the BDMA cannot cover 13 rounds, because of $2^{p+|k_f|} > 2^k$ due to a large k_f. In [BDF23], the authors modified the differential to make k_f smaller, thus enabling a differential MITM attack. However, the differential probability decreased from 2^{-86} to 2^{-126}.

All these results show that allowing flexible key guessing strategies is critical for mounting key recovery attacks efficiently. Choosing between the counting method and the enumerating method for the exhaustive search also impacts the overall efficiency. The attacks on AES-256 illustrate that by combining these methods, we can balance the time complexities of the attack steps to get more efficient attacks.

However, the time complexity of the 13-round attack does not reach $2^{|k_b \cup k_f|+p-n}$. The major reason lies in the nonlinear key schedule. k_b and k_f share some common information, but it is difficult to utilize them early on due to their complex nonlinear relationship. The 13-round attack with a lower time complexity highlights the challenge of effectively leveraging the shared key information.

4.2 Application to KATAN-32

For most block ciphers, a few rounds can be added around a differential in the key recovery attack. However, the block cipher KATAN-32 is an exception. Since it updates only 2 bits in each round, a relatively large number of rounds can be added. In attacks on KATAN-32, we can observe how the BDMA, GCDA, and GDMA perform when the number of rounds increases, and more importantly, whether there is any discernible trend. Additionally, any improvement in its differential attack is of particular interest. We start by recalling the description of the KATAN-32.

Description of KATAN. The KATAN family is composed of three variants with 32-, 48-, and 64-bit block sizes denoted as KATAN-n, $n = 32, 48, 64$, respectively. Here, we briefly revisit the KATAN-32, which is analyzed in the next. The KATAN-32 iterates 254 rounds using two non-linear feedback shift registers (NLFSR) to store and update the plaintext.

Key Schedule. The 80-bit master key $K = (k_0, k_1, \cdots, k_{78}, k_{79})$ uses the linear feedback register to generate the new round keys.

$$k_{i+80} = k_i \oplus k_{i+19} \oplus k_{i+30} \oplus k_{i+67},\ 0 \leq i \leq 427. \tag{3}$$

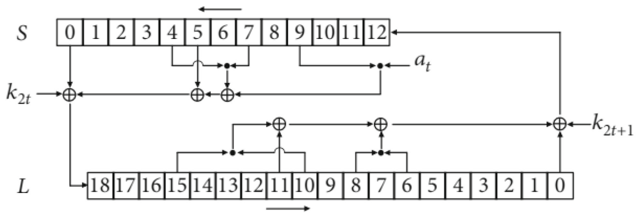


Fig. 5. The round function of KATAN-32.

Round Function. A 32-bit plaintext $X = (x_0, x_1, \cdots, x_{30}, x_{31})$ is divided into two parts with 13 and 19 bits, respectively. At round t, the two parts are denoted by $S_t = (s_t, s_t + 1, \cdots, s_{t+11}, s_{t+12})$ and $L_t = (l_t, l_t + 1, \cdots, l_{t+17}, l_{t+18})$. When $t = 0$, the plaintext is loaded as $s_i = x_i, 0 \leq i \leq 12$ and $l_i = x_{13+i}, 0 \leq i \leq 18$. When $0 \leq t \leq 253$, the round function depicted in Fig. 5 is defined as follows:

$$\begin{aligned} s_{t+13} &= l_t \oplus l_{t+11} \oplus l_{t+6} \cdot l_{t+8} \oplus l_{t+10} \cdot l_{t+15} \oplus k_{2t+1}, \\ l_{t+19} &= s_t \oplus s_{t+5} \oplus s_{t+4} \cdot s_{t+7} \oplus s_{t+9} a_t \oplus k_{2t}, \end{aligned} \tag{4}$$

where a_t is a round constant updated by the relation equation $a_t = a_t - 3 \oplus a_{t-5} \oplus a_{t-7} \oplus a_{t-8}, (t \geq 8)$ with the initial value $(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7) = (1, 1, 1, 1, 1, 1, 1, 0)$. According to the Eq. 4, we can get the expression of l_t, s_t in the decryption direction:

$$\begin{aligned} l_t &= s_{t+13} \oplus l_{t+11} \oplus l_{t+6} l_{t+8} \oplus l_{t+10} l_{t+15} \oplus k_{2t+1}, \\ s_t &= l_{t+19} \oplus s_{t+5} \oplus s_{t+4} s_{t+7} \oplus s_{t+9} a_t \oplus k_{2t}. \end{aligned} \tag{5}$$

Distinguisher. Our attacks are based on the 42-round and the 91-round distinguishers proposed in [JRS22] and [AL13] with probability 2^{-12} and $2^{-31.98}$:

$$\Delta_{in}^{42} = 0x08020040,\ \Delta_{out}^{42} = 0x00200420,$$
$$\Delta_{in}^{91} = 0x1006a880,\ \Delta_{out}^{91} = 0x00400000.$$

When the attacker conducts a key attack on KATAN, it is very time-consuming to determine the best key guessing strategy manually. It becomes even harder when the added rounds increase. Therefore, we build a MILP model to find the best key guessing strategy automatically. This model follows the same modules as [SZY+22], and its detailed description is postponed to Appendix B.1 in the full version of the paper. Given a differential, the number of rounds before the differential, the number of rounds after the differential, and the type of attack (*i.e.*, BDMA, GCDA, or GDMA), the model outputs the minimal time complexity of the attack and the parameters that lead to the attack.

Comparison and Discussion. Using the key recovery model, we can change the objective function according to the key recovery algorithm to find the best-attacking parameters such that the time complexity of the attack is optimal. Based on the key recovery model and the 42-round distinguisher, we apply the GCDA, GDMA, and BDMA to KATAN-32 from 94 to 137 rounds and compare their time complexities. The results are illustrated in Fig. 6 and analyzed as follows.

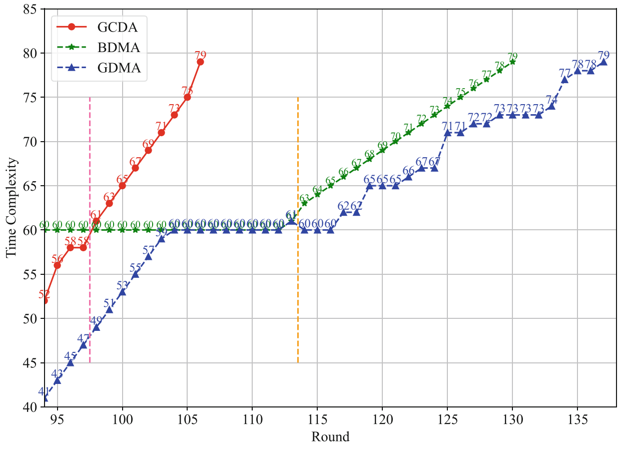


Fig. 6. The time complexity of three attacks.

– From attacks on KATAN-32 reduced to 94 to 97 rounds before the pink dashed line as shown in Fig. 6, both GCDA and GDMA have a better performance than the BDMA. The last part 2^{k-n+p} of the BDMA's time complexity is dominated, while the GCDA and GDMA can use the counting method mentioned in Sect. 3.3 to reduce the exhaustive search complexity.
– Between the pink and orange dashed lines, T_1 becomes a large one in the GCDA's time complexity as the key guessed increases. So, the GCDA is worse than the GDMA and BDMA. When $|k_b \cup k_f|$ is not the full key space, the GDMA has a lower time complexity than the BDMA. As the number of rounds increases, $|k_b \cup k_f| = 80$, which reaches the full key space. The dominant terms of the BDMA and GDMA are the same.
– After the orange dashed line, the first term $(2^{|k_b|} + 2^{|k_f|}) \cdot 2^p$ of BDMA and $T_1 = (2^{|k'_b|} + 2^{|k'_f|}) \cdot D$ of the GDMA turn to dominant terms. The GDMA has a lower time complexity than the BDMA. Because the key guessing strategy

of the BDMA is fixed and the GDMA allows the attacker to guess a part of key information.
- The GDMA always performs better than the GCDA in every round of attacks on KATAN-32. As explained in Sect. 3.3, T_1 of the GCDA is a dominant one. The partial encryption and decryption time complexity T_1 of the GDMA is lower than that of GCDA.

Attacks on KATAN-32 Reduced to 151 Rounds. We apply the BDMA and GDMA to attack 151-round KATAN-32 based on the 91-round distinguisher with 29-round E_b and 31-round E_f, as shown in Table 3. In Table 3, ? denotes a bit with the unknown difference, 0 and 1 represent the specific difference value.

BDMA*.* The parameters are $p = 31.98$, $n = 32$, $|k_b| = 41$, $|k_f| = 38$. The key information is listed as follows:

$$k_b = k_0, k_1, \cdots, k_{36}, k_{37}, k_{40}, k_{44}, k_{45},$$
$$k_f = k_{252}, k_{260}, k_{262}, k_{264}, k_{266}, k_{268}, k_{269}, k_{270}, k_{272}, k_{273}, \cdots, k_{300}, k_{301}.$$

The time complexity is

$$\begin{aligned} T &= 2^{31.98+41} + 2^{31.98+38} + 2^{41+38-0.02} + 2^{80-32+31.98} \\ &= 2^{72.98} + 2^{69.98} + 2^{78.98} + 2^{79.98} \approx 2^{79.98} \end{aligned}$$

with data and memory complexities $D = 2^{32}, M = 2^{38}$.

GDMA . The best guessing parameters are $|k'_b| = 33$, $|k'_f| = 29$, $r'_b = 25$, $r'_f = 22$ and $r_b = r_f = 32$. Namely, guessing 33-bit k'_b and 29-bit k'_f obtains 25 and 22 filters, respectively. The filtering bits in the backward and forward extended rounds are marked in blue and red in Table 3. The subkey bits guessed are:

$$k'_b = k_0, k_1, \cdots, k_{24}, k_{25}, k_{27}, k_{28}, k_{29}, k_{30}, k_{32}, k_{33}, k_{36},$$
$$k'_f = k_{264}, k_{268}, k_{272}, k_{274}, k_{275}, k_{276}, k_{278}, k_{279}, k_{280}, k_{282}, k_{283}, \cdots, k_{300}, k_{301}.$$

The data complexity and memory complexity are $D = 2^{32}, M = 2^{70}$. The time complexity is

$$\begin{aligned} T &= 2^{32+33} + 2^{32+33+32-25-1} + 2^{32+29} + 2^{32+29+32-22-1} \\ &+ 2^{32+33+29+32-25-22-1} + 2^{79-0.02} + 2^{80+31.98-32} \\ &= 2^{65} + 2^{71} + 2^{61} + 2^{70} + 2^{78} + 2^{78.98} + 2^{79.98} \approx 2^{79.98}. \end{aligned}$$

Although the GDMA and BDMA have a slight advantage over traversing all keys, our attacks primarily demonstrate that using new generic differential attacks can significantly increase the number of rounds attacked, from 115 to 151. In the attack, the time complexity of the exhaustive search is dominant. Since the differential has a probability slightly larger than 2^{-n}, there is no opportunity to trade data for time by using the counting method. Consequently, BDMA and GDMA share the same overall time complexity.

In addition, we present alternative 151-round attacks on KATAN-32 by extending 33 before and 27 rounds after the differential, respectively. More details are provided in Appendix B.2 in the full version of the paper. In this setting, the GDMA still recovers the key with a time complexity of $2^{79.98}$, while the BDMA fails. This outcome underscores the importance of flexibly guessing key information for successful key recovery attacks, highlighting the greater applicability of the GDMA.

Table 3. The 151(29 + 91 + 31)-round attack on the KATAN-32. For round t, Δ_t is the t-th round difference, $0 \leq t \leq 151$. The blue and red denote the filters by guessing a part of key information k'_b and k'_f.

Δ_0	???? ???? ???? ???? ???? ???? ???? ????	Δ_{out}	0000 0000 0100 0000 0000 0000 0000 0000
Δ_1	???? ???? ???? ???? ???? ???? ???? ????	Δ_{121}	0000 0000 1000 0000 0000 0000 0000 0001
Δ_2	???? ???? ???? ???? ???? ???? ???? ????	Δ_{122}	0000 0001 0000 0000 0000 0000 0000 0010
Δ_3	???? ???? ???? ???? ???? ???? ???? ????	Δ_{123}	0000 0010 0000 0000 0000 0000 0000 010?
Δ_4	???? ???? ???? ???? ???? ???? ???? ????	Δ_{124}	0000 0100 0000 0000 0000 0000 0000 10?0
Δ_5	???? ???? ???? ???? ???? ???? ???? ????	Δ_{125}	0000 1000 0000 ?000 0000 0000 0001 0?01
Δ_6	???? ???? ???? ???? ???? ???? ???? ????	Δ_{126}	0001 0000 000? 0000 0000 0000 0010 ?01?
Δ_7	???? ???? ???? ???? ???? ???? ???? ????	Δ_{127}	0010 0000 00?0 ?000 0000 0000 010? 01?0
Δ_8	???? ???? ???? ???? ???? ???? ???? ????	Δ_{128}	0100 0000 0?0? 0000 0000 0000 10?0 1?00
Δ_9	???? ???? ???? ???? ???? ???? ???? ???0	Δ_{129}	1000 0000 ?0?0 ?000 0000 0001 0?01 ?000
Δ_{10}	???? ???? ???? 1??? ???? ???? ???? ??0?	Δ_{130}	0000 000? 0?0? ?000 0000 0010 ?01? 0001
Δ_{11}	???? ???? ???1 ???? ???? ???? ???? ?0?1	Δ_{131}	0000 00?0 ?0?? ?000 0000 010? 01?0 001?
Δ_{12}	???? ???? ??1? ???? ???? ???? ???? 0?11	Δ_{132}	0000 0?0? 0??? ?000 0000 10?0 1?00 01?0
Δ_{13}	???? ???? ?1?? ???? ???? ???? ???0 ?110	Δ_{133}	0000 ?0?0 ???? 1000 0001 0?01 ?000 1?0?
Δ_{14}	???? ???? 1??? 0??? ???? ???? ??0? 1101	Δ_{134}	000? 0?0? ???1 ?000 0010 ?01? 0001 ?0??
Δ_{15}	???? ???1 ???0 1??? ???? ???? ?0?1 1010	Δ_{135}	00?0 ?0?? ??1? ?000 010? 01?0 001? 0???
Δ_{16}	???? ??1? ??01 ???? ???? ???? 0?11 0101	Δ_{136}	0?0? 0??? ?1?? ?000 10?0 1?00 01?0 ????
Δ_{17}	???? ?1?? ?01? 0??? ???? ???0 ?110 1010	Δ_{137}	?0?0 ???? 1??? ?001 0?01 ?000 1?0? ????
Δ_{18}	???? 1??? 01?0 0??? ???? ??0? 1101 0101	Δ_{138}	0?0? ???1 ???? ?010 ?01? 0001 ?0?? ????
Δ_{19}	???1 ???0 1?00 0??? ???? ?0?1 1010 1010	Δ_{139}	?0?? ??1? ???? ?10? 01?0 001? 0??? ????
Δ_{20}	??1? ??01 ?000 1??? ???? 0?11 0101 0100	Δ_{140}	0??? ?1?? ???? ?0?0 1?00 01?0 ???? ????
Δ_{21}	?1?? ?01? 0001 0??? ???0 ?110 1010 1000	Δ_{141}	???? 1??? ???? ??01 ?000 1?0? ???? ????
Δ_{22}	1??? 01?0 0010 0??? ??0? 1101 0101 0001	Δ_{142}	???1 ???? ???? ?01? 0001 ?0?? ???? ????
Δ_{23}	???0 1?00 0100 0??? ?0?1 1010 1010 0010	Δ_{143}	??1? ???? ???? ?1?0 001? 0??? ???? ????
Δ_{24}	??01 ?000 1000 0??? 0?11 0101 0100 0100	Δ_{144}	?1?? ???? ???? ??00 01?0 ???? ???? ????
Δ_{25}	?01? 0001 0000 0??0 ?110 1010 1000 1000	Δ_{145}	1??? ???? ???? ?000 1?0? ???? ???? ????
Δ_{26}	01?0 0010 0000 0?0? 1101 0101 0001 0000	Δ_{146}	???? ???? ???? ?001 ?0?? ???? ???? ????
Δ_{27}	1?00 0100 0000 00?1 1010 1010 0010 0000	Δ_{147}	???? ???? ???? ?01? 0??? ???? ???? ????
Δ_{28}	?000 1000 0000 0?11 0101 0100 0100 0000	Δ_{148}	???? ???? ???? ?1?0 ???? ???? ???? ????
Δ_{in}	0001 0000 0000 0110 1010 1000 1000 0000	Δ_{149}	???? ???? ???? ??0? ???? ???? ???? ????
..	....	Δ_{150}	???? ???? ???? ?0?? ???? ???? ???? ????
..	....	Δ_{151}	???? ???? ???? ???? ???? ???? ???? ????

4.3 Application on SKINNYe-64-256 V2

In this subsection, we apply our GRMA to SKINNYe-64-256 v2 to obtain a 38-round rectangle attack.

Description of SKINNYe -64-256 v2. SKINNYe-64-256 v2 [NSS20a] is one of the variants of SKINNY [BJK+16]. Similar to SKINNY, SKINNYe-64-256 v2 employs the TWEAKEY framework and the STK construction [JNP14], with modifications to the f function in STK. SKINNY supports block sizes $n \in \{64, 128\}$, and for $n = 64$, the tweakey sizes $tk \in \{64, 128, 192\}$. To support TI-friendly AE modes, Naito *et al.* extended SKINNY-64 to create SKINNYe-64-256, which features a 256-bit tweakey and a similar tweakey schedule [NSS20b]. Due to security concerns raised by Thomas Peyrin, an updated version, SKINNYe-64-256 v2, was proposed in 2020 [NSS20a].

SKINNY, SKINNYe-64-256, and SKINNYe-64-256 v2 share the same round function, applying five transformations:

1. SubCells (SC) - A 4-bit (resp. 8-bit) S-box is applied to all cells when n is 64 (resp. n is 128).
2. AddConstants (AC) - This step adds constants to the internal state.
3. AddRoundTweakey (ART) - The first two rows of the internal state absorb the first two rows of TK, where $TK = \bigoplus_{i=1}^{z} TK_i$.
4. ShiftRows (SR) - Each cell in row j is rotated to the right by j cells.
5. MixColumns (MC) - Each column of the internal state is multiplied by the matrix M whose branch number is only 2.

Tweakey Schedule of SKINNYe *-64-256 v2.* The 256-bit tweakey state is viewed as 4×4 square arrays of nibbles, denoted as (TK^1, TK^2, TK^3, TK^4). The tweakey arrays in round r ($r \geq 0$) are represented as TK_r^1, TK_r^2, TK_r^3, and TK_r^4, where $TK_0^m = TK^m$ ($1 \leq m \leq 4$). For $r \geq 1$, TK_r^m is generated in two steps:

– First, apply the permutation $P = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$ to each nibble of all tweakey arrays: $TK_r^{m,i} \leftarrow TK_{(r-1)}^{m,P[i]}$, where $1 \leq m \leq 4, 0 \leq i \leq 15$ and $r \geq 1$.
– Then, apply $LFSR_r$ to update each nibble of the first and second rows of TK_r^m for $2 \leq m \leq 4$. For the details, please refer to [NSS20a].

Distinguisher of SKINNYe-64-256 v2. At Asiacrypt 2022, Qin *et al.* [QDW+22] proposed a 26-round related-key rectangle distinguisher of SKINNYe-64-256 v2 with a probability of $2^{-57.6}$. The distinguisher is detailed in Appendix C in the full version of the paper. By modifying this distinguisher, we obtain a new version that is more suitable for GRMA. Specifically, we allow $b \rightarrow 3$ in the SC operation in round 5 with a probability of 2^{-2}. Therefore, the probability of this modified distinguisher is $P^2 = 2^{-57.6-4} = 2^{-61.6}$.

We add 6 rounds forward and backward to attack the 38-round SKINNYe-64-256 v2, as shown in Fig. 7. The corresponding parameters are: $r_b = r_f = 16 \times 4 = 64$, $|k_b| = (8 \times 3 + 6 + 2) \times 4 = 128$, and $|k_f| = (2 + 6 + 8 \times 3) \times 4 = 128$.

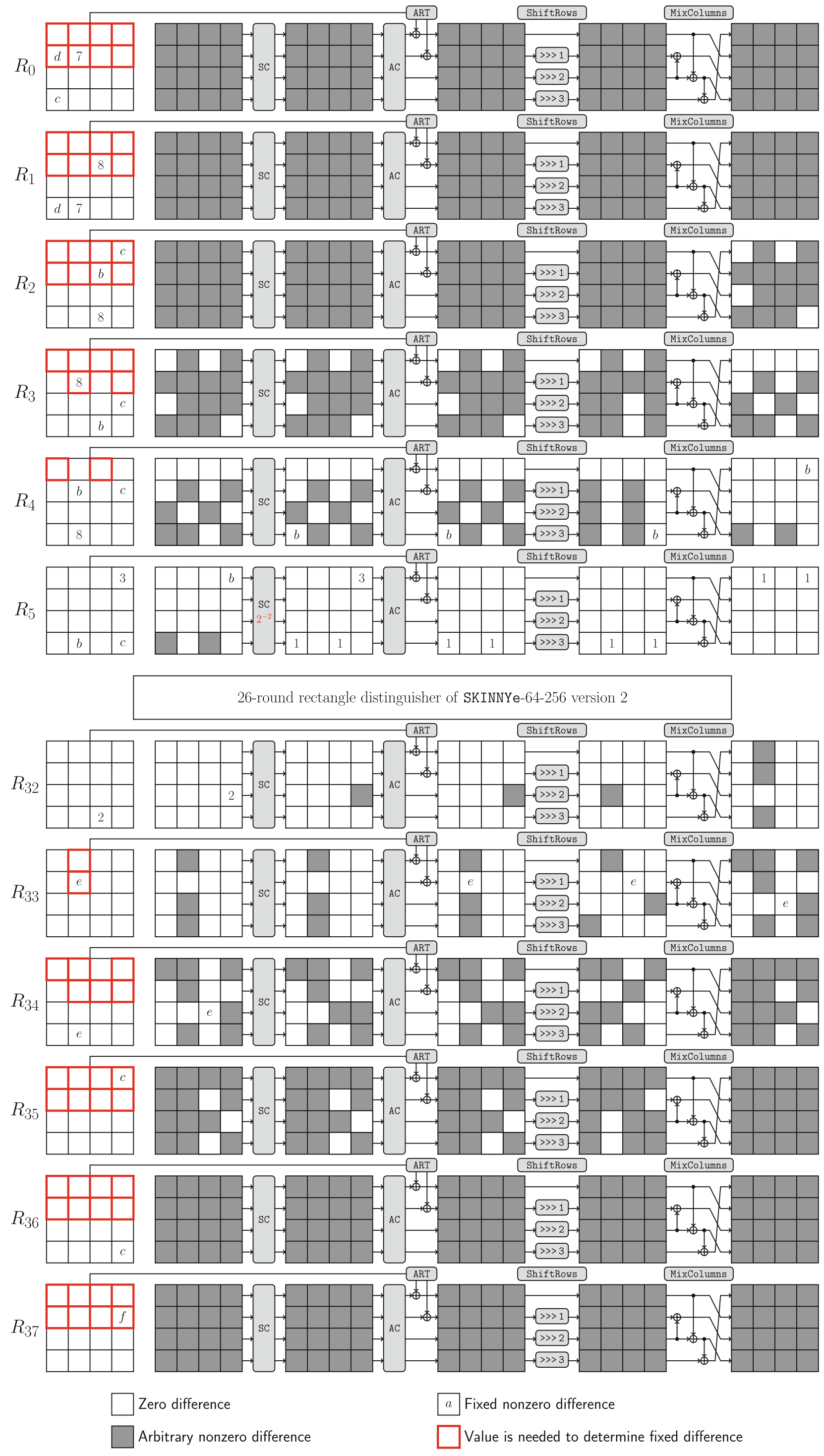


Fig. 7. Rectangle meet-in-the-middle attack on 38-round SKINNYe-64-256 v2

Parameters and Complexities. We apply GRMA to the above distinguisher. The attack is illustrated in Fig. 7, and the parameters of this attack are $|k'_b| = |k_b| = 128$ and $|k'_f| = |k_f| = 128$. Therefore, all the involved keys are guessed, and all the unknown differentials in E_b and E_f can be determined.

Given a boomerang distinguisher with probability P^2, the number of quartets satisfying the input difference of the distinguisher should be at least $P^{-2}2^n$. For the sake of clarity, we denote $2^{-2p} = P^2$, then $p = 30.8$. In our attack, $r_b = n$, so we use partial structures of plaintexts for data collection. By collecting D plaintext-ciphertext pairs, $(D^2)^2 \times 2^{-2n}$ quartets will satisfy the input difference. Therefore, $D = 2^{3n/4} \cdot 2^{p/2} = 2^{63.4}$.

Under the related-key setting, the calculation of time complexity is slightly different from that in the single-key setting introduced in Sect. 3.4. Additionally, the data of our attack will always belong to the same partial structure, so y^{-2} is omitted in T_2. We use D_R to represent the data required under the related-key setting. The complexities of our new attack are as follows:

- The data complexity is $D_R = 4 \cdot D = 2^{3n/4+p/2+2} = 2^{65.4}$.
- The memory complexity is $M = D^2 \cdot 2^{|k_b|+2r_f-2n} = 2^{254.8}$.
- The time complexity:

$$\begin{aligned}
T_0 &= 4 \cdot D_R,\\
T_1 &= (2^{|k_b|} + 2^{|k_f|}) \cdot D_R = 2^{194.4},\\
T_2 &= D^2 \cdot 2^{|k_b|} + D^4 \cdot 2^{|k_f|-2n} = 2^{255.32},\\
T_3 &= 2^{|k_b \cup k_f|} \cdot D^2 \cdot 2^{-2n} = 2^{254.8},\\
T_4 &= 2^{k-h} = 2^{256-(n-2p)} = 2^{253.6}.
\end{aligned}$$

The time complexity of our attack is $\frac{1}{38} \times (T_0 + T_1 + T_2 + T_3 + T_4) \approx 2^{251.07}$ 38-round encryption.

Comparison and Discussion. In our rectangle MITM attack, we modify one cell in the input difference. Compared with the original rectangle key recovery, the probability of the distinguisher is reduced by 2^{-4}, and the k_b has two fewer cells in E_b.

According to the GRMA algorithm, MITM attacks are more effective in situations where subkey bits are balanced on both sides. Therefore, the modified distinguisher is more suitable for the GRMA. By adding 6 rounds on both sides of the distinguisher, the number of the subkey bits is: $|k_b| = |k_f| = 128$. Using the GRMA, we achieve a 38-round rectangle attack on SKINNYe-64-256 v2, which is one more round than the previous attack. It is worth noting that for this distinguisher, the GCRA could not provide an effective 38-round rectangle attack.

5 Conclusion

In this paper, we revisit the generic rectangle key recovery attack [SZY+22] and the differential MITM attack recently proposed in [BDD+23]. Inspired by the holistic strategy and the MITM technique, we propose three new generic key recovery attacks: the classical differential attack, the differential MITM attack, and the rectangle MITM attack, respectively denoted as GCDA, GDMA, and GRMA. By selecting an appropriate strategy (including the type of key recovery attacks, key guessing strategy, etc.), the different terms of the time complexity can be more balanced, resulting in a lower overall time complexity. For application, we apply our new key recovery algorithms to the AES-256, KATAN-32, and SKINNYe-64-256 v2 block ciphers. For AES-256, we provide better results on a 12-round differential attack using the GCDA and the GDMA, and introduce a new 13-round differential attack with reduced data and time complexities. We apply the GDMA, GCDA, and BDMA to KATAN-32, building a dedicated model for key guessing with MILP. With the aid of MILP-based tools, we improve the differential attacks on KATAN-32 from 115 rounds to 151 rounds. For SKINNYe-64-256 v2, we achieve the first 38-round rectangle attack using the GRMA, which extends the previous attack by 1 round.

Acknowledgement. The authors would like to thank anonymous reviewers for their helpful comments and suggestions. The work of this paper was supported by the National Natural Science Foundation of China (Grants 62132008, 62372213, 62202460, 62332007 and U22B2028). Jian Weng is also supported by the Major Program of Guangdong Basic and Applied Research Project under Grant No. 2019B030302008, Science and Technology Major Project of Tibetan Autonomous Region of China under Grant No. XZ202201ZD0006G, Guangdong Provincial Science and Technology Project under Grant No. 2021A0505030033, National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving, Guangdong Hong Kong Joint Laboratory for Data Security and Privacy Protection, and Engineering Research Center of Trustworthy AI, Ministry of Education.

References

[AFK+08] Jean-Philippe Aumasson, Simon Fischer, Shahram Khazaei, Willi Meier, and Christian Rechberger. New features of latin dances: Analysis of Salsa, ChaCha, and Rumba.In Kaisa Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 470–488. Springer, 2008.

[AKM+24] Zahra Ahmadian, Akram Khalesi, Dounia M'foukh, Hossein Moghimi, and María Naya-Plasencia. Improved differential meet-in-the-middle cryptanalysis.In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 280–309. Springer, 2024.

[AL13] Martin R Albrecht and Gregor Leander. An all-in-one approach to differential cryptanalysis for small block ciphers. In *Selected Areas in Cryptography: 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers 19*, pages 1–15. Springer, 2013.

[BCF+21] Marek Broll, Federico Canale, Antonio Flórez-Gutiérrez, Gregor Leander, and María Naya-Plasencia. Generic framework for key-guessing improvements. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 453–483. Springer, 2021.

[BDD+23] Christina Boura, Nicolas David, Patrick Derbez, Gregor Leander, and María Naya-Plasencia. Differential meet-in-the-middle cryptanalysis. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 240–272. Springer, 2023.

[BDD+24] Christina Boura, Nicolas David, Patrick Derbez, Rachelle Heim Boissier, and María Naya-Plasencia. A generic algorithm for efficient key recovery in differential attacks - and its associated tool. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 217–248. Springer, 2024.

[BDF23] Christina Boura, Patrick Derbez, and Margot Funk. Related-key differential analysis of the aes. *IACR Transactions on Symmetric Cryptology*, 2023(4):215–243, 2023.

[BDK01] Eli Biham, Orr Dunkelman, and Nathan Keller. The rectangle attack–rectangling the Serpent. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 340–357. Springer, 2001.

[BDK02] Eli Biham, Orr Dunkelman, and Nathan Keller. New results on boomerang and rectangle attacks. In *International Workshop on Fast Software Encryption*, pages 1–16. Springer, 2002.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Annual International Cryptology Conference*, pages 123–153. Springer, 2016.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptol.*, 4(1):3–72, 1991.

[BS92] Eli Biham and Adi Shamir. Differential cryptanalysis of the full 16-round DES. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 487–496. Springer, 1992.

[DQSW21] Xiaoyang Dong, Lingyue Qin, Siwei Sun, and Xiaoyun Wang. Key guessing strategies for linear key-schedule algorithms in rectangle attacks.*IACR Cryptol. ePrint Arch.*, page 856, 2021.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[JNP14] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In Palash Sarkar and Tetsu Iwata, editors, *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 II*, volume 8874 of *Lecture Notes in Computer Science*, pages 274–288. Springer, 2014.

[JRS22] Amit Jana, Mostafizar Rahman, and Dhiman Saha. Deepand: In-depth modeling of correlated and gates for nlfsr-based lightweight block ciphers. Cryptology ePrint Archive, Paper 2022/1123, 2022. https://eprint.iacr.org/2022/1123. https://eprint.iacr.org/2022/1123.

[KMN10] Simon Knellwolf, Willi Meier, and María Naya-Plasencia.Conditional differential cryptanalysis of NLFSR-based cryptosystems. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2010.

[LKKD08] Jiqiang Lu, Jongsung Kim, Nathan Keller, and Orr Dunkelman. Improving the efficiency of impossible differential cryptanalysis of reduced Camellia and MISTY1. In Tal Malkin, editor, *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008. Proceedings*, volume 4964 of *Lecture Notes in Computer Science*, pages 370–386. Springer, 2008.

[Mat94] Mitsuru Matsui. On correlation between the order of S-boxes and the strength of DES. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 366–375. Springer, 1994.

[MP13] Nicky Mouha and Bart Preneel. A proof that the ARX cipher Salsa20 is secure against differential cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 328, 2013.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.

[NSS20a] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Lightweight authenticated encryption mode suitable for threshold implementation. Cryptology ePrint Archive, Paper 2020/542, 2020. https://eprint.iacr.org/2020/542.

[NSS20b] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Lightweight authenticated encryption mode suitable for threshold implementation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 705–735. Springer, 2020.

[QDW+22] Lingyue Qin, Xiaoyang Dong, Anyu Wang, Jialiang Hua, and Xiaoyun Wang. Mind the tweakey schedule: cryptanalysis on skinnye-64-256. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 287–317. Springer, 2022.

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In Palash Sarkar and Tetsu Iwata, editors, *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*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014.

[SLY+24] Ling Song, Huimin Liu, Qianqian Yang, Yincen Chen, Lei Hu, and Jian Weng. Generic differential key recovery attacks and beyond. Cryptology ePrint Archive, Paper 2024/1447, 2024.

[SWW21] Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the search of differential and linear characteristics with the SAT method. *IACR Trans. Symmetric Cryptol.*, 2021(1):269–315, 2021.

[SYC+24] Ling Song, Qianqian Yang, Yincen Chen, Lei Hu, and Jian Weng. Probabilistic extensions: a one-step framework for finding rectangle attacks and beyond.In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 339–367. Springer, 2024.

[SZY+22] Ling Song, Nana Zhang, Qianqian Yang, Danping Shi, Jiahao Zhao, Lei Hu, and Jian Weng. Optimizing rectangle attacks: A unified and generic framework for key recovery. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 410–440. Springer, 2022.

[WWJZ18] Ning Wang, Xiaoyun Wang, Keting Jia, and Jingyuan Zhao. Differential attacks on reduced SIMON versions with dynamic key-guessing techniques. *Sci. China Inf. Sci.*, 61(9):098103:1–098103:3, 2018.

[YSZ+24] Qianqian Yang, Ling Song, Nana Zhang, Danping Shi, Libo Wang, Jiahao Zhao, Lei Hu, and Jian Weng. Optimizing rectangle and boomerang attacks: A unified and generic framework for key recovery. *Journal of Cryptology*, 37(2):1–62, 2024.

[ZDM+20] Boxin Zhao, Xiaoyang Dong, Willi Meier, Keting Jia, and Gaoli Wang. Generalized related-key rectangle attacks on block ciphers with linear key schedule: applications to SKINNY and GIFT. *Designs, Codes and Cryptography*, 88(6):1103–1126, 2020.

Ultrametric Integral Cryptanalysis

Tim Beyne(✉) and Michiel Verbauwhede

COSIC, KU Leuven, Leuven, Belgium
{tim.beyne,michiel.verbauwhede}@esat.kuleuven.be

Abstract. A systematic method to analyze *divisibility properties* is proposed. In integral cryptanalysis, divisibility properties interpolate between bits that sum to zero (divisibility by two) and saturated bits (divisibility by 2^{n-1} for 2^n inputs). From a theoretical point of view, we construct a new cryptanalytic technique that is a non-Archimedean multiplicative analogue of linear cryptanalysis. It lifts integral cryptanalysis to characteristic zero in the sense that, if all quantities are reduced modulo two, then one recovers the algebraic theory of integral cryptanalysis. The new technique leads to a theory of trails. We develop a tool based on off-the-shelf solvers that automates the analysis of these trails and use it to show that many integral distinguishers on PRESENT and SIMON are stronger than expected.

Keywords: Geometric approach · Integral cryptanalysis · Division property

1 Introduction

Integral cryptanalysis can be approached from two opposing directions. The *structural approach* was formalized by Knudsen and Wagner [24] and stems from the 'Square attack' [15]. It is based on propagating plaintexts with some constant and some saturated parts through a cipher, ultimately resulting in a set of ciphertexts with a saturated part, or some bits that sum to zero. A part of the state is called saturated if all its possible values occur an equal number of times. The *algebraic approach* was introduced by Knudsen [23], based on the observation that the $(d + 1)^{\text{st}}$ derivative of a function of degree d is zero. This yields zero sums, but not saturation properties.

Todo [29] partially consolidated the two approaches by introducing the division property, which characterizes structured sets algebraically. More generally, it was shown by Boura and Canteaut at Crypto 2016 [10] that every set has an equivalent parity set representation. The parity set of a set X is the set of all exponents of monomials that sum to one on X.

However, as the division property is based on arithmetic over $\mathbb{F}_2$, it can describe zero sums but not saturation. The gap is significant: the probability that a uniform random Boolean function sums to zero on a set of size 2^n is $1/2$, but it is saturated with probability approximately[1] $2^{-n/2}/\sqrt{\pi}$. A saturation property consequently corresponds to a stronger filter, which is beneficial for the data and time requirements of key-recovery attacks. In spite of this, the difference is sometimes overlooked.

Thus, one might wonder if there can exist a theory of integral cryptanalysis over a field of characteristic zero rather than over $\mathbb{F}_2$, so that both zero sums and saturation properties can be described by it. In practice, zero sums are found by automated analysis of trails – there are several variants including division trails [32], monomial trails [21] and algebraic trails [4]. These concepts are more-or-less similar to trails in linear cryptanalysis, but the analogy is leaky because the 'correlations' are binary. Optimistically, a theory defined in characteristic zero might strengthen the analogy by allowing correlations 'in between' zero and one.

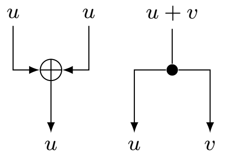
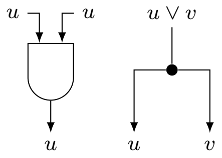
Contribution. We introduce the theory of *ultrametric integral cryptanalysis*, a non-Archimedean multiplicative analogue of linear cryptanalysis. Inspired by the idea that linear cryptanalysis simplifies additions (exclusive or), we construct an analogous theory that simplifies multiplications (bitwise and). Like linear cryptanalysis, it is defined in characteristic zero (over $\mathbb{Q}$), but to obtain a useful theory, we have to change the way distances are measured: we replace the regular (Archimedean) absolute value $|\cdot|$ on $\mathbb{Q}$ with the (non-Archimedean) 2-adic absolute value $|\cdot|_2$. Ultrametric integral cryptanalysis lifts integral cryptanalysis to characteristic zero, in the sense that if all quantities are reduced modulo two, then one obtains the algebraic theory of integral cryptanalysis over $\mathbb{F}_2$ – more precisely, its description using algebraic trails that was recently introduced in ToSC 2023 [4]. Some consequences of the analogy between linear and ultrametric integral cryptanalysis are illustrated in Table 1.

In practical terms, ultrametric integral cryptanalysis provides a systematic way to analyze divisibility properties. For example, one can use it show that the number of times a ciphertext bit equals one, is divisible by 2^ν. In our theory, this can be proven by showing that one or more *correlations* have 2-adic absolute value at most $2^{-\nu}$. Divisibility properties interpolate between zero sums (divisibility by two) and saturation (divisibility by 2^{n-1} for an input space of dimension n). We believe that these properties occur naturally in cryptanalysis, as their existence is essentially an unexplained folklore observation. For example, in Todo's invited talk at FSE 2023, divisibility by four pops up at 43:30[2].

The construction of our theory follows the geometric approach [1], which was introduced at Asiacrypt 2021 as a general description of linear cryptanalysis. In particular, we express the 'pushforward operators' that describe the propagation of states through functions as matrices relative to a carefully chosen basis. The basis is constructed in Sect. 4, and is uniquely defined by the property that it diagonalizes the matrices corresponding to multiplications $x \mapsto x \wedge k$. This is analogous to linear cryptanalysis, which diagonalizes the matrices corresponding

[1] The exact probability is equal to $\binom{2^n}{2^{n-1}}/2^{2^n}$.
[2] https://www.youtube.com/watch?v=hgHJu6Qr0Us&t=2610s.

Table 1. The analogy between linear and ultrametric integral cryptanalysis.

	Linear cryptanalysis	Ultrametric integral cryptanalysis
Field of definition	$\mathbb{Q}$ or $\mathbb{R}$ Archimedean ordinary absolute value $\lvert \cdot \rvert$	$\mathbb{Q}$ or $\mathbb{Q}_2$ non-Archimedean 2-adic absolute value $\lvert \cdot \rvert_2$
Geometric theory	'diagonalizes' additions $x \mapsto x + k$ additive characters χ^u Fourier transformation $\mathscr{F}$ $C^{\mathsf{F}} = \mathscr{F} T^{\mathsf{F}} \mathscr{F}^{-1}$	'diagonalizes' multiplications $x \mapsto x \wedge k$ multiplicative characters μ^u ultrametric integral change-of-basis $\mathscr{U}$ $A^{\mathsf{F}} = \mathscr{U} T^{\mathsf{F}} \mathscr{U}^{-1}$
Theory of trails	masks $u_1, u_2, \ldots$ correlation $\prod_{i=1}^{r} C^{\mathsf{F}_i}_{u_{i+1}, u_i}$ linear functions linear diffusion, nonlinear confusion 	exponents $u_1, u_2, \ldots$ correlation $\prod_{i=1}^{r} A^{\mathsf{F}_i}_{u_{i+1}, u_i}$ multiplicative functions nonlinear diffusion, linear confusion 

to additions $x \mapsto x+k$. Our choice of basis leads to ultrametric integral transition matrices, which are analogous to correlation matrices in linear cryptanalysis. We show that these matrices are closely related to the numerical normal form of Boolean functions. Like for correlation matrices, composition of functions corresponds to multiplication of ultrametric integral transition matrices.

Section 5 develops the theory of ultrametric integral trails. Properties can be evaluated by summing the correlations of trails, and this can be made practical using dominant trails (Theorem 8). Unlike in linear cryptanalysis, the dominant trail approximation is not heuristic in the ultrametric setting because the sum of many small numbers in a non-Archimedean field is always small.

Section 6 investigates the properties of ultrametric integral transition matrices. The main result in this section is Theorem 9, which characterizes the ultrametric integral transition matrices of low-degree functions. This result implies the Ax-Katz theorem [22] (over $\mathbb{F}_2$), which states that the number of solutions of a system of m equations of degree d in n variables is divisible by $2^{\lceil n/d \rceil - m}$. Interestingly, our proof is cryptanalytic: the result follows by analyzing ultrametric integral trails in a generic function of degree d. We also show that ultrametric integral transition matrices can be computed in time proportional to their size (up to logarithmic factors), and propagation rules for copy and xor operations are derived. Theorem 10 relates correlation matrices and ultrametric integral transition matrices, explaining and strengthening a result that was used by Canteaut and Videau [11] and Boura and Canteaut [9] to bound degrees.

Finally, in Sects. 7 and 8, we develop an automated tool to analyze ultrametric integral trails using off-the-shelf solvers and apply it to PRESENT and SIMON. Our analysis shows that the distinguishers for reduced-round PRESENT presented by Boura and Canteaut at Crypto 2016 [10] and by Wang *et al.* at Asiacrypt 2019 [31] are stronger than previously believed. For many output bits, we find divisibility by higher powers of two (ranging from 2^2 to 2^9). We also demonstrate that ultrametric integral cryptanalysis can be used to find zero-correlation linear approximations, and illustrate how divisibility properties are useful to reduce the data-complexity of key-recovery attacks. For SIMON, we reconsider the distinguishers found by Todo [29], Todo and Morii [30] and Xiang *et al.* [32] and prove higher divisibility. In addition, we slightly improve the modelling of SIMON based on the analogy between linear and multiplicative functions. This observation is also applicable to ordinary integral cryptanalysis. The source code of our tool can be found at https://github.com/MichielVerbauwhede/ultrametric-integral-cryptanalysis and an extended version of this work is available on https://ia.cr/2024/722 [5].

Future Work. Ultrametric integral cryptanalysis can be extended to all primes p and all commutative inverse monoids, including $\mathbb{F}_q^n$ with multiplication. However, this requires more technical background because the theory must be defined over the p-adic numbers $\mathbb{Q}_p$ when $p \geq 5$. Nevertheless, our results generalize to this setting. In the interest of simplicity, we focus on the case $p = q = 2$ in this paper.

2 Background

The theory of ultrametric integral cryptanalysis is based on the geometric approach, which we present (for the one-dimensional case) in Sect. 2.1 in slightly modified form. Section 2.2 describes linear cryptanalysis from this point of view, and integral cryptanalysis is discussed in Sect. 2.3.

2.1 Geometric Approach

The geometric approach to symmetric-key cryptanalysis was introduced at Asiacrypt 2021 [1] as an alternative description of linear cryptanalysis. In a subsequent paper at Crypto 2022 [3], the same approach was used to construct a fixed-key theory of differential cryptanalysis. The role of the geometric approach in this paper is similar to that in the latter work: it is used to construct a new cryptanalytic theory, analogous to the theory of linear cryptanalysis.

Let k be a field – in [1,3], k is either $\mathbb{R}$ or $\mathbb{C}$. The free k-vector space on $\mathbb{F}_2^n$ consists of all formal k-linear combinations of elements of $\mathbb{F}_2^n$, with addition defined coordinate-wise. That is, every element a of $k[\mathbb{F}_2^n]$ is of the form

$$a = \sum_{x \in \mathbb{F}_2^n} a_x \, \delta_x \,,$$

where the values a_x are arbitrary elements of k and δ_x is the formal basis vector corresponding to x. Cryptanalytically, a represents an assignment of weights (elements of k) to the elements of $\mathbb{F}_2^n$. For example, a subset $X \subseteq \mathbb{F}_2^n$ corresponds to a vector $\delta_X = \sum_{x \in X} \delta_x$ in $k[\mathbb{F}_2^n]$. Applying a function $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ to the state transforms the assignment of weights on $\mathbb{F}_2^n$ to an assignment of weights on $\mathbb{F}_2^m$. The relation is given by a linear operator.

Definition 1. *Let* $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ *be a function. The pushforward operator of* F *is the linear map* $T^{\mathsf{F}} : k[\mathbb{F}_2^n] \to k[\mathbb{F}_2^m]$ *defined by* $T^{\mathsf{F}}\, \delta_x = \delta_{\mathsf{F}(x)}$ *for all* x *in* $\mathbb{F}_2^n$*.*

The matrix representation of T^{F} with respect to the standard basis will be called the *transition matrix* of F. Its rows and columns are indexed by elements of $\mathbb{F}_2^m$ and $\mathbb{F}_2^n$ respectively. Depending on the context, the notation T^{F} either refers to the pushforward operator of F or to its transition matrix.

Transition matrices satisfy several properties, two of which are summarized in Theorem 1. In this theorem, $\bigotimes$ denotes the Kronecker product of matrices. That is, $(A \otimes B)_{y_1 \| y_2, x_1 \| x_2} = A_{y_1, x_1} B_{y_2, x_2}$, where $\|$ denotes concatenation.

Theorem 1. *For the transition matrix of a function* $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$*:*

(1) If $\mathsf{F}(x_1 \| \cdots \| x_l) = \mathsf{F}_1(x_1) \| \cdots \| \mathsf{F}_l(x_l)$*, then* $T^{\mathsf{F}} = \bigotimes_{i=1}^{l} T^{\mathsf{F}_i}$*.*
(2) If $\mathsf{F} = \mathsf{F}_r \circ \cdots \circ \mathsf{F}_2 \circ \mathsf{F}_1$*, then* $T^{\mathsf{F}} = T^{\mathsf{F}_r} \cdots T^{\mathsf{F}_2} T^{\mathsf{F}_1}$*.*

A dual way to assign weights to the elements of $\mathbb{F}_2^n$ is using functions. Let $k^{\mathbb{F}_2^n}$ be the vector space of k-valued functions on $\mathbb{F}_2^n$. Every function in $k^{\mathbb{F}_2^n}$ can be extended to a function on $k[\mathbb{F}_2^n]$ by linearity. Conversely, every linear function on $k[\mathbb{F}_2^n]$ is uniquely determined by its image on the basis vectors δ_x with x in $\mathbb{F}_2^n$. Hence, we identify $k^{\mathbb{F}_2^n}$ with the dual vector space[3] of $k[\mathbb{F}_2^n]$. The functions δ^x with $\delta^x(\delta_x) = \delta^x(x) = 1$ and $\delta^x(\delta_y) = \delta^x(y) = 0$ for $y \neq x$ are a basis for $k^{\mathbb{F}_2^n}$.

A cryptanalytic property of a function $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a pair (a, b) with a in $k[\mathbb{F}_2^n]$ and b in $k^{\mathbb{F}_2^m}$. The *evaluation* of a property (a, b) is defined as $b(T^{\mathsf{F}} a)$. This is typically a combinatorial quantity of interest, such as the correlation of a linear approximation.

Example 1. Let X and Y be subsets of $\mathbb{F}_2^n$ and $\mathbb{F}_2^m$ respectively. The evaluation of (δ_X, δ^Y), with $\delta^Y = \sum_{y \in Y} \delta^y$ the indicator function of Y, is equal to

$$\delta^Y \big(T^{\mathsf{F}} \delta_X\big) = \sum_{x \in X} \delta^Y \big(\delta_{\mathsf{F}(x)}\big) = |\{x \in X \mid \mathsf{F}(x) \in Y\}|\,.$$

If F is a permutation, then the property evaluates to $|Y \cap \mathsf{F}(X)|$. ▷

If we apply a function $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$, then functions on $\mathbb{F}_2^m$ transform to functions on $\mathbb{F}_2^n$. The relation is given by the pullback operator $T^{\mathsf{F}^\vee}$, which is the adjoint[4] of the pushforward operator. That is, $T^{\mathsf{F}^\vee} f = f \circ \mathsf{F}$. Its standard

[3] The dual vector space of $k[\mathbb{F}_2^n]$ is the space of all linear functions from $k[\mathbb{F}_2^n]$ to k.
[4] The adjoint of $L : k[\mathbb{F}_2^n] \to k[\mathbb{F}_2^m]$ is a map $L^\vee : k^{\mathbb{F}_2^m} \to k^{\mathbb{F}_2^n}$ s.t. $\big(L^\vee b\big)(a) = b\big(La\big)$.

basis matrix representation is the transpose of the transition matrix. Pullback operators also satisfy the properties listed in Theorem 1, with the order of multiplication reversed for property (2).

Different cryptanalytic theories are obtained by expressing cryptanalytic properties with respect to different pairs of dual bases for $k[\mathbb{F}_2^n]$ and $k^{\mathbb{F}_2^m}$. A pair of bases for $k[\mathbb{F}_2^n]$ and $k^{\mathbb{F}_2^m}$ consisting of vectors β_u and β^u, labeled by u in $\mathbb{F}_2^n$, is called dual if $\beta^u(\beta_u) = 1$ and $\beta^v(\beta_u) = 0$ for all $u \neq v$.

If $\mathscr{B} : k[\mathbb{F}_2^n] \to k[\mathbb{F}_2^n]$ is the change-of-basis transformation defined by $\mathscr{B}\,\beta_u = \delta_u$, then $\mathscr{B}^{-\vee}\,\beta^v = \delta^v$. That is, $\mathscr{B}^{-\vee}$ is the change-of-basis transformation for the dual basis. Let $\mathscr{B}_n$ and $\mathscr{B}_m$ be change-of-basis transformations on $k[\mathbb{F}_2^n]$ and $k[\mathbb{F}_2^m]$ respectively. For a cryptanalytic property (a, b), set $a^\wedge = \mathscr{B}_n\, a$ and $b^\wedge = \mathscr{B}_m^{-\vee}\, b$. The evaluation of (a, b) can then be expressed as

$$b\big(T^{\mathsf{F}}\, a\big) = b^\wedge\Big(\mathscr{B}_m\, T^{\mathsf{F}}\, \mathscr{B}_n^{-1} a^\wedge\Big)\,.$$

Hence, if properties are expressed in the new basis, their propagation is described by the matrix $\mathscr{B}_m\, T^{\mathsf{F}}\, \mathscr{B}_n^{-1}$. Since this matrix is similar to T^{F}, it also satisfies the properties listed in Theorem 1.

Example 2. The vectors $(\delta_0 + \delta_1)/2$ and $(\delta_0 - \delta_1)/2$ form a basis for $k[\mathbb{F}_2]$. The dual basis vectors are given by $\delta^0 + \delta^1$ and $\delta^0 - \delta^1$ and form a basis for $k^{\mathbb{F}_2}$. Hence, the change-of-basis transformation $\mathscr{B}$ and its dual $\mathscr{B}^{-\vee}$ are given by

$$\mathscr{B} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad \mathscr{B}^{-\vee} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\,.$$

For the function $\mathsf{F} : x \mapsto x + 1$, the matrix $\mathscr{B}\, T^{\mathsf{F}} \mathscr{B}^{-\vee}$ is equal to

$$\mathscr{B}\, T^{\mathsf{F}} \mathscr{B}^{-\vee} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\,.$$

The matrix $\mathscr{B}^{-\vee}\, T^{\mathsf{F}^\vee}\, \big(\mathscr{B}^{-\vee}\big)^{-1}$ is the transpose of the above matrix, which happens to be the same in this example. ▷

2.2 Linear Cryptanalysis

In linear cryptanalysis, the field k is chosen as $\mathbb{R}$ or $\mathbb{C}$ and one works in a basis of group characters and its dual. The characters of the additive group $\mathbb{F}_2^n$ are the homomorphisms $\chi^u : \mathbb{F}_2^n \to \mathbb{R}$ with $\chi^u(x) = (-1)^{u^\mathsf{T} x}$. The dual basis consists of the vectors χ_u in $\mathbb{R}[\mathbb{F}_2^n]$ with $\delta^x(\chi_u) = \chi^u(x)/2^n$. The corresponding change-of-basis transformation $\mathscr{F}_n : \mathbb{R}[\mathbb{F}_2^n] \to \mathbb{R}[\mathbb{F}_2^n]$ is called the Fourier transformation.

The Fourier transformation simultaneously diagonalizes the transition matrices T^{F} of all translations $\mathsf{F}(x) = x + t$. In fact, as shown in [1, §2.2], this is by construction: the character basis is the only basis with this property. The Fourier transformation of the transition matrix of a function F is its correlation matrix $C^{\mathsf{F}} = \mathscr{F}_m T^{\mathsf{F}} \mathscr{F}_n^{-1}$. These matrices were introduced by Daemen [14], motivated by the fact that the coordinates $C^{\mathsf{F}}_{v,u}$ are the correlations of linear approximations with input mask u and output mask v over F:

$$C^{\mathsf{F}}_{v,u} = \chi^v\big(T^{\mathsf{F}}\,\chi_u\big) = 2\Pr_{\mathbf{x}}\left[v^{\mathsf{T}}\mathsf{F}(\mathbf{x}) = u^{\mathsf{T}}\mathbf{x}\right] - 1\,,$$

with $\mathbf{x}$ uniform random on $\mathbb{F}_2^n$.

The properties listed in Theorem 1 carry over to correlation matrices. In particular, if $\mathsf{F} = \mathsf{F}_r \circ \cdots \circ \mathsf{F}_2 \circ \mathsf{F}_1$, then $C^{\mathsf{F}} = C^{\mathsf{F}_r} \cdots C^{\mathsf{F}_2} C^{\mathsf{F}_1}$. If one defines a linear trail as a tuple of $r+1$ masks, then Theorem 1 (2) implies that the correlation of a linear approximation is equal to the sum of the correlations of all linear trails with matching input and output masks:

$$C^{\mathsf{F}}_{u_{r+1},u_1} = \sum_{u_2,\ldots,u_r} \prod_{i=1}^{r} C^{\mathsf{F}_i}_{u_{i+1},u_i}\,,$$

where the product $\prod_{i=1}^{r} C^{\mathsf{F}_i}_{u_{i+1},u_i}$ is called the correlation of the trail $(u_1,\ldots,u_{r+1})$. This result is usually used in the form of the principle of dominant trails.

Theorem 2 (Dominant trails). *Let* $\mathsf{F} = \mathsf{F}_r \circ \cdots \circ \mathsf{F}_2 \circ \mathsf{F}_1$*. For all subsets* Λ *of the set* Ω *of all linear trails from* u_1 *to* u_{r+1}*,*

$$\left| C^{\mathsf{F}}_{u_{r+1},u_1} - \sum_{u\in\Lambda} \prod_{i=1}^{r} C^{\mathsf{F}_i}_{u_{i+1},u_i} \right| = \left| \sum_{u\in\Omega\setminus\Lambda} \prod_{i=1}^{r} C^{\mathsf{F}_i}_{u_{i+1},u_i} \right|,$$

where $u = (u_1, u_2, \ldots, u_{r+1})$*.*

The idea of Theorem 2 is that the trails in $\Omega\setminus\Lambda$ contribute little to $C^{\mathsf{F}}_{u_{r+1},u_1}$. In practice, Theorem 2 is used heuristically: one assumes that if the absolute values of the correlations of trails in $\Omega\setminus\Lambda$ are small, then so is the absolute value of their sum. This approach is useful for linear approximations (u, v) with large absolute correlation $|C^{\mathsf{F}}_{v,u}|$. Contrary to this, zero-correlation linear cryptanalysis [8] relies on linear approximations with $C^{\mathsf{F}}_{v,u} = 0$. Such approximations can be found using Theorem 2, but with $\Lambda = \emptyset$ and by showing that all trails in Ω have correlation zero. Bogdanov *et al.* [7] have shown that zero-correlation linear approximations and saturation properties are closely related. Following Sun *et al.* [28, Corollary 4], if U is a vector space of masks such that $C^{\mathsf{F}}_{v,u} = 0$ for all u in U, then the restriction of $v^{\mathsf{T}}\mathsf{F}$ to a coset of $U^{\perp}$ is a balanced Boolean function. This means that the linear combination of output bits corresponding to the mask v is saturated when the input set is a coset of $U^{\perp}$.

2.3 Integral Cryptanalysis

Traditionally, integral cryptanalysis is used to find affine subspaces X such that $\sum_{x\in X} f(x) = 0$ for a coordinate function f of a cryptographic primitive. As mentioned in the introduction, such properties can be approached in two different ways. Wagner and Knudsen [24] describe the propagation of structured sets with some constant and some saturated parts. In earlier work, Knudsen [23] proposed a purely algebraic approach based on higher-order derivatives.

The algebraic point of view is best understood using the algebraic normal form. This is the unique representation of a Boolean function as a polynomial in $\mathbb{F}_2[x_1, \ldots, x_n]/(x_1^2 - x_1, \ldots, x_n^2 - x_n)$. If the algebraic normal form of f does not contain any monomials $x^u = \prod_{i=1}^n x_i^{u_i}$ such that $\sum_{x \in X} x^u = 1$, then f sums to zero on X. The relation with the properties of X and the structural point of view of Wagner and Knudsen was poorly understood before the introduction of the division property by Todo [29]. The division property characterizes X by the set of exponents u such that $\sum_{x \in X} x^u = 1$. This set was called the parity set of X by Boura and Canteaut [10].

A recent paper in ToSC 2023 [4] shows that it is possible to construct a theory of integral cryptanalysis based on the geometric approach. It describes cryptanalytic properties (a, b) over the field $k = \mathbb{F}_2$. If $a = \delta_X$, then the evaluation $b(T^{\mathsf{F}} a)$ of (a, b) is equal to the sum $\sum_{x \in X} f(x)$ with $f = b \circ \mathsf{F}$. Since b is a Boolean function, there exists a change-of-basis transformation $\mathscr{M}_m$ so that $(\mathscr{M}_m b)(u)$ is the coefficient of x^u in the algebraic normal form of b. The transformation $\mathscr{M}_m$ is the binary Möbius transformation. Relative to this change-of-basis, (a, b) evaluates to

$$b\big(T^{\mathsf{F}} a\big) = b^{\wedge}\left(\mathscr{P}_m T^{\mathsf{F}} \mathscr{P}_n^{-1} a^{\wedge}\right),$$

with $a^{\wedge} = \mathscr{P}_n a$ and $\mathscr{P}_n = \mathscr{M}_n^{-\vee}$. It was shown in [4] that $\mathscr{P}_n \delta_X = \delta_Y$ with Y the parity set of X.

The matrix $A^{\mathsf{F}} = \mathscr{P}_m T^{\mathsf{F}} \mathscr{P}_n^{-1}$ is called the algebraic transition matrix of F and it satisfies the usual properties from Theorem 1. It holds that $A^{\mathsf{F}}_{v,u} = 1$ if and only if x^u occurs in the algebraic normal form of F^v. In particular, if $\mathsf{F} = \mathsf{F}_r \circ \cdots \circ \mathsf{F}_1$, then $A^{\mathsf{F}} = \prod_{i=1}^r A^{\mathsf{F}_i}$. This leads to a theory of algebraic trails $(u_1, \ldots, u_{r+1})$ with correlation $\prod_{i=1}^r A^{\mathsf{F}_i}_{u_{i+1}, u_i}$ such that

$$A^{\mathsf{F}}_{u_{r+1}, u_1} = \sum_{u_2, \ldots, u_r} \prod_{i=1}^r A^{\mathsf{F}_i}_{u_{i+1}, u_i}.$$

This yields an alternative explanation of division trails [32], monomial prediction [21] and the three-subset division property without unknown subset [19].

However, the theory of algebraic trails is not completely satisfactory. The motivation of the division property was to combine the best of the structural and algebraic approaches to integral cryptanalysis, but this was only partially achieved because saturation properties cannot be described over $\mathbb{F}_2$. For example, if zero-correlation linear cryptanalysis shows that the restriction of $f = v^{\mathsf{T}} \mathsf{F}$ to a coset X of $U^{\perp}$ is a balanced Boolean function, then $\sum_{x \in X} f(x) = 0$. However, one actually has the stronger property that $f(x) = 1$ has $|X|/2$ solutions for x in X. Hence, useful information is lost by reducing $|X|/2$ modulo two.

3 Divisibility Properties

Suppose that one of the coordinate functions f of a primitive is saturated for an affine input space X of dimension d. This implies that the number of values x in X such that $f(x) = 1$ is divisible by 2^{d-1}. If f sums to zero, then the number of such values is only divisible by two. This raises a natural question: can we find zero sums so that the number of solutions to $f(x) = 1$ in X is actually divisible by a larger power of two?

This turns out to be common. Section 3.1 describes one instance, to be used as a running example. Section 3.2 explains how divisibility properties can be described using the geometric approach.

3.1 Example for PRESENT

In their work introducing parity sets, Boura and Canteaut [10] describe the following integral property for four rounds of the block cipher PRESENT [6]. For a set of 16 plaintexts obtained by saturating the input of the rightmost S-box, every ciphertext bit sums to zero. Using zero-correlation linear cryptanalysis, one can show that first ciphertext bit is saturated, so we focus on the second bit instead.

Figure 1 shows a histogram of the number of times the second output bit is equal to one for different choices of the key. This bit is clearly not saturated, since for some keys the number of ones differs from $8 = 16/2$. The feature of interest to us are the gaps in the histogram. Indeed, the number of ones is always a multiple of four. The analysis of Boura and Canteaut explains the divisibility by two, but integral cryptanalysis cannot prove divisibility by four.

The second bit is not the only one exhibiting divisibility by four or more; some experimental results for the other bits are summarized in Appendix F of the extended version. In the remainder of this work, we develop the necessary techniques to systematically find and prove such properties. The observation in Fig. 1 will be used as a running example; a complete explanation is given in Sect. 5.2. Further results on PRESENT are contained in Sect. 7.

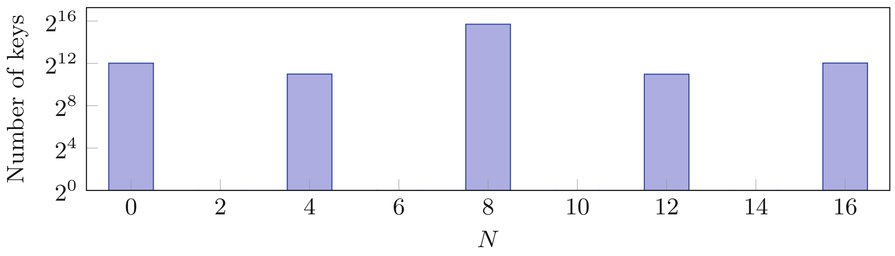


Fig. 1. Number of keys (log scale) so that the second output bit of four-round PRESENT is equal to one N times. The experiment was performed for 2^{16} keys.

3.2 Description Using the Geometric Approach

To describe arbitrary divisibility properties, one should work over the integers rather than over $\mathbb{F}_2$. Since the rational numbers are the smallest field containing the integers, let us apply the geometric approach with $k = \mathbb{Q}$ for now.

Let X be an input set, and set $a = \delta_X$ in $\mathbb{Q}\big[\mathbb{F}_2^n\big]$. Furthermore, let b in $\mathbb{Q}^{\mathbb{F}_2^m}$ be a function that maps the relevant bit (the second, for Fig. 1) to its integer value. For a function $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$, divisibility by 2^ν can be expressed as $b(T^{\mathsf{F}} a) = \sum_{x \in X} b(\mathsf{F}(x)) \equiv 0 \pmod{2^\nu}$.

Alternatively, divisibility by 2^ν is equivalent to $|b(T^{\mathsf{F}}\, a)|_2 \leq 2^{-\nu}$ with $|\cdot|_2$ the 2-adic absolute value on $\mathbb{Q}$. The 2-adic absolute value of a rational number $x = 2^\nu\, \frac{r}{s}$, with r and s odd integers, is equal to $2^{-\nu}$. One advantage of expressing divisibility using the 2-adic absolute value is that we do not need to worry about whether or not the coordinates of a and b are integers. More importantly, it suggests a strong analogy with linear cryptanalysis. Most of the time, it is not possible to evaluate cryptanalytic properties exactly. Instead, we estimate its evaluation as follows:

$$b\big(T^{\mathsf{F}} a\big) = c + \varepsilon\,,$$

where c is the estimate and ε is an error. In linear cryptanalysis, the estimate is accurate if $|\varepsilon|$ is small. For divisibility properties, the accuracy of the estimate is instead measured by $|\varepsilon|_2$. Although the discussion above focused on the case $c = 0$, there is no reason not to consider $c \neq 0$. Indeed, integral cryptanalysis can also be used to deduce constant sums modulo two and cube attacks.

Despite the analogy, the metric structure on $\mathbb{Q}$ defined by $|\cdot|_2$ is completely different from that defined by $|\cdot|$. This is because $|\cdot|_2$ satisfies the *ultrametric* triangle inequality:

$$|x + y|_2 \leq \max\big\{|x|_2, |y|_2\big\}\,.$$

The fact that this inequality is stronger than the usual triangle inequality $|x + y| \leq |x| + |y|$ plays an essential role in the theory of *ultrametric integral cryptanalysis* that is developed in the next sections. However, this is only one aspect of the theory. Another issue is the choice of basis, and this is addressed in Sect. 4.

Remark 1. As mentioned in Sect. 2.2, linear cryptanalysis is typically described over $\mathbb{R}$. For most applications $\mathbb{Q}$ is actually sufficient, but the geometry of $\mathbb{R}$ is nicer because it is metrically complete with respect to $|\cdot|$. The metric completion of $\mathbb{Q}$ with respect to $|\cdot|_2$ is the field of 2-adic numbers $\mathbb{Q}_2$. Hence, working with properties defined over $\mathbb{Q}_2$ would be somewhat nicer. However, for simplicity, we continue to work over $\mathbb{Q}$ throughout this paper. ▷

4 Lifting Integral Cryptanalysis

This section defines a suitable basis to analyze divisibility properties such as the observation from Sect. 3. Section 4.1 motivates the choice of basis by analogy with linear cryptanalysis. Whereas linear cryptanalysis simplifies addition in $\mathbb{F}_2^n$, the new basis simplifies multiplication *i.e.* bitwise and. The basis and its dual are constructed in Sect. 4.2. In Sect. 4.3, we express the pushforward operator of a function relative to the new basis. This leads to an analogue of correlation matrices that we call *ultrametric integral transition matrices.* The algebraic transition matrix of a function turns out to be the reduction of its ultrametric integral transition matrix modulo two. Hence, the theory we construct lifts integral cryptanalysis from $\mathbb{F}_2$ to $\mathbb{Q}$, or more generally $\mathbb{Q}_2$.

4.1 Motivation

From the viewpoint of the geometric approach, linear cryptanalysis is successful because it diagonalizes the transition matrices of translations (including key additions). This is achieved by working relative to the basis of characters of the additive group $\mathbb{F}_2^n$. However, $\mathbb{F}_2^n$ also has multiplicative structure with the bitwise and operation $\wedge$.

Although ciphers rarely use bitwise and with constants, nonlinear layers can often be expressed in terms of a small number of and gates. Hence, choosing a basis that maximally simplifies bitwise and is still of interest. Note, though, that $(\mathbb{F}_2^n, \wedge)$ is a monoid but not a group, because only $11\cdots1$ has an inverse. Nevertheless, the definition of characters can be extended to monoids.

Definition 2. *Let k be a field. A character of a monoid M is a homomorphism of monoids $\chi : M \to k$. That is, $\chi(1) = 1$ and $\chi(xy) = \chi(x)\chi(y)$ for all x and y in M.*

For convenience, for m in M, denote the pushforward operator of the function $x \mapsto m{\cdot}x$ by T^m. Theorem 3 shows that, like in the case of groups, the characters of M are eigenvectors of $T^{m^\vee}$. Hence, diagonalizing $T^{m^\vee}$ amounts to finding a basis of characters for the vector space of k-valued functions on M.

Theorem 3. *Let χ be a character of a finite monoid M. For all m in M, χ is an eigenvector of $T^{m^\vee}$ with eigenvalue $\chi(m)$.*

Proof. For all x in M, we have $(T^{m^\vee}\chi)(x) = \chi(m \cdot x) = \chi(m)\chi(x)$. That is, $T^{m^\vee}\chi = \chi(m)\,\chi$. Hence, χ is an eigenvector with eigenvalue $\chi(m)$. □

By a theorem of Dedekind [16, §44], characters are linearly independent. The question of whether or not there are enough characters to obtain a basis is answered by representation theory. This is possible for all finite *commutative inverse*[5] monoids, provided that k has characteristic zero and contains enough roots of unity [27, §5.2]. The monoid $(\mathbb{F}_2^n, \wedge)$ is commutative and inverse.

Having found the basis of characters $\chi^1, \ldots, \chi^{|M|}$, we can construct its dual basis $\chi_1, \ldots, \chi_{|M|}$ in $k[M]$. It is not difficult to see that $\chi_1, \ldots, \chi_{|M|}$ are eigenvectors of T^m. Indeed, we have that

$$T^m\chi_j = \sum_{i=1}^{|M|} \chi^i\big(T^m\chi_j\big)\,\chi_i = \sum_{i=1}^{|M|} \underbrace{\big(T^{m^\vee}\chi^i\big)(\chi_j)}_{\chi^i(m)\,\chi^i(\chi_j)}\,\chi_i = \chi^j(m)\,\chi_j\,.$$

[5] A monoid M is inverse if for all x in M, there exists a y such that $xyx = x$.

In Sect. 4.2, we explicitly construct the basis of characters and its dual for the monoid $(\mathbb{F}_2^n, \wedge)$ and the field $\mathbb{Q}$.

4.2 Ultrametric Integral Basis

Theorem 4 below shows that the characters of the monoid $(\mathbb{F}_2^n, \wedge)$ are given by the lifted monomial functions $\mu^v : \mathbb{F}_2^n \to \mathbb{Q}$ with $\mu^v(x) = \tau\big(x^v\big)$ for v in $\mathbb{F}_2^n$. Here, $\tau : \mathbb{F}_2 \to \mathbb{Q}$ is the embedding[6] defined by $\tau(0) = 0$ and $\tau(1) = 1$.

Theorem 4. *Every character of* $(\mathbb{F}_2^n, \wedge)$ *is equal to* μ^v *for some* v *in* $\mathbb{F}_2^n$.

Proof. The unit of $(\mathbb{F}_2^n, \wedge)$ is equal to $11\cdots1$. From the definition of μ^v, it is clear that $\mu^v(11\cdots1) = 1$ for all v. The multiplicativity of μ^v follows from the multiplicativity of τ and of monomials over $\mathbb{F}_2$. There are no other characters, because the functions μ^v are distinct, the dimension of $\mathbb{Q}^{\mathbb{F}_2^n}$ is 2^n, and characters are linearly independent. Hence, the functions of the form μ^v form a complete set of characters. $\square$

Like in the case of finite groups, the characters of a finite commutative inverse monoid themselves form a monoid under pointwise multiplication [27, Exercise 9.1]. This is called the dual monoid. The dual monoid of $(\mathbb{F}_2^n, \wedge)$ is essentially $(\mathbb{F}_2^n, \vee)$, where $\vee$ denotes bitwise-or. Indeed, $\mu^u \mu^v = \mu^{u \vee v}$.

Following Sect. 4.1, to find the eigenvectors of the pushforward operators T^m with m in $\mathbb{F}_2^n$, we construct the dual of the character basis. Theorem 5 shows that the dual basis consists of the vectors μ_v in $\mathbb{Q}\big[\mathbb{F}_2^n\big]$, with v in $\mathbb{F}_2^n$ and

$$\mu_v = \sum_{x \preccurlyeq v} (-1)^{\mathsf{wt}(x+v)}\, \delta_x \,,$$

where $\mathsf{wt}(x)$ is the Hamming weight of x and the sum is over $x \preccurlyeq v$ (bitwise order) in $\mathbb{F}_2^n$. The set of vectors μ_v will be called the *ultrametric integral basis.*

Theorem 5. *The ultrametric integral basis* $\{\mu_v \mid v \in \mathbb{F}_2^n\}$ *is dual to the character basis* $\{\mu^v \mid v \in \mathbb{F}_2^n\}$*, with in particular* $\mu^v(\mu_v) = 1$.

Proof. If $b = \sum_{x \in \mathbb{F}_2^n} b_x\, \delta_x$ is one of the dual basis vectors, then there exists a v in $\mathbb{F}_2^n$ such that $\mu^v(b) = 1$ and $\mu^u(b) = 0$ for all $u \neq v$. By linearity,

$$\mu^u(b) = \sum_{x \in \mathbb{F}_2^n} b_x\, \mu^u(x) = \sum_{x \in \mathbb{F}_2^n} b_x\, \tau(x^u) = \sum_{x \succcurlyeq u} b_x \,.$$

The sum on the right-hand side is over all elements greater than u in the partially ordered set $\mathbb{F}_2^n$. Such sums can be inverted using the Möbius inversion formula [26, Prop. 2], which is just the inclusion-exclusion principle for $\mathbb{F}_2^n$:

$$b_x = \sum_{u \succcurlyeq x} (-1)^{\mathsf{wt}(x+u)} \mu^u(b) = \begin{cases} (-1)^{\mathsf{wt}(x+v)} & \text{if } v \succcurlyeq x\,, \\ 0 & \text{else}\,. \end{cases}$$

It follows that $b = \mu_v$. $\square$

[6] The symbol τ refers to the fact that $\tau : \mathbb{F}_2 \to \mathbb{Q}_2$ is a Teichmüller character of $\mathbb{F}_2$.

Section 4.3 relies on the change-of-basis transformation $\mathscr{U}_n$ from the standard basis of $\mathbb{Q}\big[\mathbb{F}_2^n\big]$ to the ultrametric integral basis. The subscript n will be dropped when the context resolves the ambiguity. This transformation maps μ_v to δ_v for all v in $\mathbb{F}_2^n$. That is, $\mathscr{U} : \mathbb{Q}\big[\mathbb{F}_2^n\big] \to \mathbb{Q}\big[\mathbb{F}_2^n\big]$ is defined by extending $\mathscr{U}\mu_v = \delta_v$ linearly to all of $\mathbb{Q}\big[\mathbb{F}_2^n\big]$. From the definition of μ^v and μ_v, one can see that

$$\mathscr{U}_n = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^{\otimes n} \quad \text{and} \quad \mathscr{U}_n^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}^{\otimes n} . \tag{1}$$

A similar change-of-basis transformation can be defined on $\mathbb{Q}^{\mathbb{F}_2^n}$ for the character basis, mapping μ^v to δ^v. As explained in Sect. 2.1, this transformation is equal to $\mathscr{U}^{-\vee}$. Lemma 1 gives an analytic expression for $\mathscr{U}$ and its inverse.

Lemma 1. *Let a in $\mathbb{Q}\big[\mathbb{F}_2^n\big]$ be a vector and let $a^\wedge = \mathscr{U}a$. It holds that*

$$a_v^\wedge = \sum_{x \succcurlyeq v} a_x \qquad \textit{and} \qquad a_x = \sum_{v \succcurlyeq x} (-1)^{\mathsf{wt}(x+v)}\, a_v^\wedge .$$

Proof. Immediate from (1). An alternative proof is given in Appendix A.2 of the extended version. □

Similar formulas can be given for the change-of-basis transformation $\mathscr{U}^{-\vee}$ from the standard basis to the basis of monoid characters. For every cryptanalytic property (a, b), we can express a in the ultrametric integral basis and b in the basis of characters. A concrete example is worked out below.

Example 3. The indicator of the input set for the experimental property on PRESENT from Sect. 3.2 is $\delta_{u \wedge \mathbb{F}_2^n}$, with $u \wedge \mathbb{F}_2^n = \{u \wedge x \mid x \in \mathbb{F}_2^n\}$. In particular, $u = 00\cdots01111$. Using Lemma 1, we can express $\delta_{u\wedge\mathbb{F}_2^n}$ as a linear combination of the ultrametric integral basis vectors:

$$\big[\mathscr{U}\delta_{u\wedge\mathbb{F}_2^n}\big]_v = \sum_{v \preccurlyeq x \preccurlyeq u} 1 = \begin{cases} 2^{\mathsf{wt}(u)-\mathsf{wt}(v)} & \text{if } v \preccurlyeq u\,, \\ 0 & \text{else}\,. \end{cases}$$

Hence,

$$\delta_{u\wedge\mathbb{F}_2^n} = \sum_{v \preccurlyeq u} 2^{\mathsf{wt}(u)-\mathsf{wt}(v)}\, \mu_v\,.$$

Note that $\delta_{u\wedge\mathbb{F}_2^n} \equiv \mu_u \pmod 2$. ▷

The change-of-basis transformation $\mathscr{U}$ is the multiplicative analogue of the Fourier transformation $\mathscr{F}$. However, because $(\mathbb{F}_2^n, \wedge)$ is not a group, there are several important differences. Although the additive characters of $\mathbb{F}_2^n$ are orthogonal, the multiplicative characters μ^v are not. If we identify $\mathbb{R}[\mathbb{F}_2^n]$ and $\mathbb{R}^{\mathbb{F}_2^n}$ using the standard inner product, then orthogonality implies that $\mathscr{F}$ and $\mathscr{F}^{-\vee}$ are the same up to multiplication by 2^n. This fails in the multiplicative case. Since $\mathscr{U}$ and $\mathscr{U}^{-\vee}$ are quite different, identifying $\mathbb{Q}\big[\mathbb{F}_2^n\big]$ and $\mathbb{Q}^{\mathbb{F}_2^n}$ would lead to confusion. Nevertheless, the fact that $\mathscr{F}$ preserves the Euclidean norm does have an analogue in terms of the norm $\|a\|_\infty = \max\big\{|a_x|_2 \mid x \in \mathbb{F}_2^n\big\}$. The proof is given in Appendix A.1 of the extended version.

Theorem 6. *The ultrametric integral change-of-basis transformation $\mathscr{U}$ is an isometry with respect to the 2-adic maximum norm $\|\cdot\|_\infty$. That is, for all a in $\mathbb{Q}[\mathbb{F}_2^n]$, $\|\mathscr{U}a\|_\infty = \|a\|_\infty$.*

The transformations $\mathscr{U}$ and $\mathscr{U}^{-\vee}$ are closely related to integral cryptanalysis. Indeed, the characters are monomials when reduced modulo two: $\mu^v(x) \equiv x^v \pmod 2$. Hence, applying $\mathscr{U}^{-\vee} \equiv \mathscr{M} \pmod 2$ to a Boolean function yields a vector containing the coefficients of its algebraic normal form. Furthermore, for a set X, the support of $\mathscr{U}\delta_X \pmod 2$ is the parity set of X. Indeed,

$$[\mathscr{U}\delta_X]_v \equiv \sum_{\substack{x\in X\\ x\succcurlyeq v}} 1 \equiv \sum_{x\in X} x^v \pmod 2\,.$$

Hence, $\mathscr{U} \equiv \mathscr{P} \pmod 2$ and $\mathscr{U}\delta_X$ generalizes the parity set of X.

Example 4. Let X be the set $\{00, 01, 11\} \subseteq \mathbb{F}_2^2$. The representation of the indicator vector of X in the ultrametric integral basis can be computed as

$$\begin{bmatrix}1&1&1&1\\0&1&0&1\\0&0&1&1\\0&0&0&1\end{bmatrix} \cdot \begin{bmatrix}1\\1\\0\\1\end{bmatrix} = \begin{bmatrix}3\\2\\1\\1\end{bmatrix}.$$

The support of this representation reduced modulo two is the parity set of X, that is $\mathcal{U}(X) = \{00, 10, 11\}$. ▷

4.3 Ultrametric Integral Transition Matrices

The ultrametric integral transition matrix of a function is the matrix representation of the pushforward operator relative to the ultrametric integral basis. Alternatively, one can represent the pullback operator relative to the character basis. This results in the transpose of the ultrametric transition matrix.

Definition 3 (Ultrametric integral transition matrix). *For a function $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$, let $A^{\mathsf{F}} = \mathscr{U}_m T^{\mathsf{F}} \mathscr{U}_n^{-1}$. The ultrametric transition matrix of F is the coordinate representation of A^{F} with respect to the standard bases of $\mathbb{Q}[\mathbb{F}_2^n]$ and $\mathbb{Q}[\mathbb{F}_2^m]$ respectively.*

Like for correlation matrices, we will use the notation A^{F} for both the operator and its standard basis matrix representation. The notation A^{F} collides with the notation for algebraic transition matrices introduced in Sect. 2.3, but the following expression shows that this is reasonable. The coordinates of A^{F} are

$$A^{\mathsf{F}}_{v,u} = \delta^v(A^{\mathsf{F}}\,\delta_u) = \mu^v(T^{\mathsf{F}}\mu_u) = \sum_{x\preccurlyeq u} (-1)^{\mathsf{wt}(x+u)} \tau(\mathsf{F}^v(x))\,. \tag{2}$$

Row v of A^{F} contains the coefficients of the numerical normal form of the Boolean function F^v [13, §2.2.4]. This is the unique multivariate integer polynomial that interpolates F^v on $\{0,1\}^n \subseteq \mathbb{Z}^n$, and reduces to the algebraic normal form modulo two. This implies that the reduction of A^{F} modulo two is the algebraic transition matrix of F. Indeed, [4, Theorem 6] shows that row v of the algebraic transition matrix of F contains the coefficients of the algebraic normal form of F^v. A more elegant proof is given in Appendix B.1 of the extended version.

Two different extensions of the numerical normal form to vectorial Boolean functions have been proposed in the Boolean functions literature. Carlet [13, §2.2.4] considers the numerical normal form of the indicator function of the graph of F. This is not the right notion for cryptanalysis, as it is not based on a pair of dual bases. Still motivated by interpolation, Dravie *et al.* [17] define *polynomial matrices* by a formula similar to (2). Polynomial matrices are equal to ultrametric integral transition matrices, but it is unclear if this was actually intended. Indeed, [17, Proposition 7] relates the polynomial matrix to the adjacency matrix of the graph of F when $n = m$. However, this result is incorrect and correcting it would require changing the definition of polynomial matrices.

The relation between ultrametric integral transition matrices and the numerical normal form could be of independent interest, as the motivation for the ultrametric change-of-basis is quite different (diagonalization of bitwise and). The interpretation in terms of interpolating polynomials over the integers does not play a role in this paper.

Example 5 (Translation). Let $\mathsf{F} : \mathbb{F}_2 \to \mathbb{F}_2$ be defined by $\mathsf{F}(x) = x + k$, for some constant k in $\mathbb{F}_2$. The ultrametric integral transition matrix of F is

$$A^{\mathsf{F}} = \begin{bmatrix} 1 & 0 \\ \tau(k) & (-1)^k \end{bmatrix}$$

If $k = 0$, then this is just the identity matrix. ▷

The following properties are immediate consequences of the properties of transition matrices (Theorem 1, as they are invariant under change of basis.

Corollary 1. *The ultrametric transition matrix A^{F} of $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ has the following properties:*

(1) If $\mathsf{F}(x_1 \| \cdots \| x_l) = \mathsf{F}_1(x_1) \| \cdots \| \mathsf{F}_l(x_l)$, then $A^{\mathsf{F}} = \bigotimes_{i=1}^l A^{\mathsf{F}_i}$.
(2) If $\mathsf{F} = \mathsf{F}_r \circ \cdots \circ \mathsf{F}_2 \circ \mathsf{F}_1$, then $A^{\mathsf{F}} = A^{\mathsf{F}_r} \cdots A^{\mathsf{F}_2} A^{\mathsf{F}_1}$.

Proof. Both of these properties follow from Theorem 1. For the proof of property (1) , we use the fact that $\mathscr{U}_n = \mathscr{U}_1^{\otimes n}$. Indeed,

$$A^{\mathsf{F}} = \left(\bigotimes_{i=1}^l \mathscr{U}\right)\left(\bigotimes_{i=1}^l T^{\mathsf{F}_i}\right)\left(\bigotimes_{i=1}^l \mathscr{U}^{-1}\right) = \bigotimes_{i=1}^l \mathscr{U}\, T^{\mathsf{F}_i}\, \mathscr{U}^{-1} = \bigotimes_{i=1}^l A^{\mathsf{F}_i}\,.$$

For the proof of property (2) , we use Theorem 1: $T^{\mathsf{F}} = T^{\mathsf{F}_r} \cdots T^{\mathsf{F}_2} T^{\mathsf{F}_1}$. Hence,

$$\mathscr{U} T^{\mathsf{F}} \mathscr{U}^{-1} = \left(\mathscr{U} T^{\mathsf{F}_r} \mathscr{U}^{-1}\right) \cdots \left(\mathscr{U} T^{\mathsf{F}_2} \mathscr{U}^{-1}\right)\left(\mathscr{U} T^{\mathsf{F}_1} \mathscr{U}^{-1}\right).$$

The result follows by substituting $A^{\mathsf{F}_i} = \mathscr{U} T^{\mathsf{F}_i} \mathscr{U}^{-1}$. □

Example 6 (Translation). Let $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ with $\mathsf{F}(x) = x + k$, for some constant k in $\mathbb{F}_2^n$. If k_i denotes the ith bit of k, then F can be expressed as

$$\mathsf{F}(x_1 \| \cdots \| x_n) = \mathsf{F}_1(x_1) \| \cdots \| \mathsf{F}_n(x_n)\,,$$

where $\mathsf{F}_i(x_i) = x_i + k_i$ is the function that was discussed in Example 5. Hence, by Corollary 1 (1) ,

$$A^{\mathsf{F}} = \bigotimes_{i=1}^{n} A^{\mathsf{F}_i} = \bigotimes_{i=1}^{n} \begin{bmatrix} 1 & 0 \\ \tau(k_i) & (-1)^{k_i} \end{bmatrix}.$$

More explicitly, A^{F} is a lower-triangular $2^n \times 2^n$ matrix with coordinate in row v and column $u \preccurlyeq v$ equal to $(-1)^{u^{\mathsf{T}}k}\,\tau(k^{u+v})$, and zero elsewhere. ▷

The following properties are specific to ultrametric transition matrices.

Theorem 7. *The ultrametric transition matrix A^{F} of $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ has the following properties:*

(1) If F is a bijection, then A^{F} is an isometry.
(2) If F is a monoid homomorphism, then $A^{\mathsf{F}}_{v,u} = 1$ if $\mu^v \circ \mathsf{F} = \mu^u$ and 0 else.
(3) If $\mathsf{F}(x) = m \wedge x$ with m in $\mathbb{F}_2^n$, then A^{F} is diagonal with $A^{\mathsf{F}}_{u,u} = \mu^u(m)$.

Proof. For the first property, note that if F is a bijection, then T^{F} is an isometry. By Lemma 1, the ultrametric change-of-basis transformation is an isometry. Since a composition of isometries is again an isometry, A^{F} is an isometry.

If F is a monoid homomorphism, then $\mu^v \circ \mathsf{F}$ is a character of $(\mathbb{F}_2^n, \wedge)$. If $\mu^w = \mu^v \circ \mathsf{F}$, then $A^{\mathsf{F}}_{v,u} = \mu^v(T^{\mathsf{F}}\mu_u) = \mu^w(\mu_u)$. The result follows from the duality between μ_u and μ^w.

The third property is true by construction of the ultrametric change-of-basis transformation, Indeed, $A^{\mathsf{F}}_{v,u} = \mu^v(T^{\mathsf{F}}\mu_u) = \mu^u(m)\,\mu^v(\mu_u)$ since μ_u is an eigenvector of T^{F}. The result follows from the duality between μ_u and μ^v. □

Example 7. The function $\mathsf{and} : \mathbb{F}_2^{2n} \to \mathbb{F}_2^n$ defined by $\mathsf{and}(x\|y) = x \wedge y$ is a monoid homomorphism. It follows from Theorem 7 (2) that

$$A^{\mathsf{and}}_{w,u\|v} = \begin{cases} 1 & \text{if } w = u = v\,, \\ 0 & \text{else}\,. \end{cases}$$

The fact that only 2^n coordinates of this matrix are non-zero is no coincidence. The expression above is identical to that for the correlation matrix of the xor function. That is, $A^{\mathsf{and}} = C^{\mathsf{xor}}$. This is by construction, since ultrametric integral cryptanalysis is the multiplicative analogue of linear cryptanalysis. ▷

By the results of [4], algebraic transition matrices lead to a theory of trails for integral cryptanalysis (algebraic, division or monomial trails). In Sect. 5, we show how ultrametric integral transition matrices lead to a similar theory that reduces to ordinary integral cryptanalysis modulo two.

5 Ultrametric Integral Trails

Let $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a function. A pair of exponents $(u, v) \in \mathbb{F}_2^n \times \mathbb{F}_2^m$ will be called an *ultrametric integral approximation* for F. The correlation of an ultrametric integral approximation is defined as

$$A^{\mathsf{F}}_{v,u} = \mu^v \left(T^{\mathsf{F}}\, \mu_u\right).$$

In other words, there is a one-to-one correspondence between ultrametric integral approximations (u, v) and properties (μ_u, μ^v) defined by basis vectors. As shown in Sect. 4.2, the evaluation of every property can in principle be expressed as a linear combination of the evaluations of these properties. Hence, it is sufficient to compute the correlations of ultrametric integral approximations. If F is a composition of functions $\mathsf{F}_1, \ldots, \mathsf{F}_r$ with enough structure so that the coordinates of the matrices $A^{\mathsf{F}_1}, \ldots, A^{\mathsf{F}_r}$ can be determined efficiently, then correlations can be estimated (in the 2-adic sense) using *ultrametric integral trails.*

Definition 4. *An ultrametric integral trail for a function $\mathsf{F} = \mathsf{F}_r \circ \cdots \circ \mathsf{F}_2 \circ \mathsf{F}_1$ is a sequence $u_1, \ldots, u_{r+1}$ of exponents. The correlation of this trail is defined as $\prod_{i=1}^r A^{\mathsf{F}_i}_{u_{i+1},u_i}$.*

5.1 Dominant Trail Approximation

In Corollary 1 (2) , it was shown that $A^{\mathsf{F}} = A^{\mathsf{F}_r} \cdots A^{\mathsf{F}_2} A^{\mathsf{F}_1}$. Writing out this matrix product in terms of coordinates leads to the expression

$$A^{\mathsf{F}}_{u_{r+1},u_1} = \sum_{u_2,\ldots,u_r} \prod_{i=1}^{r} A^{\mathsf{F}_i}_{u_{i+1},u_i}.$$

That is, the correlation of the ultrametric integral approximation (u_1, u_{r+1}) is equal to the sum of the correlations of all trails with input and output exponent u_1 and u_{r+1} respectively. However, this result will not be used in practice because the number of trails is generally too large. Instead, similar to Theorem 2 in the case of linear cryptanalysis, we rely on a set of *dominant trails* to estimate the correlation.

Theorem 8 (Dominant trails, *cf.* Theorem 2). *Let $\mathsf{F} = \mathsf{F}_r \circ \cdots \circ \mathsf{F}_2 \circ \mathsf{F}_1$. For all subsets Λ of the set Ω of all trails from u_1 to u_{r+1},*

$$\left| A^{\mathsf{F}}_{u_{r+1},u_1} - \sum_{u\in\Lambda} \prod_{i=1}^{r} A^{\mathsf{F}_i}_{u_{i+1},u_i} \right|_2 = \left| \sum_{u\in\Omega\setminus\Lambda} \prod_{i=1}^{r} A^{\mathsf{F}_i}_{u_{i+1},u_i} \right|_2 \leq \max_{u\in\Omega\setminus\Lambda} \left| \prod_{i=1}^{r} A^{\mathsf{F}_i}_{u_{i+1},u_i} \right|_2,$$

where $u = (u_1, u_2, \ldots, u_{r+1})$.

Proof. The result follows from the following decomposition:

$$A^{\mathsf{F}}_{u_{r+1},u_1} = \sum_{u_2,\ldots,u_r} \prod_{i=1}^{r} A^{\mathsf{F}_i}_{u_{i+1},u_i} = \sum_{u\in\Lambda} \prod_{i=1}^{r} A^{\mathsf{F}_i}_{u_{i+1},u_i} + \sum_{u\in\Omega\setminus\Lambda} \prod_{i=1}^{r} A^{\mathsf{F}_i}_{u_{i+1},u_i}.$$

In particular, the equality follows by rearranging the terms and taking the absolute value of both sides of the equality. The inequality follows from the ultrametric triangle inequality. □

Theorems 2 and 8 are conceptually the same, but Theorem 8 is based on the 2-adic rather than the ordinary absolute value function. This difference has far-reaching implications. In particular, to upper bound the error term in Theorem 8, it is sufficient to bound the 2-adic absolute value of the correlation of every trails in $\Omega \setminus \Lambda$. This reflects the fact that, in $\mathbb{Q}_2$, the sum of many small numbers is always small. In contrast, Theorem 2 is used heuristically in linear cryptanalysis, because it is difficult to upper bound the error term. Indeed, in $\mathbb{R}$, the sum of many small numbers may be large.

In practice, we will use Theorem 8 as follows. If the 2-adic absolute value of the correlations of all trails in $\Omega \setminus \Lambda$ is at most 2^{-t}, then

$$A^{\mathsf{F}}_{u_{r+1},u_1} \equiv \sum_{u \in \Lambda} \prod_{i=1}^{r} A^{\mathsf{F}_i}_{u_{i+1},u_i} \pmod{2^t}\,.$$

If $\Lambda = \emptyset$, then this shows that $A^{\mathsf{F}}_{u_{r+1},u_1}$ is divisible by 2^t. This corresponds to an *approximate* zero-correlation approximation.

5.2 Example

As a first example of ultrametric integral trails, we explain and prove the property that we observed in Sect. 3.1. Throughout the analysis, we ignore the first S-box layer. Indeed, up to constant additions that can be combined with the key addition of the next round, the input set is invariant under the S-box layer. Hence, let F denote three rounds of PRESENT without the final bit-permutation, as shown in Fig. 2. As explained in Sect. 3.2, the observation corresponds to $\mu^v\big(T^{\mathsf{F}}\,\delta_X\big) \equiv 0 \pmod 4$, where the input set X consists of all values $\mathsf{00}\cdots\mathsf{0}\|x$ with x in $\mathbb{F}_2^4$, and the output exponent v is equal to $\mathsf{0000\ 0000\ 0001\ 0000}$ in hexadecimal notation. Equivalently, $\big|\mu^v\big(T^{\mathsf{F}}\,\delta_X\big)\big|_2 \leq 1/4$.

The vector δ_X is not equal to one of the basis vectors μ_u. Nevertheless, the property can be analyzed using ultrametric integral trails by writing δ_X as a linear combination of the ultrametric integral basis vectors. In particular, it was shown in Example 3 that

$$\delta_X = \sum_{u \in \mathbb{F}_2^4} 2^{4-\mathsf{wt}(u)}\, \mu_{\mathsf{00}\cdots\mathsf{0}\|u}\,.$$

Hence, the evaluation $\mu^v(T^{\mathsf{F}}\delta_X)$ of the property (δ_X, μ^v) is equal to

$$\mu^v\big(T^{\mathsf{F}}\,\delta_X\big) = \sum_{u \in \mathbb{F}_2^4} 2^{4-\mathsf{wt}(u)}\, \mu^v\big(T^{\mathsf{F}}\mu_{\mathsf{00}\cdots\mathsf{0}\|u}\big) = \sum_{u \in \mathbb{F}_2^4} 2^{4-\mathsf{wt}(u)}\, A^{\mathsf{F}}_{v,\mathsf{00}\cdots\mathsf{0}\|u}\,.$$

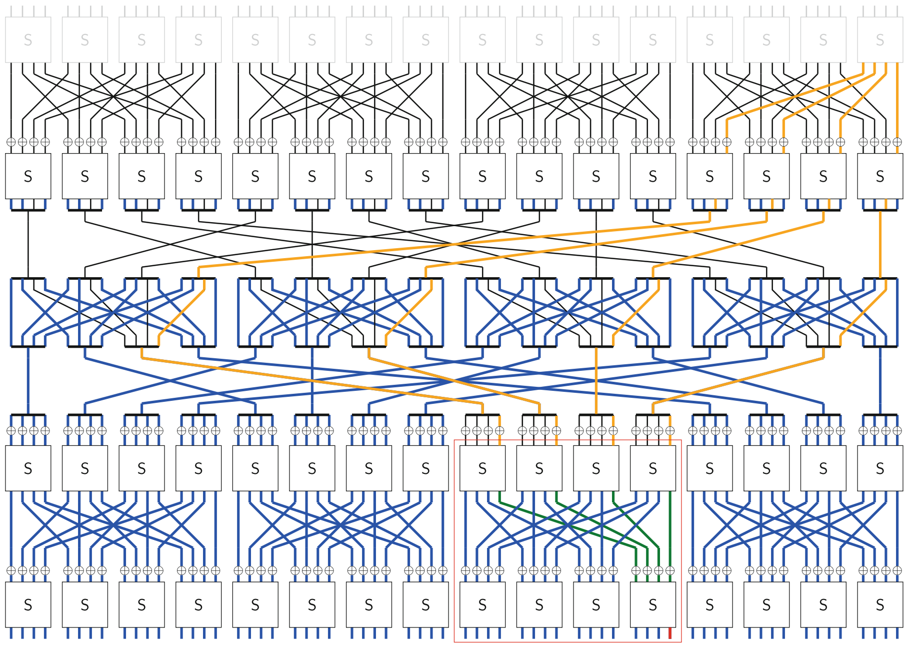


Fig. 2. Miss-in-the-middle using ultrametric integral trails for PRESENT. (Color figure online)

In particular, the 2-adic absolute value is bounded by

$$\left|\mu^v\left(T^{\mathsf{F}}\,\delta_X\right)\right|_2 \leq \max_{u\in\mathbb{F}_2^4} 2^{\mathsf{wt}(u)-4}\left|A^{\mathsf{F}}_{v,00\cdots0\|u}\right|_2 .$$

Hence, it suffices to show that $A^{\mathsf{F}}_{v,0\cdots0\|u}$ is divisible by two if $\mathsf{wt}(u) = 3$ and by four if $\mathsf{wt}(u) = 4$. To prove this, we use Theorem 8 with $\Lambda = \emptyset$.

Figure 2 illustrates the structure of ultrametric integral trails with nonzero correlation. As explained below, the colors correspond to conditions on exponent bits.

The blue lines in Fig. 2 correspond to exponent bits that are equal to zero. The orange lines correspond to a group of four bits that must have weight equal to $\mathsf{wt}(u)$. The analysis is based on a variant of the miss-in-the-middle principle: we propagate the orange set forward and the blue set backwards, in order to rule out trails with high absolute correlation.

The propagation of exponents through a bit-permutation P is straightforward. Since bit-permutations are monoid homomorphisms, Theorem 7 (2) shows that $A^{\mathsf{P}}_{v,u} \neq 0$ if and only if $v = \mathsf{P}(u)$. For $\mathsf{K}(x) = x + k$, it was shown in Example 6 that $A^{\mathsf{K}}_{v,u} \neq 0$ for at least one key k if and only if $u \preccurlyeq v$. For the S-box layer, we need Theorem 7 (2) and the ultrametric integral transition matrix of the PRESENT S-box:

$$A^{\mathsf{S}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{-2} & \mathbf{-1} & \mathbf{2} & \mathbf{1} & \mathbf{-2} & \mathbf{0} & \mathbf{0} & \mathbf{-2} & \mathbf{4} & \mathbf{2} & \mathbf{-4} \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & -1 & -1 & 1 & 0 & 2 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & -1 & -1 & 2 & 0 & 0 \\
1 & 0 & 0 & -1 & -1 & 0 & 0 & 2 & -1 & 1 & 1 & -1 & 2 & -1 & -2 & 0 \\
0 & 1 & 0 & -1 & 0 & -1 & 0 & 2 & 0 & -1 & 1 & 0 & 0 & 2 & -1 & -2 \\
0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & -1 & 0 \\
1 & -1 & -1 & 2 & 0 & 0 & 1 & -1 & -1 & 2 & 2 & -3 & 0 & -1 & -2 & 2 \\
0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & 0 & 0 & 1 & -2 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & -2 & 0 & 1 & 1 & -3 & 0 & -1 & -2 & 4 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & -2 & 0 & 0 & -1 & 2 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & 0 & -1 & 2 & 2 & -3 & 1 & -2 & -2 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -2 & 0 & -1 & -1 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1
\end{bmatrix}.$$

The row that we use in the analysis is indicated in bold.

Backward Propagation. Since $A^{\mathsf{S}}_{\mathtt{0},w} \neq 0$ if and only if $w = \mathtt{0}$, the bits of the exponent before the last S-box layer are zero except for bits 16 up to 19. Nevertheless, bits 16–19 are not arbitrary. In particular, $|A^{\mathsf{S}}_{\mathtt{1},\mathtt{f}}|_2 = 1/4$ and $|A^{\mathsf{S}}_{\mathtt{1},w}|_2 \leq 1/2$ if $\mathsf{wt}(w) = 3$. In Fig. 2, these four bits are indicated in green.

If a bit of the output exponent for the key addition operation is zero, then the corresponding bit of the input exponent must be zero as well. Propagation through the bit-permutation layer is straightforward. Hence, at the input of the third S-box layer, all exponent bits except 16–19 must be zero. Propagating this information through the middle bit-permutation, we find that every nibble of the exponent at the output of the second S-box layer must be $\mathtt{0}$ or $\mathtt{2}$.

Forward Propagation. Like in the backward direction, propagation through the first bit-permutation is straightforward. For the key-addition layer, for every bit of the input exponent equal to one, the corresponding bit of the output exponent is also equal to one. It was shown above that the output exponents on the S-boxes are either $\mathtt{0}$ or $\mathtt{2}$. Hence, since the output exponent must be nonzero if the input exponent is nonzero, the four nonzero output bits of the rightmost superbox must have weight $\mathsf{wt}(u)$. Propagating this information through the bit-permutation is straightforward.

Conclusion. To upper bound the correlation, we focus on the framed superbox in Fig. 2. Since the weight of the set of orange-colored exponent bits is $\mathsf{wt}(u)$, at least three of the first layer of four S-boxes are active. If all four S-boxes are active, then the absolute correlation is at most $1/4$. If only three S-boxes are active, then the absolute correlation is at most $1/2$. That is, the correlation is divisible by two if $\mathsf{wt}(u) = 3$ and divisible by four if $\mathsf{wt}(u) = 4$. This is what we set out to prove.

5.3 Trail Enumeration

A manual analysis of trails like in Sect. 5.2 is instructive, but it becomes tedious for larger problems. Hence, like in linear and ordinary integral cryptanalysis, we will use automated methods to find trails. This will be discussed in more detail in Sects. 7 and 8.

Theorem 8 shows that it is sufficient to upper bound the absolute correlation of every non-dominant trail, but this does not necessarily result in the best possible bound. Indeed, the absolute value of the sum of two correlations can be strictly less than the sum of their absolute values. To take into account these 'cancellations', one can try to enumerate all trails and compute

$$\left| \sum_{u \in \Omega \setminus \Lambda} \prod_{i=1}^{r} A^{\mathsf{F}_i}_{u_{i+1}, u_i} \right|_2 .$$

In practice, this is often infeasible. Nevertheless, in Sect. 7, we will encounter several properties that we can only explain using trail enumeration.

The distinction between bounding correlations of individual trails and trail enumeration also exists in ordinary integral cryptanalysis. This can be made precise using algebraic trails. As explained in Sect. 4.3, the algebraic transition matrix of F is the reduction of A^{F} modulo two. Hence, every ultrametric integral trail reduces to an algebraic trail with correlation in $\mathbb{F}_2$. The method of bounding trail correlations then amounts to showing that all algebraic trails in $\Omega \setminus \Lambda$ have correlation zero. Bit-based division property and parity sets both follow this approach. The three-subset division property without unknown subset and monomial prediction additionally take into account the parity of the number of trails with nonzero correlation. This corresponds to trail enumeration. An overview of different methods can be found in [4, §4.1] and in the survey [20].

6 Properties of Ultrametric Integral Transition Matrices

The purpose of this section is to introduce additional properties of ultrametric integral transition matrices. Some of these properties are mainly of theoretical interest, others will play an important role in Sects. 7 and 8.

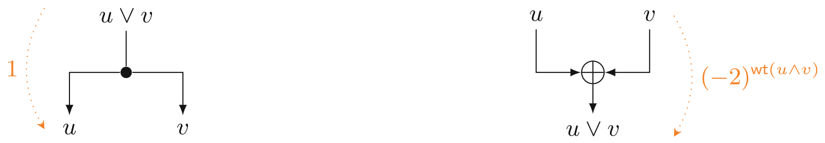
6.1 Computation

The ultrametric integral transition matrix of a function $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ can be computed in $\mathcal{O}((n+m)2^{n+m})$ time. This is the same time-complexity as for computing the correlation matrix of F, and the underlying algorithm is analogous. In particular, it exploits the fact that $\mathscr{U}_n = \mathscr{U}_1^{\otimes n}$ – see (1) on page 404. This tensor product structure leads to an $\mathcal{O}(n2^n)$ time algorithm for computing $\mathscr{U}_n\, a$ and $\mathscr{U}_n^{-1}\, a$, for a in $\mathbb{Q}[\mathbb{F}_2^n]$. Since $A^{\mathsf{F}} = \mathscr{U}_n\, T^{\mathsf{F}} \mathscr{U}_n^{-1}$, applying this algorithm to the rows and columns of T^{F} leads to an $\mathcal{O}((n+m)2^{n+m})$ time algorithm for computing A^{F}. A reference implementation is provided in the example in our code repository.

6.2 Linear Functions

Since the nonlinear functions used in most primitives only depend on a small number of state bits, it is often the linear functions that pose most difficulties in (ultrametric) integral cryptanalysis.

If the linear layer is a bit-permutation, its ultrametric integral transition matrix is easy to compute. Indeed, bit-permutations are monoid isomorphisms, so Theorem 7 (2) can be used. For most other linear functions, there is no simple formula. However, every linear function can be decomposed as a network of forking (or 'copy') and addition (or 'xor') operations. For these two operations, simple exponent propagation rules can be obtained – they are illustrated in Fig. 3 and discussed below.



(a) Copy. (b) Xor.

Fig. 3. Propagation rules for copy and xor operations.

Since copy and xor operations are bitwise operations, their ultrametric integral transition matrix can be computed using Corollary 1 (1) . As the derivation is essentially just a calculation, it is given in Appendix B.2 of the extended version. The copy operation is the function $\mathsf{copy} : \mathbb{F}_2^n \to \mathbb{F}_2^{2n}$ with $\mathsf{copy}(x) = x\|x$. The coordinates of its ultrametric integral transition matrix are

$$A^{\mathsf{copy}}_{u\|v,w} = \begin{cases} 1 & \text{if } w = u \vee v\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

This result implies the following propagation rule: if the output exponent is $u\|v$, then the input exponent must be $u \vee v$. In this case, the correlation is one. This rule is illustrated in Fig. 3a.

The xor operation is the function $\mathsf{xor}_n : \mathbb{F}_2^{2n} \to \mathbb{F}_2$ defined by $\mathsf{xor}(x\|y) = x+y$. The coordinates of its ultrametric integral transition matrix are

$$A^{\mathsf{xor}}_{w,u\|v} = \begin{cases} (-2)^{\mathsf{wt}(u \wedge v)} & \text{if } w = u \vee v\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

This result can be summarized as the propagation rule that an input exponent $u\|v$ goes to output exponent $u \vee v$ with correlation $(-2)^{\mathsf{wt}(u \wedge v)}$. This is illustrated in Fig. 3b.

It is worth mentioning that there are downsides to decomposing linear functions into copy and xor operations. Copy operations often introduce many high-correlation trails, reducing the accuracy of the principle of dominant trails and making trail enumeration more difficult. Hence, whenever dedicated formulas are available, they are usually preferable.

Finally, the propagation rules in Fig. 3 imply an interesting theoretical result: if $\mathsf{L} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a linear function, then $|A^{\mathsf{L}}_{v,u}|_2 \leq 2^{\mathsf{wt}(v)-\mathsf{wt}(u)}$. Every output bit of L can be written as a network of copy and xor operations. For a copy operation, the weight of the output exponent is always greater than the weight of the input exponent. For an xor operation, the output exponent weight *can* be lower than the input exponent weight, but if the weight decreases by Δ then the correlation is $(-2)^{\Delta}$. Hence, the correlation of an approximation (u, v) over L must be divisible by $2^{\mathsf{wt}(u)-\mathsf{wt}(v)}$. In Sect. 6.3, we generalize this result to nonlinear functions.

6.3 Low-Degree Functions

Recall from Example 7 that the propagation rule for the $\mathsf{and} : \mathbb{F}_2^{2n} \to \mathbb{F}_2^n$ function is identical to that of xor in linear cryptanalysis. This extends to the bitwise and of more than two variables, which is still a monoid homomorphism. Based on this property, the following result shows that the ultrametric integral transition matrix of a function with low algebraic degree is sparse.

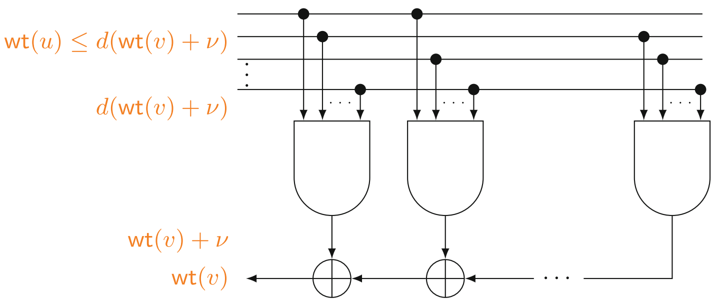
Theorem 9. *If* $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ *is a function with algebraic degree* d*, then*

$$-\log_2 \left|A^{\mathsf{F}}_{v,u}\right|_2 \geq \left\lceil \frac{\mathsf{wt}(u)}{d} \right\rceil - \mathsf{wt}(v)\,.$$

Equivalently, $\left|A^{\mathsf{F}}_{v,u}\right|_2 \geq 2^{-\nu}$ *only if* $\mathsf{wt}(v) \geq \lceil \mathsf{wt}(u)/d \rceil - \nu$*.*

Proof. The result can be proven by a somewhat technical calculation, for example using Equation (2) and splitting up the sum according to the monomials that occur in F^v by using the additive characters of $\mathbb{F}_2$. Instead, we give a more insightful 'cryptanalytic' proof based on ultrametric integral trails.

Every degree d function can be represented as a three-layer circuit, consisting of a layer of copy operations, a layer of and gates with d or fewer inputs each, and a layer of xor operations. This is illustrated in Fig. 4. Although it is not shown in Fig. 4, we allow for an exclusive-or with a constant at the output.



$\mathsf{wt}(u) \leq d(\mathsf{wt}(v) + \nu)$
$d(\mathsf{wt}(v) + \nu)$
$\mathsf{wt}(v) + \nu$
$\mathsf{wt}(v)$

Fig. 4. One of the coordinates of a function of degree d.

Figure 4 only depicts one coordinate of F, but in general we have to take into account the coordinate functions corresponding to all $\mathsf{wt}(v)$ nonzero bits in the output exponent v.

If $|A^{\mathsf{F}}_{v,u}|_2 \geq 2^{-\nu}$, then there must exist an ultrametric integral trail with correlation divisible by a power of two less than or equal to 2^{ν}. By the propagation rules for xor operations from Sect. 6.2, this implies that the weight of the exponent at the output of the and-layer is at most $\mathsf{wt}(v) + \nu$. An and operation with d inputs is a monoid homomorphism from $\mathbb{F}_2^d$ to $\mathbb{F}_2$. By Theorem 7 (2) , the input exponent is $00\cdots 0$ if the output exponent is zero and $11\cdots 1$ if it is one. Hence, using the properties of bricklayer maps, the weight of the exponent at the input of the and-layer is at most $d(\mathsf{wt}(v) + \nu)$. As shown in Sect. 2.2, the weight of the output exponent for a copy is always greater than the weight of its input exponent. Hence,

$$\mathsf{wt}(u) \leq d(\mathsf{wt}(v) + \nu)\,.$$

It follows that $\mathsf{wt}(v) \geq \lceil \mathsf{wt}(u)/d \rceil - \nu$. □

The main propagation rule for the word-based division property [29, Proposition 1] is a special case of Theorem 9. This rule states that if a multiset X has the division property of order k, then $\mathsf{F}(X)$ has the division property of order $\lceil k/d \rceil$. Indeed, by Theorem 9, $|A^{\mathsf{F}}_{v,u}|_2 = 1$ only if $\mathsf{wt}(v) \geq \lceil \mathsf{wt}(u)/d \rceil$. Recall that X has the division property of order k if and only if $[\mathscr{U}\delta_X]_u$ is divisible by two for all u with $\mathsf{wt}(u) < k$.

Theorem 9 is mostly of theoretical interest. To obtain our results in Sects. 7 and 8, more fine-grained models of nonlinear functions are necessary. Nevertheless, Theorem 9 has some interesting theoretical applications. For example, it implies the Ax-Katz theorem over $\mathbb{F}_2$.

Corollary 2 (Ax-Katz [22]). *The number of solutions of a system of m equations of degree d in n variables is divisible by $2^{\lceil n/d \rceil - m}$.*

Proof. The system of equations can be written as $\mathsf{F}(x) = 11\cdots 1$, where $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a function of degree d. That is,

$$\mu^{11\cdots 1}(T^{\mathsf{F}}\delta_{\mathbb{F}_2^n}) = \delta^{11\cdots 1}(A^{\mathsf{F}}\,\mathscr{U}\,\delta_{\mathbb{F}_2^n}) = \sum_{u \in \mathbb{F}_2^n} 2^{n-\mathsf{wt}(u)} A^{\mathsf{F}}_{11\cdots 1,u}\,.$$

By Theorem 9, the right-hand side is divisible by 2^{ν}, where

$$\nu \geq \min_{u \in \mathbb{F}_2^n} n - \mathsf{wt}(u) + \left\lceil \frac{\mathsf{wt}(u)}{d} \right\rceil - \mathsf{wt}(11\cdots 1) \geq \left\lceil \frac{n}{d} \right\rceil - m\,.$$

For the second inequality, we use that the minimum is reached for $\mathsf{wt}(u) = n$. □

There is a variant of Corollary 2 that takes into account the degrees of the individual equations. This result is given in Corollary B.1 of Appendix B of the extended version. The proof uses a variant of Theorem 9.

Finally, it is worth mentioning that Corollary 2 implies a well-known weight divisibility property of Reed-Muller codes. McWilliams and Sloane deduce this result from McEliece's theorem [25, Corollary 13].

Corollary 3. *The weights of codewords in the Reed-Muller code $\mathscr{R}(d,n)$ are divisible by $2^{\lceil n/d\rceil -1}$.*

Proof. The codewords in $\mathscr{R}(d,n)$ are truth-tables of Boolean functions of degree d. Hence, the weight of a codeword is the number of solutions of an equation of degree d in n variables. By Corollary 2, this is divisible by $2^{\lceil n/d\rceil -1}$. □

6.4 Relation with Correlation Matrices

A number of results in the Boolean functions literature relate the algebraic degree of a function to the divisibility of the coordinates of its correlation matrix (equivalently, Walsh-Hadamard transformation). Theorem 10 generalizes these results in terms of the ultrametric integral transition matrix. In doing so, we hope to clarify why such results are to be expected.

The correlation matrix C^{F} and the ultrametric integral transition matrix A^{F} of a function are both matrix representations of the pushforward operator T^{F}. In particular, C^{F} can be expressed in terms of A^{F} (and conversely):

$$C^{\mathsf{F}} = \mathscr{F}\, T^{\mathsf{F}} \mathscr{F}^{-1} = \big(\mathscr{F}\,\mathscr{U}^{-1}\big) A^{\mathsf{F}} \big(\mathscr{F}\,\mathscr{U}^{-1}\big)^{-1}.$$

Since the reduction of A^{F} modulo two is the algebraic transition matrix of F, it is not surprising that the divisibility of coordinates of C^{F} can be related to the algebraic degree. However, in general, looking at the divisibility of the coordinates of A^{F} provides finer results. The following results make this precise.

Lemma 2. *For the matrix $\mathscr{T} = \mathscr{F}\mathscr{U}^{-1}$ and its inverse $\mathscr{T}^{-1}$, we have*

$$\mathscr{T}_{v,u} = \begin{cases} (-2)^{\mathsf{wt}(u)} & \text{if } u \preccurlyeq v\,, \\ 0 & \text{else}\,, \end{cases} \quad \text{and} \quad \mathscr{T}^{-1}_{v,u} = \begin{cases} (-1)^{\mathsf{wt}(u)}\, 2^{-\mathsf{wt}(v)} & \text{if } u \preccurlyeq v\,, \\ 0 & \text{else}\,. \end{cases}$$

Proof. The matrix $\mathscr{T} = \mathscr{F}\mathscr{U}^{-1}$ and its inverse are given by:

$$\mathscr{T} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^{\otimes n} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}^{\otimes n} = \begin{bmatrix} 1 & 0 \\ 1 & -2 \end{bmatrix}^{\otimes n} \text{ and } \mathscr{T}^{-1} = \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}^{\otimes n}.$$

That is, $\mathscr{T}_{v,u}$ is equal to $(-2)^{\mathsf{wt}(u)}$ if $u \preccurlyeq v$ and zero otherwise. For the inverse, note that $\mathscr{T}^{-1}_{v,u}$ is equal to $(-1)^{\mathsf{wt}(u)}\, 2^{-\mathsf{wt}(v)}$ if $u \preccurlyeq v$ and zero otherwise. □

Together with the relation between A^{F} and C^{F}, Lemma 2 implies the following two bounds. As shown below, these bounds refine existing results about the divisibility of correlations. A comparable but different result is given for the numerical normal form of the graph indicator of a function by Carlet [13, §2.3].

Theorem 10. *Let $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a function with correlation matrix C^{F} and ultrametric integral transition matrix A^{F}. For all u in $\mathbb{F}_2^n$ and v in $\mathbb{F}_2^m$,*

$$\big|C^{\mathsf{F}}_{v,u}\big|_2 \le \max_{\substack{s \succcurlyeq u \\ t \preccurlyeq v}} 2^{\mathsf{wt}(s)-\mathsf{wt}(t)} \big|A^{\mathsf{F}}_{t,s}\big|_2 \quad \text{and} \quad \big|A^{\mathsf{F}}_{v,u}\big|_2 \le \max_{\substack{s \succcurlyeq u \\ t \preccurlyeq v}} 2^{\mathsf{wt}(v)-\mathsf{wt}(u)} \big|C^{\mathsf{F}}_{t,s}\big|_2\,.$$

Proof. For brevity, let $\mathscr{T} = \mathscr{F}\mathscr{U}^{-1}$. By the ultrametric triangle inequality,

$$\left|C^{\mathsf{F}}_{v,u}\right|_2 = \left|\left(\mathscr{T}\, A^{\mathsf{F}}\, \mathscr{T}^{-1}\right)_{v,u}\right|_2 \leq \max_{s,t} \left|\mathscr{T}_{v,t}\right|_2 \left|A^{\mathsf{F}}_{t,s}\right|_2 \left|\mathscr{T}^{-1}_{s,u}\right|_2 .$$

The result then follows from Lemma 2. Similarly, we have

$$\left|A^{\mathsf{F}}_{v,u}\right|_2 = \left|\left(\mathscr{T}^{-1}\, C^{\mathsf{F}}\, \mathscr{T}\right)_{v,u}\right|_2 \leq \max_{s,t} \left|\mathscr{T}^{-1}_{v,t}\right|_2 \left|C^{\mathsf{F}}_{t,s}\right|_2 \left|\mathscr{T}_{s,u}\right|_2 .$$

Again, the result follows from Lemma 2. □

Theorem 10 implies the well-known result that $\left|C^{\mathsf{F}}_{v,u}\right|_2 \leq 2^{n-\lceil n/d\rceil}$ if F is of degree d. The details are worked out in Appendix B.4 of the extended version. More interestingly, Theorem 10 also yields the following converse result. A weaker version of Corollary 4 (without the condition $\mathsf{wt}(u) \geq d+1$) was proven by Carlet [12, Lemma 3] and used by Canteaut and Videau [11, Proposition 2] at Eurocrypt 2002 and by Boura and Canteaut in 2013 [9] to upper bound the degree of a composition of two functions[7].

Corollary 4. *If* $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ *is a function with* $\left|C^{\mathsf{F}}_{v,u}\right|_2 \leq 2^{d-1}$ *for all* u *and* v *with* $\mathsf{wt}(u) \geq d+1$ *and* $\mathsf{wt}(v) = 1$*, then* F *has algebraic degree at most* d*.*

Proof. To show that F has degree at most d, it suffices to prove that $|A^{\mathsf{F}}_{v,u}|_2 \leq 1/2$ for all u and v with $\mathsf{wt}(u) \geq d+1$ and $\mathsf{wt}(v) = 1$. This readily follows from the second inequality of Theorem 10:

$$\left|A^{\mathsf{F}}_{v,u}\right|_2 \leq 2^{\mathsf{wt}(v)-\mathsf{wt}(u)}\, 2^{d-1} \leq 2^{d}\, 2^{d-1} = 1/2\,,$$

where we have used $\left|C^{\mathsf{F}}_{t,s}\right|_2 \leq 2^{d-1}$ for all t and s with $\mathsf{wt}(t) \leq \mathsf{wt}(v) = 1$ and $\mathsf{wt}(s) \geq \mathsf{wt}(u) \geq d+1$ □

Another application of Theorem 10 is discussed in Sect. 7.3.

7 Application to PRESENT

In this section, we apply ultrametric integral cryptanalysis to PRESENT. The analysis is automated using off-the-shelf SAT solvers. The choice of PRESENT is didactical. Indeed, integral attacks on PRESENT cover a small number of rounds compared to other methods such as linear cryptanalysis. Nevertheless, PRESENT has often served as a test-case for new ideas in integral cryptanalysis such as the division property [29, §5.3] and parity sets [10, §6].

7.1 Modelling

To automate the analysis of ultrametric trails, we construct a formula in conjunctive normal form so that each satisfying assignment corresponds to a trail with absolute correlation $2^{-\nu}$. The construction of the conjunctive normal form formula follows from the discussion in Sects. 5 and 6. The analysis of trails is based on the dominant trail approximation in Theorem 8, but with nontrivial optimizations to avoid enumerating redundant trails when analyzing properties of the form $(\delta_{u_0 \wedge \mathbb{F}_2^n}, \mu^{u_{r+1}})$ rather than $(\mu_{u_1}, \mu^{u_{r+1}})$. The details are described in the extended version of this work.

[7] It is now understood that these bounds can be proven using integral cryptanalysis.

7.2 Revisiting the Distinguishers of Boura and Canteaut

In this section we revisit the integral distinguishers on PRESENT proposed by Boura and Canteaut [10] at Crypto 2016. They showed that, for the input sets $u \wedge \mathbb{F}_2^{64}$ listed in Table 2 and 4–8 rounds of PRESENT, every bit of the state sums to zero in $\mathbb{F}_2$. For the second output bit of four-round PRESENT, we already observed divisibility by four in Sect. 3.1 and this was proven in Sect. 5.2.

To validate the efficacy of the model, we compare our results for four and five rounds with experimental results, which can be found in Appendix F of the extended version. In most cases, the divisibility predicted by our model without trail enumeration is tight. For a few bits, an additional factor of two or four was gained by enumerating trails. For this reason, trail enumeration was not used to evaluate the 6–9 round properties.

Table 2 lists our results for the first 16 output bits for the input sets chosen by Boura and Canteaut. The results for the remaining 48 bits can be found in Appendix D of the extended version. The first 16 bits give the most interesting results, though the remaining bits are not far off for six rounds and more. We also consider the eight round input set for nine rounds of PRESENT. It was shown by Wang *et al.* [31] that this results in 28 bits that sum to zero in $\mathbb{F}_2$. Our results show that four of these bits exhibit divisibility by four.

Table 2. Divisibility for the distinguishers of Boura and Canteaut [10] and Wang *et al.* [31], with input set $u \wedge \mathbb{F}_2^{64}$. The i^{th} output bit exhibits divisibility by 2^{ν_i}. Bold numbers were obtained using trail enumeration.

rounds	u	$\log_2$(data)	ν_i for bit i															
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
4	`000000000000000f`	4	**3**	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	`000000000000fff0`	12	5	5	5	5	**3**	2	2	2	3	3	3	3	**3**	2	2	2
6	`00000000ffffffff`	32	7	4	4	4	4	1	1	1	4	2	2	2	4	1	1	1
7	`fffffffffffff000`	52	9	5	5	5	4	2	2	2	5	2	2	2	4	2	2	2
8	`fffffffffffffffe`	63	8	5	5	5	5	2	2	2	5	3	3	3	5	2	2	2
9	`fffffffffffffffe`	63	2	1	1	1	2	0	0	0	2	0	0	0	2	0	0	0

7.3 Finding Zero-Correlation Distinguishers

Traditionally, zero-correlation linear approximations are found by showing that all linear trails have correlation zero. Due to the theoretical links from Sect. 6.4, the same properties can be analyzed from the point of view of ultrametric integral cryptanalysis.

To demonstrate that ultrametric integral cryptanalysis provides an alternative way to find zero-correlation linear approximations, we analyze the zero-correlation distinguishers for five and six rounds of PRESENT given by Hadipour *et al.* [18]. These zero-correlation linear approximations depend on the details of the S-box, so propagating the 'all' property as described by Knudsen and Wagner [24] is not sufficient to explain them. However, they *can* be obtained by propagating the 'all' property at the level of individual bits. Hence, these are properties that one would expect integral cryptanalysis to be able to detect, but most of the result is lost when working over $\mathbb{F}_2$.

The extended version of this work goes into more detail on the modelling of these properties. The result is that all $2^{48}-1$ output masks with correlation zero for 5-round PRESENT reported by Hadipour *et al.* can be found using ultrametric integral cryptanalysis. The zero-correlation approximation (u, v) on six rounds of PRESENT in [18, Figure 49d] follows from the five round property. Indeed, the support of the product $A^{\mathsf{F}^\vee} \mathscr{T}^\vee \delta_v$ with F the last round function lies in the set of output exponents that were analyzed for the 5-round property. The same argument can be made using linear cryptanalysis.

7.4 Improving Key Recovery Attacks

Integral distinguishers can be turned into key-recovery attacks using the last-round trick. An important parameter is then the number of incorrect candidate keys that can be filtered out based on a single input set. A single zero-sum bit filters out half of the incorrect candidate keys, but a bit with divisibility by 2^ν provides a filter of (approximately) $2^{-\nu}$. To illustrate this, a key-recovery attack on eight round PRESENT-80 using 2^{12} data and time equivalent to 2^{60} encryptions is worked out below. The time-complexity is not fully optimized.

Integral distinguishers on six round PRESENT require at least 2^8 data, for example when the input set is a coset of $\mathtt{0}\cdots\mathtt{0ff0} \wedge \mathbb{F}_2^{64}$. Since this only gives a 1-bit filter, 20 sets would be necessary to append two rounds. However, using ultrametric integral cryptanalysis, we find that every coset of $\mathtt{0}\cdots\mathtt{0eff0} \wedge \mathbb{F}_2^{64}$ – eight sets of the minimum-data property – leads to divisibility by four on the first bit. By combining both properties, every set of 2^{11} data provides a 9-bit filter. Using two such sets, one can evaluate the cipher on a coset of $\mathtt{0}\cdots\mathtt{0fff0} \wedge \mathbb{F}_2^{64}$. In this case, one has divisibility by 16, 2, 4 and 2 on the first four ciphertext bits. This results in a $2^{-18-(4-2)-1-2-1} = 2^{-24}$ filter. Hence, a single set of 2^{12} data suffices. Without using ultrametric integral cryptanalysis, one would only have a 19-bit filter. Hence, one would need 2^8 additional plaintexts.

The gain is relatively small in this example, but this is in part because only 20 key bits are guessed. If a stronger filter is required, divisibility properties become more useful.

8 Application to SIMON

Section 7 demonstrates that the ultrametric integral cryptanalysis of substitution-permutation networks such as PRESENT can be automated. The purpose of this section is to show that this also applies to ciphers with a different structure.

We use the block cipher family SIMON as an example because, like PRESENT, it has been important in the development of integral cryptanalysis [29,30,32]. As a side result, we propose a small but interesting improvement to the modelling of SIMON's round function. It also applies to ordinary integral cryptanalysis.

8.1 Modelling

Although propagation through the operations that compose SIMON is described in Sects. 6.2 and 6.3, it is worthwhile to take a closer look at the part of the SIMON round function shown in Fig. 5, corresponding to $\mathsf{F}(x) = (x \lll 1) \wedge (x \lll 8)$ with $\lll$ a rotation to the left. In previous work on SIMON [29,30,32], propagation through this function has been modeled by decomposing it into a copy and a bitwise and operation.

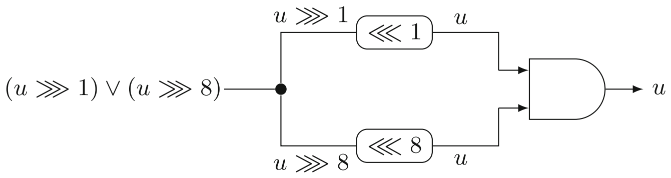


Fig. 5. A part of the round function of SIMON-32.

However, F is actually a monoid homomorphism. Hence, from the point of view of ultrametric integral cryptanalysis, it is no more difficult to handle than linear functions are in linear cryptanalysis. By Theorem 7 (2) , the input exponent is uniquely determined by the output exponent. More precisely, if the output exponent is u, then the input exponent is $\mathsf{F}^*(u) = (u \ggg 1) \vee (u \ggg 8)$. The same is clear from Fig. 5, which depicts the unique trail with output exponent u. This reduces the number of variables in the model. The function F^* is dual to F in the sense that $\wedge$ is replaced by $\vee$. This is analogous to how, in linear cryptanalysis, the output mask for a linear function $x \mapsto \mathsf{M}\,x$ propagates to the input mask by $u \mapsto \mathsf{M}^\mathsf{T} u$.

8.2 Results

The results of our analysis of SIMON are given in Table 3 and Appendix E of the extended version. The input sets are those proposed by Todo [29] and Todo and Morii [30], as well as Xiang *et al.* [32]. We find divisibility by four and higher for many of the output bits, but not for the maximum number of rounds. This is not unexpected, as previous work has focused on distinguishing a maximal number of rounds with minimal data. This is a natural goal from the point of view of ordinary integral cryptanalysis, but it does not necessarily result in the most useful properties in any given situation (such as for a key-recovery attack).

An interesting conclusion from our results is that using the same input set on a smaller number of rounds, does not just yield properties that are universally worse, as can be seen for respectively 10–13 rounds of SIMON-32 and 12–15 rounds of SIMON-48. Indeed, as one would expect, reducing the number of rounds does lead to higher divisibility.

Table 3. Divisibility for SIMON-$\{32,48\}$ distinguishers with input set $R^{-1}(u \wedge \mathbb{F}_2^{\{32,48\}})$, where R is the round function of SIMON-$\{32,48\}$ without key-addition.

SIMON-32				SIMON-48			
rounds	u	$\log_2$(data)	$\max_i \nu_i$	rounds	u	$\log_2$(data)	$\max_i \nu_i$
7	`0001ffff`	17	7	7	`00000001ffff`	17	10
8	`01ffffff`	25	7	8	`00001ffffff`	29	10
9	`1fffffff`	29	5	9	`007ffffffff`	39	8
10	`7fffffff`	31	4	10	`0ffffffffff`	44	6
11	`7fffffff`	31	3	11	`3ffffffffff`	46	5
12	`7fffffff`	31	2	12	`7ffffffffff`	47	4
13	`7fffffff`	31	1	13	`7ffffffffff`	47	3
14	`7fffffff`	31	1	14	`7ffffffffff`	47	2
15	`7fffffff`	31	1	15	`7ffffffffff`	47	1
				16	`7ffffffffff`	47	1

Acknowledgments. Some of the results in this paper were previously published in the PhD thesis of Tim Beyne [2]. This work was partially supported by the Research Council KU Leuven, C16/18/004 through the C1 on New Block Cipher Structures. In addition, this work was supported by CyberSecurity Research Flanders with reference number VR20192203. Tim Beyne is supported by a junior postdoctoral fellowship from the Research Foundation - Flanders (FWO) with reference number 1274724N.

References

1. Tim Beyne. A geometric approach to linear cryptanalysis. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part I*, volume 13090 of *LNCS*, pages 36–66. Springer, Cham, December 2021.
2. Tim Beyne. *A geometric approach to symmetric-key cryptanalysis*. PhD thesis, KU Leuven, June 2023.
3. Tim Beyne and Vincent Rijmen. Differential cryptanalysis in the fixed-key model. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 687–716. Springer, Cham, August 2022.
4. Tim Beyne and Michiel Verbauwhede. Integral cryptanalysis using algebraic transition matrices. *IACR Transactions on Symmetric Cryptology*, 2023(4):244–269, Dec. 2023.

5. Tim Beyne and Michiel Verbauwhede. Ultrametric integral cryptanalysis. Cryptology ePrint Archive, Report 2024/722, 2024.
6. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 450–466. Springer, Berlin, Heidelberg, September 2007.
7. Andrey Bogdanov, Gregor Leander, Kaisa Nyberg, and Meiqin Wang. Integral and multidimensional linear distinguishers with correlation zero. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 244–261. Springer, Berlin, Heidelberg, December 2012.
8. Andrey Bogdanov and Vincent Rijmen. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *DCC*, 70(3):369–383, 2014.
9. Christina Boura and Anne Canteaut. On the influence of the algebraic degree of f^{-1} on the algebraic degree of $g \circ f$. *IEEE Transactions on Information Theory*, 59(1):691–702, 2012.
10. Christina Boura and Anne Canteaut. Another view of the division property. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 654–682. Springer, Berlin, Heidelberg, August 2016.
11. Anne Canteaut and Marion Videau. Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 518–533. Springer, Berlin, Heidelberg, April / May 2002.
12. Claude Carlet. Two new classes of Bent functions. In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 77–101. Springer, Berlin, Heidelberg, May 1994.
13. Claude Carlet. *Boolean functions for cryptography and coding theory.*Cambridge University Press, 2021.
14. Joan Daemen, René Govaerts, and Joos Vandewalle. Correlation matrices. In Bart Preneel, editor, *FSE'94*, volume 1008 of *LNCS*, pages 275–285. Springer, Berlin, Heidelberg, December 1995.
15. Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher Square. In Eli Biham, editor, *FSE'97*, volume 1267 of *LNCS*, pages 149–165. Springer, Berlin, Heidelberg, January 1997.
16. Richard Dedekind. Ideale in Normalkörpern. In Robert Fricke, Emmy Noether, and Øystein Ore, editors, *Gesammelte mathematische Werke*. 1930.
17. Brandon Dravie, Jérémy Parriaux, Philippe Guillot, and Gilles Millérioux.Matrix representations of vectorial boolean functions and eigenanalysis.*Cryptography and Communications*, 8:555–577, 2016.
18. Hosein Hadipour, Simon Gerhalter, Sadegh Sadeghi, and Maria Eichlseder. Improved search for integral, impossible-differential and zero-correlation attacks: Application to Ascon, ForkSKINNY, SKINNY, MANTIS, PRESENT and QARMAv2. Cryptology ePrint Archive, Paper 2023/1701, 2023. https://eprint.iacr.org/2023/1701. https://eprint.iacr.org/2023/1701.
19. Yonglin Hao, Gregor Leander, Willi Meier, Yosuke Todo, and Qingju Wang. Modeling for three-subset division property without unknown subset - improved cube attacks against Trivium and Grain-128AEAD. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 466–495. Springer, Cham, May 2020.
20. Phil Hebborn, Gregor Leander, and Aleksei Udovenko. Mathematical aspects of division property. *Cryptography and Communications*, pages 1–44, 2023.

21. Kai Hu, Siwei Sun, Meiqin Wang, and Qingju Wang. An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 446–476. Springer, Cham, December 2020.
22. Nicholas M. Katz. On a theorem of Ax. *American Journal of Mathematics*, 93(2):485–499, 1971.
23. Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *FSE'94*, volume 1008 of *LNCS*, pages 196–211. Springer, Berlin, Heidelberg, December 1995.
24. Lars R. Knudsen and David Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *FSE 2002*, volume 2365 of *LNCS*, pages 112–127. Springer, Berlin, Heidelberg, February 2002.
25. Florence J. MacWilliams and Neil J. A. Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.
26. Gian Carlo Rota. On the foundations of combinatorial theory I. Theory of Möbius functions. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 2(4):340–368, Jan 1964.
27. Benjamin Steinberg. *Representation theory of finite monoids*. Springer Cham, 2016.
28. Bing Sun, Zhiqiang Liu, Vincent Rijmen, Ruilin Li, Lei Cheng, Qingju Wang, Hoda AlKhzaimi, and Chao Li. Links among impossible differential, integral and zero correlation linear cryptanalysis. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 95–115. Springer, Berlin, Heidelberg, August 2015.
29. Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 287–314. Springer, Berlin, Heidelberg, April 2015.
30. Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 357–377. Springer, Berlin, Heidelberg, March 2016.
31. SenPeng Wang, Bin Hu, Jie Guan, Kai Zhang, and Tairong Shi. MILP-aided method of searching division property using three subsets and applications. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 398–427. Springer, Cham, December 2019.
32. Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers.In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 648–678. Springer, Berlin, Heidelberg, December 2016.

Modelling Ciphers with Overdefined Systems of Quadratic Equations: Application to Friday, Vision, RAIN and Biscuit

Fukang Liu[1](✉), Mohammad Mahzoun[2], and Willi Meier[3]

[1] Institute of Science Tokyo, Tokyo, Japan
liu.f.ad@m.titech.ac.jp

[2] Eindhoven University of Technology, Eindhoven, Netherlands
mail@mahzoun.me

[3] FHNW, Windisch, Switzerland

Abstract. Overdefined polynomial systems have the potential to lead to reduced complexity in solving procedures. In this work, we study how to overdefine the system of equations to describe the arithmetic oriented (AO) ciphers Friday, Vision, and RAIN, as well as a special system of quadratic equations over $\mathbb{F}_{2^\ell}$ used in the post-quantum signature scheme Biscuit. Our method is inspired by Courtois-Pieprzyk's and Murphy-Robshaw's methods to model AES with overdefined systems of quadratic equations over $\mathbb{F}_2$ and $\mathbb{F}_{2^8}$, respectively. However, our method is more refined and much simplified compared with Murphy-Robshaw's method, since it can take full advantage of the low-degree $\mathbb{F}_2$-linearized affine polynomials used in Friday and Vision, and the overdefined system of equations over $\mathbb{F}_{2^\ell}$ can be described in a clean way with our method. For RAIN, we instead consider quadratic Boolean equations rather than equations over large finite fields $\mathbb{F}_{2^\ell}$. Specifically, we demonstrate that the special structure of RAIN allows us to set up much more linearly independent quadratic Boolean equations than those obtained only with Courtois-Pieprzyk's method. Moreover, we further demonstrate that the underlying key-recovery problem in Biscuit (NIST PQC Round 1 Additional Signatures) can also be described by solving a much overdefined system of quadratic equations over $\mathbb{F}_{2^\ell}$. On the downside, the constructed systems of quadratic equations for these ciphers cannot be viewed as semi-regular, which makes it challenging to upper bound the complexity of the Gröbner basis attack. However, such a new modelling method can significantly improve the lower bound of the complexity of the Gröbner basis attacks on these ciphers, i.e., we view the complexity of solving a random system of quadratic equations of the same scale as the lower bound. How to better estimate the upper and lower bounds of the Gröbner basis attacks on these ciphers based on our modelling method is left as an open problem.

Keywords: Friday · Vision · RAIN · Biscuit · overdefined system · algebraic attack · Gröbner basis

1 Introduction

In 2002, Courtois and Pieprzyk presented the first algebraic attack on AES in [22] by modelling it with an overdefined system of quadratic equations over $\mathbb{F}_2$ based on their observation on the inverse function $y = x^{-1}$ over $\mathbb{F}_{2^\ell}$. Subsequently at CRYPTO 2002, Murphy and Robshaw presented a similar method to model AES with an overdefined system of quadratic equations directly over $\mathbb{F}_{2^8}$ [40]. To solve such a special system of equations, Courtois and Pieprzyk proposed the so-called XSL algorithm [22], which is a variant of the XL algorithm [21]. However, it has been pointed out in [20,37] that the assumptions on the XSL algorithm are too optimistic, and that the claimed successful algebraic attacks on full-round AES in [22,40] are flawed. By these results, the community seems to have reached a consensus that AES [23] is secure against algebraic attacks.

After the seminal papers [22,40], however, there seems to be no other progress on such modelling methods. A closely related technique may be Buchmann-Pyshkin-Weinmann's modelling method proposed at FSE 2006, where AES-128 could be modelled with 200 polynomial equations of degree 254 and 152 linear equations [16]. Although this method allows them to obtain the Gröbner basis under a suitable monomial ordering directly, converting the Gröbner basis into a lexicographical order or an elimination order [8] will be too costly, and hence it cannot affect the security of AES-128. As conjectured by Buchmann-Pyshkin-Weinmann, this modelling method can be applied to various iterated block ciphers especially with rich algebraic structures. This has been confirmed by some Gröbner basis attacks [3,7,35] on AO ciphers including MiMC, Griffin, Arion, Anemoi, whose degree of the nonlinear function (a power map) or its inverse is usually low.

In this work, we will instead follow Murphy-Robshaw's main idea since Buchmann-Pyshkin-Weinmann's method is too inefficient for the inverse function, i.e., its inverse is itself and it is still of extremely high degree. Our main goal is to shed new insight into ciphers with rich algebraic structures over $\mathbb{F}_{2^\ell}$. In particular, we aim to take full advantage of the used nonlinear and linear cryptographic components to improve the modelling method.

Motivation of this Work. A number of AO ciphers have been proposed during these years [3–5,14,19,24,32,34], and some of them were also broken due to the insufficient understanding of such designs [2,11,25,38,39,43]. Almost all these successful attacks are algebraic attacks with a clever method to exploit the inner algebraic structures, while such a strategy does not usually work well for conventional block ciphers. The very first algebraic attacks on block ciphers date back to Courtois-Pieprzyk's and Murphy-Robshaw's attacks on AES by exploiting its rich algebraic structure. Unfortunately, the two attacks are flawed due to the incorrect estimation of the time complexity of the XLS algorithm. Since then, no similar attacks have been proposed for symmetric-key primitives.

Since there are some AO ciphers resembling AES, e.g., Friday [5], Vision [4] and RAIN [24], it seems important to revisit the modelling techniques to construct an overdefined system of quadratic equations describing these ciphers, and check whether some unexpected properties have been neglected by the designers. Such a work is meaningful as these AO ciphers are less studied and any potential weakness may lead to a fatal attack in the future. Therefore, we are motivated to dive into the algebraic structures of Friday, Vision and RAIN, and see whether some neglected properties can be identified.

Although Friday has been broken in [2], the same attack cannot apply to its successor Vision. It is thus meaningful to develop a more general algebraic method that can better capture their common underlying algebraic structures, e.g., the low-degree $\mathbb{F}_2$-linearized affine polynomials. Studying Friday is also important to this work since it is the simplest example to explain our new modelling method, though it has been broken.

For the cipher RAIN, it is designed to be friendly to the post-quantum signature scheme Rainier [24] built upon the MPC-in-the-head technique, whose security relies on the difficulty of the key-recovery attack on RAIN from a single plaintext-ciphertext pair. In particular, RAIN has a very small number of rounds, i.e., 3 rounds are sufficient and 4 rounds can be used for higher security. Currently, the best attacks could only reach 2 rounds [39,43], so attacking 3 or 4 rounds is on demand.

In addition to the above 3 symmetric-key primitives, we also find that the candidate Biscuit [10] in NIST PQC Round 1 Additional Signatures may be prone to our attacks, though the inverse function is not used here. Specifically, it is also built with the MPC-in-the-head technique and relies on the difficulty to solve m structured quadratic equations in n variables over $\mathbb{F}_{2^\ell}$, which is called the powAff2 problem. As a candidate in NIST PQC project, studying Biscuit is meaningful.

Our Contributions. We propose a new method to overdefine the polynomial systems describing Friday, Vision, RAIN and Biscuit. Solving such systems will either help find the preimage (e.g., Friday and Vision), or solve the secret key (e.g., RAIN and Biscuit). Specifically, we have the following new results:

1. The preimage attack on r rounds of Friday is reduced to solving $7r$ quadratic equations in $4r$ variables over $\mathbb{F}_{2^\ell}$.
2. The preimage attack on r rounds of Vision with $s \geq 2$ state words is reduced to solving $5s + 14s(r-1)$ quadratic equations in $3s + 6s(r-1)$ variables over $\mathbb{F}_{2^\ell}$.
3. The key-recovery attack on $r \geq 3$ rounds of RAIN is reduced to solving $(5r+5)\ell$ quadratic Boolean equations in $r\ell$ variables. In Gröbner basis attack, the field equation for each variable $x \in \mathbb{F}_2$, i.e., $x^2 = x$, is also useful, and hence we indeed need to consider the problem to solve $(6r+5)\ell$ quadratic equations in $r\ell$ variables with Gröbner basis. Moreover, we further reveal that the problem can also be reduced to solving $(6r+5)\ell$ quadratic equations in $r\ell$ variables over $\mathbb{F}_{2^\ell}$.
4. The powAff2 problem used in Biscuit with m quadratic equations in n variables over $\mathbb{F}_{2^\ell}$ can be overdefined as $4m + n$ quadratic equations in $2n$ variables over $\mathbb{F}_{2^\ell}$.

However, the constructed systems of quadratic equations $\{f_1(x_1, \ldots, x_n) = 0, \ldots, f_m(x_1, \ldots, x_n) = 0\}$ are not semi-regular, since there are many nontrivial syzygies, e.g., there exists a linear polynomial[1] $l_{i,j}(x_1, \ldots, x_n) = \sum_{i=1}^{n} u_i x_i$ such that $l_{i,j} \cdot f_i = f_j$, though both f_i and f_j are quadratic. Note that the trivial syzygies are caused by $f_i \cdot f_j = f_j \cdot f_i$, which can then generate many other trivial syzygies at a higher degree. Hence, the solving degree computed from the Hilbert series based on the assumption that $\{f_1, \ldots, f_m\}$ are semi-regular is just a lower bound on the actual solving degree for solving our constructed system of equations. Intuitively speaking, there are many more rows reduced to zero as the degree of the Macaulay matrix increases (see the definition in Sect. 2) for our equation system, and hence we need to consider a higher degree compared with the case when only trivial syzygies exist.

Indeed, there has been a study [6] on the cost to compute the Gröbner basis for polynomials over $\mathbb{F}_2$ where extra syzygies are taken in account, as detailed in Appendix A. We recommend to read it and believe that the lower bound on the solving degree is still meaningful and the difference between the actual solving degree and the one computed based on the semi-regular assumption is small for polynomial systems we study in this work. Additionally, we have experimentally verified that such lower bounds are indeed tight for small-scale equation systems, yet we cannot claim the same for higher dimensions as it lacks theoretic support. If using such lower bounds to estimate the complexity of the Gröbner basis attack, we obtain the following results:

1. As the first third-party analysis of Vision, we can improve the designers' attack by up to 7 rounds.
2. Using 3 rounds of RAIN with the 256-bit key is insecure, and we thus recommend to use 4 rounds.
3. All parameters of Biscuit are vulnerable to Gröbner basis attacks, and therefore they do not meet the security requirement by NIST.

Organization. We first briefly recall the Gröbner basis in Sect. 2, and then recall how to model AES with an overdefined system of quadratic equations over $\mathbb{F}_2$ and $\mathbb{F}_{2^8}$ in Sect. 3. Next, we present our new algebraic modelling methods for Friday, Vision and RAIN, Biscuit in Sect. 4, Sect. 5, Sect. 6, and Sect. 7, respectively, and give the corresponding analysis of the time complexity as well as the experimental simulation. Finally in Sect. 8, we conclude this paper by summarizing our new insight into these ciphers with our modelling method.

2 Preliminaries

Let $\mathbb{K} = \mathbb{F}_q$ be a finite field and $\mathbb{K}[x_1, \ldots, x_n]$ be a polynomial ring defined over $\mathbb{K}$ with $x_1, \ldots, x_n$ as variables. A multivariate polynomial system is defined as $\mathcal{F} = \{f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)\}$.

[1] This is just an example. A formal definition of syzygy can be referred to Sect. 2.

Let $\mathcal{I} = \langle f_1, \ldots, f_m \rangle$ be the ideal generated by the set of polynomials $\{f_1, \ldots, f_m\}$, and $V(\mathcal{I})$ be its corresponding variety. The polynomial systems describing cryptographic primitives usually generate zero-dimensional ideals. In other words, the set of points in their corresponding variety over the algebraic closure of the field is finite. Finding the corresponding variety of an ideal is an NP-hard problem and is used as a security argument in the design of many primitives. The variety $V(\mathcal{I})$ can be computed with the help of a special type of basis for the ideal $\mathcal{I}$ called Gröbner basis.

Definition 1 (Gröbner Basis [15]). *The set $G = \{g_1, \ldots, g_t\}$ is a Gröbner basis for $\mathcal{I} = \langle f_1, \ldots, f_m \rangle$ if and only if $\langle G \rangle = \mathcal{I}$ and $\langle lm(G) \rangle = \langle lm(\mathcal{I}) \rangle$ where $\langle lm(G) \rangle$ is the ideal generated by the leading monomials of the set G.*

For a monomial ordering $\prec$ and polynomial system $\mathcal{F}$, the Macaulay matrix of $\mathcal{F}$ with degree d is denoted by $\mathcal{M}_{\prec}[d](\mathcal{F})$. Columns of $\mathcal{M}_{\prec}[d](\mathcal{F})$ are labeled by monomials of degree at most d, and sorted in $\prec$-descending order from left to right. Each row of $\mathcal{M}_{\prec}[d](\mathcal{F})$ is labeled by a polynomial $m_j f_i$ where $deg(m_j) \leq d - deg(f_i)$ and m_j is a monomial in $x_1, \ldots, x_n$.

For example, let $\mathcal{F} = \{f_1, f_2\} = \{x^2 + xy, 4x + 3y\}$. Then $\mathcal{M}_{\prec}[2](\mathcal{F})$ for *degrevlex* order is defined as:

$$\mathcal{M}[2](\mathcal{F}) = \begin{array}{c} \\ \\ \\ \\ \end{array}\begin{array}{cccccc|c} x^2 & xy & y^2 & x & y & 1 & \\ 1 & 1 & 0 & 0 & 0 & 0 & f_1 \\ 0 & 0 & 0 & 4 & 3 & 0 & f_2 \\ 4 & 3 & 0 & 0 & 0 & 0 & xf_2 \\ 0 & 4 & 3 & 0 & 0 & 0 & yf_2 \end{array}$$

In [36], it was shown that for a large enough d, the row-echelon form of $\mathcal{M}_{\prec}[d](\mathcal{F})$ gives a Gröbner basis of $\mathcal{F}$. Later, F4 [26] and F5 [27] were published as more efficient algorithms to compute a Gröbner basis, which can efficiently avoid rows reduced to zero when computing $\mathcal{M}_{\prec}[d](\mathcal{F})$. Using F4/F5, the complexity of computing the Gröbner basis in *grevlex* order is as follows:

$$\mathcal{O}\left(\binom{n + D_{reg}}{D_{reg}}^{\omega}\right), \tag{1}$$

where n is the number of variables in the polynomial system, D_{reg} is the degree of regularity of the system, and ω is the linear algebra constant for Gaussian elimination.

When the polynomial system is not defined over $\mathbb{F}_2$, the Gröbner basis in *grevlex* order must be converted to a Gröbner basis in the *lex* order which has triangular form and can be solved efficiently. To convert the Gröbner basis in *grevlex* order to *lex* order, the FGLM [28] algorithm is applied and its complexity is detailed as below.

Proposition 1 (Complexity of the FGLM algorithm [9]). *Given a Gröbner basis $G_1 \subset \mathbb{K}[x_1, \ldots, x_n]$ w.r.t. a monomial ordering $\prec_1$ of a zero-dimensional system, the complexity of computing a Gröbner basis $G_2 \subset \mathbb{K}[x_1, \ldots, x_n]$ w.r.t. a monomial ordering $\prec_2$ with FGLM is*

$$\mathcal{O}(n \cdot D^{\omega}),$$

where D is the degree of the ideal generated by G_1, i.e., the number of solutions counted with multiplicity in the algebraic closure of $\mathbb{K}$.

As commented in [9], the cost of changing monomial ordering is cheaper than computing the Gröbner basis when the system has very few solutions. In our attacks, we also use the same assumption. Indeed, this is also widely used in the literature, e.g., solving the LWE problem with algebraic techniques [1], algebraic attacks on cryptographic schemes like Friday [2], Biscuit [13], UOV [9,29], and the designers' estimation of the resistance against Gröbner basis attacks [4,14, 19,24,32,34], just to name a few. The reason why the Gröbner basis attack on AES-128 [16] does not fall into this category is that the constructed polynomials are too special, i.e., they directly form a special Gröbner basis G_1 where each polynomial in G_1 has a univariate leading monomial of the same degree as of the nonlinear component, which results in an extremely high degree of the ideal generated by this G_1 according to Corollary 1 in [16]. Finally, we also mention that there is an improved variant of the FGLM algorithm called the sparse FGLM algorithm [28] with complexity $\mathcal{O}(D(N_1 + n \log D))$ where N_1 is the number of non-zero elements in a sparse multiplication matrix.

After performing the FGLM algorithm, solving the triangular system to retrieve the solutions to the system is done via factoring polynomials of degree D defined over the finite field $\mathbb{F}_q$ using the Cantor-Zassenhaus's algorithm [17] with complexity [41]:

$$\mathcal{O}(D^2(\log_2 D \log_2 \log_2 D)(\log_2 q + \log_2 D)),$$

which is also viewed as less costly than computing the Gröbner basis in our attacks.

Degree of Regularity. Computing the complexity of a Gröbner basis in Eq. 1 is difficult in general because computing the degree of regularity D_{reg} is challenging. The complexity of computing a Gröbner basis is upper bounded by the *solving degree* of the system, which we denote by D_{sol}. The solving degree of a polynomial system is the smallest integer d, such that a row-reduced echelon form of $\mathcal{M}[d](\mathcal{F})$ results in a Gröbner basis for $\mathcal{F}$. Unfortunately, computing D_{sol} without computing the Gröbner basis itself is a hard task. However, if the polynomial system $\mathcal{F}$ is semi-regular[2], i.e., $m \geq n$, D_{reg} can be upper bounded by the index of the first non-positive coefficient in the Hilbert series $S_{m,n}(z)$ [6,31]:

$$S_{m,n}(z) = \frac{\prod_{i=1}^{m}(1 - z^{\deg(f_i)})}{(1-z)^n}, \tag{2}$$

[2] It is conjectured that this holds for most cases [30].

where $\deg(f_i)$ denotes the degree of the polynomial f_i. However, as far as we know, no polynomial system can be proved to be semi-regular in Gröbner basis attacks on cryptographic primitives. It is mostly conjectured that the system is semi-regular and then experiments are run on small-scale ciphers in order to compare the theoretic solving degree derived from the Hilbert series and the actual solving degree.

Although we could also rely on a similar conjecture, this may not hold for our constructed polynomial system. To understand this, it is necessary to introduce the concept called syzygy. A syzygy on $\mathcal{F}$ is an m-tuple $(g_1, \ldots, g_m) \in \mathbb{K}^m[x_1, \ldots, x_n]$ such that

$$g_1 f_1 + \ldots + g_m f_m = 0.$$

When constructing the Macaulay matrix using the naive method, i.e., multiplying all monomials m_j of degree smaller than $d - deg(f_i)$ with f_i, there will be many syzygies generated, i.e., many rows will be reduced to 0. If the syzygies are mainly caused by the trivial ones[3] $f_i \cdot f_j = f_j \cdot f_i$, we can use the Hilbert series to upper bound the solving degree. However, if there are many non-trivial syzygies on $\mathcal{F}$, for the same degree d, much more rows of $\mathcal{M}_{\prec}[d](\mathcal{F})$ will be reduced to zero, and hence we may need to use a larger d. Hence, we conjecture that for our constructed polynomial system, the solving degree is lower bounded by the index of the first non-positive coefficient in the Hilbert series $S_{m,n}(z)$. The reader can also refer to Appendix A to better understand the above statement.

On the Algebra Constant ω. In the context of Gröbner basis attacks using F4/F5, due to the sparsity of the Macaulay matrix, some practical experiments in the literature suggest that using $\omega = 2$ to estimate the time complexity is realistic, e.g., the Gröbner basis attacks on UOV [9] and Friday [2]. Our experiments for Friday, Vision and RAIN also support $\omega = 2$. Moreover, it is also common for designers to choose $\omega = 2$ to estimate the resistance against the Gröbner basis attack [4,14,19,24,32,34].

3 Overdefined Systems of Quadratic Equations for AES

Denote the polynomial basis of the finite field $\mathbb{F}_{2^\ell}$ by $\{1, t, t^2, \ldots, t^{\ell-1}\}$. Then, each element $z \in \mathbb{F}_{2^\ell}$ can be written as $z = \sum_{i=0}^{\ell-1} \mathbf{z}_i t^i$ where $(\mathbf{z}_0, \ldots, \mathbf{z}_{\ell-1}) \in \mathbb{F}_2^\ell$. In this way, it is sufficient to only use $\overrightarrow{z} = (\mathbf{z}_0, \ldots, \mathbf{z}_{\ell-1}) \in \mathbb{F}_2^\ell$ to represent the element z in the field $\mathbb{F}_{2^\ell}$.

In the polynomial ring $\mathbb{F}_{2^\ell}[x]$, the following polynomial denoted by $B_\ell(x)$ is called an $\mathbb{F}_2$-linearized affine polynomial:

$$B_\ell(x) = \lambda_0 + \sum_{i=0}^{\ell-1} \lambda_{i+1} x^{2^i}. \tag{3}$$

[3] At a higher degree, i.e., as d of $\mathcal{M}_{\prec}[d](\mathcal{F})$ increases, we then have many other trivial syzygies caused by $p \cdot f_i \cdot f_j = p \cdot f_j \cdot f_i$ for $\forall p \in \mathbb{K}[x_1, \ldots, x_n]$.

In particular, it corresponds to an affine transform on $\overrightarrow{x}$. It is clear that this univariate polynomial is uniquely represented by its coefficients $(\lambda_0, \ldots, \lambda_\ell) \in \mathbb{F}_{2^\ell}^{\ell+1}$. Especially, when it is invertible, we can find its inverse denoted by $B_\ell^{-1}(x)$, which is also of the form:

$$B_\ell^{-1}(x) = \lambda'_0 + \sum_{i=0}^{\ell-1} \lambda'_{i+1} x^{2^i}. \tag{4}$$

As the inverse function $y = x^{-1}$ over $\mathbb{F}_{2^\ell}$ cannot take 0 as the input, it is common to use it to construct an S-box denoted by $I_\ell(x)$ in the following way:

$$I_\ell(x) = \begin{cases} x^{-1}, & \text{for } x \neq 0, \\ 0, & \text{for } x = 0. \end{cases} \tag{5}$$

For convenience, we use % to denote the modular operation, and use $[i_0, i_1]$ to denote the set of integers i satisfying $i_0 \leq i \leq i_1$ throughout this paper.

3.1 The AES Round Function

We will not give the full description of the AES algorithm [23] here. Instead, we only focus on its round function, as it is more relevant to the algebraic modelling methods in [22,40].

The AES state is a vector of 16 words in $\mathbb{F}_{2^8}$ and the used irreducible polynomial for $\mathbb{F}_{2^8}[x]$ is $x^8 + x^4 + x^3 + x + 1$. For simplicity, denote the AES state by $(a_0, \ldots, a_{15}) \in \mathbb{F}_{2^8}^{16}$, and denote its binary representation by $(\overrightarrow{a_0}, \ldots, \overrightarrow{a_{15}}) \in \mathbb{F}_2^{128}$. The round function denoted by R_A can be written as

$$R_A = M_A \circ Lin_A \circ I_A(a_0, \ldots, a_{15}),$$

where $Lin_A \circ I_A$ forms the S-box layer of AES, and M_A is the affine transform layer, i.e., the composition of ShiftRows, MixColumns, round constant additions and round key additions. In particular, $Lin_A \circ I_A(a_0, \ldots, a_{15})$ is defined as follows:

$$(I_8(a_0), \ldots, I_8(a_{15})) = I_A(a_0, \ldots, a_{15}),$$
$$(B_8(a_0), \ldots, B_8(a_{15})) = Lin_A(a_0, \ldots, a_{15}),$$

where the coefficients of $B_8(x)$, i.e., $(\lambda_0, \ldots, \lambda_8)$, satisfy $\lambda_i \neq 0$ for $i \in [0, 8]$.

3.2 Courtois-Pieprzyk's Algebraic Modelling Method

We first describe Courtois-Pieprzyk's method to construct an overdefined system of quadratic Boolean equations for AES. According to [22], the following 5 equations over $\mathbb{F}_{2^\ell}$ hold for $y = x^{-1}$:

$$xy = 1,\ x^2y = x,\ xy^2 = y,\ x^4y = x^3,\ xy^4 = y^3.$$

If we consider these 5 equations over $\mathbb{F}_{2^\ell}$, their degrees are obviously 2, 3, 3, 5 and 5, respectively. However, if using the isomorphism between $\mathbb{F}_{2^\ell}$ and $\mathbb{F}_2^\ell$, these 5 equations can be transformed into $5n$ quadratic Boolean equations in $2n$ Boolean variables $(\overrightarrow{x}, \overrightarrow{y})$. The reason is that the map $x \mapsto x^{2^i}$ over $\mathbb{F}_{2^\ell}$ for a positive integer i corresponds to a linear transform in the elements in $\overrightarrow{x}$ over $\mathbb{F}_2$. In particular, it has been proved that these $5n$ quadratic Boolean equations are linearly independent [18].

With this observation in mind, it is then trivial to construct an overdefined system of quadratic Boolean equations to describe AES by introducing intermediate variables for all outputs of $I_8(x)$. Note that except $I_8(x)$, all the remaining operations in AES are linear (or affine).

3.3 Murphy-Robshaw's Algebraic Modelling Method

In Murphy-Robshaw's method, they proposed the so-called Big Encryption System (BES), where the BES state is defined by a vector of $16 \times 8 = 128$ elements over $\mathbb{F}_{2^8}$. Note that the AES state is a vector of 16 elements over $\mathbb{F}_{2^8}$. Denote the BES state by $(a_{0,0}, \ldots, a_{0,7}, a_{1,0}, \ldots, a_{1,7}, \ldots, a_{15,7}) \in \mathbb{F}_{2^8}^{128}$, and there will be additional conditions[4] on such an enlarged state such that it is finally equivalent to AES, as specified below:

$$\forall i \in [0,15],\ j \in [0,7]:\ a_{i,(j+1)\%8} = a_{i,j}^2. \tag{6}$$

Or alternatively, $\forall i \in [0,15]$, there are

$$(a_{i,0}, a_{i,1}, a_{i,2}, a_{i,3}, a_{i,4}, a_{i,5}, a_{i,6}, a_{i,7}) = (a_{i,0}, a_{i,0}^2, a_{i,0}^{2^2}, a_{i,0}^{2^3}, a_{i,0}^{2^4}, a_{i,0}^{2^5}, a_{i,0}^{2^6}, a_{i,0}^{2^7}).$$

The round function of BES denoted by R_B is then defined as

$$R_B = M_B \circ Lin_B \circ I_B(a_{0,0}, \ldots, a_{15,7}),$$

where M_B is an affine transform in the BES state words. For I_B, it is simply defined as:

$$(I_8(a_{0,0}), \ldots, I_8(a_{15,7})) = I_A(a_{0,0}, \ldots, a_{15,7}).$$

As for Lin_B, it is a bit more technical, and it is a block diagonal matrix with 16 identical blocks L_B, i.e., $Lin_B = Diag_{16}(L_B)$, where

$$L_B = \begin{pmatrix}
(\lambda_1)^{2^0} & (\lambda_2)^{2^0} & (\lambda_3)^{2^0} & (\lambda_4)^{2^0} & (\lambda_5)^{2^0} & (\lambda_6)^{2^0} & (\lambda_7)^{2^0} & (\lambda_8)^{2^0} \\
(\lambda_8)^{2^1} & (\lambda_1)^{2^1} & (\lambda_2)^{2^1} & (\lambda_3)^{2^1} & (\lambda_4)^{2^1} & (\lambda_5)^{2^1} & (\lambda_6)^{2^1} & (\lambda_7)^{2^1} \\
(\lambda_7)^{2^2} & (\lambda_8)^{2^2} & (\lambda_1)^{2^2} & (\lambda_2)^{2^2} & (\lambda_3)^{2^2} & (\lambda_4)^{2^2} & (\lambda_5)^{2^2} & (\lambda_6)^{2^2} \\
(\lambda_6)^{2^3} & (\lambda_7)^{2^3} & (\lambda_8)^{2^3} & (\lambda_1)^{2^3} & (\lambda_2)^{2^3} & (\lambda_3)^{2^3} & (\lambda_4)^{2^3} & (\lambda_5)^{2^3} \\
(\lambda_5)^{2^4} & (\lambda_6)^{2^4} & (\lambda_7)^{2^4} & (\lambda_8)^{2^4} & (\lambda_1)^{2^4} & (\lambda_2)^{2^4} & (\lambda_3)^{2^4} & (\lambda_4)^{2^4} \\
(\lambda_4)^{2^5} & (\lambda_5)^{2^5} & (\lambda_6)^{2^5} & (\lambda_7)^{2^5} & (\lambda_8)^{2^5} & (\lambda_1)^{2^5} & (\lambda_2)^{2^5} & (\lambda_3)^{2^5} \\
(\lambda_3)^{2^6} & (\lambda_4)^{2^6} & (\lambda_5)^{2^6} & (\lambda_6)^{2^6} & (\lambda_7)^{2^6} & (\lambda_8)^{2^6} & (\lambda_1)^{2^6} & (\lambda_2)^{2^6} \\
(\lambda_2)^{2^7} & (\lambda_3)^{2^7} & (\lambda_4)^{2^7} & (\lambda_5)^{2^7} & (\lambda_6)^{2^7} & (\lambda_7)^{2^7} & (\lambda_8)^{2^7} & (\lambda_1)^{2^7}
\end{pmatrix} \tag{7}$$

[4] Indeed, the round function of BES is constructed in such a way that these conditions can hold.

In other words, Lin_B is also a linear transform[5] in the BES state words. In this way, $M_B \circ Lin_B$ forms an affine transform in the BES state words, and I_B is the only nonlinear operation in BES. The above can be similarly performed for the key schedule, and we omit the details as these are less relevant.

In a word, the internal states and round keys will become vectors of 128 elements in $\mathbb{F}_{2^8}$. Then, represent the input state of each I_8 by an intermediate variable over $\mathbb{F}_{2^8}$. In this way, each output state of I_8 is also affine in these variables. Moreover, due to the conditions specified in Eq. 6 on the BES state, the input denoted by L_{in} and output denoted by L_{out} of each I_8 can be expressed as linear functions of the following forms:

$$L_{in} = \alpha_{0,0} + \sum_{j=1}\sum_{i=0}^{7} \alpha_{j,i} v_{j,i}, \quad L_{out} = \alpha'_{0,0} + \sum_{j=1}\sum_{i=0}^{7} \alpha'_{j,i} v_{j,i}, \tag{8}$$

where $v_{j,i}$ are those introduced intermediate variables satisfying $v_{j,(i+1)\%8} = v_{j,i}^2$ for $0 \le i \le 7$, and $(\alpha_{j,i}, \alpha'_{j,i})$ are constant coefficients.

For the inverse function $y = x^{-1}$ over $\mathbb{F}_{2^\ell}$, according to Courtois-Pieprzyk's observation, there are

$$xy = 1,\ x^2y = x,\ xy^2 = y,\ x^4y = x^3,\ xy^4 = y^3,$$

which imply the following $5n$ quadratic equations over $\mathbb{F}_{2^\ell}$ if x^{2^i} and y^{2^i} for $0 \le i \le \ell - 1$ are renamed as independent variables x_i and y_i, respectively:

$$\begin{cases} (xy)^{2^i} = 1, \rightarrow x_i y_i = 1 \\ (x^2y)^{2^i} = x^{2^i} \rightarrow x_{(i+1)\%\ell} y_i = x_i, \\ (xy^2)^{2^i} = y^{2^i} \rightarrow x_i y_{(i+1)\%\ell} = y_i, , \\ (x^4y)^{2^i} = (x^3)^{2^i} \rightarrow x_{(i+2)\%\ell} y_i = x_i x_{(i+1)\%\ell}, \\ (xy^4)^{2^i} = (y^3)^{2^i} \rightarrow x_i y_{(i+2)\%\ell} = y_i y_{(i+1)\%\ell}, \end{cases} \quad \text{for } \forall i \in [0, \ell-1]. \tag{9}$$

Compared with the Boolean case, it is much easier to observe that these $5n$ equations are linearly independent as each equation contains one term that never appears in other equations.

According to the expressions of the input and output of each I_8 in BES shown in Eq. 8, and the fact that $(x^{2^{j_0}} + y^{2^{j_1}})^{2^i} = x^{2^{(i+j_0)\%\ell}} + y^{2^{(i+j_1)\%\ell}}$ holds over $\mathbb{F}_{2^\ell}$ for $\forall i, j_0, j_1 \in \mathbb{N}$, we can set up $5n$ quadratic equations for each I_8, resulting in an overdefined system of quadratic equations over $\mathbb{F}_{2^8}$ to describe BES.

Remark 1. The above explanation of BES is rather simplified. In our following new attacks on Friday and Vision, although we exploit a similar method, we never need to construct an equivalent cipher with an enlarged state, nor redefine the

[5] While there is a constant term λ_0 in $B_8(x)$ of AES, the authors do not consider it here when defining Lin_A for BES. This is because it can be moved to the definition of the affine transform M_B.

round functions. Instead, our method is much simpler and easier to understand, i.e., we will show how to construct overdefined systems of quadratic equations over $\mathbb{F}_{2^\ell}$ directly based on the ciphers' original descriptions.

4 New Algebraic Modelling Method for Friday

In this section, we present the first application of our new modelling method inspired by Murphy-Robshaw's method. As can be observed on the application to Friday, our method is much simpler and can take full advantage of the details of $B_\ell(x)$.

4.1 Description of Friday

Friday [5] is a ZK-friendly hash function over $\mathbb{F}_{2^\ell}$ ($\ell \geq 128$) based on the MP (Miyaguchi-Preneel) construction. The round function denoted by R_F of the underlying permutation denoted by $Per(x, key)$ is defined as follows:

$$R_f(x) = k_i + C \circ B^{-1} \circ I_\ell(x),$$

where $k_i \in \mathbb{F}_{2^\ell}$ is the i-th ($i \geq 1$) round key generated with the master key key, and both $B(x)$ and $C(x)$ are $\mathbb{F}_2$-linearized affine polynomials defined as below:

$$B(x) = x^4 + b_2x^2 + b_1x + b_0, \quad C(x) = x^4 + c_2x^2 + c_1x + c_0.$$

By the MP (Miyaguchi-Preneel) construction, the compression function of Friday is

$$h_{i+1} = Per(x, h_i) + x + h_i.$$

It should be mentioned that Friday was soon broken with Gröbner basis at ASIACRYPT 2019 [2], where a smart algebraic modelling method was proposed. Specifically, it is found that for r rounds of Friday where r is an even number, the preimage attack can be reduced to solving $\frac{r}{2}$ equations of degree 36 in $\frac{r}{2}$ variables over $\mathbb{F}_{2^\ell}$.

4.2 New Algebraic Modelling Method for Friday

It is found that the efficiency of the model proposed in [2] comes from a relatively small number of variables, though the degree of equations is high. However, our new modelling method is not related to it. Instead, it is more like Murphy-Robshaw's idea for AES, but it is improved in order to exploit the low degree (or sparsity) of the $\mathbb{F}_2$-linearized affine polynomials $B(x)$ and $C(x)$. Observe that all coefficients in $B_8(x)$ of AES are nonzero, while the $B_\ell(x)$ used in Friday satisfies $\lambda_i = 0$ for $i \in [4, \ell]$.

As shown in Fig. 1, let us consider the preimage attack on r rounds of Friday using a single block, i.e., the goal is to find x satisfying $h_1 = Per(x, h_0) + x + h_0$

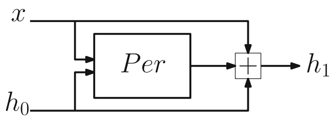


Fig. 1. Preimage attack on Friday using a single block

for a given (h_0, h_1). As shown in Fig. 2, introduce the variable x_i to denote the input of $C(x)$ at the i-th round. Then, there will be

$$\begin{aligned} \forall i \in [1, r-1] : (C(x_i) + k_i) \cdot B(x_{i+1}) &= 1, \\ B(x_1) \cdot (C(x_r) + k_r + h_1 + h_0) &= 1, \end{aligned}$$

where the last equation is to capture the relation between the input and output of the compression function.

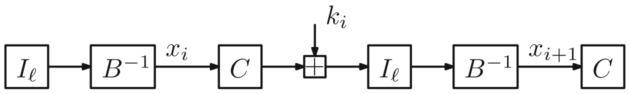


Fig. 2. Modelling the round function of Friday

In more details, these r equations are specified as follows:

$$\begin{aligned} \forall i \in [1, r-1] : (x_i^4 + c_2x_i^2 + c_1x_i + c_0 + k_i)(x_{i+1}^4 + b_2x_{i+1}^2 + b_1x_{i+1} + b_0) &= 1, \\ (x_1^4 + b_2x_1^2 + b_1x_1 + b_0)(x_r^4 + c_2x_r^2 + c_1x_r + c_0 + k_r + h_1 + h_0) &= 1. \end{aligned}$$

As can be observed, each of them is of the following form:

$$(y^4 + c_2y^2 + c_1y + \beta_1)(z^4 + b_2z^2 + b_1z + \beta_2) = 1, \tag{10}$$

where β_1, β_2 are known constants, and y, z are variables over $\mathbb{F}_{2^\ell}$.

Overdefining the Polynomial System. For an equation of the form as in Eq. 10, it is feasible to set up more quadratic equations by introducing additional variables. Specifically, let us introduce variables in the following way:

$$\forall i \in [0, i_\ell] : \; y_i = y^{2^i}, \; z_i = z^{2^i}.$$

By definition, we have the following $2i_\ell$ quadratic equations:

$$\forall i \in [0, i_\ell - 1] : \; y_{(i+1)\%\ell} = y_i^2, \; z_{(i+1)\%\ell} = z_i^2.$$

First, let us consider $i_\ell = 2$. In this case, Eq. 10 becomes

$$(y_2 + c_2y_1 + c_1y_0 + \beta_1)(z_2 + b_2z_1 + b_1z_0 + \beta_2) = 1.$$

Unfortunately, it is still a single equation and more quadratic equations cannot be generated.

However, this is not the case if $i_\ell = 3$. In this case, Eq. 10 is still

$$(y_2 + c_2y_1 + c_1y_0 + \beta_1)(z_2 + b_2z_1 + b_1z_0 + \beta_2) = 1.$$

Meanwhile, due to the extra variables (y_3, z_3), we indeed can set up 3 additional quadratic equations:

$$\begin{aligned}
&(y_2 + c_2y_1 + c_1y_0 + \beta_1)^2(z_2 + b_2z_1 + b_1z_0 + \beta_2)\\
=& (y_3 + c_2^2y_2 + c_1^2y_1 + \beta_1^2)(z_2 + b_2z_1 + b_1z_0 + \beta_2)\\
=& (y_2 + c_2y_1 + c_1y_0 + \beta_1),
\end{aligned}$$

$$\begin{aligned}
&(y_2 + c_2y_1 + c_1y_0 + \beta_1)(z_2 + b_2z_1 + b_1z_0 + \beta_2)^2\\
=& (y_2 + c_2y_1 + c_1y_0 + \beta_1)(z_3 + b_2^2z_2 + b_1^2z_1 + \beta_2^2)\\
=& (z_2 + b_2z_1 + b_1z_0 + \beta_2),
\end{aligned}$$

$$\begin{aligned}
&(y_2 + c_2y_1 + c_1y_0 + \beta_1)^2(z_2 + b_2z_1 + b_1z_0 + \beta_2)^2\\
=& (y_3 + c_2^2y_2 + c_1^2y_1 + \beta_1^2)(z_3 + b_2^2z_2 + b_1^2z_1 + \beta_2^2)\\
=& 1.
\end{aligned}$$

Therefore, for r rounds of Friday, by introducing $4r$ variables $x_{i,j}$ to represent $x_i^{2^j}$ for $i \in [1, r]$ and $j \in [0, 3]$, we can set up $(1 + 3)r = 4r$ quadratic equations in these variables according to the above method, and $3r$ quadratic equations by definition, i.e.,

$$\forall i \in [1, r], j \in [0, 2] : \; x_{i,(j+1)\%\ell} = x_{i,j}^2.$$

In total, r rounds of Friday can be modelling as $4r + 3r = 7r$ quadratic equations in $4r$ variables over $\mathbb{F}_{2^\ell}$.

4.3 Comparison and Experiments

Indeed, analyzing Friday is less interesting as it has been broken in [2], but it is a good starting point to understand our new insight into such designs with low-degree $B_\ell(x)$. Especially, the method in [2] will no longer be feasible for Vision, while our new method can still apply, even though Vision is the successor of Friday and shares a very similar structure.

For completeness, we give a comparison between the estimated time complexity to compute Gröbner basis for the two methods, as shown in Table 1. As already stated, the estimated time complexity for our modelling method is just a lower bound.

Table 1. Comparison between the time complexity of Gröbner basis attacks on Friday with different algebraic modelling methods, where the time complexity (# field operations in logarithm base 2) is estimated under $\omega \in \{2.8, 2\}$ and the complexity with $\omega = 2.8$ is given in parenthesis.

r	6	8	10	12	14	16	18	20
The algebraic modelling method in [2]								
#variables	3	4	5	6	7	8	9	10
#equations	3	4	5	6	7	8	9	10
D_{reg}	94	125	156	187	218	249	280	311
Complexity	34 (48)	47 (65)	59 (83)	72 (101)	85 (118)	97 (136)	110 (154)	123 (172)
Our new algebraic modelling method								
#variables	24	32	40	48	56	64	72	80
#equations	42	56	70	84	98	112	126	140
D_{reg}	6	7	8	9	10	11	12	13
Complexity	39 (54)	48 (67)	57 (80)	67 (93)	76 (106)	85 (119)	94 (131)	103 (144)

Experimental Verification. To verify the complexity of our new Gröbner basis attack, we implemented it on Friday using MAGMA [12] on a Linux cluster. As shown in Table 2, as r increases, the practical solving degree D_{sol} is the same with D_{reg} derived from the Hilbert series, which indicates that the lower bound is indeed tight for small-scale ciphers. Moreover, the practical running time also implies that using $\omega = 2$ to estimate the time complexity is reasonable.

Table 2. Experimental verification of the Gröbner basis attack on Friday, where the complexity (# field operations in logarithm base 2) is calculated with $\omega = 2$.

Rounds (r)	#variables	#equations	D_{sol}	D_{reg}	Time(s)	Complexity
2	8	14	3	4	0.01	15
3	12	21	4	4	0.02	22
4	16	28	5	5	1.88	29
5	20	35	5	6	40.35	32
6	24	42	6	6	3437	39

5 New Algebraic Modelling Method for Vision

After Jarvis and Friday were broken in [2], another two ciphers called Vision and Rescue were proposed for the Marvellous family in [4]. Although Rescue is more popular than Vision, we only focus on Vision in this work as it is very similar to Friday, and our technique can be efficiently applied. As Vision is mainly used for constructing the ZK-friendly hash function, we will no more consider the key-recovery attack as it is less meaningful. Instead, we only consider hash functions built on Vision. We find that the designers suggest to use the sponge construction to build the hash function, and the rate of the sponge construction as well as the length of the hash value are both set as half of the state size. In the following, we will focus on the preimage attack on such hash functions built on Vision.

5.1 Description of the Unkeyed Vision Permutation

It has been explicitly stated in [4] that the master key will be set to zero and the corresponding generated round keys are treated as round constants when a Marvellous design is used as an unkeyed permutation. As our target is the hash function, we omit the description of the keyed Vision permutation, and only focus on the unkeyed permutation.

The round function of the unkeyed permutation Vision follows the common SPN structure. The Vision state is composed of s words $(a_1, \ldots, a_s) \in \mathbb{F}_{2^\ell}^s$ where $s > 1$, which makes it different from Friday as the Friday state is simply one word over $\mathbb{F}_{2^\ell}$. For the round function of Vision at the i-th round, the Vision state will pass through 8 operations, as shown below:

$$\begin{aligned}
a_j &= I_\ell(a_j),\ \forall j \in [1, s]\\
a_j &= B_3^{-1}(a_j),\ \forall j \in [1, s]\\
(a_1, \ldots, a_s)^T &= M \cdot (a_1, \ldots, a_s)^T,\\
a_j &= a_j + \sigma_{i,j},\ \forall j \in [1, s]\\
\\
a_j &= I_\ell(a_j),\ \forall j \in [1, s]\\
a_j &= B_3(w_j),\ \forall j \in [1, s]\\
(a_1, \ldots, a_s)^T &= M \cdot (a_1, \ldots, a_s)^T,\\
a_i &= a_i + \epsilon_{i,j},\ \forall j \in [1, s],
\end{aligned}$$

where $M = (M[i][j])_{1 \le i,j \le s} \in \mathbb{F}_{2^\ell}^{s \times s}$ is an MDS matrix, $\sigma_i = (\sigma_{i,1}, \ldots, \sigma_{i,s}) \in \mathbb{F}_{2^\ell}^s, \epsilon_i = (\epsilon_{i,1}, \ldots, \epsilon_{i,s}) \in \mathbb{F}_{2^\ell}^s$ are round constants, and the definitions of $B_3(x)$ and $I_\ell(x)$ can be referred to Eq. 4 and Eq. 5, respectively. For convenience, we denote the total number of rounds of Vision by r.

As can be observed, the round function is similar to Friday, i.e., they both use the inverse function, a degree-4 $\mathbb{F}_2$-linear affine polynomial and its inverse. However, the state size is increased, and thus a mixing layer M is introduced. Moreover, $I_\ell(x)$ is applied twice in each round. In particular, the degree-4 affine polynomial and its inverse will be applied after the second and first $I_\ell(x)$ layers, respectively. Such changes make the advanced algebraic modelling method proposed in [2] infeasible, as stated in [2]. This also implies that the advanced algebraic modelling method in [2] highly relies on the special structure of Friday, and cannot work well for a more general design.

5.2 Modelling Vision with a Polynomial System

Let us consider r rounds of Vision. As shown in Fig. 3, at the i-th round where $(2 \leq i \leq r)$, we introduce $2s$ variables $(w_{i,1}, \ldots, w_{i,s})$ and $(z_{i,1}, \ldots, z_{i,s})$ to represent the outputs of the $B_3^{-1}(x)$ layer and the second $I_\ell(x)$ layer, respectively.

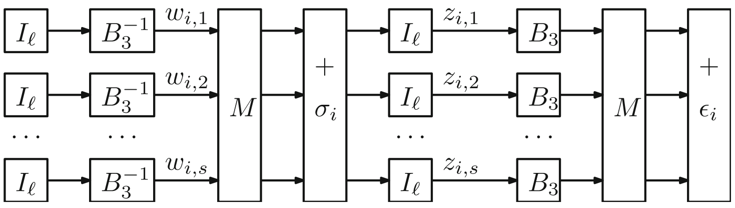


Fig. 3. Modelling the round function of Vision

Equations from the First and Second $I_\ell(x)$ in the Last $r - 1$ Rounds. According to the above way to introduce intermediate variables, we can derive the following s equations according to the first $I_\ell(x)$ layer at the i-th round $(2 \leq i \leq r)$:

$$\forall k \in [1, s]: \; B_3(w_{i,k}) \cdot \left(\sum_{j=1}^{s} M[k][j] \cdot B_3(z_{i-1,j}) + \epsilon_{i-1,k} \right) = 1. \tag{11}$$

For the second $I_\ell(x)$ layer at the i-th round $(2 \leq i \leq r)$, we similarly derive the following s equations:

$$\forall k \in [1, s]: \; z_{i,k} \cdot (\sum_{j=1}^{s} M[k][j] \cdot w_{i,j} + \sigma_{i,k}) = 1. \tag{12}$$

In other words, we can set up $2s$ equations for each round after the 1st round. Of course, it is always assumed that the input to each $I_\ell(x)$ is nonzero.

Dealing with the First Round. Let us consider the case of the sponge construction where the rate part and truncated part are both composed of h state words. The generic preimage attack on such a sponge-based hash function has time complexity $\min(2^{h\ell}, 2^{\frac{(s-h)\ell}{2}})$.

For the preimage attack, we can simply skip the first $I_\ell(x)$ layer and the $B_3^{-1}(x)$ layer, and only introduce h variables $(w_{1,1}, \ldots, w_{1,h})$. For the second $I_\ell(x)$ layer, we still introduce variables $(z_{1,1}, \ldots, z_{1,s})$ to denote the output. In this way, we can set up s equations derived from the second $I_\ell(x)$ layer at the 1st round.

$$\forall k \in [1, s]: \; z_{1,k} \cdot (\sum_{j=1}^{h} M[k][j] \cdot w_{1,j} + \sigma_{1,k} + \beta_{3,k}) = 1, \tag{13}$$

where $(\beta_{3,1}, \ldots, \beta_{3,s})$ are known constants computed from the capacity part of the input.

Equations to Match the Hash Value. In addition to the above equations, we will also have h equations to match the hash value denoted by $(\iota_1, \ldots, \iota_h)$, as shown below:

$$\forall k \in [1, h]: \sum_{j=1}^{s} M[k][j] \cdot B_3(z_{r,j}) + \epsilon_{r,k} = \iota_k. \tag{14}$$

Total Number of Equations and Variables. In total, we have introduced $2s(r-1) + h + s$ variables, and set up $2s(r-1) + s + h$ equations.

5.3 Overdefining the Polynomial System for Vision

According to the above analysis, we mainly have the two forms of equations when describing the last $r-1$ rounds of Vision with a system of equations in intermediate variables $(w_{i,k}, z_{i,k}, z_{i-1,k})$ where $1 \leq k \leq s$, as shown in Eq. 11 and Eq. 12, respectively.

Dealing with Eq. 11. For Eq. 11, it can be written in the following form:

$$B_3(v_1) \cdot \left(\alpha_1 B_3(u_1) + \ldots + \alpha_s B_3(u_s) + \beta_4\right) = 1, \tag{15}$$

where $v_1, u_1, \ldots, u_s$ are variables, and $\beta_4, \alpha_1, \ldots, \alpha_s$ are known constants. Therefore, we can introduce the following intermediate variables:

$$v_{1,i} = v_1^{2^i},\ u_{k,i} = u_k^{2^i},\ \text{for } \forall k \in [1, s], i \in [0, i_\ell].$$

Let $i_\ell = 2$, according to the definition of $B_3(x)$, Eq. 15 can be written as

$$(\lambda_3 v_{1,2} + \lambda_2 v_{1,1} + \lambda_1 v_{1,0} + \lambda_0)(\sum_{j=1}^{s}\sum_{i=0}^{2} \lambda_{j,i} u_{j,i} + \beta_5) = 1,$$

where $\lambda_j, \lambda_{j,i}, \beta_5$ are known constants. Similar to the case in Friday, it is impossible to overdefine this quadratic equation without increasing i_ℓ. However, in the case of Vision, we show that even with $i_\ell = 2$, it is still possible to construct an overdefined polynomial system, which comes from Eq. 12.

Dealing with Eq. 12. Abusing notation, for Eq. 12, it can be written in the following form:

$$u_1 \cdot \left(\alpha_1 v_1 + \ldots + \alpha_s v_s + \beta_6\right) = 1,$$

where $u_1, v_1, \ldots, v_s$ are variables, and $\beta_6, \alpha_1, \ldots, \alpha_s$ are known constants. Hence, we also have

$$\begin{cases} u_1 \cdot (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6) = 1, \\ u_1^2 \cdot (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6)^2 = 1, \\ u_1^4 \cdot (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6)^4 = 1, \\ u_1^2 \cdot (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6) = u_1, \\ u_1^4 \cdot (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6)^2 = u_1^2, \\ u_1 \cdot (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6)^2 = \alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6, \\ u_1^2 \cdot (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6)^4 = (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6)^2, \\ u_1^4 \cdot (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6) = u_1^2 \cdot u_1, \\ u_1 \cdot (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6)^4 = (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6)^{2+1}. \end{cases} \tag{16}$$

Let us introduce the following intermediate variables:

$$u_{1,i} = u_1^{2^i},\ v_{k,i} = v_k^{2^i},\ \text{for } \forall k \in [1,s], i \in [0,2].$$

According to

$$\forall i \in [0,2]:\ u_{1,i} = u_1^{2^i},\ (\alpha_1 v_{1,0} + \ldots + \alpha_s v_{s,0} + \beta_6)^{2^i} = \alpha_1^{2^i} v_{1,i} + \ldots + \alpha_s^{2^i} v_{s,i} + \beta_6^{2^i}.$$

Equation 16 indeed is a system of 9 quadratic equations in the introduced intermediate variables: $(u_{1,0}, u_{1,1}, u_{1,2}, v_{1,0}, v_{1,1}, v_{1,2}, \ldots, v_{s,0}, v_{s,1}, v_{s,2})$.

Putting all Together. According to the above analysis, if we introduce $6s(r-1)$ intermediate variables $(w_{i,k,j}, z_{i,k,j})$ as follows:

$$\forall i \in [2,r], \forall k \in [1,s], \forall j \in [0,2]:\ w_{i,k,j} = w_{i,k}^{2^j},\ z_{i,k,j} = z_{i,k}^{2^j},$$

there will be $(9+1)(r-1)s = 10s(r-1)$ quadratic equations to describe the last $r-1$ rounds, as well as the following $4s(r-1)$ quadratic equations by definition:

$$\forall i \in [2,r], \forall k \in [1,s], \forall j \in [0,1]:\ w_{i,k,j+1} = w_{i,k,j}^2,\ z_{i,k,j+1} = z_{i,k,j}^2.$$

Moreover, for the first round, we only introduce $h + 3s$ variables $(w_{1,1}, \ldots, w_{1,h})$ and $(z_{1,1,0}, \ldots, z_{1,s,2})$ where

$$\forall k \in [1,s], \forall j \in [0,1]:\ z_{1,k,j+1} = z_{1,k,j}^2,\ z_{1,k,0} = z_{1,k},$$

i.e., by definition, there are $2s$ quadratic equations. Then, we can generate $3s$ quadratic equations in these variables from Eq. 13 as follows:

$$\begin{cases} z_{1,k} \cdot (\sum_{j=1}^{h} M[k][j] \cdot w_{1,j} + \sigma_{1,k} + \beta_{3,k}) = 1, \\ z_{1,k}^2 \cdot (\sum_{j=1}^{h} M[k][j] \cdot w_{1,j} + \sigma_{1,k} + \beta_{3,k}) = z_{1,k}, \\ z_{1,k}^4 \cdot (\sum_{j=1}^{h} M[k][j] \cdot w_{1,j} + \sigma_{1,k} + \beta_{3,k}) = z_{1,k}^2 \cdot z_{1,k}. \end{cases} \tag{17}$$

At last, we need to consider h equations specified in Eq. 14 to match the hash value, which now becomes h linear equations in the introduced variables $(z_{r,1,0}, \ldots, z_{r,s,2})$.

Total Number of Quadratic Equations and Variables. According to the above analysis, finding the preimage of r rounds of Vision can be modelled as solving $10s(r-1)+4s(r-1)+2s+3s$ quadratic equations and h linear equations in $6s(r-1)+h+3s$ variables. This is equivalent to solving $5s+14s(r-1)$ quadratic equations in $3s + 6s(r-1)$ variables.

Remark 2. It should be noted that the main reason why we could overdefine the polynomial system for Vision with only $3s + 6s(r-1)$ variables are due to its special structure. More specifically, with these variables, we indeed cannot overdefine the equations describing the first inverse function in the round function, while it becomes feasible for the second inverse function. Hence, this may be an exploitable weakness for attackers to devise advanced attacks on Vision in the future.

5.4 Complexity Analysis and Experiments

Our algebraic modelling method becomes less effective if s is too large as there will be too many variables. Hence, we only focus on the small $s = \{2, 4\}$ and $h = \frac{s}{2}$. For this sponge construction, the generic time complexity to find the preimage is $2^{\frac{s\ell}{2}}$. The time complexity to compute the Gröbner basis for our modelling method is shown in Table 3. Note that the estimated complexity is still a lower bound.

New Insight into the Security Margin of Vision. The designers of Vision choose the secure number of rounds providing l bits of security based on the following formula:

$$\max(10, 2 \times \lceil \frac{l+s+8}{8s} \rceil),$$

which provides 100% security margin. Indeed, the number $\frac{l+s+8}{8s}$ is related to their estimation of the time complexity of the Gröbner basis attack under $\omega = 2$, i.e., the minimal number of rounds[6] that can resist the Gröbner basis attack under $\omega = 2$. Note that in the hash function Rescue-Prime [42], the security margin is reduced to only 50%, though nothing was mentioned for Vision in [42].

If the lower bound is tight, we shed new insight into the security margin of Vision. For $(l, s) = (128, 2)$, we could break up to 10 rounds under the same assumption that $\omega = 2$, while the claimed secure number of rounds is 18. For $(l, s) = (256, 2)$, we could break up to 24 rounds, while the claimed secure number of rounds is 34. In particular, for the instance with $(l, s) = (256, 2)$, if we consider only 50% security margin, the secure number of rounds will be $17+9 = 26$, while we could attack 24 out of 26 rounds. These results have significantly advanced the understanding of the security of the Vision, and have also demonstrated the effectiveness of our new algebraic modelling method.

Table 3. The time complexity (# field operations in logarithm base 2) of Gröbner basis attacks on Vision using $\omega = 2$, where those marked with "−" are less interesting as the corresponding time complexity is too high.

s	N	2	4	6	8	10	12	14	16	18	20	22	24	26
2	#variables	18	42	66	90	114	138	162	186	210	234	258	282	306
	#equations	38	94	150	206	262	318	374	430	486	542	598	654	710
	D_{reg}	5	7	9	11	13	15	17	19	21	23	25	26	28
	Complexity	31	53	74	95	115	136	156	176	197	217	237	250	271
4	#variables	36	84	132	180	228	276	324	−	−	−	−	−	−
	#equations	76	188	300	412	524	636	748	−	−	−	−	−	−
	D_{reg}	7	11	15	19	22	26	30	−	−	−	−	−	−
	Complexity	50	93	134	175	208	249	289	−	−	−	−	−	−

Experimental Verification. Our experimental results verify the correctness of Table 3 for $s = 2$ and $r \in \{2, 3\}$. Note that 1 round of Vision is almost equal to 2 rounds of Friday, and hence we could only practically verify the Gröbner basis attack on up to 3 rounds of Vision due to the limitations on the computational resources. The results of the experiments are summarized in Table 4. These experimental data indicate that the estimated lower bound is tight.

6 Gröbner Basis Attack on 3-Round RAIN

RAIN [24] is a symmetric-key primitive proposed at ACM CCS 2022, and it is tailored to the post-quantum signature scheme called Rainier constructed with the MPC-in-the-head technique [33]. To improve the performance of Rainier, the

[6] Note that there is not a matching attack as the designers also made a conjecture on the solving degree according to their experimental results.

Table 4. Experimental verification of the Gröbner basis attack on Vision, where the complexity (# field operations in logarithm base 2) is calculated with $\omega = 2$.

Rounds (N)	s	#variables	#equations	D_{sol}	D_{reg}	Time(s)	Complexity
2	2	18	38	5	5	1.38	31
3	2	30	66	6	6	100311	42

designers made an aggressive choice of the secure number of rounds for RAIN, i.e., 3 or 4 rounds. Specifically, it is claimed that 3 rounds are sufficient to resist algebraic attacks, and 4 rounds can be used to further increase the security margin.

It should be emphasized that the security of Rainier is based on the difficulty to recover the secret key of RAIN from a single known plaintext-ciphertext pair. In this attack setting, 2 rounds of RAIN have been shown to be insecure in [39,43]. In this section, we present new attacks on 3-round RAIN with our algebraic modelling method.

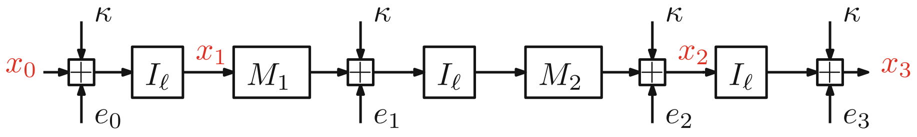


Fig. 4. Illustration of 3-round RAIN

6.1 Description of 3-Round RAIN

Similar to Friday, the RAIN state is simply one word over $\mathbb{F}_{2^\ell}$. As shown in Fig. 4, we denote the input and output of 3-round RAIN by (x_0, x_3). Moreover, we denote the output of the first $I_\ell(x)$ and the input of the last $I_\ell(x)$ by x_1 and x_2, respectively. In addition, the secret key is denoted by $\kappa \in \mathbb{F}_{2^\ell}$, and the round constants are denoted by $e_0, \ldots, e_3 \in \mathbb{F}_{2^\ell}$. For the two linear transformations denoted by $M_1(x)$ and $M_2(x)$, abusing notation, they are defined as follows:

$$M_1(x) = \sum_{j=0}^{\ell-1} \lambda_{1,j} x^{2^j}, \; M_2(x) = \sum_{j=0}^{\ell-1} \lambda_{2,j} x^{2^j},$$

where $\lambda_{1,j} \neq 0, \lambda_{2,j} \neq 0$ are known constants for $0 \leq j \leq \ell - 1$. Hence, both $M_1(x)$ and $M_2(x)$ can also be viewed as multiplying a binary matrix of size $\ell \times \ell$ with $\boldsymbol{x}$.

By these notations, 3-round RAIN can be described with the following 3 equations:

$$\begin{cases} x_1 = I_\ell(x_0 + \kappa + e_0), \\ x_2 = M_2\Big(I_\ell\big((M_1(x_1) + \kappa + e_1)\big)\Big) + \kappa + e_2, \\ x_3 = I_\ell(x_2) + \kappa + e_3. \end{cases}$$

If all inputs to $I_\ell(x)$ are nonzero, these 3 equations can also be rewritten as

$$\begin{cases} x_1 \cdot (x_0 + \kappa + e_0) = 1, \\ M_2^{-1}(x_2 + \kappa + e_2) \cdot (M_1(x_1) + \kappa + e_1) = 1, \\ x_2 \cdot (x_3 + \kappa + e_3) = 1, \end{cases} \tag{18}$$

where $M_2^{-1}(x)$ is the inverse of $M_2(x)$.

6.2 Direct Application of Existing Modelling Methods

A Direct Application of Courtois-Pieprzyk's Modelling Method. Using Courtois-Pieprzyk's observation on $xy = 1$ over $\mathbb{F}_{2^\ell}$, we can directly set up in total $3 \cdot 5\ell = 15\ell$ quadratic Boolean equations in 3ℓ Boolean unknowns $(\overrightarrow{\kappa}, \overrightarrow{x_1}, \overrightarrow{x_2}) \in \mathbb{F}_2^{3\ell}$ according to Eq. 18.

A Direct Application of Murphy-Robshaw Modelling Method. If we want to construct a polynomial system directly over $\mathbb{F}_{2^\ell}$ as in our attacks on Friday and Vision, we can introduce the following 3ℓ intermediate variables:

$$\forall i \in [0, \ell-1]: \; x_{1,i} = x_1^{2^i}, \; x_{2,j} = x_2^{2^i}, \; \kappa_i = \kappa^{2^i},$$

Then, based on Eq. 9, we can set up $3 \cdot 5\ell = 15\ell$ quadratic equations over $\mathbb{F}_{2^\ell}$ according to Eq. 18. In addition, there are 3ℓ quadratic equations over $\mathbb{F}_{2^\ell}$ by definition:

$$\forall i \in [0, \ell-1]: \; x_{1,(i+1)\%\ell} = x_{1,i}^2, \; x_{2,(i+1)\%\ell} = x_{2,i}^2, \; \kappa_{(i+1)\%\ell} = \kappa_i^2.$$

Hence, we have in total 18ℓ quadratic equations in 3ℓ variables over $\mathbb{F}_{2^\ell}$.

Do we Really have Advantages Using Equations over $\mathbb{F}_{2^\ell}$? At the first glance, there seem to be 3ℓ more quadratic equations if we overdefine Eq. 18 directly over $\mathbb{F}_{2^\ell}$. However, in the case of Gröbner basis attack, the field equations are also useful if they are of low degree. For Courtois-Pieprzyk's method, as they are over $\mathbb{F}_2$, there are indeed 3ℓ quadratic field equations neglected, i.e., the field equation for a Boolean variable $x \in \mathbb{F}_2$ is $x^2 = x$. In the case of Murphy-Robshaw's method, although the field equation $x^{2^\ell} = x$ for a variable $x \in \mathbb{F}_{2^\ell}$ is of high degree, it has been implicitly used. Specifically, we have implicitly used

$$\begin{aligned} x_{1,i} &= x_{1,i}^{2^\ell} = (x_{1,i}^{2^{\ell-1}})^2 = x_{1,(\ell-1+i)\%\ell}^2 = x_{1,(i-1)\%\ell}^2, \\ x_{2,i} &= x_{2,i}^{2^\ell} = (x_{2,i}^{2^{\ell-1}})^2 = x_{2,(\ell-1+i)\%\ell}^2 = x_{2,(i-1)\%\ell}^2, \\ \kappa_i &= \kappa_i^{2^\ell} = (\kappa_i^{2^{\ell-1}})^2 = \kappa_{(\ell-1+i)\%\ell}^2 = \kappa_{(i-1)\%\ell}^2. \end{aligned}$$

In other words, if we include the field equations for Courtois-Pieprzyk's method, the two constructed polynomial systems are of the same scale, i.e., the same number of variables and equations. However, one field operation over $\mathbb{F}_{2^\ell}$ is much more expensive than that over $\mathbb{F}_2$ for large ℓ. For this perspective, it seems that using Boolean equations for RAIN is a better choice. In our experiments, we have also confirmed this.

Gröbner Basis Attack on 3-round RAIN. Indeed, the application of Courtois-Pieprzyk's method to RAIN has been observed by the designers of AIM, another symmetric-key primitive tailored for the post-quantum signature scheme AIMer proposed at ACM CCS 2023 [34]. By including all field equations, i.e., in total $15\ell + 3\ell$ quadratic equations in 3ℓ variables, they have given the corresponding time complexity to attack different parameters of 3-round RAIN, as shown in Table 5. Note that the complexity is not necessarily accurate, as the modelling method suffers a similar problem, i.e., the polynomial system cannot be viewed as semi-regular. However, these numbers are still valuable for designers, as they could even claim the security or do comparisons under an optimistic assumption.

Table 5. The time complexity (# field operations in logarithm base 2) of the Gröbner basis attack on 3-round RAIN under $\omega = 2$ given in in [34]

ℓ	#variables	#equations	D_{reg}	Complexity
128	384	2304	14	169
192	576	3456	19	236
256	768	4608	24	304

6.3 Finding More Quadratic Equations Exploiting the Structure

We still follow Courtois-Pieprzyk's method for 3-round RAIN. However, we show that 5ℓ quadratic Boolean equations have been completely neglected if we only focus on equations from $I_\ell(x)$. Indeed, it will be clear that these 5ℓ additional equations are indeed caused by the special structure of 3-round RAIN, i.e., there is no linear transform before the first $I_\ell(x)$ nor after the last $I_\ell(x)$.

Specifically, let us consider the first and last equations in Eq. 18:

$$x_1 \cdot (x_0 + \kappa + e_0) = 1,\ x_2 \cdot (x_3 + \kappa + e_3) = 1.$$

It is easy to deduce the following equation only in the unknowns (x_1, x_2):

$$\frac{1}{x_1} + x_0 + e_0 = \frac{1}{x_2} + x_3 + e_3.$$

Let

$$\theta = x_0 + e_0 + x_3 + e_3,$$

which is a known constant to attackers, and we will have

$$x_1 + x_2 + \theta x_1 x_2 = 0. \tag{19}$$

Similar to Courtois-Pieprzyk's idea, Eq. 19 can be overdefined as follows:

$$\begin{cases} x_1^2 + x_1 x_2 + \theta x_1^2 x_2 = 0, \\ x_1 x_2 + x_2^2 + \theta x_1 x_2^2 = 0. \end{cases} \tag{20}$$

In this way, we can obtain 3ℓ new quadratic Boolean equations only in $(\overrightarrow{x_1}, \overrightarrow{x_2})$, which have been neglected if only focusing on equations from $I_\ell(x)$.

One may observe that we do not multiply the cubic monomial x_1^3 or x_2^3 with $x_1 + x_2 + \theta x_1 x_2$ as in Courtois-Pieprzyk's idea. The reason is simple. If doing so, the term $x_1^3 x_2$ or $x_1 x_2^3$ will appear, and the corresponding equation can only be converted into cubic Boolean equations in $(\overrightarrow{x_1}, \overrightarrow{x_2})$, while we only need quadratic Boolean equations. To obtain more quadratic equations, our crucial observation is that we can multiply a more complex polynomial with both sides of Eq. 19 such that the terms like $x_1^3 x_2$ or $x_1 x_2^3$ can be eliminated. With this idea in mind, we find the following 2 new equations:

$$\begin{cases} (\theta x_1^3 + x_1^2)(x_1 + x_2 + \theta x_1 x_2) = \theta x_1^4 + \theta^2 x_1^4 x_2 + x_1^3 + x_1^2 x_2 = 0, \\ (\theta x_2^3 + x_2^2)(x_1 + x_2 + \theta x_1 x_2) = \theta x_2^4 + \theta^2 x_1 x_2^4 + x_1 x_2^2 + x_2^3 = 0. \end{cases} \tag{21}$$

Hence, the 2 new equations can be converted into 2ℓ new quadratic Boolean equations in $(\overrightarrow{x_1}, \overrightarrow{x_2})$.

In summary, compared with the direct application of Courtois-Pieprzyk's method, we can set up 5ℓ more quadratic Boolean equations. Hence, recovering the secret key κ from (x_0, x_3) is reduced to solving in total $15\ell + 3\ell + 5\ell = 23\ell$ quadratic equations in 3ℓ variables with Gröbner basis.

Remark 3. Note that the 5 new equations in Eq. 19, Eq. 20, Eq. 21 can be easily overdefined to 5ℓ quadratic equations over $\mathbb{F}_{2^\ell}$ if we similarly introduce variables

$$\forall i \in [0, \ell - 1] : \; x_{1,i} = x_1^{2^i}, \; x_{2,j} = x_2^{2^i}, \; \kappa_i = \kappa^{2^i}.$$

Specifically, for each such equation $f(x_1, x_2) = 0$, we can have $f^{2^i}(x_1, x_2) = 0$ for $i \in [0, \ell - 1]$.

A Trivial Extension to 4-round RAIN. The 4 rounds of RAIN simply appends another linear transform $M_3(x)$ and $I_n(x)$, as well as another key addition and round constant addition to 3 rounds of RAIN. Our analysis can thus be trivially applied. Specifically, there are always 5ℓ new quadratic Boolean equations compared with the direct usage of Courtois-Pieprzyk's method. Hence, recovering the secret key is reduced to solving $20\ell + 4\ell + 5\ell = 29\ell$ quadratic equations in 4ℓ variables with Gröbner basis.

6.4 Experiments and Discussions

For 3 rounds of RAIN, the key-recovery attack is equivalent to solving 23ℓ Boolean equations in 3ℓ Boolean variables for $\ell \in \{128, 192, 256\}$. We have verified that these 23ℓ quadratic equations are linearly independent for $\ell \leq 20$, and hence there should be no structural linear dependency. With this polynomial system, our estimation of the time complexity of the Gröbner basis attack on 3 rounds of RAIN is shown in Table 6. Moreover, we have also considered estimating the solving degree with a different Hilbert series dedicated to Boolean polynomials [6], which is detailed in Appendix A, and find that the estimated solving degree remains the same. As already stated, we conjecture that it is a lower bound on the actual complexity. Based on these results, using 3 rounds for 256-bit security may be too aggressive, though the complexity 2^{252} is just a lower bound. However, such a method cannot attack 4-round RAIN according to our calculations.

Table 6. The time complexity of Gröbner basis attacks on 3-round RAIN, where the time complexity (# field operations in logarithm base 2) is estimated under $\omega \in \{2.8, 2\}$ and the complexity with $\omega = 2.8$ is given in parenthesis.

Rounds (N)	ℓ	#variables	#equations	D_{reg}	Complexity
3	128	384	2944	11	139 (195)
	192	576	4416	15	196 (274)
	256	768	5888	19	252 (352)

Experimental Verification. To verify our estimation of the time complexity of the 3-round Gröbner basis attack using 23ℓ quadratic equations, we have performed experiments for $\ell \leq 20$, as shown in Table 7. For comparison, we also performed the experiments for the Gröbner basis attack using only 18ℓ quadratic equations by excluding our newly observed 5ℓ quadratic equations, as shown in Table 7. It is interesting to observe from Table 7 the following facts:

1. The actual solving degree is always the same under the same ℓ in the two experiments, which indicates that the performance of the actual 3-round Gröbner basis attack using 18ℓ and 23ℓ quadratic equations may be the same. A possible explanation is that the additional 5ℓ quadratic equations can be automatically discovered when computing the Gröbner basis of the 18ℓ quadratic polynomials. However, this also indicates that there will be a degree fall when computing the Gröbner basis of the 18ℓ quadratic equations, and the solving degree will become smaller than that derived from the Hilbert series $S_{18\ell,3\ell}(z)$ shown in Eq. 2, which will result in an over-estimation of the complexity of the attack and wrong security claims.

2. As ℓ increases, the solving degree derived from the Hilbert series $S_{23\ell,3\ell}(z)$ becomes tighter. For example, when $\ell = 20$, we have $D_{sol} = D_{deg} = 4$ for 23ℓ equations, while it is $D_{sol} = 4 < D_{reg} = 5$ for 18ℓ equations. In some sense, it supports the above conclusion i.e., the actual solving degree is over-estimated using only 18ℓ quadratic equations.
3. In the cases $\ell \in \{13, 14\}$, it supports our claim that D_{reg} computed from the Hilbert series is a lower bound on the actual solving degree D_{sol}.

Since it is unclear when cases like $\ell \in \{13, 14\}$ will happen again, we cannot conclude that our estimated time complexity of the Gröbner basis attack on 3-round RAIN must be correct by using 23ℓ quadratic equations. However, our experiments also indicate that the time complexity must be underestimated as ℓ increases if only considering 18ℓ quadratic equations. We thus believe that the newly observed 5ℓ quadratic equations do help better estimate the actual solving degree, though finding a theoretic tight upper bound on the actual solving degree looks challenging.

Table 7. Experimental results for the Gröbner basis attack on 3-round RAIN, where we choose $\omega = 2$ to estimate the time complexity (# field operations in logarithm base 2).

(a) (23ℓ equations, 3ℓ variables)

ℓ	D_{reg}	D_{sol}	Time(s)	Complexity
12	3	3	0.99	27
13	3	4	4.72	34
14	3	4	9.9	35
15	4	4	17.09	36
16	4	4	33.26	37
17	4	4	62.59	37
18	4	4	121.57	38
19	4	4	262.76	38
20	4	4	2625	39

(b) (18ℓ equations, 3ℓ variables)

ℓ	D_{reg}	D_{sol}	Time(s)	Complexity
12	4	3	1.06	27
13	4	4	5.88	34
14	4	4	10.8	35
15	4	4	18.75	36
16	4	4	34.56	37
17	4	4	64.02	37
18	4	4	125.44	38
19	4	4	640.27	38
20	5	4	2911	45

7 New Algebraic Modelling Method for Biscuit

Biscuit [10] is a post-quantum signature scheme proposed at ACNS 2024, and it is also one candidate in NIST PQC Round 1 Additional Signatures. Similar to Rainier and AIMer, Biscuit is built with the MPC-in-the-head technique. However, its security relies on the hardness of the so-called powAff2 problem defined below, which is a structured variant of the MQ problem.

Definition 2 (PowAff2 Problem). *Let $d_i, a_{i,j}, b_{i,j}, c_{i,j}$ be known elements over $\mathbb{F}_q$ where $i \in [1, m]$ and $j \in [1, n]$. Given m quadratic equations $\{f_1(x_1, \dots, x_n) = 0, \dots, f_m(x_1, \dots, x_n) = 0\}$ in n variables $(x_1, \dots, x_n) \in \mathbb{F}_q^n$, where each f_i is of the following form:*

$$f_i(x_1, \dots, x_n) = d_i + \sum_{j=1}^{n} a_{i,j} x_j + \sum_{j=1}^{n} b_{i,j} x_j \times \sum_{j=1}^{n} c_{i,j} x_j,$$

find the solution of $(x_1, \dots, x_n)$.

Specifically, if the attacker can solve the powAff2 problem, the secret key of Biscuit will be recovered and Biscuit will become insecure. In particular, $q = 2^8$ is chosen in Biscuit. We should mention that $q = 2^4$ was used in its first version, but it soon got broken by a guess-and-determine (GnD) attack with time complexity $\mathcal{O}(n^3 q^{n/2})$ [13]. In this work, we only consider the later version, i.e., $q = 2^8$.
Parameters for Biscuit. Biscuit can provide 128, 192, and 256 bits of security, respectively. The choices of (n, m) for 128/192/256-bit security can be referred to Table 8. In particular, the designers have checked that these parameters are secure against the Gröbner basis attack under $\omega = 2$ and the GnD attack in [13].

7.1 New Insight Into the PowAff2 Problem over $\mathbb{F}_{2^\ell}$

Let us introduce intermediate variables x_{i+n} to represent x_i^2, i.e., $x_{i+n} = x_i^2$ for $i \in [1, n]$. In this way, we can overdefine each f_i $(1 \le i \le m)$ with 4 quadratic equations in $(x_1, \dots, x_{2n})$, as shown below:

$$\begin{cases} d_i + \sum_{j=1}^{n} a_{i,j} x_j + \sum_{j=1}^{n} b_{i,j} x_j \times \sum_{j=1}^{n} c_{i,j} x_j = 0, \\ (d_i + \sum_{j=1}^{n} a_{i,j} x_j) \times \sum_{j=1}^{n} b_{i,j} x_j + \sum_{j=1}^{n} b_{i,j}^2 x_{j+n} \times \sum_{j=1}^{n} c_{i,j} x_j = 0, \\ (d_i + \sum_{j=1}^{n} a_{i,j} x_j) \times \sum_{j=1}^{n} c_{i,j} x_j + \sum_{j=1}^{n} b_{i,j} x_j \times \sum_{j=1}^{n} c_{i,j}^2 x_{j+n} = 0, \\ d_i^2 + \sum_{j=1}^{n} a_{i,j}^2 x_{j+n} + \sum_{j=1}^{n} b_{i,j}^2 x_{j+n} \times \sum_{j=1}^{n} c_{i,j}^2 x_{j+n} = 0. \end{cases} \tag{22}$$

By the definition that $x_{i+n} = x_i^2$ for $i \in [1, n]$, we have n quadratic equations in $(x_1, \dots, x_{2n})$. Moreover, from each $f_i = 0$, we can derive 4 quadratic equations in $(x_1, \dots, x_{2n})$. As there are m equations $f_i = 0$, we can set up in total $4m + n$ quadratic equations in $2n$ variables. In other words, solving PowAff2 over $\mathbb{F}_{2^\ell}$ is reduced to solving $4m + n$ quadratic equations in $2n$ variables over $\mathbb{F}_{2^\ell}$.

To verify that the polynomials are indeed linearly independent, we performed experimental verification. We sampled 50 different polynomial systems with $(n, m) = (50, 52)$, and formed the system with $2n$ variables and $4m + n$ equations. In all 50 cases, the polynomials are indeed linearly independent. The same also holds for the parameters $(n, m) = (89, 92)$ and $(n, m) = (127, 130)$.

Complexity and Experiments. Similarly, we rely on the Hilbert series to compute the time complexity to solve $4m+n$ quadratic equations in $2n$ variables with Gröbner basis, as shown in Table 8. As already stated, it is conjectured that the estimated time complexity is a lower bound. However, we also emphasize that in our experiments on small (n, m), the solving degree computed from Hilbert series $S_{4m+n,2n}(z)$ is tight. Since 1 field multiplication over $\mathbb{F}_{2^\ell}$ is roughly equal to ℓ^2 bit operations, according to Table 8, we claim that the lower bounds are $2^{104}/2^{159}/2^{221}$ bits operations for 128/192/256-bit security levels, respectively.

Table 8. The time complexity (# field operations in logarithm base 2) of Gröbner basis attacks on Biscuit, which is estimated under $\omega = 2$.

Security	(n, m)	#variables	#equations	D_{sol}	Complexity
128	$(50, 52)$	100	258	11	98
192	$(89, 92)$	178	457	16	153
256	$(127, 130)$	254	647	22	215

In Table 9, the experimental result for computing the Gröbner basis for toy parameters is depicted. These experimental data indicate that $D_{sol} \leq D_{reg}$ for all cases, and hence our time complexity evaluation is tight for the small-scale powAff2 problem. Moreover, the running time of our experiments also indicates that using $\omega = 2$ is reasonable.

Table 9. Experimental results for Biscuit over $\mathbb{F}_{2^8}$, where the time complexity (# field operations in logarithm base 2) is computed under $\omega = 2$. The complexity of our approach is compared with the complexity of solving the polynomial system with n variables and m equations described in [10]

n	m	#variables	#equations	D_{sol}	D_{reg}	Complexity	Complexity [10]	Time(s)
11	13	22	63	4	4	28	27	2.03
12	14	24	68	4	4	29	31	19.4
13	15	26	73	5	5	35	34	471.8
14	16	28	78	5	5	36	34	3034.2
15	17	30	83	5	5	37	38	12530
16	18	32	88	5	5	37	39	48562

8 Conclusion

This study aims to deepen the understanding of how to overdefine a polynomial system with Courtois-Pieprzyk's and Murphy-Robshaw's ideas. In particular, it is found that the polynomial systems for Friday, Vision, RAIN and Biscuit can all be overdefined for different reasons. For Friday and Vision, it is mainly due to the low-degree $\mathbb{F}_2$-linearized affine polynomial. For RAIN, the system can be much more overdefined for its special structure, i.e., no linear mixing layers exist before the first nonlinear layer and after the last nonlinear layer defined by the inverse function over $\mathbb{F}_{2^\ell}$. For Biscuit, by exploiting the special structure of the powAff2 problem over $\mathbb{F}_{2^\ell}$, i.e., all quadratic terms in each quadratic equation are produced by the multiplication of two linear polynomials, an overdefined polynomial system can be efficiently set up by introducing additional variables. However, how to estimate the time complexity to solve these polynomial systems is challenging, since they are not semi-regular. We leave this as an open problem, and believe that it will have many applications in algebraic attacks.

Acknowledgment. We thank the anonymous reviewers for their insightful comments. We also thank Morten Øygarden for discussing some ideas at the early stage of this work. Fukang Liu is supported by JSPS KAKENHI Grant Numbers JP22K21282, JP24K20733. He is also funded by the commissioned research (No. JPJ012368C05801) by National Institute of Information and Communications Technology (NICT).

A Computing Gröbner Basis for Polynomials over $\mathbb{F}_2$

To compute the Gröbner basis for polynomials

$$\{f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)\}$$

over $\mathbb{F}_2$, the field equations, $x_i^2 = x_i$ for $\forall i \in [1, n]$, should also be considered. In our paper, for m quadratic Boolean equations in n Boolean variables, by including the n field equations, we indeed consider $m + n$ polynomials $\{f_1, \ldots, f_m, f_{m+1}, \ldots, f_{m+n}\}$ where $f_{m+i} = x_i^2 - x_i$ for $i \in [1, n]$, and estimate D_{sol} as the first non-positive coefficient in the following Hilbert series: $\frac{(1-z^2)^{m+n}}{(1-z)^n}$. As explained before, this holds only when trivial syzygies are caused by $f_i \cdot f_j = f_j \cdot f_i$ where $i, j \in [1, m + n]$. However, for polynomials over $\mathbb{F}_2$, including the field equations implies new syzygies, i.e., there will be new syzygies caused by $f_i^2 = f_i$ for $i \in [1, m]$. Hence, the assumption that only trivial syzygies exist for this polynomial system does not hold in practice. If taking such new syzygies into account, it has been studied in [6] that the solving degree to compute the Gröbner basis of $\{f_1, \ldots, f_m\}$ over $\mathbb{F}_2$ can be estimated as the first non-positive coefficient in the following new Hilbert series: $\frac{(1+z)^n}{(1+z^2)^m}$.

For convenience, we call the first method to compute D_{sol} Method 1 where only trivial syzygies $f_i \cdot f_j = f_j \cdot f_i$ are considered and call the second method Method 2 where new syzygies formed by field equations are also considered. We tested the degree of regularity using both Method 1 and Method 2 for all

$n < m < 500$ and observed that 1) the two methods give the same D_{sol} when the system does not involve too many variables; 2) the difference between D_{sol} obtained with the two methods tends to be the same if the system is much overdefined; 3) the gap is still small, i.e., 1 or 2, when the two methods give different D_{sol}. These may be evidence that the gap will be smaller as the system becomes much more overdefined. The difference of the degree of regularity for the two different methods is depicted in Fig. 5.

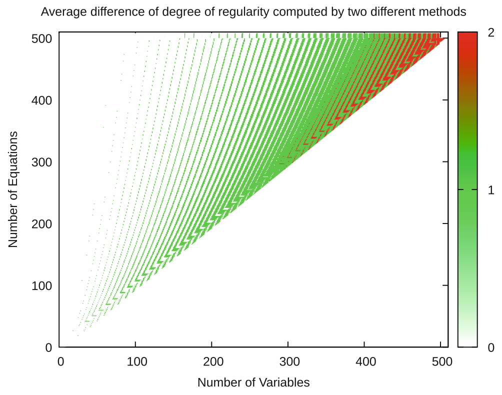


Fig. 5. The difference of D_{sol} using Method 1 and Method 2. The green points show a difference of 1, while the red points show a difference of 2. The white area represents the points with the same solving degree using both methods. (Color figure online)

The motivation to explain the above fact is to analyze the impact of newly defined syzygy relations on the solving degree of the polynomial system, as in our constructed overdefined polynomial systems (not necessarily over $\mathbb{F}_2$), there are also many new syzygies. A natural question is when the estimated solving degree D_{sol} based on the assumption that the system is semi-regular will be much larger than the actual solving degree. This is difficult to verify in practice as computing the actual solving degree is equivalent to computing the Gröbner basis. By the above fact, we may see the evidence that even if there are some new syzygies, estimating the solving degree based on the assumption that the system is semi-regular is still reliable because the difference in the estimated solving degree using the semi-regularity assumption, and the actual solving degree of the system is small, or even zero, for the polynomial systems that we study.

References

1. M. R. Albrecht, C. Cid, J. Faugère, R. Fitzpatrick, and L. Perret. Algebraic algorithms for LWE problems. *ACM Commun. Comput. Algebra*, 49(2):62, 2015.
2. M. R. Albrecht, C. Cid, L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, and M. Schofnegger. Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC. In *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 371–397. Springer, 2019.
3. M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In *ASIACRYPT (1)*, volume 10031 of *Lecture Notes in Computer Science*, pages 191–219, 2016.
4. A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec. Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. *IACR Trans. Symmetric Cryptol.*, 2020(3):1–45, 2020.
5. T. Ashur and S. Dhooghe. MARVELlous: a STARK-Friendly Family of Cryptographic Primitives. Cryptology ePrint Archive, Paper 2018/1098, 2018. https://eprint.iacr.org/2018/1098.
6. M. Bardet, J.-C. Faugère, and B. Salvy. Asymptotic Behaviour of the Index of Regularity of Semi-Regular Quadratic Polynomial Systems. In *MEGA 2005 - 8th International Symposium on Effective Methods in Algebraic Geometry*, pages 1–17, Porto Conte, Alghero, Sardinia, Italy, May 2005.
7. A. Bariant, A. Boeuf, A. Lemoine, I. M. Ayala, M. Øygarden, L. Perrin, and H. Raddum. The Algebraic Freelunch Efficient Gröbner Basis Attacks Against Arithmetization-Oriented Primitives. Cryptology ePrint Archive, Paper 2024/347, 2024. https://eprint.iacr.org/2024/347.
8. D. Bayer and M. E. Stillman. On the Complexity of Computing Syzygies. *J. Symb. Comput.*, 6:135–147, 1988.
9. L. Bettale, J. Faugère, and L. Perret. Hybrid Approach for Solving Multivariate Systems over Finite Fields. *J. Math. Cryptol.*, 3(3):177–197, 2009.
10. L. Bettale, D. Kahrobaei, L. Perret, and J. A. Verbel. Biscuit: New MPCitH Signature Scheme from Structured Multivariate Polynomials. In *ACNS (1)*, volume 14583 of *Lecture Notes in Computer Science*, pages 457–486. Springer, 2024.
11. T. Beyne, A. Canteaut, I. Dinur, M. Eichlseder, G. Leander, G. Leurent, M. Naya-Plasencia, L. Perrin, Y. Sasaki, Y. Todo, and F. Wiemer. Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems. In *CRYPTO (3)*, volume 12172 of *Lecture Notes in Computer Science*, pages 299–328. Springer, 2020.
12. W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).
13. C. Bouillaguet and J. Sauvage. Preliminary cryptanalysis of the biscuit signature scheme. *IACR Communications in Cryptology*, 1(1), 2024.
14. C. Bouvier, P. Briaud, P. Chaidos, L. Perrin, R. Salen, V. Velichkov, and D. Willems. New Design Techniques for Efficient Arithmetization-Oriented Hash Functions: Anemoi Permutations and Jive Compression Mode. In *CRYPTO (3)*, volume 14083 of *Lecture Notes in Computer Science*, pages 507–539. Springer, 2023.

15. B. Buchberger. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3):475–511, 2006. Logic, Mathematics and Computer Science: Interactions in honor of Bruno Buchberger (60th birthday).
16. J. Buchmann, A. Pyshkin, and R. Weinmann. A Zero-Dimensional Gröbner Basis for AES-128. In *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 78–88. Springer, 2006.
17. D. G. Cantor and H. Zassenhaus. A New Algorithm for Factoring Polynomials Over Finite Fields. *Mathematics of Computation*, 36(154):587–592, 1981.
18. J. H. Cheon and D. H. Lee. Resistance of S-Boxes against Algebraic Attacks. In *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 83–94. Springer, 2004.
19. J. Cho, J. Ha, S. Kim, B. Lee, J. Lee, J. Lee, D. Moon, and H. Yoon. Transciphering Framework for Approximate Homomorphic Encryption. In *ASIACRYPT (3)*, volume 13092 of *Lecture Notes in Computer Science*, pages 640–669. Springer, 2021.
20. C. Cid and G. Leurent. An Analysis of the XSL Algorithm. In *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352. Springer, 2005.
21. N. T. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer, 2000.
22. N. T. Courtois and J. Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.
23. J. Daemen and V. Rijmen. AES and the Wide Trail Design Strategy. In *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 108–109. Springer, 2002.
24. C. Dobraunig, D. Kales, C. Rechberger, M. Schofnegger, and G. Zaverucha. Shorter Signatures Based on Tailor-Made Minimalist Symmetric-Key Crypto. In *CCS*, pages 843–857. ACM, 2022.
25. M. Eichlseder, L. Grassi, R. Lüftenegger, M. Øygarden, C. Rechberger, M. Schofnegger, and Q. Wang. An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC. In *ASIACRYPT (1)*, volume 12491 of *Lecture Notes in Computer Science*, pages 477–506. Springer, 2020.
26. J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139:61–88, 1999.
27. J.-C. Faugère. A new Efficient Algorithm for Computing Grobner Bases without Reduction to Zero (F5). *ISSAC '02 : Proceedings of the 2002 international symposium on Symbolic and algebraic computation, New York, NY, USA*, pages 75–83, 2002.
28. J.-C. Faugère and C. Mou. Sparse FGLM Algorithms. *Journal of Symbolic Computation*, 80:538–569, 2017.
29. J.-C. Faugère and L. Perret. On the Security of UOV. Cryptology ePrint Archive, Paper 2009/483, 2009. https://eprint.iacr.org/2009/483.
30. R. Fröberg. An Inequality for Hilbert Series of Graded Algebras. *Mathematica Scandinavica*, 56(2):117–144, 1985.
31. R. Fröberg. An inequality for hilbert series of graded algebras. *Mathematica Scandinavica*, 56, December 1985.
32. L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *USENIX Security Symposium*, pages 519–535. USENIX Association, 2021.

33. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30. ACM, 2007.
34. S. Kim, J. Ha, M. Son, B. Lee, D. Moon, J. Lee, S. Lee, J. Kwon, J. Cho, H. Yoon, and J. Lee. AIM: Symmetric Primitive for Shorter Signatures with Stronger Security. In *CCS*, pages 401–415. ACM, 2023.
35. K. Koschatko, R. Lüftenegger, and C. Rechberger. Exploring the Six Worlds of Gröbner Basis Cryptanalysis: Application to Anemoi. Cryptology ePrint Archive, Paper 2024/250, 2024. https://eprint.iacr.org/2024/250.
36. D. Lazard. Gröbner bases, gaussian elimination and resolution of systems of algebraic equations. In J. A. van Hulzen, editor, *Computer Algebra*, pages 146–156, Berlin, Heidelberg, 1983. Springer Berlin Heidelberg.
37. C. Lim and K. Khoo. An Analysis of XSL Applied to BES. In *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 242–253. Springer, 2007.
38. F. Liu, R. Anand, L. Wang, W. Meier, and T. Isobe. Coefficient Grouping: Breaking Chaghri and More. In *EUROCRYPT (4)*, volume 14007 of *Lecture Notes in Computer Science*, pages 287–317. Springer, 2023.
39. F. Liu, M. Mahzoun, M. Øygarden, and W. Meier. Algebraic Attacks on RAIN and AIM Using Equivalent Representations. *IACR Trans. Symmetric Cryptol.*, 2023(4):166–186, 2023.
40. S. Murphy and M. J. B. Robshaw. Essential Algebraic Structure within the AES. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2002.
41. V. Shoup. Factoring Polynomials over Finite Fields: Asymptotic Complexity vs. Reality. 1993.
42. A. Szepieniec, T. Ashur, and S. Dhooghe. Rescue-Prime: a Standard Specification (SoK). Cryptology ePrint Archive, Paper 2020/1143, 2020. https://eprint.iacr.org/2020/1143.
43. K. Zhang, Q. Wang, Y. Yu, C. Guo, and H. Cui. Algebraic Attacks on Round-Reduced Rain and Full AIM-III. In *ASIACRYPT (3)*, volume 14440 of *Lecture Notes in Computer Science*, pages 285–310. Springer, 2023.

A New Security Evaluation Method Based on Resultant for Arithmetic-Oriented Algorithms

Hong-Sen Yang[1], Qun-Xiong Zheng[1](✉), Jing Yang[1](✉), Quan-Feng Liu[1], and Deng Tang[2](✉)

[1] Information Engineering University, Zhengzhou 450001, China
qunxiong_zheng@163.com, yangjingfi@163.com
[2] Shanghai Jiao Tong University, Shanghai 200240, China
dengtang@sjtu.edu.cn

Abstract. The rapid development of advanced cryptographic applications like multi-party computation (MPC), fully homomorphic encryption (FHE), and zero-knowledge (ZK) proofs have motivated the designs of the so-called arithmetic-oriented (AO) primitives. Efficient AO primitives typically build over large fields and use large S-boxes. Such design philosophy brings difficulties in the cryptanalysis of these primitives as classical cryptanalysis methods do not apply well. The generally recognized attacks against these primitives are algebraic attacks, especially Gröbner basis attacks. Thus, the numbers of security rounds are usually derived through the complexity of solving the system of algebraic equations using Gröbner bases. In this paper, we propose a novel framework for algebraic attacks against AO primitives. Instead of using Gröbner basis, we use *resultants* to solve a system of multivariate equations that can better exploit the algebraic structures of AO primitives. We employ several techniques to reduce the dimensions of the resultants and avoid rapid increases in degrees, including start-from-the-middle modeling, variable substitutions, and fast Lagrange interpolation. We apply our attack to three mainstream AO cryptographic primitives: Rescue-Prime, `Anemoi`, and Jarvis. For Rescue-Prime, we theoretically prove that the final univariate equation has a degree of at most a specific power of three and practically attack five rounds for the first time. We attack the full-round `Anemoi` with complexity $2^{110.10}$, which has been claimed to provide 127 bits of security. We also give the first practical attack against eight rounds of `Anemoi` over a 55-bit prime field. For Jarvis, we improve the existing practical attack of six rounds by a factor of 100 and practically attack eight rounds for the first time. Therefore, we point out that our analysis framework can be used as a new evaluation method for AO designs.

Keywords: Resultant · arithmetic-oriented primitives · Rescue-Prime · `Anemoi` · Jarvis · new evaluation method

1 Introduction

In recent years, with multi-party computation (MPC), fully homomorphic encryption (FHE), zero-knowledge (ZK) proofs, and other privacy computing techniques having been applied on the ground, many underlying privacy computing-friendly symmetric cryptographic primitives have emerged. While traditional symmetric ciphers like AES [25] and SHA-3 [18] are often designed and optimized for efficient software or hardware implementations, these constructions, usually referred to as *arithmetic-oriented* (AO), mainly focus on minimizing the number of non-linear arithmetic operations [23]. This is because non-linear operations pose the largest performance bottleneck in those advanced protocols, while linear computations are much cheaper compared to the former.

Till now, quite a few AO primitives have been proposed. LowMC [3] and MiMC [1] can be regarded as the first generation of AO primitives. LowMC uses partial S-box layers while MiMC works over large fields to decrease the number of multiplications. In July 2018, the Ethereum Foundation gave StarkWare a 2-year milestone-based grant to select a STARK-friendly hash (SFH) function [9]. STARK is one of the most efficient and recognized ZK proof systems. Two families of primitives received much attention in the evaluation, which are: **MARVELlous** – a family that includes Jarvis [5], Vision (over the binary field), Pepper, and Rescue (over prime fields) [4]; and **HadesMiMC** – a family that includes Starkad (over the binary field) and Poseidon (over prime fields) [22].

After the evaluation of security and efficiency, Rescue$_{122}$ was recommended as the SFH candidate for standardization by the Ethereum Foundation. Several other AO primitives have been proposed afterwards, e.g., Ciminion [17] and `Anemoi` [11]. Most of these primitives are constructed over $\mathbb{F}_p$ (p is typically a large prime number) to be consistent with many MPC/FHE/ZK-protocols that natively support operations in $\mathbb{F}_p$ to improve implementation performance. To reduce the computation overhead of nonlinear operations, many AO primitives use low-degree round functions. This introduces the most significant difference to classical symmetric ciphers that these AO primitives typically work with large S-boxes over the whole states instead of individual bytes or cells.

Cryptanalysis. Compared to the wide attention in the design of efficient AO primitives, the security analysis methods of these primitives are not mature. Classical cryptanalysis techniques, like linear and differential cryptanalysis, do not apply well to these primitives due to the adoption of large prime fields and S-boxes. To better understand the security of AO hash functions, in November 2021, the Ethereum Foundation initiated bounties[1] rewarding the best practical attacks against four round-reduced AO hash functions. These four hash functions are all sponge-based, which are Reinforced Concrete [21], Feistel–MiMC [1], Poseidon [22], and Rescue-Prime [28].

[1] These bounties were published at https://www.zkhashbounties.info/.

The security of AO primitives typically relies on the hardness of solving the constrained-input constrained-output (CICO) problem. Due to their native algebraic properties, algebraic attacks typically outperform other known cryptanalytic techniques against AO primitives. One example is the Gröbner basis attacks, which model the underlying primitive as a system of multivariate equations and solve it using off-the-shelf Gröbner basis algorithms. For instance, Gröbner basis attacks against Jarvis and Friday [5] were presented in [2], illustrating that AO primitives for MPC/FHE/ZK applications may be particularly vulnerable to algebraic attacks. The authors proposed a smart way of constructing equations that made the solving step much more efficient than initially thought and further broke the security claim of Jarvis. Therefore, the number and quality of constructed equations greatly influence the attack complexity. For some AO primitives based on the Substitution-Permutation Network (SPN) structure, authors in [8] proposed a technique that can remove all the equation modeling of the first two layers of S-boxes. Such a technique is used to analyze the security of the four AO hash functions considered in the Ethereum Foundation challenge. Recently, a new type of algebraic attack named FreeLunch was proposed in [7], which chose the monomial ordering so that the natural polynomial system encoding the CICO problem is already a Gröbner basis. The authors claimed that the FreeLunch approach challenges the security of full-round instances of `Anemoi` [11], Arion [26], and Griffin [20]. Other attacks include integral attacks against GMiMC and HadesMiMC [10], higher order differential cryptanalysis of Ciminion [30], etc.

Contributions. Since the Gröbner basis algorithms are generic solving methods that usually ignore some specific algebraic properties of a cryptographic primitive, using them to evaluate the attack complexities may overestimate the security of a certain primitive. In this paper, we propose a new analysis framework for algebraic attacks against typical AO primitives, which makes full use of the algebraic properties and employs the resultant tool to solve the system of multivariate equations. Compared to Gröbner basis attacks, our attack is much more efficient and accurate. Table 1 presents a comparison of our algebraic attacks with existing ones on three AO primitives, where the theoretical complexities are measured by numbers of basic arithmetic operations in a finite field. Our main contributions are as below.

1. We develop a novel analysis framework for analyzing the security of AO primitives. We note that typical AO primitives have special algebraic structures that are not considered in Gröbner basis attacks. We make full use of such algebraic structures and propose to use the *resultant* to solve the system of multivariate equations. To avoid a rapid increase in the degrees of variables when computing the resultants, we propose the *substitution theory* to deal with the non-linear operations, and the dimensions of the resultants are much reduced. Such substitutions also enable us to accurately estimate the degrees of variables and equations, thus the estimation of the attack complexity is much more accurate than that in Gröbner basis attacks. We also

propose to use the start-from-the-middle (SFTM) modeling technique to further simplify the resultants and use fast Lagrange interpolation to parallelize the computation. Therefore, our algebraic attack is a comprehensive analysis framework that is much more efficient than existing ones. Besides, the path of variable elimination is quite clear compared to that when using Gröbner bases, which helps to better understand the security of an AO primitive. We think the analysis framework can be used as an efficient evaluation method for new AO designs.
2. We apply the new analysis framework and techniques to the AO primitive Rescue-Prime. We propose the cubic substitution theory for Rescue-Prime and theoretically prove that the final univariate equation has a degree of at most a specific power of three. With the cubic substitution theory, we can control the degree of the equation at each step and give a more accurate security evaluation of Rescue-Prime. We find a 5-round collision which was originally thought to be "hard" in the Ethereum Foundation challenge. We also achieve a 100-fold increase in finding the 4-round collision of Rescue-Prime over the results in [8].
3. We attack the full-round of an specific instantiation of `Anemoi` with complexity $2^{110.10}$, while the version is claimed to provide 127 bits of security. We also provide an practical attack against 7-round `Anemoi` which improves the best-known practical attack by a factor of fifty [7]. Moreover, we propose a practical attack against eight rounds for the first time.
4. We further apply our attack to six rounds of Jarvis and provide a 100-fold increase in practically solving the equation system compared to the results in [2]. Moreover, we practically attack eight rounds of Jarvis for the first time.

Organization. We start by introducing the necessary mathematical background and security definitions in Sect. 2. In Sect. 3, we give an introduction to Rescue-Prime and review the work in [8]. We propose our analysis framework and instantiate it on Rescue-Prime in detail in Sect. 4. After that, we apply our attack to `Anemoi` and Jarvis in Sect. 5 and Sect. 6, respectively. We finally conclude the paper in Sect. 7 by discussing some shortcomings and potential improvements in the current design of AO primitives.

2 Preliminaries

Let $\mathbb{F}_q$ be the finite field with q elements and $\mathbb{F}_q[x_1, x_2, \ldots, x_s]$ the polynomial ring over $\mathbb{F}_q$ with indeterminates $x_1, x_2, \ldots, x_s$, where q is a prime power and s is a positive integer. Any polynomial $f(x_1, x_2, \ldots, x_s) \in \mathbb{F}_q[x_1, x_2, \ldots, x_s]$ can be represented in the form

$$f(x_1, x_2, \ldots, x_s) = \sum_{(k_1, k_2, \ldots, k_s) \in K} c_{k_1, k_2, \ldots, k_s} x_1^{k_1} x_2^{k_2} \cdots x_s^{k_s},$$

where K is a set of finitely many s-tuples $(k_1, k_2, \ldots, k_s)$ of nonnegative integers and $c_{k_1, k_2, \ldots, k_s} \in \mathbb{F}_q$. If $c_{k_1, k_2, \ldots, k_s} \neq 0$, then $x_1^{k_1} x_2^{k_2} \cdots x_s^{k_s}$ is called a term of f

Table 1. Comparison of the algebraic attacks against Rescue-Prime, **Anemoi**, and JARVIS

Primitives	Attacked rounds	Running time	Theoretical Complexities*	References
Rescue-Prime	4	258500s	-	[8]
	4	885.5s	-	Sect.4
	5	-	2^{55}	[8]
	5	$\approx$ one day	-	Sect.4
	6	-	$2^{59.96}$	Sect.4
Anemoi	7	167201s	-	[7]
	7	2968.55s	-	Sect.5
	8	10575.61s**	-	[12]
	8	38749.182s	-	Sect.5
	21	-	2^{118}	[7]
	21	-	$2^{110.10}$	Sect.5
JARVIS	6	99989s	-	[2]
	6	368.96s	-	Sect.6
	8	455650.53s	-	Sect.6

* The unit of complexities throughout the paper is one basic arithmetic operation in a finite field.
** The practical attack was performed over $\mathbb{F}_{2^{16}+1}$ in [12], while our attacks are performed over the same 55-bit prime field as that in [7].

and its degree is $k_1 + k_2 + \cdots + k_s$. The set of all terms of f is denoted by $T(f)$. For $f \neq 0$, the degree of f, denoted by $\deg(f)$, is the maximum of the degrees of the terms of f, that is,

$$\deg(f) = \max\left\{\sum_{j=1}^{s} k_j \ \middle|\ x_1^{k_1} x_2^{k_2} \cdots x_s^{k_s} \in T(f)\right\}.$$

2.1 CICO Problem

The so-called CICO (constrained-input constrained-output) problem, which is usually used to evaluate the security of AO algorithms, is defined as below.

Definition 1 (CICO problem). Let $s > 1$ be an integer and u a given positive integer smaller than s. Let F: $\mathbb{F}_q^s \to \mathbb{F}_q^s$ be a permutation. The CICO problem of F is to find a vector $(x_1, \ldots, x_{s-u}, y_1, \ldots, y_{s-u}) \in \mathbb{F}_q^{2(s-u)}$ such that

$$F\left(x_1, \ldots, x_{s-u}, \underbrace{0, \ldots, 0}_{u}\right) = \left(y_1, \ldots, y_{s-u}, \underbrace{0, \ldots, 0}_{u}\right).$$

2.2 Resultant

The resultant is a powerful tool for solving systems of polynomial equations. As it will be seen in Sect. 4–6, the polynomial equations modeled in algebraic attacks against AO algorithms (such as Rescue-Prime, `Anemoi`, and Jarvis) are particularly suitable to be solved through the resultant-based method. Compared with the Gröbner basis method, using resultants tends to give a more precise estimation of the time complexity.

Definition 2 (Resultant). Let $f(\boldsymbol{x}, y), g(\boldsymbol{x}, y) \in \mathbb{F}_q[\boldsymbol{x}, y]$ with $\boldsymbol{x} = (x_1, \ldots, x_s)$. The resultant of $f(\boldsymbol{x}, y)$ and $g(\boldsymbol{x}, y)$ with respect to the indeterminant y, denoted by $R(f, g, y)$, is defined as the determinant of the Sylvester matrix of $f(\boldsymbol{x}, y)$ and $g(\boldsymbol{x}, y)$ when considered as polynomials in the single indeterminate y. That is, if $f(\boldsymbol{x}, y) = \sum_{i=0}^{m} f_i y^i$ and $g(\boldsymbol{x}, y) = \sum_{i=0}^{n} g_i y^i$, where $f_i, g_i \in \mathbb{F}_q[\boldsymbol{x}]$, then

$$R(f,g,y) = \begin{vmatrix} f_m & f_{m-1} & \cdots & f_0 & & & & \\ & f_m & f_{m-1} & \cdots & f_0 & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & f_m & f_{m-1} & \cdots & f_0 \\ g_n & g_{n-1} & \cdots & \cdots & g_0 & & & \\ & g_n & g_{n-1} & \cdots & \cdots & g_0 & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & g_n & g_{n-1} & \cdots & \cdots & g_0 \end{vmatrix}.$$

It is well-known that the resultant is non-zero if and only if the two polynomials are algebraically independent. In this case, the resultant yields a new polynomial $h(\boldsymbol{x})$, such that if $(\boldsymbol{x_0}, y_0)$ is a root of both $f(\boldsymbol{x}, y)$ and $g(\boldsymbol{x}, y)$, then $h(\boldsymbol{x_0}) = 0$. In this way, we can remove one variable from two polynomials while retaining information about the roots of the original polynomials. Given ℓ polynomials in s variables, we can repeatedly compute resultants of the polynomials until we get a univariate polynomial. Solving for the roots of that polynomial and repeatedly substituting them back, we can derive the roots that the polynomials have in common.

The computation of a resultant involves multiplications of multivariate polynomials over $\mathbb{F}_q$. Lemma 1 can be used to give an estimation of the running time of the resultant-based method, of which the proof is omitted as it is clearly true.

Lemma 1. Let $f, g \in \mathbb{F}_q[x_1, x_2, \ldots, x_s]$ with

$$f = \sum_{\substack{0 \le i_u \le n_u \\ u=1,2,\ldots,s}} a_{i_1,i_2,\ldots,i_s} x_1^{i_1} x_2^{i_2} \cdots x_s^{i_s}, \quad g = \sum_{\substack{0 \le j_u \le m_u \\ u=1,2,\ldots,s}} b_{j_1,j_2,\ldots,j_s} x_1^{j_1} x_2^{j_2} \cdots x_s^{j_s},$$

where $n_1, n_2, \ldots, n_s$ and $m_1, m_2, \ldots, m_s$ are positive integers. Then the multiplication fg can be computed in $\mathcal{O}(d_1 d_2)$ field operations, where $d_1 = \prod_{u=1}^{s}(n_u+1)$ and $d_2 = \prod_{u=1}^{s}(m_u + 1)$.

It can be easily seen that when using the resultant-based method to solve a system of multivariate equations, each time we eliminate a variable by computing a resultant, the degrees of other variables will increase. Repeatedly computing resultants of the polynomials, we will finally get two polynomials, say $f(x, y)$ and $g(x, y)$. The degrees of x and y in $f(x, y)$ and $g(x, y)$ will inevitably be very high, especially when attacking a high-round AO algorithm. To get a univariate polynomial, one can directly compute the resultant $R(f, g, y)$ of $f(x, y)$ and $g(x, y)$. However, the degree of x may become extremely high, which introduces unaffordable memory consumption in practical attacks. To solve that, we propose to use Lagrange interpolation to compute $R(f, g, y)$, which also enables parallelization. We note that an upper bound on the degree of x in $R(f, g, y)$ can always be estimated in advance.

One can use interpolation to compute polynomial resultants as suggested by Collins [14]. We give a brief introduction to fast Lagrange interpolation based on fast multi-point evaluation in the full version in [29, Sect. 2.3] and refer to [19, Chapter 10] for more details.

3 Review of Rescue-Prime and a Recent Algebraic Attack

3.1 Description of Rescue-Prime

Rescue-Prime [28] is a family of AO hash functions of which the round function is shown in Fig. 1, where $x_{2i}, y_{2i}, \ldots$ and $x_{2i+2}, y_{2i+2}, \ldots$ are the inputs and outputs of the i-th round, respectively. Each round of Rescue-Prime consists of two similar steps: the first step involves low-degree S-boxes S, an MDS (Maximum Distance Separable) matrix M, and the addition of the round constants AddC; the second step differs in replacing S by S^{-1} and using other round constants. There is an additional AddC operation before the first round. Denote t the number of S-boxes involved in each step. The challenges from the Ethereum Foundation use $t = 3$ or $t = 2$ and $S : x \mapsto x^3$ (and so $S^{-1} : x \mapsto x^{1/3}$). Let $F : \mathbb{F}_q^3 \to \mathbb{F}_q^3$ be the permutation representing the r-round Rescue-Prime, the challenge initiated by Ethereum Foundation for Rescue-Prime with $t = 3$ is to find two pairs $(X_1, X_2), (Y_1, Y_2) \in \mathbb{F}_q^2$ satisfying $F(X_1, X_2, 0) = (Y_1, Y_2, 0)$. **In this paper, we mainly focus on $t = 3$.**

3.2 The Algebraic Attack Against Rescue-Prime in [8]

The authors of [8] introduce a smart technique to bypass the first two layers of S-boxes (two steps) of Rescue-Prime with little or even no overhead. We now give a brief review of this work.

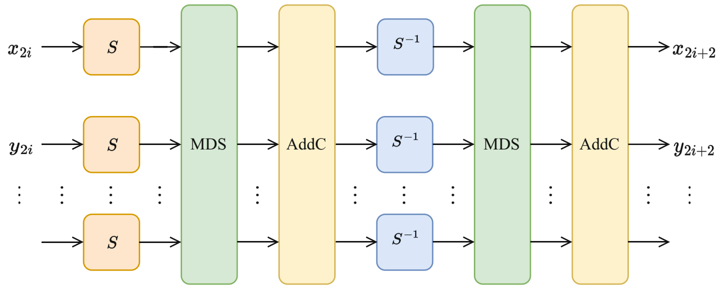


Fig. 1. The i-th round function of Rescue-Prime

Construction of Equations. There are $2r$ steps for solving the CICO problem of a r-round Rescue-Prime, which are referred to as the 0-th step, the 1-th step, ..., the $(2r-1)$-th step. Since the MDS matrix is the same in each step, it can be uniformly denoted by M. Let $AddC_k$ be the addition of the round constants in the k-th step and $L_k = AddC_k \circ M$ be the composition of M and $AddC_k$, where $0 \le k \le 2r-1$. Let $L_{k,j}$ be the j-th output of L_k with $0 \le j \le t-1$. Since S^{-1} and S have high degrees in the forward and backward directions, respectively, some intermediate variables can be introduced to build low-degree equations. More concretely, let x_{2i}, y_{2i}, z_{2i} and $x_{2i+2}, y_{2i+2}, z_{2i+2}$ be the input and output of the i-th round, respectively, where $0 \le i \le r-1$. They can be connected through the equations below as shown in Fig. 2:

$$L_{2i,j}(x_{2i}^3, y_{2i}^3, z_{2i}^3) - \left(L_{2i+1,j}^{-1}(x_{2i+2}, y_{2i+2}, z_{2i+2})\right)^3 = 0, j \in \{0,1,2\}, \qquad (1)$$

where $L_{2i+1,j}^{-1}$ is the j-th output of the inverse of the affine transformation L_{2i+1}. It is clear that each equation in Eq. (1) is of degree three since both $L_{2i,j}$ and L_{2i+1}^{-1} are of degree one. The variables z_0 and z_{2r} are both set to zero in the CICO problem, i.e. $z_0 = z_{2r} = 0$. Therefore, a system of $3r$ equations in $3r+1$ variables is derived.

Bypassing the First Two S-Box Layers. As having been observed in [8], the first two nonlinear layers can be skipped when launching an algebraic attack against Rescue-Prime. The main reason for this is the lack of a linear diffusion layer before the S-box layer. More specifically, as shown in Fig. 3, let $C_{-1,0}, C_{-1,1}, C_{-1,2}$ be the three constants of the additional AddC operation before the first round, and let $C_{0,0}, C_{0,1}, C_{0,2}$ be the three constants of $AddC_0$, and let X, Y, Z be the outputs of S-boxes in the 1-th step. To satisfy the CICO problem, the output of the third S-box in the 0-th step is $(C_{-1,2})^3$, which yields

$$(C_{-1,2})^3 = \alpha_{2,0}(X^3 - C_{0,0}) + \alpha_{2,1}(Y^3 - C_{0,1}) + \alpha_{2,2}(Z^3 - C_{0,2}), \qquad (2)$$

where $M^{-1} = (\alpha_{i,j})_{0 \le i \le 2, 0 \le j \le 2}$. It is easy to see that there are many three tuples (X, Y, Z) satisfying Eq. (2). For example, if one sets $Z = c$, where

$$c^3 = \alpha_{2,2}^{-1}(\alpha_{2,0}C_{0,0} + \alpha_{2,1}C_{0,1} + \alpha_{2,2}C_{0,2} + (C_{-1,2})^3), \qquad (3)$$

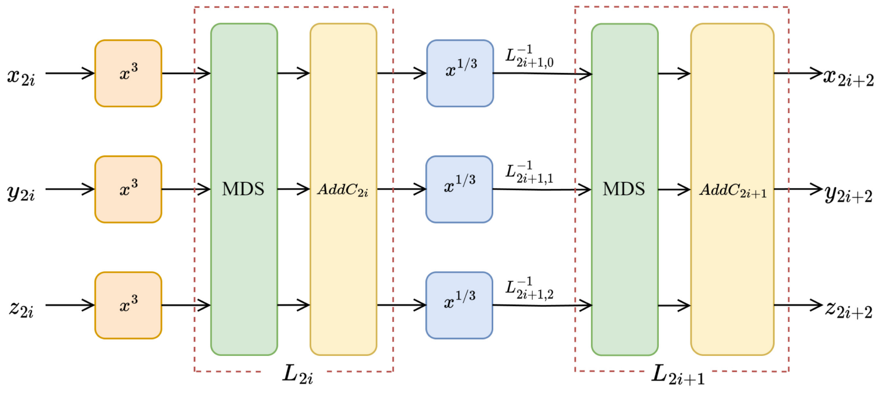


Fig. 2. One round of Rescue-Prime with $t = 3$

then Eq. (2) is simplified as $\alpha_{2,0}X^3 + \alpha_{2,1}Y^3 = 0$, and so $Y = (-\frac{\alpha_{2,0}}{\alpha_{2,1}})^{1/3}X$. The analysis above implies that if one sets

$$(X, Y, Z) = (X, (-\frac{\alpha_{2,0}}{\alpha_{2,1}})^{1/3}X, c), \tag{4}$$

then the inputs of Rescue-Prime naturally have the form $(*, *, 0)$. As a result, if there exists $X \in \mathbb{F}_q$ such that the image of $(X, (-\frac{\alpha_{2,0}}{\alpha_{2,1}})^{1/3}X, c)$ through $(r-1)$-round Rescue-Prime is equal to $(*, *, 0)$, then it is able to deduce an original input $(*, *, 0)$ of r-round Rescue-Prime with an image of the form $(*, *, 0)$.

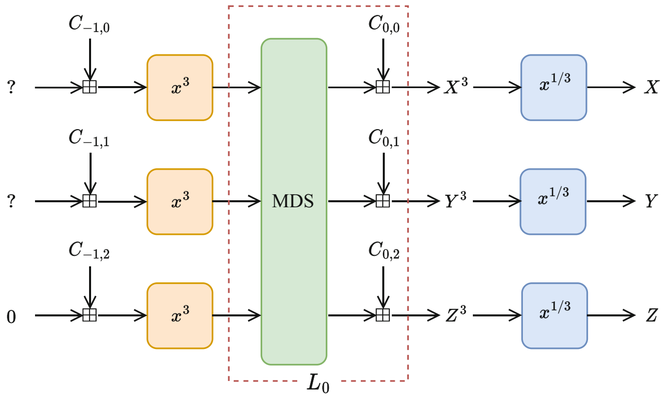


Fig. 3. Main idea of [8] on how to bypass the first round of Rescue-Prime.

Remark 1. It is worth noting that such X does not always exist since the mapping from X to the third output of r-round Rescue-Prime is not necessarily one-to-one. Then one may instead assign a value c to X (or Y), and similarly deduce a linear relation for the other two variables, say $Y = \alpha Z$ (or $X = \alpha Z$). Finally, find $Z \in \mathbb{F}_q$ such that the image of $(c, \alpha Z, Z)$ (or $(\alpha Z, c, Z)$) through $(r-1)$-round Rescue-Prime is equal to $(*, *, 0)$. However, the existence of such X or Z is closely related to the constants used in Rescue-Prime. Moreover, there do exist constants (constructable) such that desired X or Z are nonexistent.

Solving a System of Equations. With the technique of bypassing the first round, Rescue-Prime can usually be attacked one more step (the first layer of S-boxes can be skipped naturally) with little to no cost. For r-round Rescue-Prime, the authors of [8] obtain a system of $3(r-1)$ equations of degrees 3 in $3(r-1)$ variables. Theoretically, the system can be solved using the F5 and FGLM algorithms together, and the complexity can be estimated. Moreover, practical attacks on 3-round and 4-round Rescue-Prime are implemented with Magma in [8] (using F4 algorithm, not F5 algorithm, to find the grevlex Gröbner basis), which take 9.18 s and 258500 s, respectively. The time complexity for attacking 5-round Rescue-Prime is roughly estimated as 2^{57} while the memory complexity is unknown. We refer to [8, Section 3.2 and Table 3] for more details. The authors also found that the final univariate polynomial has a degree of $3^{3(r-1)}$ but did not give a proof. We will fill in the gap in the next section.

4 Optimized Algebraic Attacks Against Rescue-Prime Based on Resultant

In this section, we will give optimized algebraic attacks against Rescue-Prime based on resultant. We note that previous algebraic attacks against Rescue-Prime primarily relied on the Gröbner basis method. Compared with the Gröbner basis method, the resultant-based method tends to give more precise estimation of the time complexity. As it will be seen, the system of equations constructed in an algebraic attack has a very special structure, which clearly indicates a path for eliminating the variables by computing corresponding resultants. Based on this special structure, we propose the cubic substitution theory, with which the degrees of all but two variables at most in a multivariate polynomial derived by each resultant can be forced to remain at most equal to 2. The benefits of this are at least twofold: (1) most of the resultants can be computed with determinants of 5 by 5 matrices; (2) a tight upper bound on the degree of a multivariate polynomial obtained by each resultant can be clearly given, which together make the estimation of the time complexity more accurate. More importantly, when combining resultants with the cubic substitution theory and the fast Lagrange interpolation, practical algebraic attacks on higher rounds of Rescue-Prime become possible. For example, the existing best-known practical attack was against 4-round Rescue-Prime (with $t = 3$), which took 258500 s

[8]. However, with the resultant-based method, we can successfully attack 4-round Rescue-Prime (with $t = 3$) in 885.5 s. Moreover, we can practically attack 5-round Rescue-Prime in about one day, which was originally thought to be "hard" in the Ethereum Foundation challenge. For details, see the comparison in Table 5.

4.1 Algebraic Attack with Forward Modeling

As done in Sect. 3.2, the $2r$ steps of r-round Rescue-Prime are referred to as the 0-th step, the 1-th step, ..., the $(2r-1)$-th step.

Construction of Equations. The construction of equations roughly follows the method introduced in [8], but with the variables set at different locations (see Fig. 2 and Fig. 4 for a comparison). For $0 \le i \le r-1$, let $x_{2i+1}, y_{2i+1}, z_{2i+1}$ be the outputs of the three S-boxes (S^{-1}) in the $(2i+1)$-th step. We denote by SSS the three S-boxes arranged in parallel—that is,

$$SSS(x, y, z) = (S(x), S(y), S(z)) = (x^3, y^3, z^3).$$

Then it can be seen from Fig. 4 that

$$\begin{cases} x_{2i+3}^3 = L_{2i+2,0} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}) \\ y_{2i+3}^3 = L_{2i+2,1} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}) \\ z_{2i+3}^3 = L_{2i+2,2} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}) \end{cases} \text{for } i \in \{0, 1, \ldots, r-2\}, \tag{5}$$

where "$\circ$" denotes the composition of mappings and $L_{2i+2,j}$ is the j-th output of L_{2i+2} with $j \in \{0, 1, 2\}$.

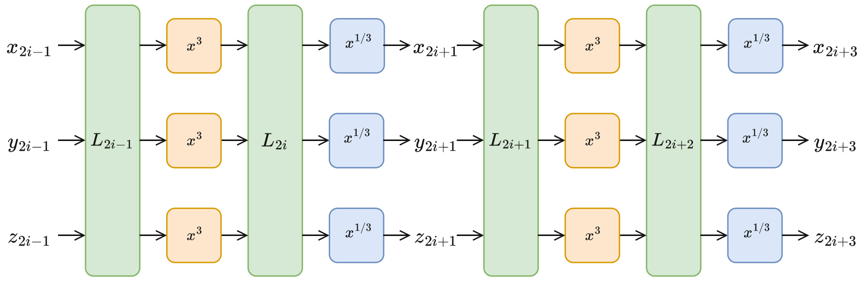


Fig. 4. Forward modeling of Rescue-Prime

Remark 2. There are $\binom{6}{3} = 20$ terms at most in the expansion of

$$L_{2i+2,j} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}).$$

Next, we use the technique of bypassing the first round of Rescue-Prime reviewed in Sect. 3.2 to construct two more equations. Specifically, as shown in Eq. (4), to make the inputs of Rescue-Prime are of the form $(*, *, 0)$,

$$(x_1, y_1, z_1) = (X, (-\frac{\alpha_{2,0}}{\alpha_{2,1}})^{1/3} X, c),$$

where c is some fixed element in $\mathbb{F}_q$ and X is any element in $\mathbb{F}_q$. Then, we get

$$y_1 = (-\frac{\alpha_{2,0}}{\alpha_{2,1}})^{1/3} x_1 \text{ and } z_1 = c. \tag{6}$$

Finally, the output of the r-round Rescue-Prime should be of the form $(*, *, 0)$ in the CICO problem, so there is one more equation, which is

$$L_{2r-1,2}\left(x_{2r-1}, y_{2r-1}, z_{2r-1}\right) = 0. \tag{7}$$

Solving the System of Equations with the Resultant-Based Method. It can be seen from Eqs. (5)–(7) that the system of equations has $3r-2$ equations in $3r-2$ unknowns (here we omit z_1 and y_1 since $z_1 = c$ is known and $y_1 = (-\frac{\alpha_{2,0}}{\alpha_{2,1}})^{1/3} x_1$): $3r-3$ equations are of degree 3 and one equations is of degree 1. Such a system of equations has a special structure: (1) it can be seen from Eq. (5) that the variable x_{2i+3} (or y_{2i+3}, or z_{2i+3}) is only associated with three other lower-subscript variables $x_{2i+1}, y_{2i+1}, z_{2i+1}$; (2) the variable x_{2r-1} (or y_{2r-1}, or z_{2r-1}) are involved only in Eq. (5) for the case $i = r-2$ and Eq. (7). This special structure makes the system of equations especially suitable for solving by the resultant-based method, since it clearly indicates a path for elimination of the variables. Specifically, let

$$f_{x_{2i+3}} = x_{2i+3}^3 - L_{2i+2,0} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}),$$
$$f_{y_{2i+3}} = y_{2i+3}^3 - L_{2i+2,1} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}),$$
$$f_{z_{2i+3}} = z_{2i+3}^3 - L_{2i+2,2} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}),$$

for $i \in \{0, 1, \ldots, r-2\}$, and let

$$f_h = L_{2r-1,2}\left(x_{2r-1}, y_{2r-1}, z_{2r-1}\right).$$

Only f_h and $f_{z_{2r-1}}$ contain the variable z_{2r-1}, and it can be eliminated by computing the resultant $R(f_h, f_{z_{2r-1}}, z_{2r-1})$. Update f_h with $R(f_h, f_{z_{2r-1}}, z_{2r-1})$, then only the updated f_h and $f_{y_{2r-1}}$ contain the variable y_{2r-1}. Then y_{2r-1} can be eliminated in the same way. Generally, following Algorithm 1, we can eliminate one variable in each computation of a resultant, and derive a univariate polynomial in $\mathbb{F}_q[x_1]$ in the end. Solving the roots of the derived univariate polynomial and substituting back, the roots of the original system of equations will be found.

Algorithm 1: Get the univariate polynomail for r-round Rescue-Prime

```
Input: $f_{x_{2i+3}}, f_{y_{2i+3}}, f_{z_{2i+3}}, f_h$ with $i \in \{0, 1, \ldots, r-2\}$.
Output: a univariate polynomial in $\mathbb{F}_q[x_1]$.
1  $i \leftarrow r-2$;
2  while $i \geq 1$ do
3      $f_h \leftarrow R(f_h, f_{z_{2i+3}}, z_{2i+3})$;
4      apply the cubic substitution to $f_h$;
5      $f_h \leftarrow R(f_h, f_{y_{2i+3}}, y_{2i+3})$;
6      apply the cubic substitution to $f_h$;
7      $f_h \leftarrow R(f_h, f_{x_{2i+3}}, x_{2i+3})$;
8      apply the cubic substitution to $f_h$;
9      $i \leftarrow i-1$;
10 end
11 $f_h \leftarrow R(f_h, f_{z_{2i+3}}, z_3)$;
12 apply the cubic substitution to $f_h$;
13 $f_h \leftarrow R(f_h, f_{y_{2i+3}}, y_3)$;
14 apply the cubic substitution to $f_h$;
15 $f_h \leftarrow R(f_h, f_{y_{2i+3}}, x_3)$;
16 return $f_h$.
```

Cubic Substitution Theory. We note that the degrees of variables in the multivariate polynomial obtained by each resultant in Algorithm 1 will increase. However, by making use of the special structure in Eq. (5), the degrees of all variables except x_1 can be forced to remain at most equal to 2. In fact, Eq. (5) has two properties: (1) the variables $x_{2i+3}, y_{2i+3}, z_{2i+3}$ are separated in the sense that they are not mixed with the lower-subscript variables $x_{2i+1}, y_{2i+1}, z_{2i+1}$; (2) the degree of x_{2i+3} (or y_{2i+3}, or z_{2i+3}) is exactly 3. For the first resultant $R(f_h, f_{z_{2r-1}}, z_{2r-1})$ in Algorithm 1, it is a multivariate polynomial in $y_{2r-1}, x_{2r-1}, x_{2r-3}, y_{2r-3}, z_{2r-3}$ with the degree of each variable at most 3. By using the following substitutions in order, i.e.,

$$y_{2r-1}^3 = L_{2r-2,1} \circ SSS \circ L_{2r-3}(x_{2r-3}, y_{2r-3}, z_{2r-3}),$$

$$x_{2r-1}^3 = L_{2r-2,0} \circ SSS \circ L_{2r-3}(x_{2r-3}, y_{2r-3}, z_{2r-3}),$$

$$z_{2r-3}^3 = L_{2r-4,2} \circ SSS \circ L_{2r-5}(x_{2r-5}, y_{2r-5}, z_{2r-5}),$$

$$y_{2r-3}^3 = L_{2r-4,1} \circ SSS \circ L_{2r-5}(x_{2r-5}, y_{2r-5}, z_{2r-5}),$$

$$x_{2r-3}^3 = L_{2r-4,0} \circ SSS \circ L_{2r-5}(x_{2r-5}, y_{2r-5}, z_{2r-5}),$$

$$\cdots$$

$$z_3^3 = L_{2,2} \circ SSS \circ L_1(x_1, kx_1, c),$$

$$y_3^3 = L_{2,1} \circ SSS \circ L_1(x_1, kx_1, c),$$

$$x_3^3 = L_{2,0} \circ SSS \circ L_1(x_1, kx_1, c),$$

where c and $k = (-\frac{\alpha_{2,0}}{\alpha_{2,1}})^{1/3}$ are two fixed elements in $\mathbb{F}_q$ as defined in Eq. (3) and Eq. (4), respectively, $R(f_h, f_{z_{2r-1}}, z_{2r-1})$ is transformed into a multivariate polynomial in y_{2r-1}, x_{2r-1}, z_{2r-3}, y_{2r-3}, $x_{2r-3}, \ldots, z_3$, y_3, x_3, x_1 with the degree of each variable except x_1 less than 3. The substitutions above are called *cubic substitutions* for convenience.

Remark 3. We note that if there is a variable, say x_{2i+1}, of which the degree is larger than 3, then we may need to use multiple substitutions of the form $x_{2i+1}^3 = L_{2i,0} \circ SSS \circ L_{2i-1}(x_{2i-1}, y_{2i-1}, z_{2i-1})$ to reduce its degree less than 3. For example, let $f_h = x_{2i+1}^7 + x_{2i+1}^3 + 1$ and $x_{2i+1}^3 = x_{2i-1}^3 + y_{2i-1}^3 + z_{2i-1}^3$, then we need two substitutions for x_{2i+1}^7 and one for x_{2i+1}^3, and derive $f_h = (x_{2i-1}^3 + y_{2i-1}^3 + z_{2i-1}^3)^2 * x_{2i+1} + x_{2i-1}^3 + y_{2i-1}^3 + z_{2i-1}^3 + 1$. The degree of x_{2i+1} in f_h now becomes less than 3.

Now we consider the second resultant $R(f_h, f_{y_{2r-1}}, y_{2r-1})$ in Algorithm 1. Since the degree of y_{2r-1} in f_h is less than 3 after the cubic substitutions, the resultant $R(f_h, f_{y_{2r-1}}, y_{2r-1})$ can be computed with the determinant of a 5 by 5 matrix. By cubic substitutions, $R(f_h, f_{y_{2r-1}}, y_{2r-1})$ is changed into a multivariate polynomial in x_{2r-1}, z_{2r-3}, y_{2r-3}, $x_{2r-3}, \ldots, z_3$, y_3, x_3, x_1 with the degree of each variable less than 3 except x_1. Similarly, when applying cubic substitutions to all the other resultants, the resultants are changed into multivariate polynomials with the degree of each variable less than 3 except x_1.

The cubic substitution has the following basic properties, which are useful for getting a tight upper bound on the degree of a polynomial obtained by each resultant.

Lemma 2. With the notations defined above, let f_h' be the polynomial obtained by cubic substitutions of f_h such that the degree of each variable in f_h' is less than 3 except x_1, then we have $\deg(f_h') \leq \deg(f_h)$.

Proof. From the cubic substitutions of the first resultant $R(f_h, f_{z_{2r-1}}, z_{2r-1})$, it can be seen that a term of degree 3 (say y_{2r-1}^3) is substituted by a polynomial (say $L_{2r-2,1} \circ SSS \circ L_{2r-3}(x_{2r-3}, y_{2r-3}, z_{2r-3})$) of degree at most 3 in each substitution, so $\deg(f_h') \leq \deg(f_h)$.

Remark 4. We note that the base field of Rescue-Prime is a large finite field, therefore, $\deg(f_h') = \deg(f_h)$ holds with a high probability. Moreover, we have experimentally verified that $\deg(f_h') = \deg(f_h)$ always holds for the parameters used in Rescue-Prime.

Lemma 3. Let R_k be the multivariate polynomial computed from the k-th resultant in Algorithm 1 with $1 \leq k \leq 3(r-1)$. Then $\deg(R_k) \leq 3^k$.

Proof. Let ω_k be the variable to be eliminated by the k-th resultant in Algorithm 1 for $1 \leq k \leq 3(r-1)$, for example, $\omega_1 = z_{2r-1}$, $\omega_2 = y_{2r-1}$, $\omega_4 = z_{2r-3}$, and so on. Now we consider the k-th resultant $R(f_h, f_{\omega_k}, \omega_k)$ in Algorithm 1. As the degree of ω_k in f_h after cubic substitutions is less than 3, we can assume that

$$f_h = u_2\omega_k^2 + u_1\omega_k + u_0, \quad f_{\omega_k} = \omega_k^3 - u_3,$$

where $u_0, u_1, u_2, u_3 \in \mathbb{F}_q[\omega_{k+1}, \ldots, \omega_{3r-3}, x_1]$ with

$$\deg(u_0) \le \deg(f_h), \deg(u_1) \le \deg(f_h) - 1, \deg(u_2) \le \deg(f_h) - 2, \text{ and } \deg(u_3) \le 3.$$

Then, it is clear that

$$R_k = R\left(f_h, f_{\omega_k}, \omega_k\right) = \begin{vmatrix} u_2 & u_1 & u_0 & 0 & 0 \\ 0 & u_2 & u_1 & u_0 & 0 \\ 0 & 0 & u_2 & u_1 & u_0 \\ 1 & 0 & 0 & -u_3 & 0 \\ 0 & 1 & 0 & 0 & -u_3 \end{vmatrix} = \begin{vmatrix} u_0 & u_2u_3 & u_1u_3 \\ u_1 & u_0 & u_2u_3 \\ u_2 & u_1 & u_0 \end{vmatrix}, \tag{8}$$

and so $R_k = u_2^3u_3^2 - 3u_0u_1u_2u_3 + u_1^3u_3 + u_0^3$. It is easy to check that

$$\deg(R_k) \le 3\deg(f_h). \tag{9}$$

Together with Lemma 2, this implies that the degree of f_h in Algorithm 1 after each update is increased over the original f_h by a factor of three. Since the input polynomial f_h of Algorithm 1 is $L_{2r-1,2}\left(x_{2r-1}, y_{2r-1}, z_{2r-1}\right)$, which is of degree 1, the desired result immediately follows from (9) and Lemma 2.

Remark 5. Since the base field of Rescue-Prime is a large finite field, it follows that $\deg(R_k) = 3^k$ with high probability. We have also experimentally verified that $\deg(R_k) = 3^k$ always holds for the parameters used in Rescue-Prime.

Combining Lemmas 2 and 3, it is clear that the output of Algorithm 1 is a univariate polynomial over $\mathbb{F}_q[x_1]$ with a degree at most 3^{3r-3}. In our practical attack against round-reduced Rescue-Prime, the maximum possible degree 3^{3r-3} is always achievable. We note that a similar result has been found by experiments (without proof) in [8, Page 87], which says "In our experiments, the system behaves like a generic system and has $d = 3^{3(r-1)}$ solutions in the algebraic closure of the field."

Complexity Analysis. It can be seen from Algorithm 1 that the time complexity of our attack consists of three parts: (1) the cubic substitutions; (2) the computations of resultants; (3) finding roots of univariate polynomials. The unit of the time complexity is one basic arithmetic operation in the finite field, which is a widely used metric for evaluating complexities for attacking AO primitives.

The Cubic Substitution. For r-round Rescue-Prime, Algorithm 1 involves a total of $3r - 3$ computations of resultants and $3r - 4$ cubic substitutions (we note that some cubic substitutions may involve multiple substitutions as explained in Remark 3). Let ω_k be the variable to be eliminated by the k-th resultant R_k in

Algorithm 1 for $1 \leq k \leq 3(r-1)$. Recall that $R_k \in \mathbb{F}_q[\omega_{k+1}, \omega_{k+2}, \ldots, \omega_{3r-3}, x_1]$ and the purpose of performing cubic substitutions for R_k is to reduce the degrees of $\omega_{k+1}, \omega_{k+2}, \ldots, \omega_{3r-3}$ to less than 3. The time complexity of all cubic substitutions in Algorithm 1 is given in Theorem 1.

Theorem 1. The time complexity of performing all the cubic substitutions in Algorithm 1 is

$$\mathcal{O}\left(\sum_{k=1}^{3r-4} 10 \cdot \left(13^3 \cdot \binom{d_k+3r-k-4}{3r-k-3} \cdot (3^k+1) + 121745\right) \cdot (3r-k-3)\right),$$

where $d_k = \min(3^k - 12, 18r - 6k - 24)$.

Proof. Detailed proof can be found in the full version [29, Theorem 1].

Computation of the Resultant. We have the following Theorem 2 to estimate the time complexities of all the computations of resultants in Algorithm 1.

Theorem 2. The time complexity of the computation of all the resultants in Algorithm 1 is

$$\mathcal{O}\left(\sum_{k=1}^{3r-3} 15 \cdot \max\left((3^k+1) \cdot 7^{3r-k}, (2 \cdot 3^{k-1}+1)(3^{k-1}+1) \cdot 15^{3r-3-k}\right)\right).$$

Proof. Detailed proof can be found in the full version [29, Theorem 2].

Finding Roots of a Univariate Polynomial. Since the output of Algorithm 1 is a univariate polynomial over $\mathbb{F}_q[x_1]$ with degree $d \leq 3^{3r-3}$, all the roots of such a univariate polynomial can be found in $\mathcal{O}(d \log(d)(\log(d) + \log(q)) \log(\log(d)))$, see [8, Sect. 3.1] for details. Once a root of x_1 is found, the corresponding CICO problem is solved.

The time complexities of the different steps of our attack against Rescue-Prime under forward modeling are presented in Table 2.

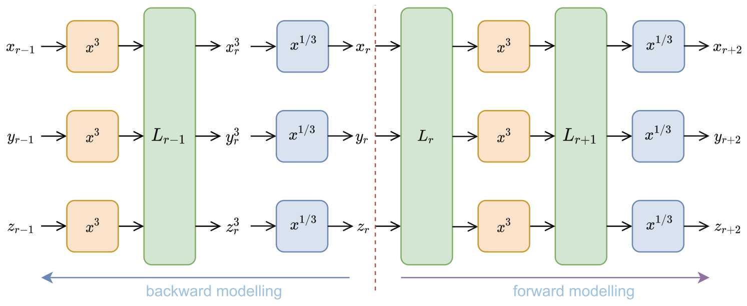
Table 2. Time complexities of our attack against Rescue-Prime under forward modeling, where f_h is the output of Algorithm 1.

r	Complexity of resultants	Complexity of cubic substitutions	$\deg(f_h)$	Complexity of roots solving	Complexity of forward modeling
4	$2^{38.45}$	$2^{40.09}$	3^9	$2^{26.12}$	$\mathbf{2^{40.49}}$
5	$2^{50.17}$	$2^{52.77}$	3^{12}	$2^{31.45}$	$2^{52.99}$
6	$2^{61.90}$	$2^{65.47}$	3^{15}	$2^{36.65}$	$2^{65.58}$
7	$2^{73.62}$	$2^{76.51}$	3^{18}	$2^{41.76}$	$2^{76.69}$
8	$2^{85.34}$	$2^{88.10}$	3^{21}	$2^{46.82}$	$2^{88.30}$

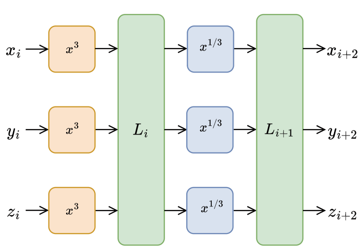
4.2 Algebraic Attack with SFTM Modeling

We use the SFTM (start-from-the-middle) modeling to build a system of equations for r-round (consists of $2r$ steps) Rescue-Prime. The SFTM modeling can be regarded as a reverse application of the meet-in-the-middle (MITM) modeling. The MITM technique is a generic cryptanalytic approach for symmetric-key primitives, which was first introduced by Diffie and Hellman in 1977 [16] for the cryptanalysis of DES. Different from the MITM modeling that starts from the two sides and meets in the middle, the SFTM modeling starts from the middle and models the constraints on the input and output as actual equations.

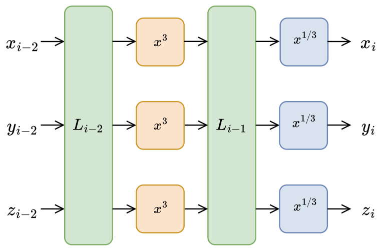
Since the case of $r = 1$ is trivial, we always assume that $r \geq 2$. It is clear that the S-boxes involved in the i-th step are S^{-1} if i is odd and S if i is even. Note that S^{-1} has a very high degree in the forward direction while S has a very high degree in the backward direction. To build low-degree equations, the main idea is to balance the number of S^{-1} layers in the forward direction and the number of S layers in the backward direction. Therefore, the constructed equations are categorized into two cases: one for odd r and the others for even r. We only take the odd r as an illustration and the modeling for even r is presented in the full version [29, Sect. 4.2].



(a) Eq.(11) modeling



(b) Eq.(10) modeling



(c) Eq.(12) modeling

Fig. 5. SFTM modeling for odd r

As shown in Fig. 5a, let x_r, y_r, z_r be the outputs of the three S-boxes (S^{-1}) in the r-th step. Now we will construct equations from the backward direction and the forward direction, respectively.

In the backward direction, for each even i with $0 < i \leq r-1$, let x_i, y_i, z_i be the inputs of the three S-boxes (S) in the i-th step. Then it can be seen from Fig. 5b and Fig. 5a that

$$\begin{cases} x_i^3 = L_{i,0}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \\ y_i^3 = L_{i,1}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \\ z_i^3 = L_{i,2}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \end{cases} \text{for } i \in \{2, 4, \ldots, r-3\}, \tag{10}$$

$$\begin{cases} x_{r-1}^3 = L_{r-1,0}^{-1}(x_r^3, y_r^3, z_r^3) \\ y_{r-1}^3 = L_{r-1,1}^{-1}(x_r^3, y_r^3, z_r^3) \\ z_{r-1}^3 = L_{r-1,2}^{-1}(x_r^3, y_r^3, z_r^3) \end{cases}, \tag{11}$$

where L_i^{-1} is the inverse of the affine transformation L_i, and $L_{i,j}^{-1}$ is the j-th output of L_i^{-1} with $j \in \{0, 1, 2\}$.

Similarly, in the forward direction, for each odd i with $r < i \leq 2r-1$, let x_i, y_i, z_i be the outputs of the three S-boxes (S^{-1}) in the i-th step. Then it can be seen from Fig. 5c that

$$\begin{cases} x_i^3 = L_{i-1,0} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \\ y_i^3 = L_{i-1,1} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \\ z_i^3 = L_{i-1,2} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \end{cases} \text{for } i \in \{r+2, r+4, \ldots, 2r-1\}. \tag{12}$$

The inputs and outputs of r-round Rescue-Prime are of the form $(*, *, 0)$ in the CICO problem, with the same notations defined above, there are two more equations,

$$(C_{-1,2})^3 = L_{0,2}^{-1} \circ SSS \circ L_1^{-1}(x_2, y_2, z_2), \tag{13}$$

$$L_{2r-1,2}\left(x_{2r-1}, y_{2r-1}, z_{2r-1}\right) = 0, \tag{14}$$

where $C_{-1,2}$ is the third constant of the additional AddC operation before the first round of Rescue-Prime.

Solving the System of Equations with the Resultant-Based Method. It can be seen from Eqs. (10)–(14) that the system of equations has $3r-1$ equations ($3r-2$ equations are of degree 3 and one equation is of degree 1) in $3r$ unknowns in total. Since the number of unknowns is one more than that of equations, the system of equations always has solutions. We can then randomly assign a value to one of the variables, say x_r, and solve the remaining variables to speed up the solving process in a practical attack. If the system has no solutions for this assignment, we can repeatedly (usually not too many times) assign another random value to x_r until we can get a solution. Similar to the case in Sect. 4.1, such a system of equations also has a special structure that clearly gives a path for the elimination of variables when solved by resultants. As presented in

Algorithm 2, we will get two bivariate polynomials $f_l, f_h \in \mathbb{F}_q[y_r, z_r]$ and further use them to compute $R(f_l, f_h, z_r)$ to eliminate z_r. When the number of rounds is high, the two polynomials would be quite complicated and directly computing the resultant $R(f_l, f_h, z_r)$ usually suffers from memory overflow. Instead, we can assign a number of values to the variable y_r and get many interpolation pairs, and further use the fast Lagrange interpolation to recover the univariate polynomial. Another benefit of using the fast Lagrange interpolation is that it can be computed in parallel, which can further reduce the attack time if there are enough threads. Once a root of the final univariate polynomial is found, the corresponding CICO problem can be solved by substituting back.

Table 3. The polynomials in SFTM modeling

r **is odd**
$\begin{cases} f_{x_i} = x_i^3 - L_{i,0}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \\ f_{y_i} = y_i^3 - L_{i,1}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \\ f_{z_i} = z_i^3 - L_{i,2}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \end{cases}$ for $i \in \{2, 4, \dots, r-3\}$, $\begin{cases} f_{x_{r-1}} = x_{r-1}^3 - L_{r-1,0}^{-1}(x_r^3, y_r^3, z_r^3) \\ f_{y_{r-1}} = y_{r-1}^3 - L_{r-1,1}^{-1}(x_r^3, y_r^3, z_r^3) \\ f_{z_{r-1}} = z_{r-1}^3 - L_{r-1,2}^{-1}(x_r^3, y_r^3, z_r^3) \end{cases}$, $\begin{cases} f_{x_i} = x_i^3 - L_{i-1,0} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \\ f_{y_i} = y_i^3 - L_{i-1,1} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \\ f_{z_i} = z_i^3 - L_{i-1,2} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \end{cases}$ for $i \in \{r+2, r+4, \dots, 2r-1\}$, $f_l = (C_{-1,2})^3 - L_{0,2}^{-1} \circ SSS \circ L_1^{-1}(x_2, y_2, z_2)$, $f_h = L_{2r-1,2}(x_{2r-1}, y_{2r-1}, z_{2r-1})$.
r **is even**
$\begin{cases} f_{x_i} = x_i^3 - L_{i,0}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \\ f_{y_i} = y_i^3 - L_{i,1}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \\ f_{z_i} = z_i^3 - L_{i,2}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \end{cases}$ for $i \in \{2, 4, \dots, r-2\}$, $\begin{cases} f_{x_{r+1}} = x_{r+1}^3 - L_{r,0}(x_r^3, y_r^3, z_r^3) \\ f_{y_{r+1}} = y_{r+1}^3 - L_{r,1}(x_r^3, y_r^3, z_r^3) \\ f_{z_{r+1}} = z_{r+1}^3 - L_{r,2}(x_r^3, y_r^3, z_r^3) \end{cases}$, $\begin{cases} f_{x_i} = x_i^3 - L_{i-1,0} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \\ f_{y_i} = y_i^3 - L_{i-1,1} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \\ f_{z_i} = z_i^3 - L_{i-1,2} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \end{cases}$ for $i \in \{r+3, r+5, \dots, 2r-1\}$, $f_l = (C_{-1,2})^3 - L_{0,2}^{-1} \circ SSS \circ L_1^{-1}(x_2, y_2, z_2)$, $f_h = L_{2r-1,2}(x_{2r-1}, y_{2r-1}, z_{2r-1})$.

Complexity Analysis. Similar to the case under forward modeling, the time complexity of solving the CICO problem under SFTM modeling consists of four parts: (1) the cubic substitutions in Algorithm 2; (2) the computations of resultants in Algorithm 2; (3) the computation of the final resultant $R(f_l, f_h, z_r)$; (4) finding roots of univariate polynomials.

It can be seen from Algorithm 2 that, for r-round Rescue-Prime, there are at most $3 \cdot \lfloor r/2 \rfloor$ variables to be eliminated in the first (lines 2–10) or the second (lines 12–20) loop. Similar to the case under forward modeling, the degree of f_l (resp. f_h) in Algorithm 2 after each update is increased over the original f_l (resp. f_h) by a factor of three. The time complexities of the first two parts are given in Theorems 3 and 4. As the deduction and proof are pretty similar to those for Theorems 1 and 2, we omit the proof here.

Algorithm 2: Get two bivariate polynomials for r-round Rescue-Prime

```
Input: $f_l, f_h, f_{x_i}, f_{y_i}, f_{z_i}$ as defined in Table 3.
Output: two bivariate polynomials.
1  $i \leftarrow 2r - 1$;
2  while $i > r$ do
3      $f_h \leftarrow R(f_h, f_{z_i}, z_i)$;
4      apply the cubic substitution to $f_h$;
5      $f_h \leftarrow R(f_h, f_{y_i}, y_i)$;
6      apply the cubic substitution to $f_h$;
7      $f_h \leftarrow R(f_h, f_{x_i}, x_i)$;
8      apply the cubic substitution to $f_h$;
9      $i \leftarrow i - 2$;
10 end
11 $i \leftarrow 2$;
12 while $i < r$ do
13     $f_l \leftarrow R(f_l, f_{z_i}, z_i)$;
14     apply the cubic substitution to $f_l$;
15     $f_l \leftarrow R(f_l, f_{y_i}, y_i)$;
16     apply the cubic substitution to $f_l$;
17     $f_l \leftarrow R(f_l, f_{x_i}, x_i)$;
18     apply the cubic substitution to $f_l$;
19     $i \leftarrow i + 2$;
20 end
21 return $f_h, f_l$.
```

Theorem 3. *The time complexity of performing all the cubic substitutions in Algorithm 2 is*

$$\mathcal{O}\left(\sum_{k=1}^{\lambda} 20 \cdot \left(13^3 \cdot \binom{d_k+\lambda-k-1}{\lambda-k} \cdot (3^k+1)^2 + 121745\right) \cdot (\lambda - k)\right),$$

where $\lambda = 3 \cdot \lfloor r/2 \rfloor$ *and* $d_k = \min\left(3^k - 12, 6 \cdot (\lambda - k - 1)\right)$.

Theorem 4. The time complexity of the computation of all the resultants in Algorithm 2 is

$$\mathcal{O}\left(\sum_{k=1}^{\lambda} 30 \cdot \max\left((3^k+1)^2 \cdot 7^{\lambda-k+3}, (2 \cdot 3^{k-1}+1)^2(3^{k-1}+1)^2 \cdot 15^{\lambda-k}\right)\right),$$

where $\lambda = 3 \cdot \lfloor r/2 \rfloor$.

For the third part of computing the final resultant $R(f_l, f_h, z_r)$, the situation is slightly different. Theorem 5 gives the time complexity of this part with a proof.

Theorem 5. Let f_l and f_h be the output polynomials of Algorithm 2 of which the degrees are denoted as d_l and d_h, respectively. Then the time complexity of computing the resultant $R(f_l, f_h, z_r)$ is

$$\mathcal{O}\left(d_l d_h \cdot (\mathcal{T}_1 + \mathcal{T}_2) + \mathcal{T}_3\right),$$

where $\mathcal{T}_1 = \left(d_l^2 \log(d_l) + d_h^2 \log(d_h)\right)$, $\mathcal{T}_2 = (d_l + d_h)^\omega$, $\mathcal{T}_3 = (d_l d_h \log(d_l d_h))$, and ω is the linear algebra exponent[2].

Proof. Detailed proof can be found in the full version [29, Theorem 5].

The time complexities of the different steps of our attack against Rescue-Prime under SFTM modeling are presented in Table 4.

Table 4. Time complexities of algebraic attacks under SFTM modeling against Rescue-Prime and the degrees of f_l, f_h, and f.

r	Complexity of resultants	Complexity of cubic substitutions	d_l	d_h	$\deg(f)$	Complexity of SFTM attacks
4	$2^{38.94}$	$2^{35.62}$	3^4	3^6	3^{10}	$\mathbf{2^{40.31}}$
5	$2^{38.94}$	$2^{35.62}$	3^7	3^6	3^{13}	$\mathbf{2^{48.37}}$
6	$2^{57.92}$	$2^{47.84}$	3^7	3^9	3^{16}	$2^{59.96}$
7	$2^{57.92}$	$2^{47.84}$	3^{10}	3^9	3^{19}	$2^{68.95}$
8	$2^{76.94}$	$2^{60.77}$	3^{10}	3^{12}	3^{22}	$2^{80.57}$

[2] A result of Coppersmith and Winograd [15] yields $\omega = 2.376$, which is asymptotic since it involves extremely large constant overheads. From a practical point-of-view, the best achievable result for ω is given by Strassen's algorithm [27], where $\omega = 2.807$.

4.3 Summary of the Resultant-Based Method

Now we give a summary about the proposed algebraic attack on AO primitives, which consists of four ingredients with the following four steps.

1. Construct a system of equations using forward modeling or SFTM modeling.
2. Combine the resultant and the substitution theory to eliminate variables in a specific order and finally get two bivariate polynomials.
3. When the number of rounds is small, one can directly compute the resultant of the last two bivariate polynomials to derive the ultimate univariate polynomial. While when the number of rounds is high, the resultant of the last two bivariate polynomials is usually hard to compute directly. Instead, one can assign several values to one of the two variables and get many interpolation pairs. Then, the fast Lagrange interpolation is used to recover the univariate polynomial.
4. Find all the roots of the derived univariate polynomial and then substitute the root values back to the original system of equations to find the collision of the algorithm.

4.4 Experimental Results

In this section, we present the experimental results of our algebraic attack on Rescue-Prime. The experiments were performed on a workstation: the operating system is Windows 10, the CPU circuit is Intel(R) Xeon(R) Gold 6248R CPU 3.00 GHz with 48 cores, and the maximum memory is 256G. We use SageMath 9.2 to construct equations for Rescue-Prime and use Maple 2023 to solve the system of equations. Finally, we use the "FpX_halfgcd" command of PARI/GP to find all the roots of a univariate polynomial.

Table 5. Attack complexities of Rescue-Prime

r	Ethereum Foundation's time complexity	Best theoretical complexity	Time complexity of SFTM	Time complexity forward modeling	Best practical time in [8]	Practical time of SFTM	Practical time of forward modeling
4	$2^{37.5}$	2^{43}	$2^{43.02}$	$2^{40.49}$	258500s	2256.7s	885.5s
5	2^{45}	2^{57}	$2^{52.93}$	$2^{52.99}$	-	$\approx$ one day	-

As mentioned, we mainly focus on the instance of $t = 3$ and the total number of rounds in this case is 11, which is derived using the code in [28, page 9, Algorithm 7]. We mainly compare our results to the benchmark results in [8], so we also use the challenge parameters published by the Ethereum foundation with

$$p = 18446744073709551557 = 2^{64} - 59.$$

We succeed to solve the CICO problem of 4-round Rescue-Prime, in which two models (forward modeling and SFTM modeling) are considered to construct the equations. Practical attacks under SFTM modeling and forward modeling take 2256.7 s and 885.5 s, respectively, introducing 100-fold improvement over the results in [8]. We also find a 5-round collision of Rescue-Prime under SFTM modeling which was originally thought as "hard" in the Ethereum Foundation challenge. For 5-round Rescue-Prime, the system of equations constructed under SFTM modeling has 14 equations in total. The 14 polynomials involved in the system are f_l, f_{x_2}, f_{y_2}, f_{z_2},f_{x_4}, f_{y_4}, f_{z_4}, f_{x_7}, f_{y_7}, f_{z_7}, f_{x_9}, f_{y_9}, f_{z_9}, and f_h. Using Algorithm 2, we get two bivariate polynomials f_l and f_h, of which the degrees are 3^7 and 3^6, respectively. Since the memory cost of computing $R(f_l, f_h, z_5)$ is too high, we use 16 threads to compute Lagrange interpolation points. Each thread computes 100,000 points, which takes an average of 70,000 s. We use 32 threads and combine a fast multi-point evaluation algorithm to finish the precomputation, which takes 9385.285 s. Recovering the univariate polynomial using the fast Lagrange interpolation takes 2486.71 s. It takes less than 10 s to solve the final single-variable equation of degree 3^{13}. Experimental results are shown in Table 5.

5 Application to `Anemoi`

We now apply our methods to a new class of AO primitives `Anemoi` [11], and provide better cryptanalysis results than existing ones.

5.1 Design Description of `Anemoi`

`Anemoi` is a new family of ZK-friendly permutations that works over $\mathbb{F}_q^{2l} (l \geq 1)$, where q is either a prime number or $q = 2^n$ with n being an odd positive integer. Different choices of parameters would affect how `Anemoi` works, and we mainly focus on the version of $l = 1$ and q being a prime number p. The original paper [11] gives two hash function instances based on `Anemoi` with $l = 1$: `AnemoiSponge`-BN-254, with a 254-bit prime p, and `AnemoiSponge`-BLS12-381, with a 381-bit prime p. Both instances are claimed to achieve 127 bits of security.

The round function of `Anemoi` has the structure of a classical substitution-permutation network, which consists of three components: the constant addition $\mathcal{A}$, the linear layer $\mathcal{M}$, and the nonlinear layer $\mathcal{H}$. The linear layer includes a diffusion layer and a pseudo-Hadamard transform, while for the version we considered ($l = 1$), there is a unique column in the internal state, and the diffusion layer can be removed. For given q, number of rounds r, and $l = 1$, the `Anemoi` permutation over $\mathbb{F}_q^2$ is described as

$$\texttt{Anemoi} = \mathcal{M} \circ R_{r-1} \circ \cdots \circ R_0,$$

where $R_i = \mathcal{H} \circ \mathcal{M} \circ \mathcal{A}_i$ for $0 \leq i \leq r-1$. The i-th round of `Anemoi` is illustrated in Fig. 6 and we below give more details about $\mathcal{A}_i, \mathcal{M}$, and $\mathcal{H}$.

1. *Constant Additions* $\mathcal{A}_i$. The operation adds round constants (c_i, d_i) to the input vector (x_i, y_i) of the i-th round.
2. *Linear Layer* $\mathcal{M}$: The Pseudo-Hadamard transform is applied to destroy some undesirable involutive patterns in the nonlinear layer, which is defined as $\mathcal{M}(x, y) = (2x + y, x + y)$.
3. *Nonlinear Layer* $\mathcal{H}$. The schematic of $\mathcal{H}$ is illustrated in Fig. 7. Let $u_i, v_i \in \mathbb{F}_q$ and $x_{i+1}, y_{i+1} \in \mathbb{F}_q$ be the inputs and outputs of $\mathcal{H}$, respectively. Then the nonlinear layer $\mathcal{H}$ can be expressed as

$$\mathcal{H}(u_i, v_i) = (u_i + gz_i^2 - 2gv_iz_i - g^{-1}, v_i - z_i), \tag{15}$$

where g is a generator of the multiplicative subgroup of the field $\mathbb{F}_q$ and z_i is an intermediate variable output of the operation $x^{1/\alpha}$ (α usually takes values $3, 5, 7$, or 11 if q is an odd prime number).

The CICO problem of `Anemoi`, which is denoted by $\mathcal{P}_{\text{CICO}}$ in [12], consisting of finding $(y_{in}, y_{out}) \in \mathbb{F}_q^2$ such that $\texttt{Anemoi}(0, y_{in}) = (0, y_{out})$.

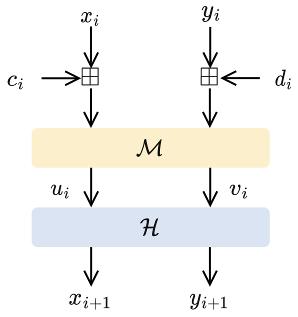


Fig. 6. Illustration of the i-th round of `Anemoi`

5.2 SFTM Attack Against `Anemoi`

We mainly focus on the `Anemoi` instance of $\alpha = 3$ with r rounds. Let the notations $x_i, y_i, c_i, d_i, u_i, v_i, z_i, x_{i+1}, y_{i+1}$ be as in Sect. 5.1 for $0 \le i \le r - 1$, then

$$(u_i, v_i) = \mathcal{M} \circ \mathcal{A}_i(x_i, y_i) = (2x_i + y_i + 2c_i + d_i, x_i + y_i + c_i + d_i).$$

Since $(x_{i+1}, y_{i+1}) = (u_i + gz_i^2 - 2gv_iz_i - g^{-1}, v_i - z_i)$, a simple computation yields

$$\begin{cases} x_{i+1} = 2x_i + y_i + gz_i^2 - 2gz_i \cdot (x_i + y_i + c_i + d_i) + \delta_i \\ y_{i+1} = x_i + y_i - z_i + c_i + d_i \\ z_i^3 = 2x_i + y_i - g \cdot (x_i + y_i + c_i + d_i)^2 + \delta_i \end{cases}, \tag{16}$$

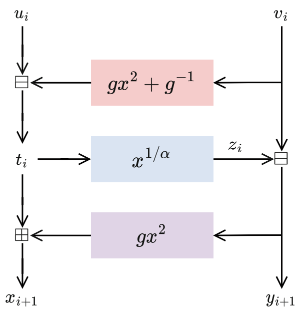


Fig. 7. The $\mathcal{H}$ function of `Anemoi` (q is odd).

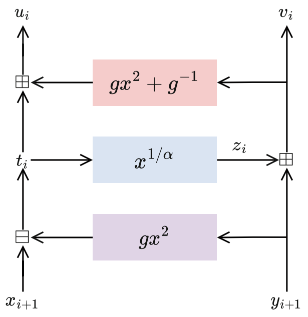


Fig. 8. The $\mathcal{H}^{-1}$ function of `Anemoi` (q is odd)

where $\delta_i = 2c_i + d_i - g^{-1}$ is a constant. Since the functions $\mathcal{A}_i$, $\mathcal{M}$, and $\mathcal{H}$ (the schematic of $\mathcal{H}^{-1}$ is illustrated in Fig. 8) in `Anemoi` are all invertible, with a similar discussion as above, we can get that

$$\begin{cases} x_i = x_{i+1} + gz_i^2 + 2gz_iy_{i+1} + g^{-1} - z_i - y_{i+1} - c_i \\ y_i = 2z_i + 2y_{i+1} - x_{i+1} - gz_i^2 - 2gz_iy_{i+1} - g^{-1} - d_i \, . \\ z_i^3 = x_{i+1} - gy_{i+1}^2 \end{cases} \tag{17}$$

Construction of Equations. Inspired by the analysis of Rescue-Prime, we still use the SFTM technique to construct the equations of the CICO problem of `Anemoi`. Due to the $x \to x^{\frac{1}{3}}$ operation in $\mathcal{H}$, the cubic substitution method is also applicable. For r-round Anemoi, we set intermediate variables $x_{\lfloor r/2 \rfloor} = x, y_{\lfloor r/2 \rfloor} = y$ as illustrated in Fig. 9. If $i > \lfloor r/2 \rfloor$, then by repeatedly using (16), those variables x_i, y_i, z_i can be repeatedly expressed by intermediate variables of the lower rounds, and in the end can be expressed in variables $x, y, z_{\lfloor r/2 \rfloor}, z_{\lfloor r/2 \rfloor+1}, \ldots, z_{r-1}$. On the other hand, if $i < \lfloor r/2 \rfloor$, then by repeatedly using (17), those variables x_i, y_i, z_i can be repeatedly expressed by intermediate variables of the higher rounds, and in the end can be expressed in variables $x, y, z_0, z_1, \ldots, z_{\lfloor r/2 \rfloor-1}$.

To solve the CICO problem of `Anemoi`, we need to find $(y_{in}, y_{out}) \in \mathbb{F}_q^2$ such that $\texttt{Anemoi}(0, y_{in}) = (0, y_{out})$, which immediately follows that $x_0 = 0$. Since

$$\texttt{Anemoi}(0, y_{in}) = \mathcal{M}(x_r, y_r) = (2x_r + y_r, x_r + y_r),$$

it follows that $2x_r + y_r = 0$. The above discussion implies that x_0 and $2x_r + y_r$ can be expressed in variables $x, y, z_0, z_1, \ldots, z_{\lfloor r/2 \rfloor-1}$ and $x, y, z_{\lfloor r/2 \rfloor}, z_{\lfloor r/2 \rfloor+1}, \ldots, z_{r-1}$, respectively. For convenience, let us denote

$$x_0 \triangleq f_l(x, y, z_0, z_1, \ldots, z_{\lfloor r/2 \rfloor-1}),$$
$$2x_r + y_r \triangleq f_h(x, y, z_{\lfloor r/2 \rfloor}, z_{\lfloor r/2 \rfloor+1}, \ldots, z_{r-1}).$$

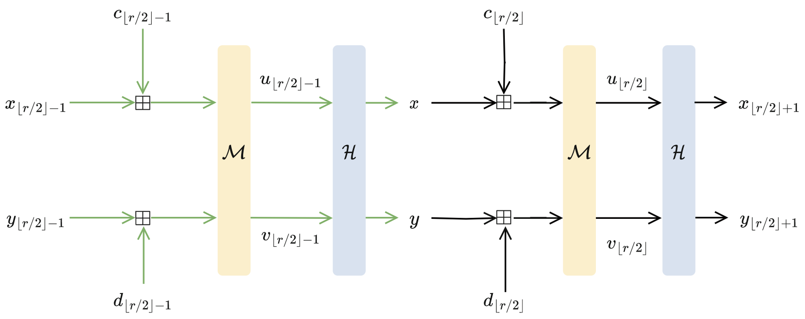

$c_{\lfloor r/2\rfloor-1}$
$x_{\lfloor r/2\rfloor-1}$
$y_{\lfloor r/2\rfloor-1}$
$d_{\lfloor r/2\rfloor-1}$
$\mathcal{M}$
$u_{\lfloor r/2\rfloor-1}$
$v_{\lfloor r/2\rfloor-1}$
$\mathcal{H}$
x
y
$c_{\lfloor r/2\rfloor}$
$d_{\lfloor r/2\rfloor}$
$\mathcal{M}$
$u_{\lfloor r/2\rfloor}$
$v_{\lfloor r/2\rfloor}$
$\mathcal{H}$
$x_{\lfloor r/2\rfloor+1}$
$y_{\lfloor r/2\rfloor+1}$

Fig. 9. SFTM modeling for `Anemoi`

Similarly, let us denote

$$z_i^3 - x_{i+1} + gy_{i+1}^2 \triangleq g_i(x, y, z_i, z_{i+1}, \ldots, z_{\lfloor r/2\rfloor-1})$$

for $0 \le i \le \lfloor r/2\rfloor - 1$, and

$$z_i^3 - (2x_i + y_i - g\cdot(x_i + y_i + c_i + d_i)^2 + \delta_i) \triangleq g_i(x, y, z_{\lfloor r/2\rfloor}, z_{\lfloor r/2\rfloor+1}, \ldots, z_i)$$

for $\lfloor r/2\rfloor < i \le r-1$. Then the CICO problem of `Anemoi` can be modeled with the following system of equations

$$\begin{cases} f_l(x, y, z_0, z_1, \ldots, z_{\lfloor r/2\rfloor-1}) = 0 \\ g_0(x, y, z_0, z_1, \ldots, z_{\lfloor r/2\rfloor-1}) = 0 \\ g_1(x, y, z_1, z_2, \ldots, z_{\lfloor r/2\rfloor-1}) = 0 \\ \quad\cdots \\ g_{\lfloor r/2\rfloor-1}(x, y, z_{\lfloor r/2\rfloor-1}) = 0 \\ g_{\lfloor r/2\rfloor}(x, y, z_{\lfloor r/2\rfloor}) = 0 \\ \quad\cdots \\ g_{r-2}(x, y, z_{\lfloor r/2\rfloor}, z_{\lfloor r/2\rfloor+1}, \ldots, z_{r-2}) = 0 \\ g_{r-1}(x, y, z_{\lfloor r/2\rfloor}, z_{\lfloor r/2\rfloor+1}, \ldots, z_{r-1}) = 0 \\ f_h(x, y, z_{\lfloor r/2\rfloor}, z_{\lfloor r/2\rfloor+1}, \ldots, z_{r-1}) = 0 \end{cases}, \tag{18}$$

which has $r+2$ equations with $r+2$ unknowns in total.

Solving the System of Equations with Resultant-Based Method. Equations in Eq. (18) indicate a path for eliminating the intermediate variables by the resultant. After computing a series of resultants, we can get two bivariate polynomials f_l and f_h only in variables x and y in the end. Then, we compute the roots of the univariate polynomial $R(f_l, f_h, y)$, and the CICO problem is solved.

Complexity Analysis. As computing $R(f_l, f_h, y)$ takes far more time than the other resultants, we take its time complexity as the main time complexity.

Table 6. Time complexities of our attack against **Anemoi** using SFTM modeling and the degrees of f_l, f_h, and f.

r	Number of equations	Highest degree of f_l	Highest degree of f_h	Degree of a single f	Time complexity
3	5	5^1	5^2	5^3	$2^{19.56}$
4	6	5^2	5^2	5^4	$2^{23.32}$
5	7	5^2	5^3	5^5	$2^{29.60}$
6	8	5^3	5^3	5^6	$2^{33.38}$
7	9	5^3	5^4	5^7	$2^{39.58}$
8	10	5^4	5^4	5^8	$2^{43.42}$
9	11	5^4	5^5	5^9	$2^{49.57}$
10	12	5^5	5^5	5^{10}	$2^{53.46}$
11	13	5^5	5^6	5^{11}	$2^{59.59}$
12	14	5^6	5^6	5^{12}	$2^{63.53}$
13	15	5^6	5^7	5^{13}	$2^{69.64}$
14	16	5^7	5^7	5^{14}	$2^{73.63}$
15	17	5^7	5^8	5^{15}	$2^{79.72}$
16	18	5^8	5^8	5^{16}	$2^{83.74}$
17	19	5^8	5^9	5^{17}	$2^{89.83}$
18	20	5^9	5^9	5^{18}	$2^{93.87}$
19	21	5^9	5^{10}	5^{19}	$2^{99.96}$
20	22	5^{10}	5^{10}	5^{20}	$2^{104.01}$
21	23	5^{10}	5^{11}	5^{21}	$2^{110.10}$

Authors in [13] proved that the degree of the ideal induced by $\mathcal{P}_{\text{CICO}}$ of r-round **Anemoi** is $(\alpha+2)^r$. For the case $\alpha = 3$, we have a similar observation that $\deg(f_l) = 5^{\lfloor r/2 \rfloor}$ and $\deg(f_h) = 5^{\lceil r/2 \rceil}$, which is also confirmed by our experiments. By Theorem 5, the time complexities of our attacks are presented in Table 6.

5.3 Experimental Results for Anemoi

The experiment environment for **Anemoi** is exactly the same to that for Rescue-Prime. We use the same prime number $p = $ 0x64ec6dd0392073 as that in [7]. For 8-round Anemoi, the system of equations constructed under SFTM modeling in total has 10 equations in 10 unknowns. The 10 polynomials corresponding to the system are $f_l, g_0, \ldots, g_7, f_h$. We compute the resultants of f_l with g_0, g_1, g_2, g_3 in turn and f_h with g_7, g_6, g_5, g_4 in turn. After the computations of above resultants, $\deg(f_l) = \deg(f_h) = 3^4$, and then the univariate polynomial obtained by the final resultant $R(f_l, f_h, y)$ is of degree 5^8. We use four threads to compute Lagrange interpolation points, each of which computes 100,000 points. The running time of the four threads is 29642.521 s, 27547.542 s, 27853.072 s, and 27785.910 s, respectively. We use eight threads and combine a fast multi-point evaluation algorithm to finish the precomputation, which takes 9535.405 s. Recovering the univariate polynomial using the fast Lagrange interpolation takes 1006.516 s. The practical attack time is presented in Table 7, which greatly improves the running time compared to that in [7].

Table 7. Comparison with [7] in practical attack time of `Anemoi`

r	The attacks in [7]	Our attacks
3	$< 0.04s$	0.423s
4	0.58s	0.973s
5	30.97s	7.113s
6	2421.52s	296.568s
7	167201s	2968.55s
8	–	38749.182s

6 Application to Jarvis

In this section, we apply our algebraic attack on the block cipher Jarvis, which is one member of the MARVELlous family of cryptographic primitives that are specifically designed for STARK efficiency [5]. Detailed descriptions of Jarvis and the attack in [2] can be found in the full version [29, Sect. 6] and we only give the attack results here.

The experiment environment for Jarvis is the same to that for Rescue-Prime, except that we use Magma V2.28-3 to solve the system of equations. We use the same finite field $\mathbb{F}_{2^{128}} = \mathbb{F}_2[y]/(p(y))$ of Jarvis-128 as defined in [5], where $p(y) = y^{128} + y^7 + y^2 + y + 1$. We construct the equations using the method given in [5] but use the resultant-based method instead of the Gröbner basis method to solve the system of equations. For the practical attack on six rounds

Table 8. Comparison with [5] in practical attack time of Jarvis

r	Time for other resultants	Time for the final resultant	Total practical time	Time in [5]
6	11.2s	357.76s	368.96s	99989.0s
8	606.76s	455043.77s	455650.53s	–

of Jarvis, a 100-fold increase in the running time is achieved over that in [2]. We also practically attack eight rounds of Jarvis for the first time, which consumes about 5.27 days (the memory consumption is about 68 GB). The comparison of the running time with that in [5] is presented in Table 8.

7 Conclusions and Discussions

This paper presents a novel analysis framework of algebraic attacks against AO primitives that we think can serve as a new evaluation method. We make full use of the algebraic properties of AO primitives and propose to use resultants to solve systems of multivariate equations. We further use SFTM modeling, variable substitutions, and the fast Lagrange interpolation to simplify the derived multivariate system and accelerate the solving procedure. We apply the analysis framework to analyze the security of Rescue-Prime, `Anemoi`, and Jarvis, and achieve much faster practical attacks than existing ones for all the three primitives. Besides, the estimation of time complexity is more accurate than that in Gröbner basis attacks as the degrees of variables can be estimated more accurately and the elimination path for variables is definite. Based on our analysis and experiments, we have the following discussions that may deserve some attention.

7.1 Why SFTM Modeling Is Better Than Forward Modeling

We take the 4-round Rescue-Prime as an example. In SFTM modeling, the resultant of the final two bivariate polynomials takes up most of the time (about 99.8%), but in forward modeling, the time spent for the cubic substitutions is the major cost (about 76.4%). In forward modeling, there are a total of 10 variables in the system of equations, and to get a univariate polynomial, 9 resultant computations and 36 cubic substitutions are required. However, each operation involves all the uneliminated variables that will increase memory and time consumption. It is the memory overflow problem that hinders higher-round attacks against Rescue-Prime under forward modeling even if it can bypass one round with no cost. Under SFTM modeling, it takes 3 resultant computations and 3 cubic substitutions to get f_l, 6 resultant computations and 15 cubic substitutions to get f_h. Each operation involves fewer variables than that under the forward modeling. For the most difficult part to compute $R(f_l, f_h, z_4)$, it can be parallelized using the fast Lagrange interpolation. Therefore, memory and time consumption are reduced, making practical higher-round attacks possible.

7.2 Why not Combine SFTM Modeling with First-Round Bypassing

We now show how to do it if we want to combine SFTM modeling with the idea of bypassing the first round. We use the notations defined in Sect. 3 and take the SFTM modeling for 4-round Rescue-Prime as an example, in which case we have 12 variables $x_i, y_i, z_i,\ i \in [2, 4, 5, 7]$. As shown in Fig. 3 and Eq. 4, to bypass the first round, the output of the $x^{1/3}$ operations, denoted as (X, Y, Z), should have the following form,

$$(X, Y, Z) = (X, (-\frac{\alpha_{2,0}}{\alpha_{2,1}})^{1/3}X, c). \tag{19}$$

The variables X, Y, Z also denote the input of the L_1 operation, i.e., $X = L^{-1}_{1,0}(x_2, y_2, z_2), Y = L^{-1}_{1,1}(x_2, y_2, z_2)$, and $Z = L^{-1}_{1,2}(x_2, y_2, z_2)$. Then we can get two polynomials f_{l1} and f_{l2} defined as

$$f_{l1} = L^{-1}_{1,1}(x_2, y_2, z_2)/L^{-1}_{1,0}(x_2, y_2, z_2) - (-\frac{\alpha_{2,0}}{\alpha_{2,1}})^{1/3},$$

$$f_{l2} = L^{-1}_{1,2}(x_2, y_2, z_2) - c.$$

To get the final univariate polynomial, we need to first get three polynomials f_h, f_{l1}, f_{l2} in variables x_4, y_4, z_4. For f_h, we repeatedly use resultants to eliminate $z_7, y_7, x_7,\ z_5, y_5, x_5$ in order and f_h turns into a polynomial in the three variables. Each resultant computation will eliminate one variable. While for f_{l1} and f_{l2}, we need to first compute $R(f_{l1}, f_{z_2}, z_2)$ and $R(f_{l2}, f_{z_2}, z_2)$ to eliminate z_2 (one can also eliminate y_2 or x_2 instead), which takes two resultant computations. Therefore, the first disadvantage of combining the two methods is the increase in resultant computations.

Now, we target to get the univariate polynomial based on f_h, f_{l1}, and f_{l2}. In the original SFTM modeling, we have two bivariate polynomials and use fast Lagrange interpolation to get the univariate polynomial. But here, we need to perform one more resultant computation to eliminate one more variable of x_4, y_4, z_4 to get the bivariate polynomials. What is worse, the degrees of x_4, y_4, z_4 are all high and the resultant computation involves high overhead, and this gives the second reason for not combining the SFTM modeling with the idea of bypassing the first round.

7.3 Discussion on Our Attack

We present some potential weaknesses of AO primitives that are susceptible to our attacks and give some potential improvements.

1. The time complexities of different steps in SFTM modeling are not balanced. For example, in our resultant-based attack, the most challenging part of attacking the algorithm is to compute the final resultant. However, the fast Lagrange interpolation can parallelize this step, and the practical result can be better than the theoretical estimation if there are enough threads in parallel.
2. The variable isolation and substitution are very critical techniques in our attack, AO primitives that do not suffer from such problems can potentially resist our attack. Using nonlinear structures that do not result in variable isolation and substitution, e.g., Toffoli-like gates [17], big S-boxes over extension finite fields [6] and the unified structure [24], may be a feasible design idea that deserves further investigation.

3. The round constant add operation $+$ can be replaced by $\oplus$, which will add difficulties in establishing and solving equations.

Acknowledgements. We would like to thank the anonymous reviewers for their positive reviews, valuable comments and questions. We would like to thank Topsec Technologies Group Inc for organizing the National Crypto-Math Challenge and Jiamin Cui from Shandong University for bringing us into the field of Arithmetic-Oriented algorithms. Thanks also go to Zhongxiao Wang and Ruijie Wang for their inspiration of interpolation ideas. This work was supported by the National Natural Science Foundation of China (Nos. 12371526, 62202492, and 62272303) and by the National Key Research and Development Program of China (No. 2024YFB4504700).

References

1. Albrecht, M., Grassi, L., Rechberger, C., Roy, A., Tiessen, T.: Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 191–219. Springer (2016)
2. Albrecht, M.R., Cid, C., Grassi, L., Khovratovich, D., Lüftenegger, R., Rechberger, C., Schofnegger, M.: Algebraic cryptanalysis of stark-friendly designs: application to marvellous and mimc. In: Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part III 25. pp. 371–397. Springer (2019)
3. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for mpc and fhe. In: Advances in Cryptology–EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I 34. pp. 430–454. Springer (2015)
4. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of symmetric-key primitives for advanced cryptographic protocols. IACR Transactions on Symmetric Cryptology pp. 1–45 (2020)
5. Ashur, T., Dhooghe, S.: Marvellous: a stark-friendly family of cryptographic primitives. Cryptology ePrint Archive (2018)
6. Ashur, T., Kindi, A., Mahzoun, M., Bhati, A.S.: Xhash8 and xhash12: Efficient stark-friendly hash functions. Cryptology ePrint Archive (2023)
7. Bariant, A., Boeuf, A., Lemoine, A., Ayala, I.M., Øygarden, M., Perrin, L., Raddum, H.: The algebraic freelunch efficient gröbner basis attacks against arithmetization-oriented primitives. Annual International Cryptology Conference, CRYPTO 2024, accepted (2024), https://eprint.iacr.org/2024/347, https://crypto.iacr.org/2024/acceptedpapers.php
8. Bariant, A., Bouvier, C., Leurent, G., Perrin, L.: Algebraic attacks against some arithmetization-oriented primitives. IACR Transactions on Symmetric Cryptology pp. 73–101 (2022)
9. Ben-Sasson, E., Goldberg, L., Levit, D.: Stark friendly hash–survey and recommendation. Cryptology ePrint Archive (2020)

10. Beyne, T., Canteaut, A., Dinur, I., Eichlseder, M., Leander, G., Leurent, G., Naya-Plasencia, M., Perrin, L., Sasaki, Y., Todo, Y., et al.: Out of oddity–new cryptanalytic techniques against symmetric primitives optimized for integrity proof systems. In: Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part III 40. pp. 299–328. Springer (2020)
11. Bouvier, C., Briaud, P., Chaidos, P., Perrin, L., Salen, R., Velichkov, V., Willems, D.: New design techniques for efficient arithmetization-oriented hash functions: anemoi permutations and jive compression mode. In: Annual International Cryptology Conference. pp. 507–539. Springer (2023)
12. Bouvier, C., Briaud, P., Chaidos, P., Perrin, L., Salen, R., Velichkov, V., Willems, D.: New design techniques for efficient arithmetization-oriented hash functions:anemoi permutations and jive compression mode. Cryptology ePrint Archive, Paper 2022/840 (2022), https://eprint.iacr.org/2022/840
13. Briaud, P.: A note of `Anemoi` gröbner bases. Cryptology ePrint Archive (2024)
14. Collins, G.E.: The calculation of multivariate polynomial resultants. Journal of the Acm **18**(4), 515–532 (1971)
15. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. In: Proceedings of the nineteenth annual ACM symposium on Theory of computing. pp. 1–6 (1987)
16. Diffie, W., Hellman, M.: Special feature exhaustive cryptanalysis of the nbs data encryption standard. Computer **10**(6), 74–84 (1977). https://doi.org/10.1109/C-M.1977.217750
17. Dobraunig, C., Grassi, L., Guinet, A., Kuijsters, D.: Ciminion: symmetric encryption based on toffoli-gates over large finite fields. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 3–34. Springer (2021)
18. Dworkin, M.J.: Sha-3 standard: Permutation-based hash and extendable-output functions (2015)
19. von zur Gathen, J., Gerhard, J.: Modern computer algebra. Cambridge: Cambridge University Press, 2nd ed. edn. (2003)
20. Grassi, L., Hao, Y., Rechberger, C., Schofnegger, M., Walch, R., Wang, Q.: Horst meets fluid-spn: Griffin for zero-knowledge applications. In: Annual International Cryptology Conference. pp. 573–606. Springer (2023)
21. Grassi, L., Khovratovich, D., Lüftenegger, R., Rechberger, C., Schofnegger, M., Walch, R.: Reinforced concrete: a fast hash function for verifiable computation. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 1323–1335 (2022)
22. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for {Zero-Knowledge} proof systems. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 519–535 (2021)
23. Grassi, L., Khovratovich, D., Schofnegger, M.: Poseidon2: A faster version of the poseidon hash function. In: International Conference on Cryptology in Africa. pp. 177–203. Springer (2023)
24. Liu, J., Sun, B., Liu, G., Dong, X., Liu, L., Zhang, H., Li, C.: New wine old bottles: Feistel structure revised. IEEE Transactions on Information Theory **69**(3), 2000–2008 (2023https://doi.org/10.1109/TIT.2022.3223139
25. Rijmen, V., Daemen, J.: Advanced encryption standard. Proceedings of federal information processing standards publications, national institute of standards and technology **19**, 22 (2001)

26. Roy, A., Steiner, M.J., Trevisani, S.: Arion: Arithmetization-oriented permutation and hashing from generalized triangular dynamical systems (2023)
27. Strassen, V.: Gaussian elimination is not optimal. Numerische mathematik **13**(4), 354–356 (1969)
28. Szepieniec, A., Ashur, T., Dhooghe, S.: Rescue-prime: a standard specification (sok). Cryptology ePrint Archive, Paper 2020/1143 (2020), https://eprint.iacr.org/2020/1143
29. Yang, H.S., Zheng, Q.X., Yang, J., feng Liu, Q., Tang, D.: A new security evaluation method based on resultant for arithmetic-oriented algorithms. Cryptology ePrint Archive, Paper 2024/886 (2024), https://eprint.iacr.org/2024/886
30. Zhang, L., Liu, M., Li, S., Lin, D.: Cryptanalysis of ciminion. In: International Conference on Information Security and Cryptology. pp. 234–251. Springer (2022)

Author Index

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15490, pp. 491–492, 2025.
https://doi.org/10.1007/978-981-96-0941-3