

Privacy-Enhancing Techniques in Distributed Systems

Mahdi Sedaghat

Supervisor:
Prof. dr. ir. Bart Preneel

Dissertation presented in partial
fulfillment of the requirements for the
degree of Doctor of Engineering
Science (PhD): Electrical Engineering

June 2024

Privacy-Enhancing Techniques in Distributed Systems

Mahdi SEDAGHAT

Examination committee:

Prof. dr. ir. Omer Van der Biest, chair

Prof. dr. ir. Bart Preneel

Prof. dr. Nigel Smart

Prof. dr. ir. Wouter Joosen

Dr. Svetla Petkova-Nikova

Dr. Aysajan Abidin

Prof. dr. Tom Van Cutsem

Prof. dr. Markulf Kohlweiss

(University of Edinburgh, UK)

Prof. dr. Daniel Slamanig

(Universität der Bundeswehr München,
Germany)

Dissertation presented in partial
fulfillment of the requirements for
the degree of Doctor of Engineering
Science (PhD): Electrical Engineering

June 2024

© 2024 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Mahdi Sedaghat, Kasteelpark Arenberg 10, bus 2452, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

To Niloofar.

Preface

This thesis represents the result of years of support from many people. I am incredibly grateful for the chance to acknowledge everyone who played a crucial role in my academic journey. First and foremost, I want to express my gratitude to my wonderful supervisor, Bart. Your guidance, wisdom, and encouragement have been invaluable throughout this process. Bart has always been helpful in keeping me passionate and motivated about my projects, and he has connected me with many people from whom I have learned a lot. I am thankful for your continuous support and the freedom you have given me in my research.

I am also deeply thankful to the chairman, Omer Van der Biest, and my examinee committee, Daniel Slamanig, Markulf Kohlweiss, Nigel Smart, Wouter Joosen, Svetla Nikova, Aysajan Abidin and Tom Van Custom for reading my thesis and for the insightful discussions and useful feedbacks that significantly enhanced the quality of this thesis. A special thanks to Daniel and Markulf for always being available to answer my scientific and non-scientific questions and enlightening me with fresh ideas. Daniel spent a lot of time reviewing different versions of this thesis, and I am truly grateful for his time and his suggestions.

I want to thank each single one of my co-authors: Farhad Aghili, Christian Badertscher, Karim Bagheri, Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Kelong Cong, Elizabeth Crites, Francois Garillot, Maanak Gupta, Yan Ji, Philipp Jovanovic, Markulf Kohlweiss, Jonas Lindstrom, Akash Madhusudan, Deepak Maram, Axel Mertens, Aikaterini Mitrokotsa, Sayantan Mukherjee, Bart Preneel, Ben Riva, Arnab Roy, Daniel Slamanig, Dave Singelée, Alberto Sonnino, Samarth Tiwari, Jenit Tomy, Hendrik Waldner, Joy Wang, and Pun Waiwitlikhit. Your collaborative spirit and great contributions have been crucial to the success of the research presented in this thesis. A special thanks to Karim for his patience and for taking the time to have long discussions, even when he had his own deadlines. I am excited to continue working with all of you.

Before joining COSIC, I spent a year visiting the Computer Science Institute of Charles University in Prague, working with Pavel Hubacek. Though the journey was short, it was a fantastic experience, and I greatly appreciated Pavel's support in helping me to find my future career.

During my PhD, I had the opportunity to visit three outstanding research groups: ZK Labs, hosted by Markulf Kohlweiss, the Department of Information Engineering at CUHK, hosted by Sherman S. M. Chow, and the Foundations of Cryptography group at ETH Zürich, hosted by Dennis Hofheinz. I am grateful to these wonderful hosts and all the members of these groups for their hospitality and engaging discussions. In the summer of 2023, I had an amazing experience interning at Mysten Labs. It was a fantastic opportunity to work with the great Mysten's cryptography team, from whom I learned a lot. Special thanks to George Danezis for interviewing me and linking me with Kostas.

I want to dedicate this paragraph to say thanks to all members of the COSIC Bullying Club (CBC). You turned COSIC into a great and friendly environment, which was exactly what we all needed after the tough post-pandemic days. To save space and avoid extra printing costs, I have encoded your names with SHA-256 (7QQcY9S2JTfmLUyrF0GJ9c9ncEvo0hyfeorNtdNUaUo=). I will reveal the names later, or you can try to break the one-wayness and figure them out yourselves. Special thanks to my office-mates, Amit Singh Bhati and Roozbeh Sarenche, and to Jannik Spiessens for helping me with a proper translation of the abstract into Dutch. Special thanks to Péla for her admin supports and organizing the defense and patiently answering my numerous questions, and to Elsy, Wim, and Anouk for their assistance with many reimbursements requests.

This journey would not have been possible without the love and support of my soulmate, Niloofar. You have always been there for me, motivating and encouraging me through every situation. I am incredibly lucky to have had you by my side for over 14 years. To my parents, Baba, Maman, and my parents-in-law, Baba Joon, and Maman Joon, your unwavering belief in my abilities has been a great source of strength and motivation. I will never forget the day my dad told me his only wish is to see my success. You are my heroes.

Completing this thesis has been a challenging yet rewarding experience, and I am deeply grateful to everyone who has been part of this journey. This work is as much yours as it is mine.

Thank you all!

Mahdi

Abstract

Recent advancements in distributed ledger technologies and decentralized identity management systems have emphasized the need for transparent and privacy-preserving methods. In this context, Threshold cryptography and Non-Interactive Zero-Knowledge (NIZK) proofs are two prominent and fundamental cryptographic concepts that play a critical role. By distributing cryptographic keys among multiple parties, threshold cryptography enhances the security, builds trust, and reduces the risk of key compromise. On the other hand, NIZK proofs enhance privacy in decentralized and distributed applications. Particularly, threshold signatures and Zero-Knowledge Succinct Non-interactive ARGument of Knowledge (zk-SNARK) proofs have gained importance in both academia and industry. However, their combined application often remains non-trivial.

The first part of this thesis is the study of Threshold Structure-Preserving Signatures (TSPS). These schemes are particularly interesting, as they are compatible with efficient NIZK proof systems such as Groth-Sahai proofs. We present the details of two TSPS schemes: one offers the shortest possible signature size for a restricted message space and relies on idealized models. The other, despite having slightly larger signatures, is proven secure under standard assumptions in a stronger model and supports messages being arbitrary group element vectors. We summarize the main properties of these constructions and discuss how these schemes can be applied to Threshold-Issuance Anonymous Credentials.

The second part of this thesis highlights the importance of universal and updatable NIZKs, particularly zk-SNARKs, focusing on their impact on reducing trust in a single party and enhancing privacy in large-scale applications. We further discuss how the security properties of updatable NIZKs can be generically lifted to stronger security models targeting security in composable security frameworks such as the Universal Composability (UC) framework.

Beknopte samenvatting

Recente ontwikkelingen in gedistribueerde blockchaintechnologieën, cryptomunten en gedecentraliseerde identiteitsbeheersystemen hebben de noodzaak benadrukt van transparante en privacybeschermende methoden. In deze context zijn drempelcryptografie en Niet-Interactieve Nul kennisbewijzen (Non-Interactive Zero-Knowledge (NIZK)) twee prominente en fundamentele cryptografische concepten die een cruciale rol spelen. Drempelcryptografie verbetert de beveiliging en vermindert het risico op het lekken van sleutels door cryptografische sleutels te verdelen onder meerdere partijen. Aan de andere kant verbeteren NIZK-bewijzen de privacy in gedecentraliseerde en gedistribueerde toepassingen. Specifiek drempel-handtekeningen en Beknopte Nul-Kennis Niet-Interactief Argument van Kennis (Zero-Knowledge Succinct Non-interactive ARGument of Knowledge (zk-SNARK)) bewijzen hebben zowel in de academische wereld als in de industrie aan belang gewonnen. Hun gecombineerde toepassing blijft echter vaak niet triviaal.

Het eerste deel van deze thesis richt zich op de studie van Drempel Structuur-Bewarende Handtekeningen (Threshold Structure-Preserving Signatures (TSPS)). Ze zijn bijzonder interessant omdat ze compatibel zijn met efficiënte NIZK-bewijssystemen zoals Groth-Sahai bewijzen. We bespreken twee TSPS-schema's in detail: het eerste biedt de kortst mogelijke handtekening voor een beperkte berichtenruimte en vertrouwt op geïdealiseerde modellen. Het tweede heeft iets grotere handtekeningen maar is bewezen onder standaard aannames in een sterker model en ondersteunt berichten die bestaan uit willekeurige groeps-elementen. Vervolgens voorzien we een overzicht van de belangrijkste eigenschappen van deze constructies en bespreken we hoe deze schema's kunnen worden aangewend in toepassingen zoals Threshold-Issuance Anonymous Credential (TIAC).

Het tweede deel van deze thesis benadrukt het belang van universele en bijwerkbare, bewerkbare of aanpasbare NIZK's, in het bijzonder zk-SNARK's, waarbij we de focus leggen op hun impact op het verminderen van het vertrouwen in één enkele partij en het versterken van de privacybescherming in scenario's met veel gebruikers. We bespreken verder hoe de beveiligingseigenschappen van bijwerkbare, bewerkbare of aanpasbare NIZK's generiek kunnen worden opgetild naar sterkere beveiligingsmodellen die gericht zijn op beveiliging in samenstelbare beveiligingsraamwerken zoals het universele samenstelbaarheidsraamwerk (Universal Composability framework).

Contents

Abstract	iv
Beknopte samenvatting	v
Contents	vi
List of Figures	viii
List of Tables	x
List of Abbreviations	xii
I Privacy-Preserving Tools in Distributed Systems	1
1 Introduction	2
1.1 Motivation	2
1.2 Overview of Contributions	6
1.3 Outline of This Thesis	7
1.4 Other Publications	8
2 Background	11
2.1 Basic Notation	11
2.2 Bilinear Groups	12
2.3 Cryptographic Assumptions	14
2.4 Digital Signatures	20
2.5 Threshold Signatures	22
2.6 Secret Sharing Schemes	24
2.7 Non-Interactive Zero-Knowledge Proofs	26
2.8 Public Key Encryption	31
3 Threshold Structure-Preserving Signatures	33
3.1 Technical Challenges Towards Building a TSPS	36
3.2 A Threshold-Friendly SPS	39

3.3	An Introduction to EUF-CiMA Security	41
3.4	TSPS over iDH Message Spaces	42
3.5	TSPS from Standard Assumptions	43
3.6	Application to Threshold-Issuance Anonymous Credential (TIAC)	46
4	Universal and Updatable NIZK	50
4.1	NIZK in the Universal and Updatable SRS Model	53
4.2	On the Setup of Updatable zk-SNARKs	56
4.3	Key-Updatable Public Key Encryption Schemes	60
4.4	A General Framework for Lifting to Upd-BB-SE	62
4.5	Application to Privacy-Preserving Smart Contracts	64
5	Conclusion and Open Problems	66
5.1	Summary of Research	66
5.2	Future Work	67
II	Publications	70
6	Threshold Structure-Preserving Signatures (TSPS)	71
7	TSPS: Strong and Adaptive Security under Standard Assumptions	104
8	Benchmarking the Setup of Updatable zk-SNARKs	136
9	Tiramisu: Black-Box Simulation Extractable NIZKs in the Updatable CRS Model	173
	Publications	194
	Bibliography	194
	Curriculum	223

List of Figures

2.1	Polynomial Interpolation.	25
2.2	Simulation Oracle.	29
2.3	Simulation Oracle for Simulation Extractability Property. . . .	30
4.1	Simulation Oracle for Updatable Zero-Knowledge property. . .	56
4.2	Simulation Oracle for Updatable Simulation Extractability property.	56
6.1	Game Defining the PS Assumption.	82
6.2	Game Defining the GPS ₃ Assumption.	83
6.3	Encoding Function of iDH Message Space in the ROM.	84
6.4	From \tilde{M} to M and back again	85
6.5	Game Defining $\mathbf{G}_{\mathcal{A}}^{\text{EUF-CiMA}}(\kappa)$	87
6.6	Our Indexed Message SPS Construction IM-SPS.	87
6.7	The proposed constructions and underlying assumptions.	88
6.8	Encoding function of iDH multi-message space in the ROM. . .	89
6.9	Our Indexed Multi-Message SPS Construction IMM-SPS. . . .	90
6.10	Game $\mathbf{G}_{\mathcal{A}}^{\text{T-EUF-CiMA}}(\kappa)$	93
6.11	Our Threshold SPS Construction TSPS.	94
6.11	Our Threshold SPS Construction TSPS.	95
6.11	A Threshold Blind Signature with straight-line extraction. . . .	100
6.11	A Threshold Blind Signature with straight-line extraction. . . .	101
7.1	Unforgeability games for threshold signatures.	118
7.2	Game defining the core lemma.	119
7.3	Our proposed TSPS construction.	120
7.4	Game_0	121
7.5	Modifications in Game_1	122
7.6	Modifications in Game_3	123
7.7	Reduction to the core lemma in Lemma 1.	124
7.8	Modification from Game_3 to Game_4	125
7.9	Game'_3 in the proof of Theorem 6.	127
7.10	Security game Game_0	130
7.11	Modifications in Game_1	131

7.12	Game'_3 in the proof of Theorem 7.	132
8.1	Setup in the updatable zk-SNARKs: SG, SU, and SV by P or V.	140
8.2	SG algorithm for SONIC.	147
8.3	SU and SV algorithms for SONIC.	149
8.4	Slightly modified SG algorithm of Marlin.	153
8.5	SV and SU algorithms for Marlin with the slightly modified SRS.	155
8.6	SG algorithm for LunarLite.	156
8.7	SU and SV algorithms for LunarLite.	157
8.8	SG algorithm for Basilisk.	160
8.9	SU and SV algorithms for Basilisk.	161
8.10	BSV: The Batched version of SV algorithm for Sonic.	163
8.11	BSV: The Batched version of SV algorithm for Marlin.	165
8.12	BSV: The Batched version of SV algorithm for LunarLite.	166
8.13	The BSV algorithm for Basilisk.	167
8.14	Table of comparisons for the existing universal zkSNARKs.	168
8.15	Recursive execution of BSV to identify a malicious SRS updater.	170
9.1	Using C0C0 and TIRAMISU to Build BB-SE	178
9.2	Comparison of the Performance of Key Update and Key Verify algorithms.	185
9.3	TIRAMISU	193

List of Tables

- 4.1 Cost of Basic Group Operations. 58
- 6.1 Table of pairing-based non-interactive threshold signature schemes. 77
- 6.2 Some Structure-Preserving Signatures (SPS). 78
- 7.1 Overview of security notions and our results. 108
- 7.2 Comparison of the existing threshold structure-preserving signature.109
- 8.1 An efficiency comparison of our proposed SU, SV and BSV algorithms. 140
- 9.1 A comparison of TIRAMISU with related works. 179
- 9.2 A comparison of BB-SE NIZK arguments built with the C0C0 and TIRAMISU 191

List of Abbreviations

AC	Anonymous Credentials
ACE	Access Control Encryption
AGM	Algebraic Group Model
BDH-KE	Bilinear Diffie-Hellman Knowledge of Exponent
CRHF	Collision Resistant Hash Function
CRS	Common Reference String
DAP	Decentralized Anonymous Payment
DDH	Decisional Diffie-Hellman
DH	Diffie-Hellman
DKG	Distributed Key Generation
DL	Discrete Logarithm
DPC	Decentralized Private Computation
DS	Digital Signatures
EUF-CiMA	Existential Unforgeability against Chosen indexed Message Attack
EUF-CMA	Existential Unforgeability against Chosen Message Attack
GGM	Generic Group Model
GPS	Generalized Pointcheval-Sanders
GS	Groth-Sahai
iDH	indexed Diffie-Hellman
IND-CPA	Indistinguishability under Chosen-Plaintext Attack

KEA	Knowledge of Exponent Assumption
kerMDH	Kernel Matrix Diffie-Hellman
KS	Knowledge Soundness
MDDH	Matrix Decisional Diffie-Hellman
MPC	Multi-Party Computation
NIZK	Non-Interactive Zero-Knowledge
NP	Nondeterministic Polynomial Time
PKE	Public-Key Encryption
PPE	Pairing-Product Equation
PPT	Probabilistic Polynomial-Time
PS	Pointcheval-Sanders
ROM	Random Oracle Model
SE	Simulation Extractability
SPS	Structure-Preserving Signatures
SPS-EQ	Structure-Preserving Signatures on Equivalence-Classes
SRS	Structured Reference String
Sub-ZK	Subversion Zero-Knowledge
SXDH	Symmetric External Diffie-Hellman
TIAC	Threshold-Issuance Anonymous Credential
TS	Threshold Signatures
TSPS	Threshold Structure-Preserving Signatures
UC	Universally Composable
Upd-BB-SE	Updateable Black-Box Simulation Extractability
Upd-KS	Updatable Knowledge Soundness
Upd-nBB-SE	Updateable non-Black-Box Simulation Extractability
Upd-ZK	Updateable Zero-Knowledge
ZK	Zero-Knowledge
zk-SNARK	Zero-Knowledge Succinct Non-interactive ARgument of Knowledge

Part I

Privacy-Preserving Tools in Distributed Systems

CHAPTER

1

Introduction

The garden of the world has no limits, except in your mind.

Molana Rumi

1.1 Motivation

Through the advancement of mobile technologies and the internet, people are now able to communicate and use online services from anywhere at any time. However, in doing so they often share their personal information in real time with centralized online service providers. Personal information can include a person's name, age, geographical location, or real-life behavior, and oversharing these details can lead to identity theft, online harassment, physical threats and mass surveillance. This highlights the importance of data privacy within the digital world. Generally speaking, data privacy refers to the ability to control the flow of personal information: *when*, *how*, and *to what extent* data can be shared. Specifically, privacy refers to the right of individuals to selectively reveal sensitive details about themselves in a fair and legal manner [Swi97].

While the most prominent privacy protection regulations such as the General Data Protection Regulation (GDPR) [Uni16] aim to restrict the use of personal data to safeguard individuals' online privacy, cryptographic privacy-preserving tools play a central role in implementing these regulations. These methods are considered fundamental techniques for enforcing regulations such as GDPR, as they enable privacy by design. Enhancing the privacy using these methods is significant, as a recent survey [MKW22] indicates that over 130 000 personal data breaches across Europe were reported to regulators between January 2021

and January 2022, an average of 356 breaches per day. Meanwhile, it has been evident that regulatory fines against big tech (centralized) companies were not sufficient to change their behavior.¹ These days, the top 1% of social networks hold 95% of social web traffic and 86% of social mobile app usage. Similarly, the top 1% of search engines capture 97% of search traffic, while the top 1% of e-commerce sites account for 57% of e-commerce traffic [Dix24].

On the one hand, decentralized and distributed systems aim to reduce users' sensitive data availability to big and centralized service providers such as Meta and Google. On the other hand, privacy-preserving tools as a cryptographic solution aim to protect users' sensitive information, even in the event of data breaches [Tro+17].

Privacy-preserving cryptographic tools enable the extraction of useful information from a potentially sensitive set of data without providing full access to the data. There are many real-world applications for such schemes in scenarios where access to data can be beneficial to those with exclusive access or in cases where plain data access may be harmful for the users, such as in health care, electronic voting, and notably in the context of decentralized anonymous authentication systems. Zero-Knowledge (ZK) proofs [GMR85] play a central role in many privacy-preserving systems, particularly in decentralized and distributed systems.

ZK proofs involve two parties (the prover and verifier) and enable the prover to prove the validity of a claim to a verifier, usually in multiple rounds of communication, without disclosing any information beyond the validity of the claim. Non-Interactive Zero-Knowledge (NIZK) proofs [BFM88] remove the interaction between prover and verifier. They allow a prover to convince a verifier about the truth of a statement in a single round of communication. NIZK proofs have three main fundamental security properties: completeness, zero-knowledge and knowledge soundness. Completeness ensures correctly generated proofs always verify, zero-knowledge captures the fact that no information beyond the validity of the proof is leaked during the verification phase, and the knowledge soundness guarantees that the prover cannot convince a verifier without knowing the hidden witness.

NIZK proofs are invaluable within many systems such as Anonymous Credentials (AC), allowing users to confirm their identity or specific attributes to others without giving away any extra personal details, thereby protecting privacy and mitigating the risk of identity fraud. To be more precise, users frequently need to prove aspects of their identity or status without compromising their anonymity. This might include demonstrating they reside in a specific region to access geo-restricted services, proving membership in a loyalty program

¹<https://analyticsindiamag.com/big-techs-dont-care-about-lawsuits/>.

to receive discounts, or confirming they belong to a particular professional group to access specialized forums without being tracked. In these systems, a verifier can be assured nothing beyond that a trusted entity, known as the credential issuer, has already certified the list of attributes claimed by the user. Anonymous credentials could be a very valuable tool for the European Digital Identity Wallet, which the European Commission plans to roll out by 2026 [EU24; AF24].

A credential in this context is defined by a digital signature on a set of attributes that the users possesses (e.g., affiliation, role, age). The users can demonstrate their possession of a credential (digital signature) that satisfies a specific access policy without revealing any details about the real identity of the user, except that they meet the criteria of the access policy using NIZK proofs. Despite many advantages that NIZK proofs offer to privacy-focused systems, they often bring along both communication and computational overheads. Over the past decades, there has been significant work towards building digital signatures that are compatible with efficient NIZK proofs.

Structure-preserving primitives [Abe+10], in particular Structure-Preserving Signatures (SPS) keep an algebraic structure over bilinear groups and avoid non-linear operations in their designs; as a consequence, they are compatible with efficient NIZKs such as Groth-Sahai (GS) proof systems [GS08]. Since GS proofs are straight-line extractable, i.e. extraction works without rewinding, in the standard model, they are particularly interesting for constructions targeting security in composable security frameworks such as the Universally Composable (UC) framework [Can01], which is a formal method for analyzing and proving the security of cryptographic protocols. However, GS proofs only can handle a restricted class of relations known as quadratic equations, particularly Pairing-Product Equation (PPE).

As discussed earlier, the credentials, i.e. signatures on attributes, are often issued by central authorities to manage user identities. However, this approach has two main drawbacks: the reliance on a single credential issuer, which creates an availability concern, and the potential risk of compromising the single key, representing a vulnerability and a single point of failure. To be more precise, a corrupted issuer can generate fake credentials, thereby compromising the integrity and security of the entire system. Consequently, there is a growing interest in developing decentralized identity management systems which typically rely on threshold cryptography, particularly Threshold Signatures (TS).

By sharing a single secret signing key among multiple parties, TS [Des90] enable a sufficiently large group of signers to collaboratively generate a signature. In this case, the users obtain their credentials if a quorum of credential issuers agrees on the authenticity of the claimed attributes. The combined (i.e. aggregated)

credential provides the same security properties as a centrally issued credential, and subsequent use of the credential, i.e. signature, in this context remains the same.

While NIZKs with a limited class of functionality such as Groth-Sahai (GS) proofs are a solid design choice for the aforementioned applications, they are not suitable for all types of complex and arbitrary relations. Zero-Knowledge Succinct Non-interactive ARgument of Knowledge (zk-SNARK) [Par+13; Gro16] are an efficient form of the (pre-processed) NIZKs, with succinct (short) proofs and efficient verification. In contrast to GS proofs, they can be applied efficiently to any Nondeterministic Polynomial Time (NP)-language, described as an arithmetic circuit. However, zk-SNARKs require a complex trusted setup. This phase needs a high degree of trust in a single entity for the generation of a set of public parameters, known as Common Reference String (CRS), which can be used to create and validate a proof. Given the fact that a maliciously executed setup by a subverted CRS generator can undermine the security guarantees of these systems, assuming the existence of a trustworthy entity in most of decentralized and distributed applications is difficult.

To overcome this challenge, various trust mitigation techniques have been proposed, including subversion-resistant zk-SNARKs [BFS16; Abd+17; Fuc18; Bag19b] or generating the CRS parameters with Multi-Party Computation (MPC) protocols [Ben+15; BGM17; Koh+21]. In the former case, a one-time validity check can be performed by the prover, but the verifier still needs to trust the generator of the CRS. In contrast, in the latter case, the setup can be performed by a (fixed and) sufficiently large number of parties, denoted by n , instead of a single party, thereby reducing the trust level to one out of n in the setup for both the prover and verifier. Powers-of-Tau ceremonies [BGM17] are a prime example for this scenario that are being deployed in real-world applications. However, this trust assumption is fixed throughout the entire operational lifespan of the system. Moreover, this heavy setup phase in some zk-SNARKs is circuit-dependent, and any minor change to the initial circuit necessitates a complete redo of the setup process.

As an alternative to these trust mitigation mechanisms, Groth et al. [Gro+18] proposed universal and updatable zk-SNARKs. The universal nature of these systems enables to reuse the same CRS for any circuit up to a certain size, while in non-universal zk-SNARKs such as Groth16 [Gro16], the CRS components solely depend on a fixed circuit. Moreover, the updatability allows parties—including provers and verifiers—to take the most recent CRS and continuously update it in a verifiable way, enabling the engagement of a more diverse set of participants over time. Similar to the MPC-based settings, the knowledge soundness remains as long as one of the contributors (updaters) acts honestly.

1.2 Overview of Contributions

The main contributions of this thesis can be summarized in two main research areas listed below.

1.2.1 Threshold Structure-Preserving Signatures

As already mentioned, the compatibility of Structure-Preserving Signatures (SPS) with Groth-Sahai (GS) proofs [GS08] makes them an attractive building block for many complex (privacy-preserving) cryptographic protocols. However, even though more than ten years have passed since the introduction of SPS, the literature still lacks Threshold Structure-Preserving Signatures (TSPS) as a natural extension. By sharing secret keys among multiple parties, threshold cryptography reduces the potential risk of compromising the single key which represents a single point of failure. In this case, a secret key-dependent operation can be performed if a sufficiently large number of parties collaborates.

In [Cri+23], we define TSPS as a novel cryptographic primitive and propose an efficient construction over a new message space called the indexed Diffie-Hellman (iDH) message space. We prove its security under the hardness of a modified version of an interactive assumption, called the Generalized Pointcheval-Sanders (GPS) assumption, against static adversaries and a weaker unforgeability notion, called T-UF-0 [Bel+22]. However, we obtain the shortest possible TSPS based on the impossibility results for SPS shown by Abe et al. [Abe+11b]. This paper leaves several interesting open questions, including the development of a TSPS for any vector of group elements and its construction with stronger security guarantees and supporting adaptive adversaries.

As a subsequent work, in [Mit+24], we answer some of the open questions in [Cri+23]. In this paper, we propose a TSPS based on standard and non-interactive assumptions, namely Matrix Decisional Diffie-Hellman (MDDH) and Kernel Matrix Diffie-Hellman (kerMDH) assumptions. We prove the unforgeability of this construction under the strongest notion of security for fully non-interactive Threshold Signatures (TS), called T-UF-1 [Bel+22] and for adaptive adversaries.

1.2.2 Non-Interactive Zero-Knowledge proofs

The updatability feature of universal and updatable zk-SNARK allows parties—including provers and verifiers—to take the most recent CRS and continuously

update it in a verifiable way, enabling the engagement of more diverse set of participants over time. However, except for Groth et al.’s initial study [Gro+18], none of the follow-up works on universal and updatable zk-SNARKs discussed how it can be set up, i.e., how the CRS components can be updated and be verified. To fill the existing gap, in [BMS23], we show how the CRS in some existing universal and updatable zk-SNARKs, including Sonic [Mal+19], Plonk [GWC19], Marlin [Chi+19], Lunar [Cam+21], and Basilisk [RZ21] can be updated and their consistency can be checked. As an additional contribution, we propose a batched and aggregated form of CRS validity checks, leading to more practical results.

In [BS21], we elaborate on the fact that the traditional definitions of knowledge soundness in the context of universal and updatable NIZKs are not well adapted to the decentralized settings. Zk-SNARK’s use-cases within the UC framework [Can01], require stronger security guarantees such as Simulation Extractability (SE) along with straight-line extraction and zk-SNARKs in the universal and updatable CRS model are not an exception. SE ensures the knowledge soundness even if the adversary has already seen many simulated proofs. This definition matches better real-world applications such as Decentralized Anonymous Payment (DAP) and privacy-preserving smart contracts in which the adversary has already access to many valid proofs and still should not be able to generate a fake proof. In NIZKs with updatable CRS, an adversary can also update the CRS and observe proofs generated under different CRSs, thus the generalization of this security property needs some care.

1.3 Outline of This Thesis

This thesis has two parts. The first part of this thesis includes the following chapters:

Chapter 2 provides a summary of the notation and recalls the definition of various cryptographic primitives along with their security properties. Additionally, in this chapter, we recall some mathematical assumptions on problems that are assumed to be hard and form the foundation for the design of the constructions discussed in the next chapters.

Chapter 3 covers the main contributions of this thesis on Threshold Structure-Preserving Signatures (TSPS). We start by motivating these constructions. We briefly summarize the main challenges towards building them and discuss two instantiations.

In Chapter 4, the emphasis is on the setup process for universal and updatable zk-SNARKs. Additionally, in this chapter we discuss a generic framework for lifting the standard updatable knowledge-sound NIZKs to a stronger security notion, called updatable black-box simulation extractability.

Finally, Chapter 5 concludes the main contributions of this thesis and lists some interesting open problems for future study.

In the second part (Part II), we list the original text of our papers.

1.4 Other Publications

Additionally, we published several articles that complement this thesis.

- C. Badertscher, M. Sedaghat, and H. Waldner. “Unlinkable Policy-Compliant Signatures for Compliant and Decentralized Anonymous Payments”. In: *Cryptology ePrint Archive, Paper 2023/1070 (to appear in PETS’2024)* (2024). URL: <https://eprint.iacr.org/2023/1070>

SUMMARY OF RESULTS: Blockchain as the main technology behind the cryptocurrencies is essentially a decentralized ledger which operates on a distributed network of nodes. However, the flagship cryptocurrencies, such as Bitcoin and Ethereum, have received a great public attention, they come with privacy issues. Decentralized Anonymous Payment (DAP) systems overcome this challenge and hides the transaction values as well as the identifiers of the parties involved in each transaction, resulting in a fully privacy-preserving payment system. Meanwhile, their potential for full privacy has raised concerns about regulatory issues and illicit activities, such as money laundering, tax evasion, trading of illegal substances and terrorist funding. In this paper, we propose Unlinkable Policy-Compliant Signatures (ul-PCS) that can enhance the existing proposals towards a proactive accountable private system. We subsequently integrate this scheme with the existing DAPs, ensuring that the validity of signature on a transaction is determined by a predefined policy, with both the sender’s and receiver’s attributes playing a role in finality of the transactions.

- F. Baldimtsi, K. K. Chalkias, Y. Ji, J. Lindstrøm, D. Maram, B. Riva, A. Roy, M. Sedaghat, and J. Wang. “zkLogin: Privacy-Preserving Blockchain Authentication with Existing Credentials”. In: *The Science of Blockchain Conference (SBC) 2024 (To be presented)* (2024). <https://arxiv.org/pdf/2401.11735>

SUMMARY OF RESULTS: To receive and send cryptocurrencies, the users need to create a wallet. Private key-based wallets, often accessed via mnemonics or hardware wallets, pose onboarding challenges, impacting blockchain adoption. In this paper, we propose zkLogin, that utilizes identity tokens from OpenID Connect platforms (like Google, Microsoft, Meta, etc) for transaction authentication, allowing users to sign with their existing accounts, enhancing user experience by eliminating the need to remember new secrets. The zkLogin system ensures strong security and privacy by integrating platform-based authentication and by using zk-SNARKs, it links the connection between users' off-chain and on-chain identities, without additional trusted entities.

- A. Madhusudan, M. Sedaghat, S. Tiwari, K. Cong, and B. Preneel. "Reusable, Instant and Private Payment Guarantees for Cryptocurrencies". In: *Information Security and Privacy - 28th Australasian Conference, ACISP 2023, Brisbane, QLD, Australia, July 5-7, 2023, Proceedings*. Ed. by L. Simpson and M. A. R. Baee. Vol. 13915. Lecture Notes in Computer Science. Springer, 2023, pp. 580–605. DOI: [10.1007/978-3-031-35486-1_25](https://doi.org/10.1007/978-3-031-35486-1_25)

SUMMARY OF RESULTS: Public decentralized cryptocurrencies, in particular the proof of work-based cryptocurrencies such as Bitcoin, have scalability issues such as low throughput and high transaction latency. In this paper, we present a payment guarantee system that provides conditionally-reusable collateral enabling instant payment confirmation. Based on a novel cryptosystem called randomness-reusable Threshold Encryption (rrTE) scheme, we propose a state-less and proactive deanonymizing mechanism which prevents malicious users from misusing the system, namely double-spending their collatorals. We also propose a Threshold-Issuance Anonymous Credential (TIAC) instantiated by the Groth-Sahai (GS) proofs [GS08] using the proposed TSPS in [Cri+23].

- S. F. Aghili, M. Sedaghat, D. Singelée, and M. Gupta. "MLS-ABAC: Efficient Multi-Level Security Attribute-Based Access Control scheme". In: *Future Generation Computer Systems* (2022). ISSN: 0167-739X. DOI: <https://doi.org/10.1016/j.future.2022.01.003>. URL: <https://www.sciencedirect.com/science/article/pii/S0167739X22000115>

SUMMARY OF RESULTS: In this paper, we propose an outsourceable and lightweight multi-level secure attribute-based access control scheme as a followup to [SP21]. In our prior work, we proposed a Ciphertext-Policy Attribute-Based Encryption (CP-ABE) which benefits from constant ciphertext and constant key sizes, however its ciphertext contains target group elements. Our method uses the Ascon cryptosystem [Dob+21] to compress the ciphertext further and replace the target group element in

the ciphertext with a substantially shorter field element. Additionally, based on the architecture, we reduced the decryption cost by using an honest-but-curious intermediary Cloud. To grant access to the data, this paper combines dynamic and static attributes, unlike prior works that only considered static attributes.

SUMMARY OF RESULTS: Multi-Signatures have received much attention due to their application in blockchain settings and more specifically in proof of stake-based cryptocurrencies like Eth 2.0 and Sui. The most efficient multi-signature schemes like BLS-based schemes [BDN18], however, require to compute the aggregated public keys for each block. This paper defines the subset-optimized multi-signatures, which enables secure signing for any subset of a pre-fixed set signers. As a result, we evaluate the unforgeability of the scheme using an information theoretical assumption, namely the Random Modular Subset Sum (RMSS) problem in the Algebraic Group Model (AGM) and Random Oracle Model (ROM).

- M. Sedaghat and B. Preneel. “Cross-Domain Attribute-Based Access Control Encryption”. In: *Cryptology and Network Security (CANS)*. ed. by M. Conti, M. Stevens, and S. Krenn. Springer International Publishing, 2021, pp. 3–23. DOI: [10.1007/978-3-030-92548-2_1](https://doi.org/10.1007/978-3-030-92548-2_1)

SUMMARY OF RESULTS: Access Control Encryption (ACE) was introduced by Damgård et al. [DHO16] as a novel cryptographic solution to the problem of information flow control, i.e., to enforce who can read and write which data. It allows to enforce a fine-grained access control by giving different rights to different users in terms of which messages they are allowed to send and receive (i.e., decrypt). In this paper, we extend the existing identity-based ACEs in which the predicate functions can handle attributes of the users and improve the granularity of the policy in a cross-domain setting [WC21], called Cross-Domain Attribute-Based Access Control Encryption.

The complete list of publications can be found in the [Publications](#) section on page 194.

CHAPTER

2

Background

The beauty of mathematics only shows itself to more patient followers.

Maryam Mirzakhani

In this section, we first introduce several background notions used in the first part (Part I) to understand the main contributions of this thesis. Then we review some well-known cryptographic primitives and their security properties. These primitives are building blocks for the constructions discussed in the next chapters. Considering that some algorithms in different primitives share identical names, we use $\mathcal{PRI}.\text{Alg}$ to show algorithm Alg for the primitive \mathcal{PRI} . Additionally, we elaborate on certain cryptographic hardness assumptions and the models ensuring their security.

2.1 Basic Notation

Throughout the first part of this thesis, let $\kappa \in \mathbb{N}$ denote some general security parameter and 1^κ its unary representation. A function $\nu : \mathbb{N} \rightarrow \mathbb{R}^+$ is called *negligible* if for all large enough κ there exists $\kappa_0 \in \mathbb{N}$ such that for all $\kappa > \kappa_0$ and $c > 0$ it holds that $\nu(\kappa) < 1/\kappa^c$. All the algorithms are randomized and Probabilistic Polynomial-Time (PPT) unless it is explicitly stated. For a non-empty set S , $x \leftarrow \$ S$ denotes sampling an element from S uniformly at random and assigning its output to variable x . Let $y \leftarrow \$ \text{Alg}(z; r)$ denote running probabilistic algorithm Alg on input z and an implicit random input r and assigning its output to variable y . An algorithm Alg is called deterministic if $r = \perp$ and if we do not specify r , i.e. $y \leftarrow \$ \text{Alg}(z)$ the random input r is sampled uniformly at random. A vector \vec{r} is denoted by \mathbf{r} and a matrix A is

denoted by \mathbf{A} . We use $[1, n]$ to denote the set of integers $\{1, \dots, n\}$. $\{0, 1\}^*$ denotes the set of all binary strings of arbitrary length.

We denote the output of a security game \mathbf{G}^{GAME} between a challenger and a PPT adversary \mathcal{A} by $\mathbf{G}_{\mathcal{A}}^{\text{GAME}}$, where \mathcal{A} wins the game if $\mathbf{G}_{\mathcal{A}}^{\text{GAME}} = 1$. Within the security games, the adversary could potentially have access to optional oracles such as $\mathcal{O}^{\text{oracle}}(\cdot)$. A cryptographic scheme, Pri , is called computationally secure or perfectly secure if for all PPT adversaries \mathcal{A} , we have $\text{Adv}_{\mathcal{A}}^{\text{Pri}} \leq \nu(\kappa)$ or $\text{Adv}_{\mathcal{A}}^{\text{Pri}} = 0$, where $\text{Adv}_{\mathcal{A}}^{\text{Pri}} := \Pr[\mathbf{G}_{\mathcal{A}}^{\text{GAME}, \text{Pri}} = 1]$. The scheme is called statistically secure, if the above condition holds for unbounded adversaries.

Let q be a κ -bit prime. We denote the group of integers module p by $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, its multiplicative group of units by \mathbb{Z}_q^* , and the polynomial ring over \mathbb{Z}_q by $\mathbb{Z}_q[X]$. For a group $\mathbb{G} = \langle \mathbf{G} \rangle$ of prime order q with generator \mathbf{G} and identity element $1_{\mathbb{G}}$, we use \mathbb{G}^* to denote the set $\mathbb{G} \setminus 1_{\mathbb{G}}$. Throughout this thesis, we treat most groups using additive notation and use an implicit representation of group elements, in which for an integer $\alpha \in \mathbb{Z}_q$, its implicit representation in group \mathbb{G} is defined by $[\alpha]_{\mathbf{M}} := \alpha \mathbf{M} \in \mathbb{G}$, where \mathbf{M} can be any arbitrary group element in \mathbb{G} . To be more general, the implicit representation of a matrix $\mathbf{A} = (\alpha_{ij}) \in \mathbb{Z}_q^{m \times n}$ in \mathbb{G} is defined by $[\mathbf{A}]_{\mathbf{M}}$, and we have:

$$[\mathbf{A}]_{\mathbf{M}} := \begin{pmatrix} \alpha_{1,1} \mathbf{M} & \cdots & \alpha_{1,n} \mathbf{M} \\ \alpha_{2,1} \mathbf{M} & \cdots & \alpha_{2,n} \mathbf{M} \\ \vdots & \ddots & \vdots \\ \alpha_{m,1} \mathbf{M} & \cdots & \alpha_{m,n} \mathbf{M} \end{pmatrix}.$$

To simplify, when \mathbf{M} is a generator of \mathbb{G} , i.e. \mathbf{G} , we use $[\alpha]$ instead of $[\alpha]_{\mathbf{G}}$.

2.2 Bilinear Groups

In real-world applications, elliptic curve groups over prime fields with large characteristics are commonly used. Within this setting, the cyclic group, represented as \mathbb{G} , is composed of point pairs $(x, y) \in \mathbb{Z}_p^2$. These pairs satisfy the elliptic curve equation $y^2 = x^3 + ax + b$, where the variables $a, b \in \mathbb{Z}_p$ are fixed parameters.

Example 1. For example, the Secp256k1 elliptic curve used to generate Bitcoin public keys is defined by the equation $y^2 = x^3 + 7$, i.e., $a = 0$ and $b = 7$. As another example, the popular pairing-friendly BLS12-381 curve, proposed by Barreto, Lynn and Scott in [BLS03], is defined by $y^2 = x^3 + 4$, i.e., $a = 0$ and $b = 4$ over a coordinate field of size 384 bits.

Pairings, also known as bilinear maps, are efficiently computable functions that operate over three cyclic groups, \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T , each having a prime order p and generators \mathbf{G}_1 , \mathbf{G}_2 , and \mathbf{G}_T , respectively. The pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ maps a pair of elliptic curve points from *source groups* \mathbb{G}_1 and \mathbb{G}_2 to the specific *target group* \mathbb{G}_T as a multiplicative subgroup of a finite field such that the exponents of source group elements multiply.

A pairing is termed *symmetric* if \mathbb{G}_1 and \mathbb{G}_2 are the same; otherwise, it is *asymmetric*. In this thesis, we exclusively concern with asymmetric bilinear groups, specifically Type-III bilinear groups. These groups are distinct in that there is no efficient isomorphism between \mathbb{G}_1 and \mathbb{G}_2 . Type-III bilinear groups are the most efficient choice for certain security levels [GPS08]. BLS12-381 [BLS03] and BN-254 [BN06] elliptic curves are two prime examples of this type.

Definition 1 (Type-III Bilinear Groups). More formally, an asymmetric bilinear group generator, $\text{BGgen}(1^\kappa)$, returns a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{G}_1, \mathbf{G}_2, e)$, such that $\mathbb{G}_1 = \langle \mathbf{G}_1 \rangle$, $\mathbb{G}_2 = \langle \mathbf{G}_2 \rangle$ and $\mathbb{G}_T = \langle e(\mathbf{G}_1, \mathbf{G}_2) \rangle$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map with the following properties:

- *Bilinearity:* $\forall \alpha, \beta \in \mathbb{Z}_p, e([\alpha]_1, [\beta]_2) = [\alpha\beta]_T = e([\beta]_1, [\alpha]_2)$,
- *Non-degeneracy:* $e(\mathbf{G}_1, \mathbf{G}_2) \neq 1_{\mathbb{G}_T}$,

where for simplicity we use $[\alpha]_\zeta$ instead of $[\alpha]_{\mathbb{G}_\zeta}$ for $\zeta \in \{1, 2, T\}$. Additionally, we use $[\alpha, \dots, \alpha^\ell]_\zeta$ instead of $([\alpha]_\zeta, [\alpha^2]_\zeta, \dots, [\alpha^\ell]_\zeta)$. We use additive notation for the source groups of \mathbb{G}_1 and \mathbb{G}_2 while we use multiplicative notation for the target group \mathbb{G}_T .

We further extend the pairing operation such that for two matrices \mathbf{A} and \mathbf{B} with matching dimensions we define $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T$. To be more precise,

$$e([\mathbf{A}]_1, [\mathbf{B}]_2) = e \left(\begin{pmatrix} \alpha_{1,1}\mathbf{G}_1 & \cdots & \alpha_{1,n}\mathbf{G}_1 \\ \alpha_{2,1}\mathbf{G}_1 & \cdots & \alpha_{2,n}\mathbf{G}_1 \\ \vdots & \ddots & \vdots \\ \alpha_{m,1}\mathbf{G}_1 & \cdots & \alpha_{m,n}\mathbf{G}_1 \end{pmatrix}, \begin{pmatrix} \beta_{1,1}\mathbf{G}_2 & \cdots & \beta_{1,n}\mathbf{G}_2 \\ \beta_{2,1}\mathbf{G}_2 & \cdots & \beta_{2,n}\mathbf{G}_2 \\ \vdots & \ddots & \vdots \\ \beta_{m,1}\mathbf{G}_2 & \cdots & \beta_{m,n}\mathbf{G}_2 \end{pmatrix} \right) = [\mathbf{AB}]_T.$$

2.3 Cryptographic Assumptions

In cryptography, the security proofs mainly rely on mathematical hardness assumptions, i.e., the fact that some problems are believed to be hard to solve. These assumptions can be categorized into different types, ranging from standard to non-standard, falsifiable to non-falsifiable and interactive to non-interactive, etc. In the following, we provide a list of hardness assumptions that form the basis for the constructions we will discuss in the next chapters.

Note 1. (Standard Assumptions) Standard assumptions are well-accepted mathematical problems, such as the factoring problem or the discrete logarithm problem, and may involve multiple rounds of interaction (i.e. interactive) or no interaction at all between the adversary and the challenger (i.e. non-interactive). An assumption is termed falsifiable if it can be structured as a game, possibly interactive, where a challenger can effectively check if the adversary has achieved their goal.

Discrete Logarithm (DL)-based problems. The Discrete Logarithm (DL) problem as a classical problem in number theory forms a foundation for many Public Key Cryptography (PKC) systems.

Definition 2 (Discrete Logarithm (DL) problem). Let a cyclic group $\mathbb{G} = \langle G \rangle$ of prime order q , the DL problem is hard if for all PPT adversaries \mathcal{A} , we have:

$$\Pr [\forall Z \leftarrow \mathbb{G} \mid z \leftarrow \mathcal{A}(\mathbb{G}, Z) : Z = [z]] \leq \nu(\kappa).$$

The (q_1, q_2) -Discrete Logarithm assumption as formally defined below, expands the set of known group elements to the adversary.

Definition 3 ((q_1, q_2) -Discrete Logarithm Assumption). Given an asymmetric bilinear group description $\mathcal{BG} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{G}_1, \mathbf{G}_2, e) \leftarrow \text{BGgen}(1^\kappa)$, the (q_1, q_2) -discrete logarithm assumption holds if for all PPT adversaries \mathcal{A} , we have: $\Pr [z \leftarrow \mathbb{Z}_p^*, z' \leftarrow \mathcal{A}(\mathcal{BG}, [z, \dots, z^{q_1}]_1, [z, \dots, z^{q_2}]_2) : z' = z] < \nu(\kappa)$.

This assumption is hard in the bilinear Generic Group Model (GGM).

Note 2. (Computational vs. Decisional Assumptions) The assumptions such as Discrete Logarithm (DL) are computational, meaning the adversary's objective is to compute and obtain a challenge output. Conversely, an assumption is termed decisional when the adversary is only required to distinguish between two potential scenarios.

Definition 4 (Decisional Diffie-Hellman Assumption [Bon98]). Given a cyclic group \mathbb{G} of prime order q with generator G , the Decisional Diffie-Hellman (DDH) assumption holds if for all PPT adversaries \mathcal{A} , and uniformly random integers $x, y, z \leftarrow \mathbb{Z}_q^*$, we have: $Adv_{\mathcal{A}}^{\text{DDH}}(\kappa) := |\varepsilon_1 - \varepsilon_0| \leq \nu(\kappa)$, where $\varepsilon_\beta := \Pr[\mathcal{A}([x], [y], [xy + \beta z]) = 1]$.

It is straightforward that the DDH assumption over the source group \mathbb{G} becomes trivial when there exists a symmetric bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. The adversary can verify by checking if the Pairing-Product Equation (PPE) $e([x], [y]) = e([xy + \beta z], G)$ holds. However, in the Type-III bilinear setting, the Symmetric External Diffie-Hellman (SXDH) assumption [Jou00; BF01] ensures that DDH holds in both source groups \mathbb{G}_1 and \mathbb{G}_2 . Looking ahead, in Chapter 4, we use the SXDH assumption to construct a key-updatable public key encryption scheme from the well-known ElGamal cryptosystem [ElG84].

PS assumptions. Next, we recall the family of Pointcheval-Sanders (PS) assumptions that play a central role in the design of the initial Threshold Structure-Preserving Signatures (TSPS) scheme, discussed in Chapter 3.

Definition 5 (Pointcheval-Sanders (PS) Assumption [PS16]). Given a tuple $([x]_2, [y]_2)$ and an oracle $\mathcal{O}^{\text{PS}}(m) \rightarrow (h, [x + my]_h) \in \mathbb{G}_1^2$, where $x, y \leftarrow \mathbb{Z}_p^*$, and $h \leftarrow \mathbb{G}_1$. The PS game, i.e., \mathbf{G}^{PS} , is to find the tuple (m^*, h^*, s^*) such that (1) $h^* \neq 1_{\mathbb{G}_1}$, $m^* \neq 0$, (2) $s^* = [x + m^*y]_{h^*}$ and (3) $m^* \notin \mathcal{Q}$, where \mathcal{Q} is the list of all queried messages to the $\mathcal{O}^{\text{PS}}(m)$ oracle. The advantage of an adversary \mathcal{A} against this game can be defined as $Adv_{\mathcal{A}}^{\text{PS}}(\kappa) := \Pr[\mathbf{G}_{\mathcal{A}}^{\text{PS}}(\kappa) = 1]$. The PS assumption holds if for all PPT adversaries \mathcal{A} , we have: $Adv_{\mathcal{A}}^{\text{PS}}(\kappa) < \nu(\kappa)$.

It is easy to verify the validity of the challenge tuple (m^*, h^*, s^*) by checking the PPE, $e(s^*, G_2) = e(h^*, [x]_2 + m^*[y]_2)$. Note that the PS assumption is an interactive assumption, meaning that during the security game, there is communication between the challenger and the adversary and its hardness is proved in the GGM [PS16].

Informal Definition 1. (GGM vs. AGM) The Generic Group Model (GGM), introduced by Shoup [Sho97], assumes a strong characteristic of adversaries: they can only interact with the group in a black-box manner. The Algebraic Group Model (AGM), introduced by Fuchsbaauer, Kiltz and Loss [FKL18], lies between the GGM and the standard model. The AGM is similar to the standard model, but it differs from the GGM in that the security game relies on the hardness of a mathematical problem. The AGM, on the other hand, is similar to GGM but differs from the standard model as an algebraic algorithm can only produce group elements by employing group operations on the given elements. Although an algebraic algorithm does not need to interact with an oracle to perform a computation, it must output a record of group operations performed, called representation vector.

Kim et al. [Kim+20] developed an expanded form of the PS assumption, termed the Generalized Pointcheval-Sanders (GPS) assumption. This version differs from the original PS assumption by dividing the oracle $\mathcal{O}^{\text{PS}}(\cdot)$ into two distinct oracles, $\mathcal{O}_0^{\text{GPS}}(\cdot)$, which randomly selects h from \mathbb{G}_1 , and $\mathcal{O}_1^{\text{GPS}}(\cdot)$, which, given h and $m \in \mathbb{Z}_p$, computes the value $[x + my]_h$. Subsequently, Kim et al. [Kim+22] further expand the GPS assumption to what is now referred to as GPS_2 , which modifies the oracle $\mathcal{O}_1^{\text{GPS}}(\cdot)$ such that it accepts group element messages of $[m]$ instead of field elements m as inputs.

Definition 6 (GPS_2 Assumption [Kim+22]). Given a tuple $([x]_2, [y]_2)$ and two oracles $\mathcal{O}_0^{\text{GPS}_2}() \rightarrow h \in \mathbb{G}_1$ and $\mathcal{O}_1^{\text{GPS}_2}(h, M_1, M_2) \rightarrow ([x]_h + [y]_{M_1}) \in \mathbb{G}_1$, where $x, y \leftarrow \mathbb{Z}_p$ and $\text{dlog}_h(M_1) = \text{dlog}_{\mathbb{G}_1}(M_2)$, the GPS_2 game i.e., $\mathbf{G}^{\text{GPS}_2}$, is to find (M_1^*, M_2^*, h^*, s^*) such that (1) $h^* \neq 1_{\mathbb{G}_1}$, $M^* \neq 1_{\mathbb{G}_1}$, (2) $s^* = [x]_{h^*} + [y]_{M_1^*}$, (3) $\text{dlog}_{h^*}(M_1^*) = \text{dlog}_{\mathbb{G}_1}(M_2^*)$ and (4) $(\star, M_1^*) \notin Q_1$, where Q_1 is the list of pairs (h, M_1) queried to $\mathcal{O}_1^{\text{GPS}_2}(\cdot)$. The advantage of an adversary \mathcal{A} against this game can be defined as $\text{Adv}_{\mathcal{A}}^{\text{GPS}_2}(\kappa) := \Pr [\mathbf{G}_{\mathcal{A}}^{\text{GPS}_2}(\kappa) = 1]$. The GPS_2 assumption holds if for all PPT adversaries \mathcal{A} , we have: $\text{Adv}_{\mathcal{A}}^{\text{GPS}_2}(\kappa) < \nu(\kappa)$.

The GPS_2 assumption is hard based on the (2,1)-DL assumption (cf. Definition 3) in the AGM and ROM.

Informal Definition 2. (Random Oracle Model (ROM)) Hash functions are a common design choice for many cryptographic schemes. The Random Oracle Model (ROM), introduced by Bellare and Rogaway in [BR93], is used to model these functions in the security proofs. A random oracle refers to a function that generates uniformly random outputs for each new query, while maintaining consistency by always producing the same output for an identical query. The main idea in a security proof in the ROM is to replace cryptographic hash functions with random oracles to prove the security, then transitioning back to actual hash functions for real-world deployments. While this model leads to more efficient cryptosystems, random oracles do not really exist in real world [PS00].

Definition 7 (Collision Resistant Hash Function (CRHF)). For a given security parameter κ and a family of functions $\mathcal{H} : \{0, 1\}^{\ell_{in}(\kappa)} \rightarrow \{0, 1\}^{\ell_{out}(\kappa)}$, where $\ell_{out}(\kappa) > \ell_{in}(\kappa)$, \mathcal{H} is a family of CRHF, if for any $H \in \mathcal{H}$ it is difficult to find a pair (X_1, X_2) such that $X_1 \neq X_2$ and $H(X_1) = H(X_2)$.

Example 2. SHA-256 is a well-known and practical hash function which produces outputs of 256 bits.

Matrix Assumptions. Matrix assumptions extend the DL-based assumptions and play a central role in the construction of many schemes [Esc+13; Her+14; MRV16; Esc+17].

Definition 8 (Matrix Distribution). For any natural numbers k and ℓ where $k \leq \ell$, consider $\mathcal{D}_{\ell,k}$ as a matrix distribution. This distribution produces matrices from $\mathbb{Z}_p^{\ell \times k}$ of full rank k in polynomial time. W.l.o.g., we let the first k rows of matrix $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}$ form an invertible matrix. When $\ell = k + 1$, we simply refer to the distribution as \mathcal{D}_k .

Example 3. As a simple example, let $k = 3$ and $\ell = 4$, meaning the matrix \mathbf{A} has 4 rows and 3 columns. Given $k = 3$, $\ell = 4$, and a finite field of prime order p , the following matrix \mathbf{A} satisfies the requirements:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & 3 & 4 \end{pmatrix}.$$

In this example, the first 3 rows form an identity matrix, which is clearly invertible, satisfying the requirement that the first k rows form an invertible matrix. The matrix \mathbf{A} has full rank $k = 3$, as the first k rows are linearly independent, and the addition of any row does not affect the rank.

Definition 9 ($\mathcal{D}_{\ell,k}$ -Matrix Decisional Diffie-Hellman ($\mathcal{D}_{\ell,k}$ -MDDH) Assumption [Esc+17]). For a given security parameter κ , consider two positive integers $k \in \mathbb{N}^*$ and $\ell \in \mathbb{N}^*$ s.t. $k < \ell$ and $\mathcal{D}_{\ell,k}$ be a matrix distribution as defined in Definition 8. The $\mathcal{D}_{\ell,k}$ -MDDH assumption for \mathbb{G}_ζ with $\zeta = \{1, 2\}$ holds, if for all PPT adversaries \mathcal{A} we have: $\text{Adv}_{\mathcal{D}_{\ell,k}, \mathbb{G}_\zeta, \mathcal{A}}^{\text{MDDH}}(\kappa) := |\varepsilon_1 - \varepsilon_0| \leq \nu(\kappa)$, where $\varepsilon_\beta := \Pr[\mathcal{A}(\mathcal{BG}, [\mathbf{A}]_\zeta, [\mathbf{Ar} + \beta \mathbf{u}]_\zeta) = 1]$, where $\mathcal{BG} \leftarrow \$ \text{BGgen}(1^\kappa)$, $\mathbf{A} \leftarrow \$ \mathcal{D}_{\ell,k}$, $\mathbf{r} \leftarrow \$ \mathbb{Z}_p^k$, $\mathbf{u} \leftarrow \$ \mathbb{Z}_p^\ell$.

Intuitively, this assumption extends the SXDH assumption and states that two distributions $([\mathbf{A}]_\zeta, [\mathbf{Ar}]_\zeta)$ and $([\mathbf{A}]_\zeta, [\mathbf{u}]_\zeta)$ are computationally indistinguishable.

Definition 10 (\mathcal{D}_k -Kernel Matrix Diffie-Hellman (\mathcal{D}_k -KerMDH) Assumption [MRV16]). For a given security parameter κ , consider a positive integer $k \in \mathbb{N}^*$, and let \mathcal{D}_k be a matrix distribution as defined in Definition 8. The \mathcal{D}_k -KerMDH assumption for \mathbb{G}_ζ with $\zeta = \{1, 2\}$ holds, if for all PPT adversaries \mathcal{A} , we have: $\text{Adv}_{\mathcal{D}_k, \mathbb{G}_\zeta, \mathcal{A}}^{\text{KerMDH}}(\kappa) = \Pr[\mathbf{c} \in \text{orth}(\mathbf{A}) \mid [\mathbf{c}]_{3-\zeta} \leftarrow \mathcal{A}(\mathcal{BG}, [\mathbf{A}]_\zeta)] \leq \nu(\kappa)$.

Definition 11 (Orthogonal of a Matrix). For any full rank matrix $\mathbf{A} \in \mathbb{Z}_p^{\ell \times k}$, let $\mathbf{A}^\perp \in \mathbb{Z}_p^{\ell \times (\ell-k)}$ be a matrix with $\mathbf{A}^\top \mathbf{A}^\perp = \mathbf{0}$ and $\text{rank}(\mathbf{A}^\perp) = \ell - k$. The set of all these matrices are denoted as follows:

$$\text{orth}(\mathbf{A}) := \{\mathbf{A}^\perp \in \mathbb{Z}_p^{\ell \times (\ell-k)} \mid \mathbf{A}^\top \mathbf{A}^\perp = \mathbf{0} \wedge \text{rank}(\mathbf{A}^\perp) = \ell - k\}.$$

The Kernel Matrix Diffie-Hellman assumption can be seen as a computational analogue to the Matrix Decisional Diffie-Hellman (MDDH) assumption. It is recognized that for any $k \geq 1$, \mathcal{D}_k -MDDH $\Rightarrow \mathcal{D}_k$ -KerMDH [KPW15; MRV16].

Example 4. As an example for the \mathcal{D}_2 -KerMDH assumption, let the random matrix $\mathbf{A} \leftarrow \$ \mathcal{D}_2$ be defined as follows:

$$\mathbf{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix} \in \mathbb{Z}_p^{3 \times 2},$$

where $a_1, a_2 \leftarrow \$ \mathbb{Z}_p^*$. Given $[\mathbf{A}]_\zeta$, i.e.,

$$[\mathbf{A}]_\zeta = \begin{pmatrix} [a_1]_\zeta & 0 \\ 0 & [a_2]_\zeta \\ [1]_\zeta & [1]_\zeta \end{pmatrix},$$

it is computationally hard to find $[\mathbf{c}]_{3-\zeta}$, where $\mathbf{c} := (c_1 \ c_2 \ c_3) \neq \mathbf{0}$, such that,

$$(c_1 \ c_2 \ c_3) \cdot \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix} = (a_1 c_1 + c_3 \ a_2 c_2 + c_3) = \mathbf{0}.$$

Knowledge of Exponent Assumption (KEA). Next, we recall the well-known Knowledge of Exponent Assumption (KEA), which is known as a basis for efficient Non-Interactive Zero-Knowledge (NIZK) proofs. Note that this assumption is categorized as a non-falsifiable assumption, meaning the validity of the adversary's outputs cannot be efficiently checked by the challenger.

Note 3. (Non-Falsifiable Assumptions) A common form of a non-falsifiable assumption is a statement of the form,

$$\forall \mathcal{A}, \exists \text{Ext}_{\mathcal{A}}(\cdot) : \Pr[\mathbf{G}_{\mathcal{A}}^{\text{GAME}} = 1 \wedge \text{Ext}_{\mathcal{A}}(\cdot) \text{ work}] \geq 1 - \nu(\kappa).$$

To break an assumption of this type we need to show adversary \mathcal{A} wins such that no algorithm $\text{Ext}_{\mathcal{A}}(\cdot)$ exists that works. Proving the non-existence of such an algorithm is challenging without additional assumptions.

The Knowledge of Exponent Assumption (KEA) as the first hardness assumption of this type was introduced by Damgård in [Dam92]. It states that it is infeasible to obtain a pair of elements C and \hat{C} from \mathbb{G} and $[\alpha]$ in a group \mathbb{G} , such that $\hat{C} = [\alpha]_C$, without prior knowledge of a , where $C = [a]$ and $\hat{C} = a[\alpha]$.

Next, we recall the Bilinear Diffie-Hellman Knowledge of Exponent (BDH-KE) assumption [Abd+17] which is a version of the original KEA over asymmetric bilinear groups.

Definition 12 (BDH-KE Assumption [Abd+17]). Consider an asymmetric bilinear group \mathcal{BG} , the BDH-KE assumption for a given relation $\mathbf{R} \in \mathcal{R}(1^\kappa)$, and PPT adversary \mathcal{A} , states that there exists a PPT extractor $\text{Ext}_{\mathcal{A}}(\cdot)$, s.t. we have:

$$\Pr \left[\begin{array}{l} r \leftarrow \$ \text{RND}(\mathcal{A}), ([\alpha_1]_1, [\alpha_2]_2) \leftarrow \mathcal{A}(\mathbf{R}, \mathcal{BG}, r), a \leftarrow \text{Ext}_{\mathcal{A}}(\mathbf{R}, r) : \\ e([\alpha_1]_1, \mathbf{G}_2) = e(\mathbf{G}_1, [\alpha_2]_2) \rightarrow (a = \alpha_1 \wedge a = \alpha_2) \end{array} \right] \geq 1 - \nu(\kappa).$$

Example 5. As a simple example, let $\mathcal{BG} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{G}_1, \mathbf{G}_2, e)$ be a Type-III bilinear group description. Given \mathcal{BG} , if adversary \mathcal{A} returns $([a]_1, [a]_2)$ such that the PPE, $e(\mathbf{G}_1, [a]_2) = e([a]_1, \mathbf{G}_2)$ holds, then we can conclude \mathcal{A} knows a . The main reason for this is that \mathcal{A} can choose a random element $[a]_1$ without knowing a . However, because the discrete logarithm in \mathbb{G}_1 is hard, \mathcal{A} cannot obtain a to compute $[a]_2$.

2.4 Digital Signatures

Digital Signatures (DS) emulate handwritten signatures which are commonly used in our society to signify the acceptance of some conditions. There are different reasons to sign a physical document, for example to agree about the details of a bank transaction or to confirm the postman has delivered an item to us. With the progress of the digital world, many physical documents have been substituted with digital content. Similarly, DS let us confirm the identity of the signer and validate that they have approved the content of a digital document.

A DS is typically defined with four processes. Initially, the setup phase generates the set of public parameters and the key generation produces a key pair, a private key and a public key. Following this, in the signing phase, the signer, using its private key, creates a signature for a given message. Lastly, the verification process involves checking the signature's authenticity by using the public key, the message, and the signature itself.

Definition 13 (Digital Signatures (DS)). More formally, a DS over message space \mathcal{M} consists of the following PPT algorithms:

- $\text{pp} \leftarrow \mathcal{DS}.\text{Setup}(\kappa)$: the probabilistic algorithm setup takes the security parameter κ as input and outputs the set of public parameters pp .¹
- $(\text{sk}, \text{vk}) \leftarrow \mathcal{DS}.\text{KeyGen}(\text{pp})$: the probabilistic algorithm key generation takes pp as input and outputs a pair of signing/verification keys (sk, vk) .
- $\sigma \leftarrow \mathcal{DS}.\text{Sign}(\text{pp}, \text{sk}, m)$: the probabilistic algorithm sign takes pp , a secret signing key sk , and a message $m \in \mathcal{M}$ as inputs, and outputs a signature σ .
- $0/1 \leftarrow \mathcal{DS}.\text{Verify}(\text{pp}, \text{vk}, m, \sigma)$: the deterministic algorithm verify takes pp , a public verification key vk , a message $m \in \mathcal{M}$, and a signature σ as inputs, and outputs either 1 (accept) or 0 (reject).

A DS is deemed secure if it satisfies two properties: *Correctness* and *Existential Unforgeability against Chosen Message Attack (EUF-CMA)*. Correctness ensures any correct signature must be verifiable exclusively with the signer's public key. However, EUF-CMA guarantees the process of creating the signature should be solely under the signer's control, and any changes to either the signature or the corresponding message must be easily identifiable.

Definition 14 (Correctness). A digital signature is correct for all public parameters $\text{pp} \leftarrow \text{Setup}(\kappa)$, key-pairs $(\text{sk}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp})$ and messages $m \in \mathcal{M}$, if: $\Pr [\text{Verify}(\text{pp}, \text{vk}, m, \text{Sign}(\text{pp}, \text{sk}, m)) = 1] \geq 1 - \nu(\kappa)$.

Definition 15 (Existential Unforgeability against Chosen Message Attack (EUF-CMA) [GMR88]). Given the security parameter κ , public parameters pp , verification key vk such that $(\text{sk}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp})$, and oracle $\mathcal{O}^{\text{Sign}}(m) \rightarrow \sigma$, the goal of the EUF-CMA game for a DS, i.e. $\mathbf{G}^{\text{EUF-CMA}}$, is to find a pair (m^*, σ^*) such that (1) $\text{Verify}(\text{pp}, \text{vk}, m^*, \sigma^*) = 1$ and (2) $m^* \notin \mathcal{Q}$, where \mathcal{Q} is the list of queried messages to oracle $\mathcal{O}^{\text{Sign}}(\cdot)$. A DS is called EUF-CMA-secure if for all PPT adversaries \mathcal{A} : $\text{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}(\kappa) := \Pr [\mathbf{G}_{\mathcal{A}}^{\text{EUF-CMA}}(\kappa) = 1] \leq \nu(\kappa)$.

Note 4. (Strong Unforgeability) A signature is said strongly unforgeable when the adversary is not only unable to generate a valid signature for a fresh message (i.e. passes the conditions of existential unforgeability) but also incapable of producing a new signature for a challenge message m^* , even after observing a valid signature for the same message m^* . In simpler terms, in a strongly unforgeable signature scheme, the signature itself cannot be altered or re-randomized.

Below we overview the fundamentals of Threshold Signatures (TS) as the core field of study for the constructions discussed in Chapter 3.

¹The setup algorithm is typically an auxiliary algorithm for digital signatures, and in this thesis we expect it to generate the bilinear group description.

2.5 Threshold Signatures

While Digital Signatures (DS) involve a single signer, an (n, t) -threshold signature scheme allows for distributed signing among n signers. In this case, any subgroup of at least t members can collectively sign a message, but a group smaller than t cannot successfully create a signature.

A Threshold Signature (TS) typically consists of five main phases. The initial stage, known as the setup phase, is responsible for generating a set of public parameters, while the key generation phase involves creating a unique key pair for each signer, consisting of a private share and its corresponding verification key and a universal verification key. During the partial signing phase, signers utilize their private shares to sign a message, producing a partial signature. The aggregation phase then combines a sufficient number of these valid partial signatures to produce a unified aggregated signature. The process concludes with the verification phase, where the integrity of the aggregated signature is checked using the universal verification key. When a signature/message pair successfully completes the verification phase, it signifies to the verifier that at least t out of n signers have participated in the partial signing process.

Example 6. As a simple example, when a student completes their academic program and is eligible to receive a university certificate, they request a certificate. The university may form a threshold signing group, which consists of multiple authorized administrators or officials, for example the administration office, the dean and the head of the ICT division. To issue a certificate, they can collectively generate Threshold Signatures (TS) on the certificate. This signature is only valid if a sufficient number of administrators agrees to sign.

Threshold signatures were proposed by Desmedt in 1989 to distribute the trust among multiple signers and to improve the key availability [Des90]. Since then, there have been many advancements in enhancing the security and efficiency of these schemes [Gen+01; MR01; GGN16; Lin17; Doe+18; Doe+19]. Additionally, due to the rise of cryptocurrencies, blockchain technology, and self-sovereign identity management, threshold signatures attracted significant research interest, e.g., [Bol03; KG20; Can+20; Kon+21; CKM23]. Currently, there is a standardization effort in the broader domain of threshold cryptography by NIST [BDV+20; BP23].

Note 5. (Non-Interactive Threshold Signatures) A TS is called non-interactive if the group of signers do not need to interact during the partial signing phase and no pre-processing phase is required. In this thesis, we focus on non-interactive threshold signatures over bilinear groups, which we will refer to simply as Threshold Signatures (TS).

Definition 16 ((n, t) -Threshold Signatures). Given a security parameter κ and a message space \mathcal{M} , an (n, t) -TS contains the following PPT algorithms:

- $\text{pp} \leftarrow \mathcal{TS.Setup}(\kappa)$: the setup algorithm takes the security parameter κ as input and outputs the set of public parameters pp .
- $(\{\text{sk}_i, \text{vk}_i\}_{i \in [1, n]}, \text{vk}) \leftarrow \mathcal{TS.KeyGen}(\text{pp}, n, t)$: the key generation algorithm takes the public parameters pp along with two integers n, t s.t. $1 \leq t \leq n$ as inputs. It then returns secret/verification keys $(\text{sk}_i, \text{vk}_i)$ for $i \in [1, n]$ along with a global verification key vk as output.
- $\Sigma_i \leftarrow \mathcal{TS.ParSign}(\text{pp}, \text{sk}_i, M)$: the partial signing algorithm takes pp , the i^{th} party's secret key, sk_i , and a message $M \in \mathcal{M}$ as inputs. It then returns a partial signature Σ_i as output.
- $0/1 \leftarrow \mathcal{TS.ParVerify}(\text{pp}, \text{vk}_i, M, \Sigma_i)$: the partial verification algorithm as a deterministic algorithm, takes pp , the i^{th} verification key, vk_i , and a message $M \in \mathcal{M}$ along with partial signature Σ_i as inputs. It then returns 1 (accept), if the partial signature is valid and 0 (reject), otherwise.
- $\Sigma \leftarrow \mathcal{TS.CombineSign}(\text{pp}, T, \{\Sigma_i\}_{i \in T})$: the combine algorithm takes a set of partial signatures Σ_i for $i \in T$ along with $T \subseteq [1, n]$ and $|T| \geq t$. It then returns an aggregated signature Σ as output.
- $0/1 \leftarrow \mathcal{TS.Verify}(\text{pp}, \text{vk}, M, \Sigma)$: the verification algorithm as a deterministic algorithm, takes pp , the global verification key, vk , and message $M \in \mathcal{M}$ along with an aggregated signature Σ as inputs. It then returns 1 (accept), if the aggregated signature is valid and 0 (reject), otherwise.

Note 6. (Distributed Key Generation (DKG)) While the key generation algorithm can be run by a DKG protocol [Ped92], throughout this thesis we assume the presence of a trusted dealer for sharing the pair of secret/verification keys among signers. DKG is an interactive protocol that allows for the secure creation of key pairs among multiple parties without a centralized dealer.

Security Properties. A threshold signature is called secure, i.e., unforgeable, if no one can come up with a valid signature even if $t-1$ of the signers are corrupted, i.e. act maliciously. Bellare et al. in [Bel+22] studied multiple notions of security for threshold signatures, particularly non-interactive threshold signatures. They discussed two primary security notions for non-interactive threshold signatures, labeled TS-UF-0 and TS-UF-1. In the TS-UF-0 notion, the adversary is limited in that it cannot observe any partial signatures of the challenge message for which it creates a forgery. On the other hand, the TS-UF-1 notion, as a stronger security notion, permits the adversary to make partial signing queries up to $(t - |\# \text{ corrupted signers}| - 1)$ times on the challenge message.

In finalizing our discussion on the security aspects of threshold signatures, we briefly explore how *static* and *adaptive* corruptions differ. A threshold signature is called secure against static adversaries when adversaries must declare all corrupted parties at the beginning of the security game, before having seen their verification keys. However, in practical scenarios, corruption often occurs over time. This necessitates protection against stronger adversaries, specifically those capable of adaptive corruption. In these situations, adversaries have the flexibility to select which signers to corrupt as the security game progresses, and this selection can be influenced by information gained from observing the set of public parameters and also having access to the partial signing oracle.

The formal definition of these unforgeability notions can be found in Definition 54 on page 117.

2.6 Secret Sharing Schemes

Secret sharing schemes allow to divide a secret among several parties, where only a quorum of a pre-determined size has the ability to reconstruct the secret. In this thesis, we use the Shamir secret sharing scheme [Sha79] as a well-known secret sharing mechanism that employs Lagrange Interpolation.

Shamir Secret Sharing Scheme. In an (n, t) -Shamir secret sharing, to share a secret $s \in \mathbb{Z}_p$ among n parties, the dealer forms a random polynomial, $f(x) \in \mathbb{Z}_p[X]$ of degree $(t-1)$ such that $f(0) = s$. To share the secret, the dealer evaluates $f(x)$ on each shareholder's index and securely provides each shareholder with $s_i = f(i), i \in [1, n]$. To reconstruct the secret, for a subset $S \subseteq [1, n]$ one can compute the Lagrange coefficients $\lambda_i := \Lambda_i(0) = \prod_{j \in S, j \neq i} \frac{j}{j-i}$, and then combine them with the shares, i.e., $s = f(0) = \sum_{i \in S} s_i \lambda_i$. The need for at least t polynomial evaluations arises from the degree of the initial polynomial and fewer than t cannot learn any information about s .

Definition 17 (Lagrange Interpolation). Given ℓ distinct interpolation points and evaluation points (α_i, ϕ_i) , using Lagrange interpolation we can find the unique polynomial $\Phi(x)$ of degree $\ell - 1$ such that $\Phi(\alpha_i) = \phi_i$ for all $i \in [1, \ell]$. In doing so we first compute Lagrange polynomials $\Lambda_i(x) := \prod_{j \in [1, \ell], j \neq i} \frac{x - \alpha_j}{\alpha_i - \alpha_j}$ for all $i \in [1, \ell]$.

It is easy to observe that for all $i \in [1, \ell]$, we have $\Lambda_i(\alpha_i) = 1$ and also for all $i \in [1, \ell]$ and $j \neq i$ we have $\Lambda_i(\alpha_j) = 0$. Then we can obtain $\Phi(x)$ as $\Phi(x) = \sum_{i \in [1, \ell]} \Lambda_i(x) \phi_i$.

Example 7. Taking a simple example where $t = 2$ and considering shares as points on a plane, with x-coordinate being the index and y-coordinate being the point evaluated by a random polynomial $f(x)$ of degree 1, i.e. $t - 1 = 1$, at this index. As shown in Figure 2.1, it is a basic fact that multiple lines can pass through any single point, yet only one unique line can connect two distinct points. Without knowing the second point it is not possible to interpolate the line and obtain the secret, which is its intersection with y -axis.

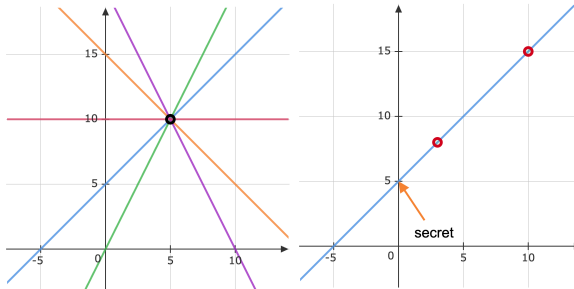


Figure 2.1: Polynomial Interpolation: the unique line through two points.

We further extend the Shamir secret sharing to enable sharing of matrices of dimension $a \times b$, essentially running ab instances of Shamir secret sharing in parallel. It is easy to observe that in the standard Shamir secret sharing scheme we have $a = b = 1$.

Definition 18 (Secret Sharing). More formally, for any two positive integers $n, t \leq n$, a secret-sharing scheme over $\mathbb{Z}_p^{a \times b}$ (for $a, b \in \mathbb{N}$) involves two main algorithms: **Share** and **Rec**. **Share** is a probabilistic function that, given a secret matrix \vec{M} from $\mathbb{Z}_p^{a \times b}$, produces n shares $(\vec{M}_1, \dots, \vec{M}_n)$, each belonging to $\mathbb{Z}_p^{a \times b}$. This scheme is defined by two security properties:

1. **Correctness** guarantees that from any t or more shares, the original secret \vec{M} can be accurately reconstructed.
2. **Security** ensures that any set of shares fewer than t provides no information about \vec{M} , maintaining its confidentiality against adversaries.

For more details, check Definition 47 on page 112.

2.7 Non-Interactive Zero-Knowledge Proofs

The concept of Zero-Knowledge (ZK) proofs, introduced by Goldwasser et al. in 1985 [GMR85], had a great impact on the field of cryptography. In simple terms, ZK proofs offer an efficient means of showing the validity of a statement without disclosing any additional information. The main security requirements of a ZK proof are that it should require the prover to know the secret to be proven, called knowledge soundness, and that no extra information is leaked during the proof process, called zero-knowledge. In this thesis, we focus on Non-Interactive Zero-Knowledge (NIZK) proofs, primarily due to their capability to eliminate the need for interaction between parties. This is important, especially when dealing with large-scale systems where achieving smooth and efficient interactions can be challenging.

Before exploring the formal definition of NIZKs, let's review some foundational notations.

Informal Definition 3. (Relations vs. Languages) For a security parameter κ , let \mathcal{R} be a relation generator, such that $\mathcal{R}(\kappa)$ returns an efficiently computable binary relation $\mathbf{R} = \{(x, w)\}$, where x is the statement and w is the corresponding witness (secret input). Let $\mathbf{L}_{\mathbf{R}} = \{x : \exists w \mid (x, w) \in \mathbf{R}\}$ be a language consisting of the statements in relation \mathbf{R} .

A relation can be defined as a set of pairs (x, w) and in a provable statement, i.e., Nondeterministic Polynomial Time (NP) relation, x is called instance and w is called witness. However, the language defined by a relation \mathbf{R} consists of all the valid instances such that there exists a witness satisfying the relation \mathbf{R} .

The following example inspired by the blog post by Mohnblatt², gives a better understanding of the differences between a statement (x), a witness (w), a relation (\mathbf{R}) and a language ($\mathbf{L}_{\mathbf{R}}$).

²https://nmohnblatt.github.io/zk-jargon-decoder/intro_to_zk/what_is_proving.html.

Example 8. We know some mathematical statements, like checking if “17 is a prime number!” or “99 is a prime number!”, are easily decidable. However, finding if the following Sudoku puzzle has an answer and finding the solution^a requires more effort.

1		3
3		
	3	

(★)

In the Sudoku puzzle, the relation $\mathbf{R}_{\text{Sudoku}}$ involves a grid consisting of integers and blanks, e.g. puzzle (★), as instance. However, the witness in this case refers to a vector of integers that correctly fills all the blanks in the puzzle, ensuring the Sudoku rules are met. Thus, we can write $(x := (\star), w := [2, 1, 2, 2, 1]) \in \mathbf{R}_{\text{Sudoku}}$.

The Sudoku puzzle (★) belongs to the language $\mathbf{L}_{\mathbf{R}_{\text{Sudoku}}}$. However, any Sudoku puzzle like the one below that does not have a solution, i.e. witness, cannot be in this NP-language.

1		3
2		
	3	

^aOne can fill in the blanks in such a way that each row and each column, contain numbers from 1 to 3 exactly once.

Definition 19 (Non-Interactive Zero-Knowledge (NIZK) proofs). Formally, a NIZK proof for an NP-relation $\mathbf{R} \in \mathcal{R}(\kappa)$ consists of the following PPT algorithms:

- $(\vec{c\mathbf{r}}\mathbf{s}, \vec{\mathbf{t}}\mathbf{s}, \vec{\mathbf{t}}\mathbf{e}) \leftarrow \mathcal{NIZK}.\text{Gen}_{\text{crs}}(\kappa, \mathbf{R})$: the CRS generation, i.e. setup, is a randomized algorithm that takes the security parameter κ and the relation \mathbf{R} as input(s), and outputs a Common Reference String (CRS), $\vec{c\mathbf{r}}\mathbf{s}$, along with a simulation and extraction trapdoors $\vec{\mathbf{t}}\mathbf{s}, \vec{\mathbf{t}}\mathbf{e}$. Note that the input \mathbf{R} is an arbitrary input element, where in some NIZK schemes the CRS needs to be generated under a certain relation, while others can be established without direct dependency on the relation itself.
- $(\pi, \perp) \leftarrow \mathcal{NIZK}.\text{Prove}(\mathbf{R}, \vec{c\mathbf{r}}\mathbf{s}, x, w)$: the prove algorithm is a probabilistic algorithm that takes the tuple $(\mathbf{R}, \vec{c\mathbf{r}}\mathbf{s}, x, w)$ as input. It then returns a proof π when $(x, w) \in \mathbf{R}$, otherwise it returns \perp .
- $0/1 \leftarrow \mathcal{NIZK}.\text{Verify}(\mathbf{R}, \vec{c\mathbf{r}}\mathbf{s}, x, \pi)$: the verify algorithm is a deterministic algorithm that takes the tuple $(\mathbf{R}, \vec{c\mathbf{r}}\mathbf{s}, x, \pi)$ as input. It either accepts (1) or rejects (0), checking whether $x \in \mathbf{L}_{\mathbf{R}}$.

- $\pi' \leftarrow \mathcal{NIZK}.\text{Sim}(\mathbf{R}, \text{crs}, \vec{\text{ts}}, x)$: the simulator algorithm takes the tuple $(\mathbf{R}, \text{crs}, \vec{\text{ts}}, x)$ as input and *without knowing the corresponding secret witness*, outputs a simulated proof π' such that it is computationally indistinguishable from π .³

Informal Definition 4. (Zero-Knowledge Succinct Non-interactive ARgument of Knowledge (zk-SNARK)) Generally speaking a zk-SNARK is essentially a knowledge sound NIZK with certain feature of succinctness. More concretely, the prover is (quasi-)linear, the proof is asymptotically short, i.e. for any κ we have $|\pi| \in O(\text{polylog}(|x| + |w|))$ and the verification time is fast, i.e. in the order of $O(|x|)$.

The most efficient zk-SNARK to date is proposed by Groth in 2016 [Gro16], Groth16 in short, and has constant proof size, only 3 source group elements and 3 pairing operations in the verification phase.

Perhaps the most powerful cryptographic technology to come out of the last decade is general-purpose succinct zero-knowledge proofs.

Vitalik Buterin, Inventor of Ethereum

The primary security properties of NIZK arguments are completeness, Zero-Knowledge (ZK), soundness and Knowledge Soundness (KS).

Note 7. (NIZK proofs vs. NIZK arguments) In a NIZK argument the security properties in Definitions 20 to 24 are computationally bounded, meaning the adversary's advantage is limited by a negligible function, $\nu(\kappa)$. However in a NIZK proof the winning conditions are deterministically equal to 0. In this thesis, we use the terms NIZK argument and NIZK proof, interchangeably.

Definition 20 (Completeness). A NIZK argument is called *complete* for relation $\mathbf{R} \in \mathcal{R}(\kappa)$, if for all κ and $(x, w) \in \mathbf{R}$, we have:

$$\Pr \left[(\text{crs}, \vec{\text{ts}}, \vec{\text{te}}) \leftarrow \text{Gen}_{\text{crs}}(\kappa, \mathbf{R}) : \text{Verify}(\mathbf{R}, \text{crs}, x, \text{Prove}(\mathbf{R}, \text{crs}, x, w)) \right] \geq 1 - \nu(\kappa) .$$

Definition 21 (Zero-Knowledge). A NIZK argument is called ZK, if for all security parameters κ , all PPT adversaries \mathcal{A} and relations $\mathbf{R} \in \mathcal{R}(\kappa)$ we have: $|\varepsilon_0 - \varepsilon_1| \leq \nu(\kappa)$, where,

$$\varepsilon_b := \Pr \left[(\text{crs}, \vec{\text{ts}}, \vec{\text{te}}) \leftarrow \text{Gen}_{\text{crs}}(\kappa, \mathbf{R}) : \mathcal{A}^{\mathcal{O}_{b, \vec{\text{ts}}}(\cdot, \cdot)}(\mathbf{R}, \text{crs}) = 1 \right] ,$$

³Note that the simulation is exclusively used within the security definitions and security proofs, as (ideally) there is no trapdoor in real-world deployments.

where the oracle $\mathcal{O}_{b,\vec{ts}}(\cdot, \cdot)$ is defined in Figure 2.2.

$\mathcal{O}_{b,\vec{ts}}(x, w) :$
1 : if $(x, w) \notin \mathbf{R}$ return \perp else :
2 : if $b = 0$: return $\pi \leftarrow \text{Prove}(\text{c}\vec{r}s, x, w)$
3 : else return $\pi \leftarrow \text{Sim}(\text{c}\vec{r}s, \vec{t}s, x)$

Figure 2.2: Simulation Oracle.

The main idea behind this definition is that if a simulator, with access to a secret trapdoor, can create a proof (without knowing the witness) that looks exactly like an original proof generated by the proving algorithm, then we can say that the proof itself does not contain information about the witness.

Definition 22 (Soundness). A NIZK argument is *sound* for any relation $\mathbf{R} \in \mathcal{R}(\kappa)$, if for all adversaries \mathcal{A} , we have:

$$\Pr \left[\begin{array}{l} (\text{c}\vec{r}s, \vec{t}s, \vec{t}e) \leftarrow \text{Gen}_{\text{crs}}(\kappa, \mathbf{R}), (x, \pi) \leftarrow \mathcal{A}(\mathbf{R}, \text{c}\vec{r}s) : \\ \text{Verify}(\mathbf{R}, \text{c}\vec{r}s, x, \pi) \wedge x \notin \mathbf{L}_{\mathbf{R}} \end{array} \right] \leq \nu(\kappa) .$$

Usually, we use the stronger notion of Knowledge Soundness (KS) which enables extracting a witness from a valid proof. Note that KS implies soundness.

Definition 23 (Knowledge Soundness (KS)). A NIZK argument is called *knowledge-sound*, if for all security parameters κ , all PPT adversaries \mathcal{A} and relations $\mathbf{R} \in \mathcal{R}(\kappa)$, there exists an extractor $\text{Ext}_{\mathcal{A}}$, and we have:

$$\Pr \left[\begin{array}{l} (\text{c}\vec{r}s, \vec{t}s, \vec{t}e) \leftarrow \text{Gen}_{\text{crs}}(\kappa, \mathbf{R}), (x, \pi) \leftarrow \mathcal{A}(\mathbf{R}, \text{c}\vec{r}s), \\ w \leftarrow \text{Ext}_{\mathcal{A}}(\mathbf{R}, \text{c}\vec{r}s, \vec{t}e, \pi) : \text{Verify}(\mathbf{R}, \text{c}\vec{r}s, x, \pi) \wedge (x, w) \notin \mathbf{R} \end{array} \right] \leq \nu(\kappa) .$$

Intuitively, in a knowledge sound NIZK, it is impossible for someone to create a fake proof that verifies, but the witness does not belong to the relation with a high probability. To show this, we can define an extractor in two possible settings: *Black-Box* and *Non-Black-Box*, that with the help of a secret trapdoor, can extract the witness from any verifiable proof.

Informal Definition 5. (Black-Box vs. Non-Black-Box Extraction)

We say an efficient extractor $\text{Ext}(\cdot)$ has black-box access to the adversary \mathcal{A} if it can run \mathcal{A} externally without having access to its source code. In contrast, in a non-black-box scenario^a, the extractor $\text{Ext}_{\mathcal{A}}(\cdot)$ has direct access to the adversary's source code, private coins, and hardcoded secrets. Note that in a non-black-box scenario, there is a specific extractor for each adversary, whereas in a black-box scenario, a single extractor works for all adversaries.

^aAlso known as white-box.

Below we recall a stronger notion of Simulation Extractability (SE) in two possible scenarios; strong SE and weak SE. This definition strengthens the KS since the adversary can observe a polynomially bounded number of simulated proofs.

Definition 24 (Weak and Strong Simulation Extractability (SE) [Gro06; Bag+21]). A NIZK argument (cf. Definition 19) is called weak strong *simulation extractable*, if for all security parameters κ , all PPT adversaries \mathcal{A} and relations $\mathbf{R} \in \mathcal{R}(1^\kappa)$, there exists an extractor $\text{Ext}_{\mathcal{A}}(\cdot)$, and we have:

$$\Pr \left[\begin{array}{l} (\vec{c\vec{r}s}, \vec{t\vec{s}}, \vec{t\vec{e}}) \leftarrow \text{Gen}_{\text{crs}}(\kappa, \mathbf{R}), (x, \pi) \leftarrow \mathcal{A}^{\mathcal{O}_{\vec{c\vec{r}s}, \vec{t\vec{s}}}}(\mathbf{R}, \vec{c\vec{r}s}), \\ w \leftarrow \text{Ext}_{\mathcal{A}}(\mathbf{R}, \vec{c\vec{r}s}, \vec{t\vec{e}}, \pi) : (x, w) \notin \mathbf{R} \wedge \text{Verify}(\mathbf{R}, \vec{c\vec{r}s}, x, \pi) \wedge \\ \left(\left[\begin{array}{c} \vdots \\ (x, \cdot) \notin Q_S \\ \vdots \end{array} \right] \vee \left[\begin{array}{c} \vdots \\ (x, \pi) \notin Q_S \\ \vdots \end{array} \right] \right) \end{array} \right] \leq \nu(\kappa) ,$$

where the oracle $\mathcal{O}_{\vec{c\vec{r}s}, \vec{t\vec{s}}}(\cdot)$ is defined in Figure 2.3.

$\mathcal{O}_{\vec{c\vec{r}s}, \vec{t\vec{s}}}(x) :$
1 : $\pi \leftarrow \text{Sim}(\vec{c\vec{r}s}, \vec{t\vec{s}}, x)$
2 : $Q_S \leftarrow Q_S \cup \{(x, \pi)\}$
3 : return π

Figure 2.3: Simulation Oracle for Simulation Extractability Property.

The idea behind this definition is that in a strong simulation extractable NIZK an adversary wins the Simulation Extractability (SE) game, if the proof verifies and then either confirms the extractor's success or if the proof has already

been simulated by the simulation oracle. To show this, we can define an extractor that, using the extraction trapdoor, is capable of extracting the witness from any verifiable proof that has not been previously simulated by the oracle, i.e. $(x, \pi) \notin Q_S$. However, in the weak SE, we permit randomization of proofs. Consequently, we adjust the requirement for a fresh instance and proof condition to ensure that only the instance has not been queried before to the simulation oracle. This adjustment comes primarily from the realization that in a NIZK system with proof randomization, the adversary can easily re-randomize the proof and pass the conditions in the strong SE. To elaborate more, the adversary can request the simulation oracle for an instance x and obtain the proof π and the simulation oracle updates $Q_S \leftarrow Q_S \cup \{(x, \pi)\}$. Due to the proof's re-randomizability, the adversary can alter π to a valid proof π' and return (x, π') , while the extractor fails since this is a simulated proof, and as a result, (x, π') does not belong to the set Q_S . Thus the adversary wins the strong SE game and we can conclude that the NIZKs with randomizable proof cannot fulfill the strong SE property.

Intuitively, the weak and strong notions of SE can be likened to the EUF-CMA and strong EUF-CMA security properties of Digital Signatures (DS) (cf. Definition 15). In the EUF-CMA (weak SE), it is infeasible to obtain a valid signature (a proof) for a fresh message (a fresh instance) even after observing multiple simulated signatures (proofs). Moreover, within the strong notion of EUF-CMA, it becomes impossible to modify a signature into a new form and succeed in the game. A NIZK with SE property is also often called Signature of Knowledge (SoK) [CL06; GM17].

Example 9. As a prime example, the proof in Groth'16 zk-SNARK [Gro16] is re-randomizable (i.e. its proof is malleable), thus as shown by Bagheri et al. [Bag+21] this famous zk-SNARK can only achieve weak SE.

2.8 Public Key Encryption

Public-Key Encryption (PKE) is a well-known cryptographic primitive that allows a user to encrypt a plaintext using a public key. The corresponding secret key holder can then decrypt the ciphertext and retrieve the plaintext.

Definition 25 (Public-Key Encryption (PKE)). More formally, a PKE consists of the following PPT algorithms:

- $(\text{sk}, \text{pk}) \leftarrow \mathcal{PKE}.\text{KeyGen}(\kappa)$: the key generation algorithm takes κ as input and outputs a secret/public key pair (sk, pk) .
- $\text{ct} \leftarrow \mathcal{PKE}.\text{Encrypt}(\text{pk}, m)$: the encryption algorithm takes a public key pk and a message m as inputs and outputs ct .
- $m/\perp \leftarrow \mathcal{PKE}.\text{Decrypt}(\text{sk}, \text{ct})$: the decryption algorithm takes a secret key sk and a ciphertext ct as inputs, and outputs either a message m or \perp .

The main security properties of a PKE scheme are *completeness* and *confidentiality*, defined by Indistinguishability under Chosen-Plaintext Attack (IND-CPA).⁴

Definition 26 (Correctness). A PKE satisfies correctness if for all $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(\kappa)$ and messages m we have:

$$\Pr[\text{Decrypt}(\text{sk}, \text{Encrypt}(\text{pk}, m)) = m] \geq 1 - \nu(\kappa) .$$

Definition 27 (Indistinguishability under Chosen-Plaintext Attack (IND-CPA)). A PKE meets IND-CPA security, if for all κ and PPT adversaries \mathcal{A} , we have:

$$\Pr \left[\begin{array}{l} (\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(\kappa), (m_0, m_1) \leftarrow \mathcal{A}(\text{pk}), b \leftarrow_{\$} \{0, 1\}, \\ \text{ct}_b \leftarrow \text{Encrypt}(\text{pk}, m_b), b' \leftarrow \mathcal{A}(\text{pk}, \text{ct}_b) : b' == b \end{array} \right] \leq \frac{1}{2} + \nu(\kappa) .$$

⁴Note that for a PKE, stronger notions of security, such as Indistinguishability under Chosen-Ciphertext Attack (IND-CCA), are also defined. However, in this thesis, we only focus on CPA-security.

CHAPTER

3

Threshold Structure-Preserving Signatures

Individually, we are one drop. Together, we are an ocean.

Ryunosuke Satoro

CONTENT SOURCE:

1. E. Crites, M. Kohlweiss, B. Preneel, M. Sedaghat, and D. Slamanig. “Threshold Structure-Preserving Signatures”. In: *Advances in Cryptology – ASIACRYPT 2023*. Ed. by J. Guo and R. Steinfeld. Singapore: Springer Nature Singapore, 2023, pp. 348–382. ISBN: 978-981-99-8724-5. DOI: [10.1007/978-981-99-8724-5_11](https://doi.org/10.1007/978-981-99-8724-5_11)

Contributions: I was the main author of this paper, focusing on the protocol's design and the formal security proofs. My contributions were critical in formally defining the primitive and security definitions. I also contributed to formally prove the hardness of the underlying assumption in the AGM and ROM models.

2. A. Mitrokotsa, S. Mukherjee, M. Sedaghat, D. Slamanig, and J. Tomy. “Threshold Structure-Preserving Signatures: Strong and Adaptive Security Under Standard Assumptions”. In: *Public-Key Cryptography – PKC 2024*. Ed. by Q. Tang and V. Teague. Cham: Springer Nature Switzerland, 2024, pp. 163–195. ISBN: 978-3-031-57718-5. DOI: [10.1007/978-3-031-57718-5_6](https://doi.org/10.1007/978-3-031-57718-5_6)

Contributions: In this work, I contributed as an experienced researcher in this field and helped the co-authors re-defining the formal definition of TSPS schemes and their security properties under a wider range of security notions. I was involved in the design of the proposed construction and describing the technical challenges during its design.

Structure-Preserving Signatures (SPS) were introduced by Abe et al. in [Abe+10]: these are standard digital signatures (cf. Section 2.4) over bilinear groups (cf. Definition 1) with certain properties. In these schemes, the message, signature, and verification keys are composed of elements from source groups \mathbb{G}_1 and \mathbb{G}_2 , and the verification of a signature is limited to group membership checks and the evaluation of some Pairing-Product Equation (PPE). As previously noted in Chapter 1, these features make them compatible with efficient proof systems such as Groth-Sahai (GS) proofs [GS08] and this compatibility makes them a key candidate for the development of efficient privacy-preserving cryptographic primitives in the standard model such as group signatures [Abe+10; LPY15], traceable signatures [Abe+11a], blind signatures [Abe+10; FHS15; Fuc+16], policy-compliant signatures [BMW21; BSW24] and anonymous credentials [Fuc11; Cam+15].

Informal Definition 6. (Groth-Sahai proofs [GS08]) GS proofs can prove the satisfiability of some quadratic equations over bilinear groups. This proof system is notably capable of proving the satisfiability of Pairing Product Equations (PPE) of the following form:

$$\prod_{i=1}^n e(A_i, \mathcal{Y}_i) \prod_{i=1}^m e(\mathcal{X}_i, B_i) \prod_{j=1}^m \prod_{i=1}^n \gamma_{i,j} e(\mathcal{X}_j, \mathcal{Y}_i) = T ,$$

where $\mathcal{X}_1, \dots, \mathcal{X}_m \in \mathbb{G}_1$, $\mathcal{Y}_1, \dots, \mathcal{Y}_n \in \mathbb{G}_2$ are the witnesses (\mathbf{w}), and $T \in \mathbb{G}_T$, $A_1, \dots, A_n \in \mathbb{G}_1$, $B_1, \dots, B_m \in \mathbb{G}_2$ and $\Gamma := \{\gamma_{i,j}\}_{j \in [1,m], i \in [1,n]} \in \mathbb{Z}_p^{m \times n}$ are constant public values, known as instances.

GS proofs are commit-and-prove systems, in which the prover proves that a quadratic equation is satisfying the committed assignments. This process involves two steps: initially, the prover commits to the hidden values, and then, it validates their correctness through a pre-defined relation.

Since their initial development, SPS schemes have seen numerous enhancements in areas such as achieving shorter signatures [Abe+11b; Abe+14; Gha16; Gha17b; AGO11; Abe+18a], proposing schemes under standard assumptions [Abe+12; CDH12; HJ12; KPW15; LPY15; JR17], and tight security reductions [Abe+17; JOR18; Gay+18; Abe+18b; Abe+19; CH20]. Despite all these advancements, Threshold Structure-Preserving Signatures (TSPS) were missing in the literature. In [Cri+23], we define them and propose the first TSPS and further enhance their design in [Mit+24].

3.1 Technical Challenges Towards Building a TSPS

In the process of developing the first TSPS, we encountered a series of technical challenges primarily due to the unique properties of these signatures. Next, we focus on building a *threshold-friendly SPS* which can serve as a basis for the design of TSPS. Once more, a digital signature is called SPS if,

1. The message, signature and verification keys are only source group elements.
2. To verify a signature of this type, only group membership test and evaluation of pairing product equations are allowed.

Note 8. While these two conditions are essential for a digital signature to be classified as a SPS, the second condition emphasizes that no non-linear operations, such as hash functions, are used during the verification phase. We refer to this requirement as the third condition.

Given the list of some existing Threshold Signatures and their properties in Table 6.1 on page 77, we recall the well-known BLS signatures [BLS01], which operates within Type-III bilinear groups (cf. Definition 1 on page 13). Note that BLS signatures can be configured in two ways: either emphasizing short signatures, where the signatures are elements of the first source group, \mathbb{G}_1 , and the verification keys with the second source group, \mathbb{G}_2 , or focusing on short verification keys, the opposite configuration. In below, we specifically recall the short signature approach.

BLS Signatures [BLS01]. Given a hash function¹, $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$, the BLS signature scheme is defined over the message space $\mathcal{M} := \{0, 1\}^*$ and consists of the following PPT algorithms:

- $\text{pp} \leftarrow \text{Setup}(1^\kappa)$: run $\text{BGGen}(\kappa)$, output $\text{pp} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{G}_1, \mathbf{G}_2, e)$.
- $(\text{sk}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp})$: sample $\text{sk} \leftarrow \mathbb{Z}_p^*$ and set $\text{vk} := [\text{sk}]_2$. Output (sk, vk) .
- $\sigma \leftarrow \text{Sign}(\text{pp}, \text{sk}, m)$: compute $\sigma := [\text{sk}]_{H(m)}$. Output σ .
- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, m, \sigma)$: if the Pairing-Product Equation (PPE), $e(H(m), \text{vk}) = e(\sigma, \mathbf{G}_2)$ holds, output 1 (accept); else, output 0 (reject).

¹This is a hash-to-curve function that can be modeled in the random oracle model.

Boldyreva [Bol03] proposed threshold BLS signature schemes. It is easy to observe that the single private signing key sk can be divided into multiple shares, sk_i , using Shamir secret sharing (cf. Section 2.6). To partially sign a message, the signer computes $\sigma_i := [\text{sk}_i]_{\text{H}(m)}$. If there are enough partial signatures then one can aggregate them and obtain $\sigma := [\text{sk}]_{\text{H}(m)}$ which is verifiable by the general verification key vk .

Although the BLS signature can easily be extended to a threshold signature, it does not meet the conditions for a threshold structure-preserving signature due to its reliance on hash functions during the verification phase, which are non-linear operations and fail to meet the third condition described in Note 3.1. Meanwhile, designing a BLS-style signature without hash functions is not trivial, as discussed in [Cri+23]. To construct a TSPS, the study turns to Pointcheval-Sanders signatures [PS16], PS in short, as the starting point. PS16 signatures also operate within Type-III bilinear groups, similar to BLS signatures.

Pointcheval-Sanders Signatures [PS16]. The PS16 signature scheme is defined over the message space \mathcal{M} of scalar messages $m \in \mathbb{Z}_p^*$ and consists of the following PPT algorithms:

- $\text{pp} \leftarrow \text{Setup}(\kappa)$: run $\text{BGgen}(\kappa)$, output $\text{pp} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \text{G}_1, \text{G}_2, e)$.
- $(\text{sk}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp})$: sample $x, y \leftarrow \mathbb{Z}_p^*$ and set $\text{sk} := (\text{sk}_1, \text{sk}_2) = (x, y)$ and $\text{vk} := (\text{vk}_1, \text{vk}_2) = ([x]_2, [y]_2)$. Output (sk, vk) .
- $\sigma \leftarrow \text{Sign}(\text{pp}, \text{sk}, m)$: sample $r \leftarrow \mathbb{Z}_p^*$ and compute $\sigma := (h, s) = ([r]_1, [x + my]_h)$. Output σ .
- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, m, \sigma)$: if $h \in \mathbb{G}_1, h \neq 1_{\mathbb{G}_1}$, and the pairing product equation $e(h, \text{vk}_1 + [m]_{\text{vk}_2}) = e(s, \text{G}_2)$ holds, output 1 (accept); else, output 0 (reject).

As formally proved in [PS16], PS signatures are EUF-CMA secure under the PS assumption (cf. Definition 5 on page 15).

PS signatures are not structure-preserving because their message is a field element, however in an SPS the message should be a source group element. Ghadafi [Gha16] proposed an SPS from the PS signatures over Diffie-Hellman message spaces.

Informal Definition 7. (Diffie-Hellman Message Space [Fuc09; Abe+10]) Over a Type-III bilinear group (cf. Definition 1), a message pair $(M_1, M_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ is called a Diffie-Hellman (DH) message space^a, i.e., \mathcal{M}_{DH} if $e(M_1, \mathbb{G}_2) = e(\mathbb{G}_1, M_2)$.

^aAlso known as Dual message space.

Ghadafi SPS [Gha16]. The Ghadafi SPS is defined over the DH message space $\mathcal{M}_{\text{DH}} := \mathbb{G}_1 \times \mathbb{G}_2$ and consists of the following PPT algorithms:

- $\text{pp} \leftarrow \text{Setup}(\kappa)$: run $\text{BGGen}(\kappa)$, output $\text{pp} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e)$.
- $(\text{sk}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp})$: sample $x, y \leftarrow \mathbb{Z}_p^*$ and set $\text{sk} := (\text{sk}_1, \text{sk}_2) = (x, y)$ and $\text{vk} := (\text{vk}_1, \text{vk}_2) = ([x]_2, [y]_2)$. Output (sk, vk) .
- $\sigma \leftarrow \text{Sign}(\text{pp}, \text{sk}, M_1, M_2)$: sample $r \leftarrow \mathbb{Z}_p^*$ and compute $\sigma := (h, s, t) = ([r]_1, [r]_{M_1}, [x]_h + [y]_s)$. Output σ .
- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, \sigma, M_1, M_2)$: if $h, s, t \in \mathbb{G}_1, h \neq 1_{\mathbb{G}_1}$, and both pairing product equations $e(h, M_2) = e(s, \mathbb{G}_2)$ and $e(t, \mathbb{G}_2) = e(h, \text{vk}_1)e(s, \text{vk}_2)$ hold, output 1 (accept); else, output 0 (reject).

As mentioned in Table 6.2 on page 78, and the fact that the Ghadafi SPS meets all the conditions to be considered a structure-preserving, it does not lead to a non-interactive TSPS. This limitation primarily comes from the variation in randomness sources among signers, rendering the secret reconstruction process in one round of communication impossible.

By merging these three digital signatures, and acknowledging that the source of randomness in BLS signatures is derived from hash functions which forms a consistent basis for partial signatures, we now propose our threshold-friendly SPS.

Note 9. (SPS Impossibility Results) Abe et al. in [Abe+11b] provide a list of impossibility results for SPS schemes that have directly influenced the design of our TSPS schemes. The authors showed that no unilateral SPS exists. Unilateral SPS are ones where signatures exclusively contain elements of one source group. It is known that it is impossible to even build unilateral SPS that are secure against random message attacks [Abe+11b]. Additionally, it is shown that no SPS with signatures of less than three group elements exists. However, in an asymmetric bilinear setting and over a DH message space [Abe+10], where the message space is dual in both source groups, Ghadafi [Gha16] has shown that building a unilateral SPS is indeed possible. Finally, it is also formally shown in [Abe+11b] that to check the validity of a SPS signature at least two PPEs are required.

3.2 A Threshold-Friendly SPS

In [Cri+23], we propose the first TSPS over a new but restricted classes of messages called indexed Diffie-Hellman (iDH) message space. We take Ghadafi SPS and slightly modify it to make it threshold-friendly. To be more precise, from Ghadafi SPS we can observe that the randomness is message agnostic and each signer samples distinct randomnesses to partially sign the message. However, we extend DH message spaces to iDH such that the randomness, similar to BLS signatures, comes from a hash function (random oracle in the security proof) instead.

Note 10. (Threshold-Friendly Digital Signatures) In this thesis, a digital signature is called threshold-friendly if it avoids certain forbidden operations in its signature elements. These restrictions include avoiding the inversion of the secret signing key or randomness, r , in the groups, e.g., $[1/r]_1$, not combining the secret key and fresh randomness in a group element, e.g., $[rsk]_1^a$, and excluding any powers of the secret signing key or randomness in the groups for powers greater than one, e.g., $[sk^2]_1$. These criteria help ensure the compatibility with non-interactive threshold schemes.

^aNote that this is nonlinear because, in the non-interactive threshold setting, each signer samples distinct randomness during the partial signing phase, and this prevents signature aggregation through Lagrange interpolation.

As an example, Ghadafi's SPS [Gha16] is not considered threshold-friendly. This is because in its design, the last element of the signature involves the multiplication of the secret signing key with fresh randomness, i.e. $[rsk_1]_1 + [rsk_2]_{M_1}$. However, the BLS signature is a threshold-friendly digital signature.

The hash function in BLS signatures take the message m as input and this might cause privacy issues when we want to build a *threshold blind signature* using this scheme. Thus, we use a more general term of "index" instead of the plain message as the input for this source of randomness. The indexing function can be an injective function such as commitments. For the sake of generality we can define the indexing function as $f : \mathcal{M} \rightarrow \mathcal{I}$, where \mathcal{M} is the message space and \mathcal{I} is the index space.

Informal Definition 8. (Blind Signatures) A digital signature is called a blind signature when during the signing phase the signer learns nothing about the message. The signing process takes a blinded format of the message, typically a commitment to the message. These signatures include an additional unblinding algorithm, which transforms a signature on the blinded message into a signature on the original message.

From DH to iDH. To bring the index to the message space, in [Cri+23] we extend the Diffie-Hellman (DH) message spaces to indexed Diffie-Hellman (iDH), defined as follows:

Informal Definition 9. (Indexed Diffie-Hellman message space [Cri+23]) Given a Type-III bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e) \leftarrow \text{BGgen}(1^\kappa)$, an index set \mathcal{I} , and a random oracle $H : \mathcal{I} \rightarrow \mathbb{G}_1$, $\mathcal{M}_{\text{iDH}}^H$ is an indexed Diffie-Hellman (iDH) message space if we have:

$$\mathcal{M}_{\text{iDH}}^H \subset \{(id, \tilde{M}) \mid id \in \mathcal{I}, m \in \mathbb{Z}_p, \tilde{M} = ([m]_{H(id)}, [m]_2) \in \mathbb{G}_1 \times \mathbb{G}_2\},$$

where for all $(id, \tilde{M}) \in \mathcal{M}_{\text{iDH}}^H, (id', \tilde{M}') \in \mathcal{M}_{\text{iDH}}^H, id = id' \Rightarrow \tilde{M} = \tilde{M}'$.

Figure 6.4 on page 85 depicts how one can obtain an iDH from an integer message $m \in \mathbb{Z}_p$. Next, we recall our proposed threshold-friendly SPS, called message-indexed SPS, from [Cri+23], which is crucial for the design of our TSPS.

Our Proposed Threshold-Friendly SPS [Cri+23]. Our proposed threshold-friendly SPS is called message-indexed SPS over the iDH message space $\mathcal{M}_{\text{iDH}}^H$, and consists of the following PPT algorithms:

- $\text{pp} \leftarrow \text{Setup}(\kappa)$: run $\text{BGGen}(1^\kappa)$, output $\text{pp} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e)$.
- $(\text{sk}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp})$: sample $x, y \leftarrow \mathbb{Z}_p^*$ and set $\text{sk} := (\text{sk}_1, \text{sk}_2) = (x, y)$ and $\text{vk} := (\text{vk}_1, \text{vk}_2) = ([x]_2, [y]_2)$. Output (sk, vk) .
- $\sigma \leftarrow \text{Sign}(\text{pp}, \text{sk}, \text{id}, M_1, M_2)$: run $H(\text{id})$ to obtain h' . If $e(h', M_2) = e(M_1, G_2)$, compute $\sigma := (h, s) = (h', [x]_h + [y]_{M_1})$.
- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, \sigma, M_1, M_2)$: if $h, s \in \mathbb{G}_1, h \neq 1_{\mathbb{G}_1}$, and both pairing product equations $e(h, M_2) = e(M_1, G_2)$ and $e(s, G_2) = e(h, \text{vk}_1)e(M_1, \text{vk}_2)$ hold, output 1 (accept); else, output 0 (reject).

In Section 6.4.4 on page 89, we further expand this SPS to support multi-messages iDH denoted by $\mathcal{M}_{\text{iDH}}^H$.

The above threshold-friendly SPS allows for partial re-randomization. As the verification process does not depend on the hash function H , it is possible to re-randomize the base h and M_1 individually. It is straightforward to see that verification remains valid even when the message and signature are randomized.

3.3 An Introduction to EUF-CiMA Security

Given the fact that the message and signature pairs in our proposed threshold-friendly SPS are partially re-randomizable, its unforgeability proof under the EUF-CMA security (cf. Definition 15 on page 21) is not trivial. This is mainly because the forgery checks in EUF-CMA only cover the freshness of the queried messages, however we need to check their freshness within their equivalence classes to avoid trivial forgeries.

Informal Definition 10. (Equivalence-Class of iDH messages)

Motivated by the notion of unforgeability in Structure-Preserving Signatures on Equivalence-Classes (SPS-EQ) [HS14; FHS19], the equivalence class for each iDH message $\tilde{M} = (M_1, M_2) \in \mathcal{M}_{\text{iDH}}^H$ can be defined as $\text{EQ}_{\text{iDH}}(M_1, M_2) = \{(M_1^r, M_2) \mid \forall r \in \mathbb{Z}_p\}$.

Additionally, the Existential Unforgeability against Chosen indexed Message Attack (EUF-CiMA) security limits the adversary to querying each index only once. The primary reason for this restriction is that, in our message-indexed

SPS scheme, the randomness is derived from the hash function, leading to identical indices producing the same random basis h . However, if the adversary were to query two different messages with the same index, it could lead to the second trivial forgery.

The formal definition of the EUF-CiMA security notion can be found in Definition 41 on page 86.

GPS₃ Assumption. Given the fact that our proposed SPS scheme is defined over iDH message spaces, we introduce an extension to the GPS₂ assumption (cf. Definition 6), called the GPS₃ assumption. This assumption generalizes GPS₂ to dual message spaces iDH by allowing that the message consists of both source groups of \mathbb{G}_1 and \mathbb{G}_2 . While this modification caused several challenges in the security proof, it enabled us to circumvent one of the SPS impossibility results, particularly avoiding the constraint of being a unilateral signature.

The formal definition of the GPS₃ assumption can be found in Definition 39 on page 82. Later in Theorem 1, we prove under the hardness of $(2, 1)$ -DL problem (cf. Definition 3) that this assumption holds in the AGM and ROM. Figure 6.7 on page 88 shows the relationship between these assumptions and the theorem.

3.4 TSPS over iDH Message Spaces

Given the formal definition of Threshold Signatures (TS) (cf. Definition 16 on page 23), next we recall the proposed TSPS over iDH message spaces from [Cri+23].

- $\text{pp} \leftarrow \text{Setup}(\kappa)$: run $\text{BGgen}(1^\kappa)$, output $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e)$.
- $(\{\text{sk}_i, \text{vk}_i\}_{i \in [1, n]}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp}, n, t)$: sample $x, y \xleftarrow{\$} \mathbb{Z}_p$ and run $\{x_i\}_{i \in [1, n]} \leftarrow \text{Share}(x, \mathbb{Z}_p, n, t)$ and $\{y_i\}_{i \in [1, n]} \leftarrow \text{Share}(y, \mathbb{Z}_p, n, t)$. output $\{\text{sk}_i := (x_i, y_i)\}_{i \in [1, n]}$ and $\{\text{vk}_i := (\text{vk}_{1i}, \text{vk}_{2i}) = ([x_i]_2, [y_i]_2)\}_{i \in [1, n]}$ along with $\text{vk} := ([x]_2, [y]_2)$.
- $\Sigma_i \leftarrow \text{ParSign}(\text{pp}, \text{sk}_i, M)$: parse $M := (id, M_1, M_2)$ and run $\text{H}(id)$ to obtain basis h' . If $e(h', M_2) = e(M_1, G_2)$, compute $\Sigma_i := (h, s_i) = (h', [x_i]_h + [y_i]_{M_1})$. Output Σ_i .
- $0/1 \leftarrow \text{ParVerify}(\text{pp}, \text{vk}_i, M, \Sigma_i)$: if $h, s_i \in \mathbb{G}_1, h \neq 1_{\mathbb{G}_1}$, and both PPEs $e(h, M_2) = e(M_1, G_2)$ and $e(s_i, G_2) = e(h, \text{vk}_{1i})e(M_1, \text{vk}_{2i})$ hold, output 1 (accept); else, output 0 (reject).

- $\Sigma \leftarrow \text{CombineSign}(\text{pp}, T, \{\Sigma_i\}_{i \in T})$: if for all $i \in T$ the basis h is the same and all partial signatures Σ_i pass the partial verification algorithm, output $\Sigma := (h, \prod_{i \in T} [\lambda_i]_{s_i})$.
- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, M, \Sigma)$: if $h, s \in \mathbb{G}_1, h \neq 1_{\mathbb{G}_1}$, and both PPEs $e(h, M_2) = e(M_1, G_2)$ and $e(s, G_2) = e(h, \text{vk}_1)e(M_1, \text{vk}_2)$ hold, output 1 (accept); else, output 0 (reject).

This TSPS scheme is TS-UF-0 secure against threshold EUF-CiMA unforgeability notion (cf. Definition 46 on page 93) based on the unforgeability of the proposed message indexed SPS discussed in Section 3.2 and static adversaries.

However, this TSPS scheme is the shortest possible scheme based on the SPS's impossibility results of [Abe+11b]. Its unforgeability is guaranteed based on a weak security notion of TS-UF-0 and under an interactive assumption. Next, we will briefly outline the key findings of the subsequent research and show how it resolves these issues.

3.5 TSPS from Standard Assumptions

The design of the initial TSPS was inspired by Ghadafi's SPS. However, for building a TSPS capable of handling arbitrary group vectors and being proved based on standard assumptions, we shifted our perspective and chose to build a TSPS starting from the SPS scheme by Kiltz et al. [KPW15], KPW15 in short. This SPS combines a one-time SPS with a randomized Pseudo-Random Function (PRF), offering a new basis for our construction. This approach differs from our initial starting point, enabling us to solve the open problems discussed in [Cri+23].

Informal Definition 11. (Pseudo-Random Functions (PRF) [GGM84]) A PRF produces random looking outputs in a deterministic way. These functions take a public input along with a secret key and generate an output in such a way that the output seems random, even though it is generated under a repeatable function.

KPW15 SPS [KPW15]. Given the formal definition of Digital Signatures (DS) (cf. Definition 13 on page 20), the KPW15 SPS is defined over arbitrary group vectors of \mathbb{G}_1 of size ℓ , i.e., $\mathcal{M} := \mathbb{G}_1^\ell$, and defined as follows:

- $\text{pp} \leftarrow \text{Setup}(\kappa)$: run $\text{BGgen}(1^\kappa)$. Output $\text{pp} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e)$.

- $(\text{sk}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp})$: sample $\mathbf{A}, \mathbf{B} \leftarrow_{\$} \mathcal{D}_k$ (cf. Definition 8 on page 17), $\mathbf{K} \leftarrow_{\$} \mathbb{Z}_p^{(\ell+1) \times (k+1)}$ and $\mathbf{U}, \mathbf{V} \leftarrow_{\$} \mathbb{Z}_p^{(k+1) \times (k+1)}$. Set $\text{sk} := (\mathbf{K}, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1, [\mathbf{B}]_1)$ and $\text{vk} := ([\mathbf{A}]_2, [\mathbf{U}\mathbf{A}]_2, [\mathbf{V}\mathbf{A}]_2, [\mathbf{K}\mathbf{A}]_2)$. Output (sk, vk) .
- $\sigma \leftarrow \text{Sign}(\text{pp}, \text{sk}, [\mathbf{m}]_1)$: sample random vector $\mathbf{r} \leftarrow_{\$} \mathbb{Z}_p^k$ and random integer $\tau \leftarrow_{\$} \mathbb{Z}_p$. Compute $\sigma := (\sigma_1, \sigma_2, \sigma_3, \sigma_4) := ([(1 \quad \mathbf{m}^\top)]_1 \mathbf{K} + \mathbf{r}^\top [\mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1, [\mathbf{r}^\top \mathbf{B}^\top \tau]_1, [\tau]_2)$. Output σ .
- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, \sigma, [\mathbf{m}]_1)$: if the following PPEs hold output 1 (accept); else, output 0 (reject).
 1. $e(\sigma_1, [\mathbf{A}]_2) = e([(1 \quad \mathbf{m}^\top)]_1, [\mathbf{K}\mathbf{A}]_2) e(\sigma_2, [\mathbf{U}\mathbf{A}]_2) e(\sigma_3, [\mathbf{V}\mathbf{A}]_2)$,
 2. $e(\sigma_2, \sigma_4) = e(\sigma_3, \mathbf{G}_2)$.

The EUF-CMA security of this SPS is proved under the hardness of the Kernel Matrix Diffie-Hellman (kerMDH) assumption (cf. Definition 10 on page 18) and the MDDH assumption (cf. Definition 9 on page 18) which are known as standard assumptions.

Meanwhile, to make the above SPS threshold-friendly we need to make a few small modifications to the KPW15 SPS. Motivated by seminal work of Kiltz and Wee [KW15], these adjustments involve shifting certain elements from the secret and verification keys to public parameters created during the setup phase. Additionally, to avoid non-linear operations in the signature components, the tag τ is derived from a Collision Resistant Hash Function (CRHF) applied to the message $[\mathbf{m}]_1$, rather than being randomly sampled by the signer. This approach streamlines the structure to support threshold capabilities and still remain secure. The modified version of KPW15 SPS is defined as follows:

Modified KPW15 SPS [KPW15]. The modified KPW15 SPS is defined over the same message space $\mathcal{M} := \mathbb{G}_1^\ell$, and it assumes the existence of a CRHF, $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$; it consists of the following PPT algorithms:

- $\text{pp} \leftarrow \text{Setup}(\kappa)$: run $\mathcal{BG} \leftarrow \text{BGgen}(1^\kappa)$, sample $\mathbf{A}, \mathbf{B} \leftarrow_{\$} \mathcal{D}_k$ and $\mathbf{U}, \mathbf{V} \leftarrow_{\$} \mathbb{Z}_p^{(k+1) \times (k+1)}$. Set and output $\text{pp} := (\mathcal{BG}, [\mathbf{A}]_2, [\mathbf{U}\mathbf{A}]_2, [\mathbf{V}\mathbf{A}]_2, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1, [\mathbf{B}]_1)$.
- $(\text{sk}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp})$: sample $\mathbf{K} \leftarrow_{\$} \mathbb{Z}_p^{(\ell+1) \times (k+1)}$ and set $\text{sk} := \mathbf{K}$ and $\text{vk} := [\mathbf{K}\mathbf{A}]_2$. Output (sk, vk) .
- $\sigma \leftarrow \text{Sign}(\text{pp}, \text{sk}, [\mathbf{m}]_1)$: sample the random vector $\mathbf{r} \leftarrow_{\$} \mathbb{Z}_p^k$. Obtain $\tau := H([\mathbf{m}]_1)$, and compute $\sigma := (\sigma_1, \sigma_2, \sigma_3, \sigma_4) :=$

$([(1 \quad \mathbf{m}^\top)]_1 \mathbf{K} + \mathbf{r}^\top [\mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1, [\mathbf{r}^\top \mathbf{B}^\top \tau]_1, [\tau]_2)$. Output σ .

- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, \sigma, [\mathbf{m}]_1)$: if the following PPEs hold output 1 (accept); else, output 0 (reject).

1. $e(\sigma_1, [\mathbf{A}]_2) = e([(1 \quad \mathbf{m}^\top)]_1, \text{vk}) \ e(\sigma_2, [\mathbf{U}\mathbf{A}]_2) \ e(\sigma_3, [\mathbf{V}\mathbf{A}]_2)$,
2. $e(\sigma_2, \sigma_4) = e(\sigma_3, \mathbf{G}_2)$.

This SPS is considered threshold-friendly as it avoids any forbidden operations in its signature elements. It serves as basis for the first TSPS with arbitrary group element messages proved in the standard model. We recall this TSPS below; for an in-depth details of this design, we refer the readers to Section 7.3.3 on page 118.

A new TSPS based on standard assumptions [Mit+24]. The first TSPS under standard assumptions is defined over the message space $\mathcal{M} := \mathbb{G}_1^\ell$ and assumes the existence of a CRHF, $\mathbf{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$; it consists of the following PPT algorithms:

- $\text{pp} \leftarrow \text{Setup}(\kappa)$: run $\mathcal{BG} \leftarrow \text{BGgen}(1^\kappa)$, sample $\mathbf{A}, \mathbf{B} \leftarrow \mathcal{D}_k$ and $\mathbf{U}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)}$. Set and output $\text{pp} := (\mathcal{BG}, [\mathbf{A}]_2, [\mathbf{U}\mathbf{A}]_2, [\mathbf{V}\mathbf{A}]_2, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1, [\mathbf{B}]_1)$.
- $(\{\text{sk}_i, \text{vk}_i\}_{i \in [1, n]}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp}, n, t)$: sample $\mathbf{K} \leftarrow \mathbb{Z}_p^{(\ell+1) \times (k+1)}$ and run $\mathbf{K}_1, \dots, \mathbf{K}_n \leftarrow \text{Share}(\mathbf{K}, \mathbb{Z}_p^{(\ell+1) \times (k+1)}, n, t)$. Set $\text{vk} := [\mathbf{K}\mathbf{A}]_2$ and $(\text{sk}_i, \text{vk}_i) := (\mathbf{K}_i, [\mathbf{K}_i \mathbf{A}]_2)$.
- $\Sigma_i \leftarrow \text{ParSign}(\text{pp}, \text{sk}_i, [\mathbf{m}]_1)$: sample the random vector $\mathbf{r}_i \leftarrow \mathbb{Z}_p^k$ and obtain $\tau := \mathbf{H}([\mathbf{m}]_1)$. Compute $\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4) := ([(1 \quad \mathbf{m}^\top)]_1 \mathbf{K}_i + \mathbf{r}_i^\top [\mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{r}_i^\top \mathbf{B}^\top]_1, [\mathbf{r}_i^\top \mathbf{B}^\top \tau]_1, [\tau]_2)$. Output Σ_i .
- $0/1 \leftarrow \text{ParVerify}(\text{pp}, \text{vk}_i, [\mathbf{m}]_1, \Sigma_i)$: parse $\Sigma_i := (\sigma_{1,i}, \sigma_{2,i}, \sigma_{3,i}, \sigma_{4,i})$. If the following PPEs hold output 1 (accept); else, output 0 (reject).
 1. $e(\sigma_{1,i}, [\mathbf{A}]_2) = e([(1 \quad \mathbf{m}^\top)]_1, \text{vk}_i) \ e(\sigma_{2,i}, [\mathbf{U}\mathbf{A}]_2) \ e(\sigma_{3,i}, [\mathbf{V}\mathbf{A}]_2)$,
 2. $e(\sigma_{2,i}, \sigma_{4,i}) = e(\sigma_{3,i}, \mathbf{G}_2)$.
- $\Sigma \leftarrow \text{CombineSign}(\text{pp}, T, \{\Sigma_i\}_{i \in T})$: parse $\Sigma_i = (\sigma_{1,i}, \sigma_{2,i}, \sigma_{3,i}, \sigma_{4,i})$ for all $i \in T$. Compute the Lagrange polynomials λ_i for $i \in T$. Output $\Sigma := (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4)$, where $\hat{\sigma}_1 := \sum_{i \in T} \lambda_i \sigma_{1,i} = \left[(1 \quad \mathbf{m}^\top) \sum_{i \in T} \lambda_i \mathbf{K}_i \right]_1 +$

$$\sum_{i \in T} \lambda_i \mathbf{r}_i^\top [\mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, \hat{\sigma}_2 := \sum_{i \in T} \lambda_i \sigma_{i,2} = \left[\sum_{i \in T} \lambda_i \mathbf{r}_i^\top \mathbf{B}^\top \right]_1 = [\mathbf{r}^\top \mathbf{B}^\top]_1,$$

$$\hat{\sigma}_3 := \sum_{i \in T} \lambda_i \sigma_{i,3} = \left[\sum_{i \in T} \tau \lambda_i \mathbf{r}_i^\top \mathbf{B}^\top \right]_1 = [\tau \mathbf{r}^\top \mathbf{B}^\top]_1 \text{ and } \hat{\sigma}_4 := \sigma_4.$$

- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, \Sigma, [\mathbf{m}]_1)$: if the following PPEs hold output 1 (accept); else, output 0 (reject).

1. $e(\sigma_1, [\mathbf{A}]_2) = e([\mathbf{1} \quad \mathbf{m}^\top]_1, [\mathbf{KA}]_2) \cdot e(\sigma_2, [\mathbf{UA}]_2) \cdot e(\sigma_3, [\mathbf{VA}]_2)$,
2. $e(\sigma_2, \sigma_4) = e(\sigma_3, \mathbf{G}_2)$.

As formally proved in Theorem 6 on page 126, this TSPS is TS-UF-1-secure against static adversaries (cf. Definition 54 on page 117) under the hardness of the \mathcal{D}_k -MDDH assumption in \mathbb{G}_1 (cf. Definition 9 on page 18) and the \mathcal{D}_k -KerMDH assumption in \mathbb{G}_2 (cf. Definition 10 on page 18). In Theorem 7 on page 128, we show TS-UF-1-security against adaptive adversaries.

However, achieving these advanced security features results in certain trade-offs, such as increased sizes of signatures and verification keys, along with more PPEs during the verification phase. Table 7.2 on page 109 compares these two TSPS schemes.

3.6 Application to Threshold-Issuance Anonymous Credential (TIAC)

Safeguarding our digital identities is shifted from the basic username and password combinations to sophisticated multi-factor authentication mechanisms. Authentication ensures that individuals accessing online services are who they claim to be, and it forms the foundation of trust in digital interactions. However, existing authentication methods that involve human factors such as fingerprint scanning, facial recognition and more, allow service providers (verifiers) to gather and exchange user-specific data, creating comprehensive user profiles without the user's consent. Anonymous Credentials (AC) were introduced by Chaum [Cha85], as a now well-studied cryptographic tool which enables a reliable and privacy-preserving authentication mechanism.

The “ACS protocol”², Idemix³, and U-Prove⁴ are some prime examples of open-source AC systems. Although there are multiple ways to design an

²<https://github.com/facebookresearch/acs>.

³<https://github.com/IBM/idemix>.

⁴<https://www.microsoft.com/en-us/research/project/u-prove/>.

AC, a prominent approach is to use re-randomizable signatures [CL03; CL04; Lib+16; PS16]. Additionally, there are similar techniques such as redactable signatures [Cam+15; San20], and equivalence class signatures [HS14; FHS19; CL19; HS21; CLP22] which yields more efficient designs.

Example 10. To better understand AC systems, consider a simple scenario where a user possesses a set of attributes certified by a central authority, such as being under 26 years old ($[\text{age}] < \text{"26"}$), affiliated with KU Leuven ($[\text{affiliation}] = \text{"KU Leuven"}$), and being a PhD student ($[\text{role}] = \text{"PhD_Student"}$). Imagine this user needs to show it is under a certain age to be eligible for a public transport discount, e.g. $[\text{age}] < \text{"26"}$.

Now, consider a different situation where the same user wants to prove its status as a PhD student at KU Leuven, without disclosing its age, to register a conference with student rate ($[\text{affiliation}] = \text{"KU Leuven"} \ \& \ [\text{role}] = \text{"PhD_Student"}$).

In AC, selective disclosures of user information is important; it enables to reveal specific details only when necessary to meet particular requirements. Additionally, unlinkability ensures that engaging in multiple interactions with either the same or different verifiers remains untraceable. This is crucial because failing to do so could lead to the de-anonymization of users within the system, potentially compromising their privacy. To achieve this, most of the existing AC systems in the literature rely on Non-Interactive Zero-Knowledge (NIZK) (see Section 2.7 on page 26).

In the context of AC systems, NIZK proofs enable users to prove their knowledge of a credential, i.e., a valid digital signature on a specific subset of certified attributes, without revealing the credential itself. The zero-knowledge property of NIZK proofs ensures that during the authentication phase and even when observing multiple requests, verifier(s) cannot extract any extra information beyond the validity of the statements. It is important to note that while NIZK proofs offer unique and valuable features, they can introduce computational overheads in the system, potentially slowing down the authentication process. Exploring ways to reduce dependency on NIZK proofs or even eliminate their use altogether is an interesting field of study, as it may lead to further improvements in the system's performance and functionality.

On the other hand, AC constructions are prone to compromise since they rely on a single credential issuer authority such as Google, Meta or governments, to manage user identities, which is a single point of failure. Consequently, there is a growing attention towards developing decentralized AC systems, such as Threshold-Issuance Anonymous Credential (TIAC), called Coconut [Son+19],

that enables a subset of credential issuers to jointly generate credentials. This improves the availability and also solves the single point of failure problem of centralized AC schemes.

Example 11. (Fake Vaccine Certificates) A notable example of credential fraud occurred during the COVID-19 pandemic, with some entities in various countries issuing false vaccination certificates [Geo+23]. This highlights the challenges and risks associated with managing certification processes.

Coconut, and some follow-up works such as PEReDi [KKS22], Coconut⁺⁺ [RP22; RP23] and PARscoin [SKK23], rely on a threshold version of PS signatures [PS16] (cf. Section 3.1 on page 37) and distributes the signing phase among n credential issuers where a cooperation of any t of them is required to generate a valid credential. However, as we discuss in Section 6.6 on page 99, we can replace this threshold signature with our proposed TSPS schemes. By retaining the structure of the scheme and avoiding structure-destroying primitives such as hash functions, our suggested TSPS is compatible with Groth-Sahai (GS) proofs [GS08], which makes them an attractive building block for many more complex (privacy-preserving) cryptographic protocols. Since GS proofs are straight-line extractable, they are particularly interesting for constructions targeting security in composable security frameworks such as the Universally Composable (UC) framework [Can01].

Informal Definition 12. (Universally Composable (UC) framework)

A UC protocol [Can01] operates without interfering with other protocols and can be arbitrarily composed with other protocols. In simpler terms, a protocol Ψ is considered UC-secure if you cannot tell the difference between interacting with it and interacting with an ideal process that performs the same task perfectly. Consider an adversary \mathcal{A} attempting to interfere with the protocol and the participating parties. In an ideal scenario, this adversary and some dummy parties would interact perfectly with an ideal functionality, let's call it F_Ψ . Hence for a protocol to be truly UC-secure, no one should be able to figure out if they are dealing with the real-world protocol or this perfect, idealized version. Formally, this means that the outcomes of these two scenarios, i.e. the ideal world and real world should look identical for all environments \mathcal{Z} .

Note that proving the knowledge of a PS-style signature requires Schnorr-style proofs. The Fiat-Shamir transform is typically used to make these proof systems non-interactive in the random oracle model. However, to achieve a proof of knowledge, the extractor must be able to rewind the adversarial prover.

This approach, known as a non-black-box technique, compromises concurrent composability and UC security. As shown by Rial and Piotrowska [RP22], for Coconut to be proven in the UC framework, we need to either use the Fischlin transform or traditional black-box extraction techniques, such as encrypting the witness. The primary obstacles to using the Fischlin transform in practice are its computational cost and implementation complexity. However, very recently, [CL24] evaluated the running time of this transform and suggested several optimizations. Despite these improvements, it still presents an additional overhead compared to proof systems that support straight-line extractability, such as GS proofs, which inherently support black-box extraction.

In summary, the introduced TSPS schemes, especially the later scheme that supports arbitrary group vector messages [Mit+24], can serve as a drop-in replacement to current SPS schemes. Consequently, any complex system such as AC, e-cash, electronic voting and others designed using these schemes can be instantiated with this TSPS. It then reduces the risk of key compromise and additionally make the output of the system GS proofs-friendly. This capability enables achieving UC security with minimal modifications in the initial design.

CHAPTER

4

Universal and Updatable
NIZKs: Security & Trust

A hair divides what is False and True.

Omar Khayyam

CONTENT SOURCE:

1. K. Bagheri, A. Mertens, and M. Sedaghat. “Benchmarking the Setup of Updatable zk-SNARKs”. In: *8th International Conference on Cryptology and Information Security in Latin America, LATINCRYPT*. ed. by A. Aly and M. Tibouchi. Vol. 14168. Lecture Notes in Computer Science. Springer, 2023, pp. 375–396. DOI: [10.1007/978-3-031-44469-2_19](https://doi.org/10.1007/978-3-031-44469-2_19)

Contributions: In this paper, my contributions revolved around addressing technical challenges to achieve the main achievements and novel ideas. Additionally, I assisted the co-authors on forming the main PPE checks for the setup of existing universal and updatable zk-SNARKs. Furthermore, I participated in developing an early version of proof-of-concept implementations.

2. K. Bagheri and M. Sedaghat. “Tiramisu: Black-Box Simulation Extractable NIZKs in the Updatable CRS Model”. In: *Cryptology and Network Security (CANS)*. ed. by M. Conti, M. Stevens, and S. Krenn. Cham: Springer International Publishing, 2021, pp. 531–551. ISBN: 978-3-030-92548-2. DOI: [10.1007/978-3-030-92548-2_28](https://doi.org/10.1007/978-3-030-92548-2_28)

Contributions: In this paper, I mainly worked on the definition of a novel primitive, called the key-updatable public key encryption scheme and its security proofs. Moreover, I actively participated in the proof of theorems in this paper. Additionally, I developed an open source proof-of-concept for the proposed schemes.

As discussed in the previous chapters, Non-Interactive Zero-Knowledge (NIZK) proofs enable a prover to reveal nothing beyond the truth of a statement to a verifier. Non-interactivity implies that the prover can generate a single proof element π without requiring any challenge or response from the verifier. Conversely, the verifier can check the proof's validity using only the proof itself, i.e. π , and certain public parameters known as Common Reference String (CRS).¹

Note 11. (CRS, URS and SRS) In the rest of this thesis, we primarily use the term of Structured Reference String (SRS) instead of Common Reference String (CRS). However, as noted in [Ben+22], the CRS covers a broader category, representing any output obtained from the setup algorithm, in two possible formats: Uniform Random String (URS) and Structured Reference String (SRS). URS refers to any random output obtained from a specific distribution without including of any secrets during its generation which yields to a transparent setup. SRS specifically denotes an output that involves a secret trapdoor during its creation either in *universal* or *non-universal* setting.

As formally defined in Definition 19, a NIZK is designed to fulfill several security properties: (1) Completeness ensures that an honest prover can always successfully convince any verifier. (2) Soundness prevents a dishonest prover from convincing an honest verifier on wrong statements. (3) Zero-Knowledge ensures that a proof, created honestly, does not reveal any information about the hidden witness.

Stronger Notions of Soundness. Although completeness and zero-knowledge are well-defined, in some applications the soundness definition (cf. Definition 22 on page 29) is not sufficient. For this aim, some stronger variations for soundness definition have been discussed: (4) Simulation Soundness ensures a dishonest prover cannot convince an honest verifier on a fake proof even after observing a number of simulated proofs. (5) Knowledge Soundness (KS) (cf. Definition 23 on page 29) guarantees that a dishonest prover cannot convince an honest verifier if it does not *know* a witness for the given statement. (6) Simulation Extractability (SE) (cf. Definition 24 on page 30), also known as Simulation Knowledge Soundness, strengthens the previous security notions by ensuring a dishonest prover cannot convince an honest verifier, even after observing many

¹Note that another family of NIZKs are defined in the Random Oracle Model (ROM), however in this thesis we only focus on NIZKs in the Structured Reference String (SRS) model.

simulated proofs, unless it knows a valid witness for the given statement in two possible scenarios: *Black-Box* and *non-Black-Box*.

NIZKs with subverted SRS. As discussed in Chapter 1, (pre-processing) NIZK requires a trusted setup phase. Particularly, constructing zk-SNARKs in the SRS model necessitates both the prover and verifier to trust the SRS generator. To address this trust concern, in 2015, Ben Sasson et al. [Ben+15] introduced an efficient Multi-Party Computation (MPC) protocol capable of computing a SRS for some pairing-based zk-SNARKs. In this case, both prover and verifier need to trust only 1 out of n parties, rather than relying entirely on a single party, where n denotes the number of participating parties in the MPC protocol [BGM17; BGG19; Abd+19]. In a separate research direction, Bellare et al. [BFS16] examined the security of NIZK arguments in the event of subverted SRS. They expanded the security properties of NIZKs in the following way: (7) Subversion Soundness (Sub-SND) guarantees the protocol’s soundness even if \mathcal{A} manipulates the SRS, while (8) Subversion Zero-Knowledge (Sub-ZK) ensures Zero-Knowledge (ZK) even if \mathcal{A} manipulates the SRS.

Bellare et al. [BFS16] show an impossibility result: achieving both Sub-ZK (Zero-Knowledge without reliance on a third party) and Black-Box Knowledge Soundness (KS) simultaneously is impossible.

4.1 NIZK in the Universal and Updatable SRS Model

At CRYPTO’18, Groth et al. [Gro+18] introduced a new family of NIZKs known as NIZK in the *universal* and *updatable* SRS model. In these constructions, universality allows for the creation of SRS components without knowing the circuit (circuit agnostic), while the feature of updatability extends the MPC-based setup ideas such that it permits continued participation in the setup phase and ensures the engagement of a wide range of contributors. In addition, the authors extend the security properties of NIZKs by considering the scenarios where the SRS components could be set up or updated maliciously. However, the Groth et al. construction results in a SRS size that is quadratic in the maximum number of constraints in a circuit; many follow-up works such as Sonic [Mal+19], Plonk [GWC19], Marlin [Chi+19], Lunar [Cam+21], Basilisk [RZ21], and Counting Vampires [LSZ22] reduced this to linear size.

Note 12. (Updatable NIZKs) The family of NIZKs in the universal and updatable SRS model is valuable due to their dual attributes of universality and the ability to update the SRS components. In the subsequent sections of this thesis, our primary emphasis lies on exploring this updatability feature and we usually refer to them as updatable NIZKs.

In such a setting, one may also need to modify the properties in order to capture the updatability feature.

- ⑨ Updatable Knowledge Soundness (Upd-KS) guarantees KS as long as the initial SRS or one of SRS updates is performed honestly, and
- ⑩ Updateable Zero-Knowledge (Upd-ZK) ensures Zero-Knowledge (ZK) under similar conditions.
- ⑪ Updateable non-Black-Box Simulation Extractability (Upd-nBB-SE) and Updateable Black-Box Simulation Extractability (Upd-BB-SE) ensure Simulation Extractability (SE) with non-Black-Box and Black-Box extractions, respectively, such that either the initial SRS or at least one of the updates are computed, honestly. For the sake of completeness, below we overview the formal definition of NIZKs in the universal and updatable SRS model, along with their main security properties defined by Groth et al. [Gro+18].

Definition 28. (NIZK in the Universal and Updatable SRS model [BMS23, Section 4] [Gro+18]). More formally, a NIZK in the updatable SRS model for \mathcal{R} consists of the following PPT algorithms:

- $(\text{srs}_0, \Pi_{\text{srs}_0}, \vec{\text{ts}}_0, \vec{\text{te}}_0) \leftarrow \text{Gen}_{\text{srs}}(\kappa, \underline{\mathbf{R}})$: given security parameter κ and $\underline{\mathbf{R}} \in \mathcal{R}(1^\kappa)$, sample the trapdoors $\vec{\text{ts}}'_0$ and $\vec{\text{te}}'_0$ and then generate srs_0 along with Π_{srs_0} as a proof for its well-formedness. Then, store the trapdoors associated with srs_0 including the simulation trapdoor $\vec{\text{ts}}_0 := \vec{\text{ts}}'_0$, and the extraction trapdoor $\vec{\text{te}}_0 := \vec{\text{te}}'_0$. Finally, return $(\text{srs}_0, \Pi_{\text{srs}_0})$ as the output.
- $(\text{srs}_i, \Pi_{\text{srs}_i}, \vec{\text{ts}}_i, \vec{\text{te}}_i) \leftarrow \text{SU}(\underline{\mathbf{R}}, \text{srs}_{i-1})$: given the tuple of $\underline{\mathbf{R}}$ and the most recent SRS, srs_{i-1} , where srs_{i-1} is an input SRS, return the pair of $(\text{srs}_i, \Pi_{\text{srs}_i})$, where srs_i is the updated SRS and Π_{srs_i} is a proof of correct updating. Note that after each update, the simulation and extraction trapdoors are updated, for instance $\vec{\text{ts}}_i := \vec{\text{ts}}_{i-1} + \vec{\text{ts}}'_i$, and $\vec{\text{te}}_i := \vec{\text{te}}_{i-1} + \vec{\text{te}}'_i$.
- $(\perp, 1) \leftarrow \text{SV}(\text{srs}_i, \Pi_{\text{srs}_i})$: given a potentially updated srs_i , and Π_{srs_i} return either \perp on the condition that the srs_i is incorrectly formed or 1.

- $(\pi, \perp) \leftarrow \text{Prove}(\mathbf{R}, \text{srs}_i, \mathbf{x}, \mathbf{w})$: for $\text{SV}(\text{srs}_i, \Pi_{\text{srs}_i}) = 1$, given the tuple of $(\mathbf{R}, \text{srs}_i, \mathbf{x}, \mathbf{w})$, such that $(\mathbf{x}, \mathbf{w}) \in \mathbf{R}$, output an argument π . Otherwise, return \perp .
- $(0, 1) \leftarrow \text{Verify}(\mathbf{R}, \text{srs}_i, \mathbf{x}, \pi)$: for $\text{SV}(\text{srs}_i, \Pi_{\text{srs}_i}) = 1$, given the set of parameters as $(\mathbf{R}, \text{srs}_i, \mathbf{x}, \pi)$, return either 0 (reject π) or 1 (accept π).
- $(\pi) \leftarrow \text{Sim}(\mathbf{R}, \text{srs}_i, \vec{\text{ts}}_i, \mathbf{x})$: for $\text{SV}(\text{srs}_i, \Pi_{\text{srs}_i}) = 1$, given the tuple $(\mathbf{R}, \text{srs}_i, \vec{\text{ts}}_i, \mathbf{x})$, where $\vec{\text{ts}}_i$ is the simulation trapdoor associated with the latest SRS, namely srs_i , output a simulated argument π .

Below we recall various security requirements that a NIZK argument can satisfy in the *updatable* SRS model [Gro+18]. Note that in the following definitions, i is the index of the final update, and w.l.o.g, we assume the initial SRS generation is done honestly while \mathcal{A} can maliciously update it to $\{\text{srs}_j\}_{j=1}^i$. We denote the malicious updater by Sub .

Definition 29 (Updatable Completeness). A NIZK with Updatable SRS is said to be *updatable complete* for \mathcal{R} , if for all $\mathbf{R} \in \mathcal{R}(1^\kappa)$, and $(\mathbf{x}, \mathbf{w}) \in \mathbf{R}$, we have,

$$\Pr \left[\begin{array}{l} (\text{srs}_0, \Pi_{\text{srs}_0}) \leftarrow \text{Gen}_{\text{srs}}(\mathbf{R}), \\ (\{\text{srs}_j, \Pi_{\text{srs}_j}\}_{j=1}^i) \leftarrow \mathcal{A}(\mathbf{R}, \text{srs}_0), \{\text{SV}(\text{srs}_j, \Pi_{\text{srs}_j}) = 1\}_{j=0}^i : \\ (\mathbf{x}, \pi) \leftarrow \text{Prove}(\mathbf{R}, \text{srs}_i, \mathbf{x}, \mathbf{w}) \wedge \text{Verify}(\mathbf{R}, \text{srs}_i, \mathbf{x}, \pi) \end{array} \right] \geq 1 - \nu(\kappa).$$

Definition 30 (Updateable Zero-Knowledge (Upd-ZK)). A NIZK with Updatable SRS meets the *Updateable Zero-Knowledge (Upd-ZK)* for \mathcal{R} , if for all $\mathbf{R} \in \mathcal{R}(1^\kappa)$, any subvector Sub , and computationally unbounded \mathcal{A} , we have $|\varepsilon_0 - \varepsilon_1| \leq \nu(\kappa)$, where

$$\varepsilon_b := \Pr \left[\begin{array}{l} ((\text{srs}_0, \Pi_{\text{srs}_0}) \parallel \vec{\text{ts}}_0' := \vec{\text{ts}}_0') \leftarrow \text{Gen}_{\text{srs}}(\mathbf{R}), r_s \leftarrow \$ \text{RND}(\text{Sub}), \\ ((\{\text{srs}_j, \Pi_{\text{srs}_j}\}_{j=1}^i, \xi_{\text{Sub}}) \parallel \{\vec{\text{ts}}_j'\}_{j=1}^i) \leftarrow (\text{Sub} \parallel \text{Ext}_{\text{Sub}})(\text{srs}_0, \Pi_{\text{srs}_0}, r_s) : \\ \{\text{SV}(\text{srs}_j, \Pi_{\text{srs}_j}) = 1\}_{j=0}^i \wedge \mathcal{A}^{\mathcal{O}_{\text{srs}_i, \vec{\text{ts}}_i, b}(\cdot, \cdot)}(\mathbf{R}, \xi_{\text{Sub}}, \text{srs}_i) = 1 \end{array} \right].$$

The simulation oracle $\mathcal{O}_{\text{srs}_i, \vec{\text{ts}}_i, b}(\cdot, \cdot)$ for $b \in \{0, 1\}$ is defined in Figure 4.1.

We skip recalling the definition of updatable knowledge soundness and proceed directly to discussing the Updateable non-Black-Box Simulation Extractability (Upd-nBB-SE) property which plays a central role in the remaining parts of this chapter.

$\mathcal{O}_{\vec{srs}_i, \vec{ts}_i, b}(x, w) :$
1 : if $(x, w) \notin \mathbf{R}$ return \perp else :
2 : if $b = 0$: return $\pi \leftarrow \text{Prove}(\mathbf{R}, \vec{srs}_i, x, w)$
3 : else return $\text{Sim}(\mathbf{R}, \vec{srs}_i, x, \vec{ts}_i := \{\vec{ts}'_j\}_{j=0}^i)$

Figure 4.1: Simulation Oracle for Updatable Zero-Knowledge property.

Definition 31 (Updateable non-Black-Box Simulation Extractability (Upd-nBB-SE) [ARS20]). A NIZK satisfies *Upd-nBB-SE* for \mathcal{R} , if for every PPT \mathcal{A} and any subverter Sub , there exists an efficient extractor $\text{Ext}_{\mathcal{A}}$, and we have,

$$\Pr \left[\begin{array}{l} (\vec{srs}_0, \Pi_{\vec{srs}_0}) \leftarrow \text{Gen}_{\text{srs}}(\mathbf{R}), r_s \leftarrow \$ \text{RND}(\text{Sub}), \\ (\{\vec{srs}_j, \Pi_{\vec{srs}_j}\}_{j=1}^i, \xi_{\text{Sub}}) \leftarrow \text{Sub}(\vec{srs}_0, \Pi_{\vec{srs}_0}, r_s), \\ \{\text{SV}(\vec{srs}_j, \Pi_{\vec{srs}_j}) = 1\}_{j=0}^i, r_{\mathcal{A}} \leftarrow \$ \text{RND}(\mathcal{A}), \\ ((x, \pi) \parallel w) \leftarrow (\mathcal{A}^{\mathcal{O}_{\vec{srs}_i, \vec{ts}_i}(\cdot)} \parallel \text{Ext}_{\mathcal{A}})(\mathbf{R}, \xi_{\text{Sub}}, \vec{srs}_i; r_{\mathcal{A}}) : \\ (x, \pi) \notin Q_S \wedge (x, w) \notin \mathbf{R} \wedge \text{Verify}(\mathbf{R}, \vec{srs}_i, x, \pi) \end{array} \right] \leq \nu(\kappa),$$

where the simulation oracle $\mathcal{O}_{\vec{srs}_i, \vec{ts}_i}(\cdot)$ is defined in Figure 4.2.

$\mathcal{O}_{\vec{srs}_i, \vec{ts}_i}(x) :$
1 : $\pi \leftarrow \text{Sim}(\mathbf{R}, \vec{srs}_i, x, \vec{ts}_i := \{\vec{ts}'_j\}_{j=0}^i)$
2 : $Q_S \leftarrow Q_S \cup \{(x, \pi)\}$
3 : return π

Figure 4.2: Simulation Oracle for Updatable Simulation Extractability property.

4.2 On the Setup of Updatable zk-SNARKs

Considering the formal definition of standard NIZKs (cf. Section 2.7 on page 26) and NIZKs with universal and updatable SRS (cf. Definition 28 on page 54), it

is easy to observe that the latter schemes include two additional algorithms: SRS update, i.e. SU and SRS verify, i.e. SV. In [BMS23], it was noted that none of the subsequent universal and updatable zk-SNARKs, except for Groth et al.’s initial work [Gro+18], define these algorithms. The authors often seem content with a single sentence from Groth et al. [Gro+18], which suggests that since the SRS consists of monomials, it can be demonstrated to be updatable. However, we customized them and conducted a thorough comparison to assess their efficiency. Next, we briefly summarize the main findings and techniques used.

SRS Update & SRS Verify algorithms. W.l.o.g. the SRS in most of the universal and updatable zk-SNARKs over bilinear groups except the initial work typically consist of monomials of the form $\{[\alpha^k]_\zeta\}_{k \in [0, \ell]}$, where $\alpha \in \mathbb{Z}_p$, $\zeta \in \{1, 2\}$, and $\ell \in \mathbb{N}$. To update and verify the integrity of this set of monomials, one can randomly sample an integer $\tau \leftarrow \mathbb{Z}_p^*$ and compute $\{\tau^k [\alpha^k]_\zeta\}_{k \in [0, \ell]}$. The new (updated) randomness for these monomials can be denoted by $\alpha' := \tau\alpha$, and we have $\{[\alpha'^k]_\zeta\}_{k \in [0, \ell]}$.

However, a malicious updater (i.e., subverter) can assert any random group elements as the set of updated monomials. Given the hardness of the DDH problem in both \mathbb{G}_1 and \mathbb{G}_2 , it is not possible to verify whether this party has executed the update phase correctly. Therefore, we need to require each updater to provide a proof of the form of $\Pi = ([\tau]_1, [\tau]_2)$ to confirm its knowledge of the randomizing factor τ . This is crucial because, in the security definitions of NIZK proofs in the updatable SRS model, the extractor must be capable of extracting the trapdoors. For instance, in Definition 30 on page 55, the extractor Ext_{Sub} should be able to extract $\{\vec{\text{ts}}'_j\}_{j=0}^i$ in order to execute the simulation oracle defined in Figure 4.1.

By following this approach, one can update the random terms while ensuring their secrecy of the final randomness as long as at least one participant behaves, honestly. However, to meet the security requirements discussed for updatable NIZKs (cf. Definition 28 on page 54), the randomness used in the malicious updates must be extractable. For this case, the verifier checks the consistency of updates by running the SRS verify algorithm.

Batched Verification Techniques. Suppose a verifier needs to verify ℓ distinct PPEs of the form $e(A_j, H_2) = e(H_1, B_j)$, where $j \in [0, \ell]$ and $H_1 \in \mathbb{G}_1$, $H_2 \in \mathbb{G}_2$ are fixed source group elements. Using the standard batching techniques from [BGR98], we can batch all these equations to a single pairing equation of

the following form,

$$e \left(\sum_{j \in \ell} [r_j]_{A_j}, H_2 \right) = e \left(H_1, \sum_{j \in \ell} [r_j]_{B_j} \right),$$

where the verifier samples $r_j \leftarrow \mathbb{Z}_p$ for all $j \in [0, \ell]$.

This technique works because of the bilinearity property of pairings, as outlined in Definition 1. According to the DeMillo-Lipton-Schwartz-Zippel lemma, the soundness error of this batching method is $1/p$. In simpler terms, there exists a small possibility that the batched equation holds, while at least one of the original pairing equations is invalid.

Definition 32 (DeMillo-Lipton-Schwartz-Zippel Lemma²). Given a non-zero polynomial $F(x_1, \dots, x_n) \in \mathbb{Z}_p[X_1, \dots, X_n]$ with total degree $d \geq 0$, and a finite and non-empty subset $S \subset \mathbb{Z}_p$ we have,

$$\Pr[\forall r_1, \dots, r_n \leftarrow S : F(r_1, \dots, r_n) = 0] \leq \frac{d}{|S|}.$$

In this scenario, we manage to decrease the number of pairing operations from 2ℓ to just 2. Despite the fact that the verifier still needs to compute ℓ source group exponentiations in \mathbb{G}_1 and \mathbb{G}_2 , the cost of source group exponentiations is much lower compared to pairing operations. Table 4.1 compares the basic operations on the widely-used BN-254 curve using the Charm-Crypto library³ ran on the HP Zbook 15 G6 laptop with 16 GB of RAM, an Intel Core i7-9850H CPU 2.60 GHz. The reported execution times are averaged over 100 runs without preprocessing.

Table 4.1: Cost of Basic Group Operations. \mathbf{M}_i and \mathbf{E}_i denote multiplication and exponentiation costs in source groups \mathbb{G}_i for $i \in \{1, 2, T\}$, and \mathbf{P} denotes pairing cost.

Operation	$\mathbf{M}_1/\mathbf{M}_2/\mathbf{M}_T$	$\mathbf{E}_1/\mathbf{E}_2/\mathbf{E}_T$	\mathbf{P}
Time	3.3/7.1/21.4 (μs)	0.9/1.6/4.8 (ms)	18.5 (ms)

To further enhance the efficiency in [BMS23], we took the approach to sample the randomnesses r_j from smaller subsets such as $\{0, 1\}^{50}$ instead of sampling

²Also known as the Schwartz-Zippel lemma. For a detailed history of this lemma, we refer the readers to <https://rjlipton.wpcomstaging.com/2009/11/30/the-curious-history-of-the-schwartz-zippel-lemma/>.

³<https://github.com/JHUISI/charm>.

from \mathbb{Z}_p . Even with these smaller subsets, the soundness error for this batching technique remains at $1/2^{50}$ (approximately 1 in 1,125,899,906,842,624). This simple batching technique is the central idea for reducing the cost of SRS verification algorithms in the existing universal and updatable zk-SNARKs.

Marlin’s SRS is not extractable! In our research [BMS23], we also identified that the SRS of Marlin [Chi+19], one of the studied constructions, is not extractable. We proposed a solution to address this limitation as an additional contribution. Marlin [Chi+19] is proposed by Chiesa et al. (cf. Section 8.3.2 on page 151), a prominent zk-SNARK in the universal and updatable SRS model. It stands out for its constant-size proofs and advancements in efficiency compared to both earlier and subsequent designs. The authors prove Marlin achieves KS in the Algebraic Group Model (AGM), and to show its Upd-KS, it is necessary to show that the SRS trapdoors are extractable from a subverted or maliciously updated SRS.

Note that this might not be a practical concern since the current deployments for Marlin⁴ are unaffected by this. However, a potential theoretical concern could still remain. In the security model, it is more realistic to let an adversary be able to hash to an elliptic curve point and then generate the SRS components as $([x]_1, [\gamma x]_1, [1]_2, [x]_2)$ without knowing γ . For example, they could sample a group element from \mathbb{G}_1 without knowing its exponent and subsequently utilize a known x to compute $([x]_1, [\gamma x]_1, [1]_2, [x]_2)$ for an undisclosed γ . This attack can be executed by a malicious SRS updater such that the extractor in extracting the trapdoors fails which is against the updatability definition.

Note 13. (Algebraic Group Model (AGM) with hashing) One could argue that Marlin (and some subsequent schemes) are proven within the original AGM framework [FKL18]. As discussed in Chapter 2, in the AGM adversaries are purely algebraic and cannot generate random group elements without knowing their discrete logarithms (i.e. representation vector). While this argument holds true, real-world challenges persist. Such constructions might not inherently achieve Sub-ZK, as adversaries can leverage elliptic curve hashing [Ica09] to produce random group elements without recovering the exponents.

To address these concerns, earlier Sub-ZK zk-SNARKs [Abd+17; Lip22] were designed and validated in more practical models. These include the proofs in the GGM with *hashing* [BFS16; Abd+17] and the AGM with *hashing* [Lip22]. The term of “with hashing” aspect signifies that adversaries are permitted

⁴<https://github.com/arkworks-rs/marlin>

to sample random group elements without the need for exponent knowledge, using methods such as elliptic curve hashing [Ica09]. At TCC’23, Lipmaa et al. [LPS23a] extended this concept to AGM with Oblivious Sampling. This extension allows the adversary to gain access to an oracle in order to sample group elements obliviously from a certain distribution.

4.3 Key-Updatable Public Key Encryption Schemes

In TIRAMISU [BS21], we utilize Public-Key Encryption (PKE) schemes with key updatability features. While similar definitions have been recently proposed for zk-SNARKs [Gro+18] and digital signatures [ARS20], drawing on previous definitions [CHK03; Fau+19], to the best of our knowledge, this type of PKE was introduced for the first time in [BS21].

In contrast to subversion-resilient encryption schemes [ABK18], where the key-generation phase might be subverted, here we consider scenarios where the output of the key-generation phase is updatable, allowing parties to update the keys. Our aim is to meet the standard security requirements of a Public-Key Encryption (PKE) scheme (cf. Section 2.8 on page 31), as long as either the original key generation or at least one of the updates was carried out honestly. The formal definition of this cryptosystem can be found in Definition 59 on page 180.

The primary security requirements for a public-key cryptosystem with updatable keys are *updatable correctness*, *updatable key hiding* and *Updatable Indistinguishability under Chosen-Plaintext Attack (Upd-IND-CPA)*. For more detail we refer the readers to Section 9.3 on page 180.

ElGamal Encryption. The ElGamal cryptosystem [ElG84] as a Public-Key Encryption (PKE) scheme is a common basis for many systems. Given the formal definition of Public-Key Encryption (PKE) schemes (cf. Section 2.8 on page 31), let \mathbb{G} be a finite group of prime order p and let the message space $\mathcal{M} := \mathbb{G}$; then the ElGamal encryption can be defined as follows:

- $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\kappa)$: take security parameter κ as input. Sample $\text{sk} \leftarrow_{\$} \mathbb{Z}_p^*$. Set $\text{pk} := [\text{sk}]$ and return (sk, pk) .
- $\text{ct} \leftarrow \text{Encrypt}(\text{pk}, M; r)$: take pk and message $M \in \mathbb{G}$. If $r = \perp$, sample $r \leftarrow_{\$} \mathbb{Z}_p^*$. Output $\text{ct} := (c_1, c_2) = ([r], M + r[\text{sk}])$.
- $M \leftarrow \text{Decrypt}(\text{sk}, \text{ct})$: parse $(c_1, c_2) \leftarrow \text{ct}$. Output $M \leftarrow c_2 - \text{sk}c_1$.

The Indistinguishability under Chosen-Plaintext Attack (IND-CPA) security of ElGamal encryption is proved under the hardness of Decisional Diffie-Hellman (DDH) assumption (cf. Definition 4 on page 15) [ELG84].

Informal Definition 13. (Lifted ElGamal encryption). In another version of the ElGamal cryptosystem, referred to as the lifted ElGamal^a, the message space is defined over field elements, denoted as $\mathcal{M} := \mathbb{Z}_p$. The encryption process operates the same as above, but first, we must convert the message to its group format $[m]$.

However, there is a constraint on the size of the message m ; the bit-length of the original message must be sufficiently small (usually less than 40 bits) to allow for efficient computation of the discrete logarithm. This limitation primarily arises from the requirement of the decryption algorithm to return the original message m rather than $[m]$.

^aAlso known as exponent ElGamal encryption.

The ElGamal encryption can also be defined over (Type-III) bilinear groups. For a given asymmetric bilinear setting description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{G}_1, \mathbf{G}_2, e) \leftarrow \text{BGgen}(\kappa)$ and message space $\mathcal{M} := \mathbb{G}_T$, the ElGamal encryption is defined as follows:

- $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\kappa)$: Take the security parameter κ in its unary representation, and sample $\text{sk} \leftarrow \mathbb{Z}_p^*$. Set $\text{pk} := ([\text{sk}]_1, [\text{sk}]_2)$ and return (sk, pk) .
- $\text{ct} \leftarrow \text{Encrypt}(\text{pk}, M; r)$: Take pk and message $M \in \mathbb{G}_T$. If $r = \perp$, sample $r \leftarrow \mathbb{Z}_p^*$. Output $\text{ct} := (c_1, c_2) = ([r]_T, Me([\text{sk}]_1, [r]_2))$.
- $M \leftarrow \text{Decrypt}(\text{sk}, \text{ct})$: parse $(c_1, c_2) \leftarrow \text{ct}$. Output $M \leftarrow c_2/c_1^{\text{sk}}$.

We can show that the above encryption scheme is IND-CPA secure (cf. Definition 27 on page 32) under the hardness of Symmetric External Diffie-Hellman (SXDH) assumption (cf. Section 8.2.2 on page 145).

Key updatability of ElGamal encryption. ElGamal secret/public key pairs are updatable to a new pair and one can check the consistency of the update, efficiently. More precisely, given the formal definition of key-updatable PKE (cf. Definition 59 on page 180) we can extend the ElGamal algorithms as follows:

- $(\text{pk}_i, \Pi_{\text{pk}_i}) \leftarrow \text{KU}(\text{pk}_{i-1})$: Given a correctly formed ElGamal's public key $\text{pk}_{i-1} := (\text{pk}_{i-1,1}, \text{pk}_{i-1,2}) := ([\text{sk}_{i-1}]_1, [\text{sk}_{i-1}]_2)$, one can update it to $\text{pk}_i := (\text{pk}_{i,1}, \text{pk}_{i,2}) := (\tau_i[\text{sk}_{i-1}]_1, \tau_i[\text{sk}_{i-1}]_2)$, where $\tau_i \leftarrow \mathbb{Z}_p^*$. It then

outputs the new public key along with the proof $\Pi_{\mathbf{pk}_i} = ([\tau_i]_1, [\tau_i]_2)$. Note that the corresponding secret key \mathbf{sk}_i can be formulated as $\mathbf{sk}_i := \tau_i \mathbf{sk}_{i-1}$ and to derive \mathbf{sk}_i , it is necessary to know both τ_i and \mathbf{sk}_{i-1} .

- $(1, \perp) \leftarrow \text{KV}(\mathbf{pk}_i, \Pi_{\mathbf{pk}_i})$: To check the consistency of the updates, one can verify if the following PPE holds.

$$e(\mathbf{pk}_{i-1,1}, [\tau_i]_2) = e(\mathbf{G}_1, \mathbf{pk}_{i,2}) = e([\tau_i]_1, \mathbf{pk}_{i-1,2}) = e(\mathbf{pk}_{i,1}, \mathbf{G}_2) \cdot$$

As discussed in Theorem 8 on page 182, the ElGamal encryption with key updatability meets all the security properties of perfect updatable completeness, updatable key hiding and updatable IND-CPA (cf. Section 9.3 on page 180).

We use this PKE with key updatability as a main tool to our generic framework, called TIRAMISU to lift NIZKs in the updatable SRS model to stronger notion of Updateable Black-Box Simulation Extractability (Upd-BB-SE).

4.4 A General Framework for Lifting to Upd-BB-SE

TIRAMISU is a framework designed to lift the Knowledge Sound NIZK in the updatable SRS model to a stronger notion of security called Updateable Black-Box Simulation Extractability (Upd-BB-SE). This security feature extends the Updateable non-Black-Box Simulation Extractability (Upd-nBB-SE) feature defined in [ARS20] (cf. Definition 31 on page 56). Similarly, we assume i is the index of the final update.

Note 14. (Non-Black-Box to Black-Box SE lifting Compilers) Zk-SNARKs in the SRS model cannot naturally achieve Black-Box SE.^a However, certain compilers, such as [Kos+15; Bag19b; AB19], are designed to lift these schemes to a stronger notion of Black-Box SE. Broadly speaking, these methods use to the traditional approach of encrypting the witness with the extractor’s public key. Consequently, the extractor $\text{Ext}(\cdot)$ only needs access to the corresponding secret key to decrypt (extract) the witness. Looking ahead, we aim to extend this technique to achieve Updateable Black-Box Simulation Extractability (Upd-BB-SE) in the next sections.

^aA recent work [CF24] has shown some widely used zk-SNARKs in the ROM can achieve UC-security unconditionally without modification.

Definition 33. (Updateable Black-Box Simulation Extractability (Upd-BB-SE) [BS21, Definition 6]) A NIZK satisfies Upd-BB-SE for \mathcal{R} if there is an

extractor $\text{Ext}(\cdot)$ that for any PPT adversary \mathcal{A} and subverter Sub , another extractor $\text{Ext}_{\text{Sub}}(\cdot)$ exists, ensuring that the following probability is $\leq \nu(\kappa)$:

$$\Pr \left[\begin{array}{l} ((\vec{s}\vec{r}\vec{s}_0, \Pi_{\vec{s}\vec{r}\vec{s}_0}), \vec{t}\vec{s}_0 := \vec{t}\vec{s}'_0, \vec{t}\vec{e}_0 := \vec{t}\vec{e}'_0) \leftarrow \text{Gen}_{\text{srs}}(\mathbf{R}), r_s \leftarrow \text{\$ RND}(\text{Sub}), \\ ((\{\vec{s}\vec{r}\vec{s}_j, \Pi_{\vec{s}\vec{r}\vec{s}_j}\}_{j=1}^i, \xi_{\text{Sub}}) \parallel \{\vec{t}\vec{s}'_j, \vec{t}\vec{e}'_j\}_{j=1}^i) \leftarrow (\text{Sub} \parallel \text{Ext}_{\text{Sub}})(\vec{s}\vec{r}\vec{s}_0, \Pi_{\vec{s}\vec{r}\vec{s}_0}, r_s), \\ \{\text{SV}(\vec{s}\vec{r}\vec{s}_j, \Pi_{\vec{s}\vec{r}\vec{s}_j}) = 1\}_{j=0}^i, r_{\mathcal{A}} \leftarrow \text{\$ RND}(\mathcal{A}), \\ (x, \pi) \leftarrow \mathcal{O}_{\vec{s}\vec{r}\vec{s}_i, \vec{t}\vec{s}_i}(\cdot)(\mathbf{R}, \vec{s}\vec{r}\vec{s}_i, \xi_{\text{Sub}}; r_{\mathcal{A}}), w \leftarrow \text{Ext}(\mathbf{R}, \vec{s}\vec{r}\vec{s}_i; \vec{t}\vec{e}_i) : \\ (x, \pi) \notin Q_S \wedge (x, w) \notin \mathbf{R} \wedge \text{Verify}(\mathbf{R}, \vec{s}\vec{r}\vec{s}_i, x, \pi) \end{array} \right],$$

where the simulation oracle $\mathcal{O}_{\vec{s}\vec{r}\vec{s}_i, \vec{t}\vec{s}_i}(\cdot)$ is defined in Figure 4.2.

This definition draws inspiration from the standard SE definition (cf. Definition 24), as defined by Groth [Gro06], which involves two extractors, one for the setup phase and the other for the subsequent arguments. However, our definition necessitates a non-black-box extractor in the setup phase, which appears to be an unavoidable requirement for constructing Upd-BB-SE NIZKs without relying on a trusted third party [BFS16]. Using arguments or assumptions with non-black-box extraction techniques, such as rewinding [Dam+12] or knowledge assumptions [BFS16; Abd+17; Gro+18], is a common approach to mitigate or eliminate the need to trust the parameters of various cryptographic protocols.

Note 15. Note that in Definition 33, the extractor $\text{Ext}_{\text{Sub}}(\cdot)$ is a non-Black-Box extractor, based on rewinding or Knowledge of Exponent Assumption (KEA), meanwhile $\text{Ext}(\cdot)$ is a black-box extractor, e.g., using a PKE with key updatability (cf. Section 4.3).

Figure 9.1 on page 178 depicts how TIRAMISU compares with other similar frameworks, namely the C0C0 framework [Kos+15] and the Lamassu framework [ARS20].

In summary, the Black-Box extractor, $\text{Ext}(\cdot)$, can retrieve the witness by decrypting the ciphertext without relying on the adversary's source code or rewinding techniques. However, in order to extend this technique to NIZKs in the updatable SRS model, we had to introduce a new primitive to enable the PKE key pairs' updatability. Additionally, we needed to ensure that these schemes satisfy the security properties, particularly the Upd-BB-SE, which also requires secret key extraction for PKE trapdoors.

4.5 Application to Privacy-Preserving Smart Contracts

Blockchain as the main technology behind cryptocurrencies is a decentralized ledger which operates on a distributed network of nodes. Blockchain is a publicly-visible and append-only log protected by a consensus protocol, such as Proof-of-Work (PoW) and Proof-of-Stake (PoS). Instead of centralized entities, a group of validators, known as miners, collaboratively validate and record the transactions in a tamper-resistant manner.

Note 16. (Programmable Blockchains and Smart Contracts)

While cryptocurrencies are the most popular application of blockchain, they can enable functionalities beyond keeping track of funds in a ledger. Programmable blockchains offer broader capabilities, particularly when transactions must be initiated based on predefined events such as “Bob should send x USD to Alice if Cond. A happens!”. Smart contracts enable such functionalities within a blockchain environment.

In 2006, Groth [Gro06] has shown that a NIZK argument capable of achieving Black-Box Simulation Extractability (BB-SE) can realize the ideal NIZK-functionality $\mathcal{F}_{\text{NIZK}}$ [GOS06]. Unfortunately, the default security of zk-SNARKs is insufficient for direct deployment in UC protocols. This is because a zk-SNARK achieves nBB extraction, and the extractor $\text{Ext}_{\mathcal{A}}(\cdot)$ requires access to the source code and random coins of \mathcal{A} . However, in UC-secure NIZKs, the simulator of the *ideal-world* should be able to simulate the corrupted parties, which necessitates the ability to extract witnesses without access to the source code of the environment’s algorithm.

Following this, Kosba et al. [Kos+15] introduced the $\mathcal{C}\emptyset\mathcal{C}\emptyset$ framework, along with various constructions, enabling the transformation of a sound NIZK argument into a Black-Box Simulation Extractability (BB-SE) NIZK argument. In short, given a sound NIZK argument for language $\mathbf{L}_{\mathbf{R}}$, the $\mathcal{C}\emptyset\mathcal{C}\emptyset$ framework defines a new extended language $\tilde{\mathbf{L}}_{\mathbf{R}}$ appended with some primitives and provides a NIZK argument capable of achieving BB-SE. This lifted version can then be used in UC protocols.

Moving forward, in applications aiming for Universally Composable (UC) security, such as Hawk [Kos+16], Gyges [JKS16], and Ouroboros Cryptsinous [Ker+19], the underlying zk-SNARK is lifted by the $\mathcal{C}\emptyset\mathcal{C}\emptyset$ framework [Kos+15] to achieve BB-SE. Furthermore, KACHINA [KKK21] provides a comprehensive UC model for privacy-preserving smart contracts. It aims to serve as a framework capturing ZEXE [Bow+20], Hawk [Kos+16],

Zether [Bün+20], and others, without compromising their privacy guarantees. While the C0C0 and KACHINA frameworks operate within the standard SRS model, i.e. achieving BB-SE necessitate a trusted setup phase. However, prior to TIRAMISU [BS21] it was not formalized how to achieve this notion of security for the zk-SNARKs in the universal and updatable SRS model.

In summary, TIRAMISU [BS21] made one step towards achieving Upd-BB-SE which serves as a basis for UC-security in the practical applications that employ updatable zk-SNARKs. For example, VERIZEXE [Xio+23] employs Plonk [GWC19] as a well-known zk-SNARK in the universal and updatable SRS model, to develop an efficient Decentralized Private Computation (DPC) system that can support a diverse range of applications. VERIZEXE improves ZEXE [Bow+20], which is limited to supporting only specific applications. While the authors emphasized Plonk’s universality, VERIZEXE can be set up in the updatable SRS model too which is a more realistic trust assumption, and then one can prove its security in the UC model using TIRAMISU [BS21] along with some recent works such as [Abd+23] to avoid non-black-box Ext_{Sub} using the Fischlin transform.

Conclusion and Open Problems

5.1 Summary of Research

In this thesis, we discussed two fundamental cryptographic building blocks with a direct impact on enhancing the privacy and reducing the trust in distributed systems. The main results of this thesis are the invention of a new cryptographic primitive, called Threshold Structure-Preserving Signatures (TSPS) which is compatible with the efficient Groth-Sahai (GS) NIZKs which forms a basis for the design of distributed and complex privacy-preserving systems. Furthermore, we delved into a crucial category of NIZKs termed universal and updatable NIZKs. We emphasized how these can be used in scenarios where there is no trusted entity available to generate trustworthy SRS.

In Chapter 3, we discussed two TSPS schemes. The first with the shortest signature size and the most efficient verification, although supporting only a restricted class of messages (iDH message space). This scheme is proven secure under a weaker unforgeability notion called TS-UF-0. Moreover, we explored the main techniques used in the subsequent TSPS scheme, designed in the standard model. This latter scheme supports messages of arbitrary group element vectors and offers enhanced unforgeability against adaptive adversaries under the stronger notion of TS-UF-1. Additionally, we touched upon how these schemes could be applied in certain privacy-preserving systems, such as Threshold-Issuance Anonymous Credential (TIAC).

In Chapter 4, we emphasized the importance of trust mitigation techniques in pre-processing NIZKs. We provided an overview of stronger notions of knowledge soundness, focusing particularly on updatable NIZKs, where adversaries can also join during the update process. Notably, except for the initial work of Groth et al. [Gro+18], subsequent zk-SNARKs in the universal and updatable

SRS model often neglect discussions on how their construction could be set up and updated. Consequently, we summarized some techniques used for further optimization such as smaller randomness distributions and batched verification technique. Furthermore, we discussed TIRAMISU as a general framework to lift non-black-box updatable KS and Upd-nBB-SE to a stronger notion of Upd-BB-SE. Additionally, we discussed on how this security property is important in proving the security of some complex systems such as privacy-preserving smart contracts in the Universally Composable (UC) framework.

5.2 Future Work

TSPS. The TSPS is the first multi-party structure-preserving primitive. Considering that these primitives are not exclusively limited to Digital Signatures (DS), a natural question arises: how can we expand this concept to other cryptographic primitives?

Moreover, a TSPS with the same security advantages as [Mit+24] while shorter signatures and fewer PPEs in the verification phase stands as another potential future work. This may be feasible since we relied on matrix assumptions that usually come with higher costs. In some parallel research e.g., [Bac+24; DR23], the authors have obtained threshold signatures based on the standard DDH assumption. One potential research on combining these two directions might be how we can use their techniques to propose more efficient TSPS schemes in the standard model.

Fuchsbauer et al. (AC'14, JoC'19) [HS14; FHS19] introduced SPS-EQ, which enable signature adaptivity—a controlled form of malleability over both message and signatures. This allows anyone to publicly transform any signature of this type into a uniformly random signature on the equivalence class of the initial message, thus avoiding complex zero-knowledge proofs. Considering that the main applications for SPS-EQ involve a central signer, which introduces vulnerabilities in the form of single points of failure, an interesting open problem is to define and design threshold SPS-EQ. This could potentially enhance the efficiency of TSPS in specific applications by eliminating the need for complex NIZK proofs. Very recently, Nanri et al. [Nan+24] proposed an interactive Mercurial signature. Their approach begins with the initial SPS-EQ, leading to multiple rounds of communication to generate a partial signature. Meanwhile, a non-interactive threshold structure-preserving signatures on equivalence-classes remains as an interesting future work.

Standard Threshold Signatures (TS) lack accountability. Specifically, an aggregated signature passes the verification phase if a sufficient number of

signers agree by partially signing the message. With an aggregated signature, it is impossible to determine which group of signers participated in the partial signing phase. Accountable threshold signatures [BK22; BPR22] offer signer traceability in two potential scenarios: private accountability, which allows specific entities to identify the set of signers, and public accountability, which provides public traceability. Thus we leave accountable TSPS schemes as an interesting open problem.

As mentioned earlier, there are several works, such as [Gay+18; Abe+19], focused on improving the tightness of security for SPS. An interesting direction for future work is exploring how these techniques can be applied to develop a tightly-secure TSPS.

Updatable NIZKs. Given that the results in Chapter 4 have been obtained three years ago, we first summarize the findings of several subsequent works that attempt to address the open problems in this field of study.

As we discussed in Chapter 4, in [BMS23] we found that Marlin’s SRS is not extractable, however its knowledge soundness is formally proved in the Algebraic Group Model (AGM). We then briefly discussed why the AGM for this family of zk-SNARKs is not realistic and referred to a more realistic notion of security, called AGM with hashing [Lip22]. Upon taking this observation and highlighting the inadequacy of the AGM for such constructions in [BMS23], Lipmaa et al. [LPS23b] (TCC’23) formally introduced this concept.

Additionally, in [BMS23], we provided a detailed benchmarking of the setup phase for some popular universal and updatable zk-SNARKs. Our findings indicated that the setup of Basilisk [RZ21] (CRYPTO’21) outperforms other constructions. However, an interesting extension would be to evaluate the efficiency of these schemes based on other factors such as efficiency of proof generation and verification. Recently, Ernstberger et al. in [Ern+23] introduced zkBench, the first benchmarking framework and estimator tool for public key cryptography. They provided a comprehensive comparison of universal and updatable zk-SNARKs using this tool.

Moreover, as we noted in Definition 33, the extractor $\text{Ext}_{\text{Sub}}(\cdot)$ is still a *non-black-box* extractor, and we need to either use rewinding or Knowledge of Exponent Assumption (KEA) which contradicts universality, and we left this as an open problem in [BS21]. To be more precise, this is a limitation on achieving both universality and updatability for the NIZKs in the updatable SRS model. Some subsequent works e.g. [Gan+22; Cam+23; Abd+23] address this problem using techniques that we briefly discuss below.

Ganesh et al. [Gan+22] (SCN’22) show the simulation extractability of universal

zk-SNARKs such as Plonk, Sonic, and Marlin in the Random Oracle Model (ROM) without additional overhead. The authors study the Fiat-Shamir (FS) transformation which plays a central role in making these schemes non-interactive. They extend the FS transform to multi-round settings and updatable SRS setting, yet do not ensure Updateable Black-Box Simulation Extractability (Upd-BB-SE). The Fiat-Shamir transformation still depends on rewinding techniques, contradicting the universality of these systems. However, Campanelli et al. [Cam+23] (Africacrypt’23) formally proved that non-adaptive black-box extractable zk-SNARKs for NP relations are impossible.

Recently, Abdolmaleki et al. [Abd+23] (IEEE CSF’24) show that composability and updatability can be achieved in the updateable SRS model by relaxing the succinctness property of the zk-SNARKs. The authors introduced a framework that ensures the resulting construction is circuit-succinct, although it might not necessarily yield a witness-succinct outcome.

However, the design of an updatable NIZK with full extractability with succinct proofs remains as an interesting open problem.

Open-Sourced Implementations

All the implementations are available on my GitHub account under the username <https://github.com/Mahdi171>.

- **Unlinkable Policy-Compliant Signatures**
Prototyping several ul-PCS schemes for [BSW24].
 - **Groth-Sahai Proofs**
An efficient implementation for the Groth-Sahai proof system [GS08].
 - **Nirvana Payment**
An anonymous and reusable payment guarantee system [Mad+23].
 - **Attribute-Based Access Control Encryption**
Proof of concept for [SP21].
 - **Ciphertext-Policy Attribute-Based Encryption**
A constant size CP-ABE proposed in [SP21, Section 5].
 - **Multi-Level Secure Attribute-Based Access Control**
A proof of concept for [Agh+22].

Python, Docker







Python

Python

Python

Python

Python



Part II

Publications

Threshold Structure-Preserving Signatures

Source. E. Crites, M. Kohlweiss, B. Preneel, M. Sedaghat, and D. Slamanig. “Threshold Structure-Preserving Signatures”. In: *Advances in Cryptology – ASIACRYPT 2023*. Ed. by J. Guo and R. Steinfeld. Singapore: Springer Nature Singapore, 2023, pp. 348–382. ISBN: 978-981-99-8724-5. DOI: [10.1007/978-981-99-8724-5_11](https://doi.org/10.1007/978-981-99-8724-5_11)

Abstract. Structure-preserving signatures (SPS) are an important building block for privacy-preserving cryptographic primitives, such as electronic cash, anonymous credentials, and delegatable anonymous credentials. In this work, we introduce the first *threshold* structure-preserving signature scheme (TSPS). This enables multiple parties to jointly sign a message, resulting in a standard, single-party SPS signature, and can thus be used as a replacement for applications based on SPS.

We begin by defining and constructing SPS for indexed messages, which are messages defined relative to a unique index. We prove its security in the random oracle model under a variant of the generalized Pointcheval-Sanders assumption (PS). Moreover, we generalize this scheme to an indexed multi-message SPS for signing vectors of indexed messages, which we prove secure under the same assumption. We then formally define the notion of a TSPS and propose a construction based on our indexed multi-message SPS. Our TSPS construction is *fully non-interactive*, meaning that signers simply output partial signatures without communicating with the other signers. Additionally, signatures are short: they consist of 2 group elements and require 2 pairing product equations to verify. We prove the security of our TSPS under the security of our indexed multi-message SPS scheme. Finally, we show that our TSPS may be used as a drop-in replacement for UC-secure Threshold-Issuance Anonymous Credential (TIAC) schemes, such as Coconut, without the overhead of the Fischlin transform.

6.1 Introduction

Threshold cryptography [Des90; DF90; De +94] was designed to reduce the trust in single entities and improve the availability of keying material. It allows a secret key to be shared among a set of parties [Sha79; Bla79] such that the task involving the key can only be performed if some threshold of them collaborates. Threshold signatures [Sho00; DK01], threshold encryption [SG98; Can+99], and threshold verifiable unpredictable functions [Gur+21] enable distributed protocols, such as e-voting systems [Cra+96; CGS97] and multi-party computation [CDN01; DN03].

Threshold signatures in particular have attracted significant interest recently, in part because of advances in distributed ledger technologies, cryptocurrencies, and decentralized identity management [Doe+19; KG20; Can+20; Kon+21; CKM23]. They are also the subject of current standardization efforts by NIST [BDV+20; BP23]. Signatures used by certification authorities to issue credentials or to secure digital wallets make attractive targets for misuse or forgery. To mitigate these risks, an (n, t) -threshold signature scheme distributes the signing key among n parties such that any quorum of at least t signers can jointly generate a signature, but the scheme remains secure as long as fewer than t key shares are known to the adversary.

A threshold signature that is *fully non-interactive* consists of a single round of communication. On input the message, each signer computes its partial signature independently of other signers, and aggregation of at least t partial signatures results in a single signature representing the group. Interactive signing protocols involving two or more rounds add complexity and are error prone [TS21; Dri+19]. Thus, fully non-interactive schemes are preferable, the canonical example being threshold BLS [BLS04; Bol03].

Structure-preserving signatures. Structure-preserving signatures (SPS) [Abe+10] are pairing-based signatures where the message, signature, and verification key consist of source group elements only (in one or both source groups), and signature verification consists of group membership checks and pairing product equations only. SPS have been studied extensively, with a main focus on short signatures [Abe+11b; Abe+14; Gha16; Gha17b], lower bounds [Abe+11b; AGO11; Abe+18a], and (tight) security under well-known assumptions [Abe+12; HJ12; KPW15; LPY15; JR17; Gay+18; Abe+19].

SPS are compatible with Groth-Sahai non-interactive zero-knowledge proofs (NIZKs) [GS08] and, more generally, help to avoid the expensive extraction of exponents in security proofs. This makes them attractive for the modular

design of protocols relying on signatures and NIZKs. Indeed, SPS have seen widespread adoption in privacy-preserving applications, such as group signatures [Abe+10; LPY15], traceable signatures [Abe+11a], blind signatures [Abe+10; FHS15], attribute-based signatures [EGK14], malleable signatures [ALP12], anonymous credentials [Fuc11; Cam+15; FHS19], delegatable anonymous credentials [Bel+09; CL19], and anonymous e-cash [Bla+11].

For such signature-based applications, compromise of the signing key represents a single point of attack and failure. Replacing the use of SPS with TSPS together with distributed key generation (DKG) would help to reduce the trust in a single authority and increase the availability of the respective signing service. While many of the aforementioned applications of SPS would benefit from thresholdization, until now there was no known threshold construction of SPS that could serve as their basis. We provide the first candidate TSPS scheme as the main contribution of this work.

Towards constructing a threshold SPS. Our goal is to construct threshold SPS that are fully non-interactive, i.e., there is no coordination among signers. This puts some requirements on the used SPS and in particular prevents the use of nonlinear operations of the signing randomness and secret keys (cf. Section 6.2), which existing SPS fail to satisfy. Thus, as a starting point for our TSPS, we consider the pairing-based Pointcheval-Sanders signature scheme (PS) [PS16] (cf. Section 6.3.1), as its randomness is simply a random base group element and it avoids hashing during verification. We recall that the PS scheme is defined over an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g})$ with signing key $\text{sk} = (x, y) \in (\mathbb{Z}_p^*)^2$ and corresponding verification key $\text{vk} = (\hat{g}^x, \hat{g}^y) \in \mathbb{G}_2^2$. The signing algorithm takes as input a scalar message $m \in \mathbb{Z}_p$ and outputs a signature

$$\sigma = (h, s) = (g^r, h^{x+my}) \in \mathbb{G}_1^2.$$

Importantly, the nonce r (or equivalently the base h) is sampled fresh for each signature. This scheme fails to be an SPS because the message is not a group element (or elements). Ghadafi [Gha16] made the observation that a PS-like SPS scheme can be constructed for a group element message $(M_1, M_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ for which there exists a scalar message $m \in \mathbb{Z}_p$ such that $M_1 = g^m$ and $M_2 = \hat{g}^m$. This is referred to as a Diffie-Hellman (DH) message. (cf. Section 6.1 for more on this message space.) A Ghadafi SPS signature (cf. Section 6.3.1) has the form:

$$\sigma = (h, s, t) = (g^r, M_1^r, h^x s^y) \in \mathbb{G}_1^3.$$

Let us see how one might construct a threshold version of this scheme. Suppose each signer possesses a share $\text{sk}_i = (x_i, y_i)$ of the secret key $\text{sk} = (x, y)$. A first

(non-interactive) attempt might have each signer output a partial signature of the form:

$$\sigma_i = (h_i, s_i, t_i) = (g^{r_i}, M_1^{r_i}, h_i^{x_i} s_i^{y_i}) ,$$

with aggregation of the third term having the form:

$$t = \prod_{i \in \mathcal{T}} t_i^{\lambda_i} = \prod_{i \in \mathcal{T}} g^{r_i x_i \lambda_i} M_1^{r_i y_i \lambda_i} ,$$

where λ_i is the Lagrange coefficient for party i in the signing set \mathcal{T} of size at least t (the threshold). As with other existing SPS, this however does not allow reconstruction via Lagrange interpolation because each term in the exponent is multiplied by a distinct random integer r_i . To overcome this, due to the specific form of the signatures, the signers would have to agree on a common random element $h = g^r$. Indeed, this will be our approach to solve this issue.

A second (interactive) attempt might have each signer output randomness shares $h_i = g^{r_i}$ and corresponding $s_i = M_1^{r_i}$ in a first round of signing, followed by a second round in which each signer computes aggregate values $h = g^r = \prod_{i \in \mathcal{T}} h_i = g^{\sum_{i \in \mathcal{T}} r_i}$ and $s = \prod_{i \in \mathcal{T}} s_i$ and outputs a partial signature of the form:

$$\sigma_i = (h, s, t_i = h^{x_i} s^{y_i}) , \tag{6.1}$$

with aggregation of the third term having the form:

$$t = \prod_{i \in \mathcal{T}} t_i^{\lambda_i} = \prod_{i \in \mathcal{T}} g^{r_i x_i \lambda_i} M_1^{r_i y_i \lambda_i} .$$

This allows reconstruction via Lagrange interpolation. In terms of security, the unforgeability of this threshold scheme may be reduced to the unforgeability of single-party Ghadafi SPS signatures. However, the reduction needs to obtain the corrupt h_j, s_j values before revealing honest values h_i, s_i . The addition of a third signing round could achieve this, whereby all values h_i, s_i are committed to in the first round as $H(h_i), H'(s_i)$, for H and H' modeled as random oracles, and then revealed in the second round. However, the reduction needs to obtain the *nonces* r_j of the corrupt parties, which may be extracted from zero-knowledge proofs appending the outputs h_i, s_i in round two. These additional rounds and zero-knowledge proofs add significant overhead.

Our approach is clean and straightforward: we instead have signers obtain shared randomness $h = g^r$ via a random oracle, yielding a fully non-interactive scheme. But observe that if partial signatures have the form of Equation (6.1),

then $s = M_1^r$ cannot be computed without knowledge of the discrete logarithm $\text{dlog}_h(M_1)$. Thus, we borrow techniques from Sonnino et al. [Son+19] and Camenisch et al. [Cam+20], which implicitly sign *indexed* Diffie-Hellman messages (id, M_1, M_2) , a concept we define and formalize rigorously in this work. Indexing can be understood as requiring the existence of an injective function f that maps each scalar message $m \in \mathbb{Z}_p$ to an index $id = f(m)$. We then have $h = H(id)$, where H is modeled as a random oracle, and $M_1 = H(id)^m$. Then each partial signature has the form:

$$\sigma_i = (h, s_i) = (H(id), h^{x_i} M_1^{y_i}) ,$$

and the aggregated signature has the form:

$$\sigma = (h, s) = (H(id), h^x M_1^y) . \quad (6.2)$$

This is exactly our TSPS construction, with the underlying SPS signature defined by Equation (6.2). We extend these techniques to vectors of indexed Diffie-Hellman messages $(id, \vec{M}_1, \vec{M}_2)$, which allows additional elements to be signed, e.g., attributes when used within anonymous credential systems [PS16; Son+19]. It is important to note that the index is not needed for verification (and therefore $H(id)$ is not computed), so our schemes are indeed structure preserving.

We define an appropriate notion of unforgeability for indexed messages: existential unforgeability under chosen indexed message attack (EUF-CiMA) and prove the security of our constructions under this notion. We discuss various ways of defining the index function, depending on the application. For example, if privacy is not required and the message and public key are known, the index function may simply be the identity function: $id = f(m) = m$, capturing the intuitive notion that each nonce r is associated with a single scalar message m .

Why Diffie-Hellman messages?

Diffie-Hellman messages can be traced back to the introduction of automorphic signatures [Fuc09] and SPS [Abe+10], and have since appeared in various other SPS constructions [Gha16; Gha17b; Gha17a; Gha19]. Their use is largely motivated by an impossibility result by Abe et al. [Abe+11b], which proves that any SPS in the Type-III setting must have at least 3 group elements and 2 pairing product equations in the verification. Furthermore, the result rules out unilateral signatures (those containing elements from only one source group) meeting this lower bound. However, if messages are in *both* source groups, it is possible to construct a unilateral SPS meeting this lower bound. This is what

Diffie-Hellman messages and the Ghadafi construction [Gha16] achieve. We follow the same approach to construct efficient TSPS.

Constructing a TSPS over standard, unilateral message spaces remains an interesting open problem. However, such a scheme would necessarily contain more group elements in the signature and more pairing product equations to verify, due to this impossibility result. This is an important consideration when combining with Groth-Sahai NIZK proofs in applications, as the number of pairings required for verification scales linearly with the number of source pairings.

6.1.1 Our Contributions

Our contributions can be summarized as follows:

- We formalize the concept of indexed message spaces and formally define the notion of structure-preserving signatures (SPS) over indexed message spaces and corresponding notion of security: existential unforgeability under chosen indexed message attack (EUF-CiMA).
- We propose a concrete SPS construction over indexed Diffie-Hellman messages, called IM-SPS, and prove its EUF-CiMA security under a new variant of the generalized Pointcheval-Sanders assumption. We reduce this assumption to the hardness of the $(2, 1)$ -discrete logarithm problem in the algebraic group model (AGM).
- We provide an indexed multi-message SPS construction, called IMM-SPS, which allows vectors of indexed Diffie-Hellman messages to be signed, and prove its EUF-CiMA security under the same assumption.
- We introduce the notion of a threshold structure-preserving signature (TSPS) scheme and propose a fully non-interactive TSPS based on our EUF-CiMA secure SPS scheme. Signatures contain only 2 group elements and verification consists of 2 pairing product equations. We prove the security of our TSPS under the EUF-CiMA security of IMM-SPS.
- We discuss applications of our TSPS construction and, in particular, blind signing of messages. This represents a core functionality in Threshold-Issuance Anonymous Credential (TIAC) systems. We outline how our TSPS can be used in TIAC systems as a drop-in replacement that avoids rewinding extractors for the required non-interactive zero-knowledge (NIZK) proofs.

6.2 Related Work

We provide an overview of pairing-based non-interactive threshold signature schemes in Table 6.1 and structure-preserving signature schemes (SPS) in Table 6.2 and discuss how these schemes fail to meet our requirements.

Table 6.1: Table of pairing-based non-interactive threshold signature schemes. iDH refers to indexed Diffie-Hellman messages (Definition 40). \checkmark : Satisfied. \times : Not satisfied.

Scheme	Message Space	Signature Size	Structure-Preserving
BLS [Bol03; BL22]	$\{0, 1\}^*$	$1\mathbb{G}_1$	\times
LJY $\S 1$ [LJY16]	$\{0, 1\}^*$	$2\mathbb{G}_1$	\times
LJY $\S 2$ [LJY16]	$\{0, 1\}^*$	$4\mathbb{G}_1 + 2\mathbb{G}_2$	\times
GJMMST [Gur+21]	$\{0, 1\}^*$	$4\mathbb{G}_1 + 2\mathbb{G}_2$	\times
PS [Son+19; Tom+22]	\mathbb{Z}_p	$2\mathbb{G}_1$	\times
Our TSPS	iDH	$2\mathbb{G}_1$	\checkmark

Threshold Signatures. BLS [BLS04] and its threshold version [Bol03; BL22] are not structure preserving, as they map bitstring messages $\{0, 1\}^*$ to the group using a random oracle. Libert et al. [LJY16] propose a secure non-interactive threshold signature scheme based on linearly-homomorphic SPS (LHSPS) [Lib+13]. While this construction meets many of our requirements, the resulting threshold signature is not structure preserving. It either relies on random oracles to hash bitstring messages to group elements ($\S 1$ [LJY16]) or, when avoiding random oracles, a bit-wise encoding of the message is required ($\S 2$ [LJY16]). Gurkan et al. [Gur+21] propose a pairing-based threshold Verifiable Unpredictable Function (VUF), which is essentially a unique threshold signature [MRV99]. However, their construction is not structure preserving: it hashes bitstring messages to the group using a random oracle. Sonnino et al. [Son+19] and Tomescu et al. [Tom+22] present non-interactive threshold versions of Pointcheval-Sanders (PS) signatures; however, verification takes place over scalar vectors, and is thus not structure preserving. We note that signatures for scalar vectors are intuitively closer to SPS than ones for bitstring messages, as evidenced, for example, by Ghadafi’s scheme [Gha16]. We do not know of a general conversion technique, however.

Structure-Preserving Signatures. Most structure-preserving signatures in the literature fail to be good candidates for thresholdization due to nonlinear operations of signer-specific randomness and secret key elements, which are not amenable to Lagrange interpolation (e.g., [Abe+10; Abe+11b; Abe+14; Bar+15; Gha17b; Gro15]). However, there are two promising approaches: linearly-homomorphic SPS (LHSPS) [Lib+13] and the SPS by Ghadafi [Gha16]. The former is a one-time signature, meaning that a key pair can only sign a single message¹. The SPS by Ghadafi [Gha16] lends itself to thresholdization, but it requires multiple communication rounds and incurs significant overhead.

Table 6.2: Table of structure-preserving signature schemes (SPS). DH refers to Diffie-Hellman messages (Definition 35), and iDH refers to indexed Diffie-Hellman messages (Definition 40). Avoids Nonlinearity refers to operations of the signing randomness and secret keys. ✓: Satisfied. ✗: Not satisfied.

Scheme	Message Space	Signature Size	Avoids Nonlinearity
AFGHO[Abe+10]	\mathbb{G}_1	$5\mathbb{G}_1 + 2\mathbb{G}_2$	✗
AGHO [Abe+11b]	$\mathbb{G}_1 \times \mathbb{G}_2 / \mathbb{G}_2$	$2\mathbb{G}_1 + 1\mathbb{G}_2$	✗
AGOT [Abe+14]	\mathbb{G}_1	$2\mathbb{G}_1 + 1\mathbb{G}_2$	✗
BFFSST [Bar+15]	\mathbb{G}_2	$1\mathbb{G}_1 + 2\mathbb{G}_2$	✗
Ghadafi [Gha17b]	DH	$2\mathbb{G}_1$	✗
Ghadafi [Gha16]	DH	$3\mathbb{G}_1$	✗
Groth [Gro15]	\mathbb{G}_2	$1\mathbb{G}_1 + 2\mathbb{G}_2$	✗
LPJY [Lib+13]*	\mathbb{G}_1	$2\mathbb{G}_1$	✓
Our SPS	iDH	$2\mathbb{G}_1$	✓

*One-time: a key pair can only sign a single message.

6.3 Preliminaries and Definitions

General. Let $\kappa \in \mathbb{N}$ denote the security parameter and κ its unary representation. Let p be a κ -bit prime. For all positive polynomials $f(\kappa)$, a function $\nu : \mathbb{N} \rightarrow \mathbb{R}^+$ is called *negligible* if $\exists \kappa_0 \in \mathbb{N}$ such that $\forall \kappa > \kappa_0$ it holds that $\nu(\kappa) < 1/f(\kappa)$. We denote by \mathbb{G}^* the set $\mathbb{G} \setminus 1_{\mathbb{G}}$, where $1_{\mathbb{G}}$ is the

¹Note that the LHSPS in [Lib+13] is designed over symmetric bilinear groups with signatures consisting of 3 group elements. The authors in [LJY16] extend this LHSPS over asymmetric bilinear groups with signatures of size 2.

identity element of the group \mathbb{G} . We denote the group of integers mod p by $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$, its multiplicative group of units by \mathbb{Z}_p^* , and the polynomial ring over \mathbb{Z}_p by $\mathbb{Z}_p[X]$. For a group \mathbb{G} of order p with generator g , we denote the discrete logarithm $m \in \mathbb{Z}_p$ of $M \in \mathbb{G}$ base g by $\text{dlog}_g(M)$ (i.e., $M = g^m$). We denote the set of integers $\{1, \dots, n\}$ by $[1, n]$ and the vector A by \vec{A} . Let $Y \leftarrow \$ F(X)$ denote running probabilistic algorithm F on input X and assigning its output to Y . Let $x \leftarrow \$ \mathbb{Z}_p$ denote sampling an element of \mathbb{Z}_p uniformly at random. All algorithms are randomized unless expressly stated otherwise. PPT refers to probabilistic polynomial time. We denote the output of a security game \mathbf{G}^{GAME} between a challenger and a PPT adversary \mathcal{A} by $\mathbf{G}_{\mathcal{A}}^{\text{GAME}}$, where \mathcal{A} wins the game if $\mathbf{G}_{\mathcal{A}}^{\text{GAME}} = 1$.

Definition 34 (Bilinear Group). A bilinear group generator $\mathcal{BG}(\kappa)$ returns a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g})$ such that \mathbb{G}_1 , \mathbb{G}_2 and \mathbb{G}_T are finite groups of the same prime order p , $g \in \mathbb{G}_1$ and $\hat{g} \in \mathbb{G}_2$ are generators, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficiently computable bilinear pairing, which satisfies the following properties:

1. $e(g, \hat{g}) \neq 1_{\mathbb{G}_T}$ (non-degeneracy).
2. $\forall a, b \in \mathbb{Z}_p, e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab} = e(g^b, \hat{g}^a)$ (bilinearity).

We rely on bilinear groups \mathbb{G}_1 and \mathbb{G}_2 with no efficiently computable isomorphism between them [GPS08], also called Type-III or asymmetric bilinear groups. To date, they are the most efficient choice for relevant security levels.

Definition 35 (Diffie-Hellman Message Space [Fuc09; Abe+10]). Over an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g})$, a pair $(M_1, M_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ belongs to the Diffie-Hellman (DH) message space \mathcal{M}_{DH} if there exists $m \in \mathbb{Z}_p$ such that $M_1 = g^m$ and $M_2 = \hat{g}^m$.

One can efficiently verify whether $(M_1, M_2) \in \mathcal{M}_{\text{DH}}$ by checking $e(M_1, \hat{g}) = e(g, M_2)$.

Definition 36 (Algebraic Group Model [FKL18]). An adversary is *algebraic* if for every group element $h \in \mathbb{G} = \langle g \rangle$ that it outputs, it is required to output a representation $\vec{h} = (\eta_0, \eta_1, \eta_2, \dots)$ such that $h = g^{\eta_0} \prod g_i^{\eta_i}$, where $g, g_1, g_2, \dots \in \mathbb{G}$ are group elements that the adversary has seen thus far.

The original definition of the algebraic group model (AGM) [FKL18] only captured regular cyclic groups $\mathbb{G} = \langle g \rangle$. Mizuide et al. [MTT19] extended this definition to include symmetric pairing groups $(\mathbb{G}_1 = \mathbb{G}_2)$, such that the adversary is also allowed to output target group elements (in \mathbb{G}_T) and their representations. Recently, Couteau and Hartmann [CH20] defined the Algebraic

Asymmetric Bilinear Group Model, which extends the AGM definition for asymmetric pairings by allowing the adversary to output multiple elements from all three groups. The definition can be found in the full version [Cri+22].

6.3.1 Schemes

Pointcheval-Sanders Signatures [PS16]. The PS signature scheme is defined over the message space \mathcal{M} of scalar messages $m \in \mathbb{Z}_p$ and consists of the following PPT algorithms:

- $\text{pp} \leftarrow \text{Setup}(\kappa)$: Output $\text{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}) \leftarrow \mathcal{BG}(\kappa)$.
- $(\text{sk}, \text{vk}) \leftarrow \text{KGen}(\text{pp})$: Sample $x, y \leftarrow \mathbb{Z}_p^*$ and set $\text{sk} = (\text{sk}_1, \text{sk}_2) = (x, y)$ and $\text{vk} = (\text{vk}_1, \text{vk}_2) = (\hat{g}^x, \hat{g}^y)$. Output (sk, vk) .
- $\sigma \leftarrow \sigma(\text{pp}, \text{sk}, m)$: Sample $r \leftarrow \mathbb{Z}_p^*$ and compute $\sigma = (h, s) = (g^r, h^{x+my})$. Output σ .
- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, m, \sigma)$: If $h \in \mathbb{G}_1, h \neq 1_{\mathbb{G}_1}$, and the pairing product equation $e(h, \text{vk}_1 \text{vk}_2^m) = e(s, \hat{g})$ holds, output 1 (accept); else, output 0 (reject).

Pointcheval-Sanders signatures are EUF-CMA secure under the PS assumption (Definition 38) [PS16].

Ghadafi SPS [Gha16]. The Ghadafi structure-preserving signature scheme is defined over the message space \mathcal{M}_{DH} of Diffie-Hellman pairs $(M_1, M_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ such that $e(M_1, \hat{g}) = e(g, M_2)$ and consists of the following PPT algorithms:

- $\text{pp} \leftarrow \text{Setup}(\kappa)$: Output $\text{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}) \leftarrow \mathcal{BG}(\kappa)$.
- $(\text{sk}, \text{vk}) \leftarrow \text{KGen}(\text{pp})$: Sample $x, y \leftarrow \mathbb{Z}_p^*$ and set $\text{sk} = (\text{sk}_1, \text{sk}_2) = (x, y)$ and $\text{vk} = (\text{vk}_1, \text{vk}_2) = (\hat{g}^x, \hat{g}^y)$. Output (sk, vk) .
- $\sigma \leftarrow \sigma(\text{pp}, \text{sk}, M_1, M_2)$: Sample $r \leftarrow \mathbb{Z}_p^*$ and compute $\sigma = (h, s, t) = (g^r, M_1^r, h^x s^y)$. Output σ .
- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, \sigma, M_1, M_2)$: If $h, s, t \in \mathbb{G}_1, h \neq 1_{\mathbb{G}_1}$, and both pairing product equations $e(h, M_2) = e(s, \hat{g})$ and $e(t, \hat{g}) = e(h, \text{vk}_1)e(s, \text{vk}_2)$ hold, output 1 (accept); else, output 0 (reject).

The Ghadafi SPS is weakly EUF-CMA secure in the generic group model (GGM) [Gha16].

Shamir Secret Sharing [Sha79]. An (n, t) -Shamir secret sharing divides a secret s among n shareholders such that each subset of at least t shareholders can reconstruct s , but fewer than t cannot (and s remains information-theoretically hidden). A dealer who knows the secret s forms a polynomial $f(x)$ of degree t with randomly chosen coefficients from \mathbb{Z}_p such that $f(0) = s$. The dealer then securely provides each shareholder with $s_i = f(i), i \in [1, n]$. Let $\vec{s} \leftarrow \text{Share}(s, p, n, t)$ denote the process of computing shares $\vec{s} = (s_1, \dots, s_n)$ of a secret s . Each subset $\mathcal{T} \subset [1, n]$ of size at least t can pool their shares to reconstruct the secret s using Lagrange interpolation, as $s = f(0) = \sum_{i \in \mathcal{T}} s_i \lambda_i$, where $\lambda_i = \prod_{j \in \mathcal{T}, j \neq i} \frac{j}{j-i}$.

6.3.2 Assumptions

Definition 37 ((2,1)-Discrete Logarithm Assumption [BFL20]). Let $\text{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}) \leftarrow \mathcal{BG}(\kappa)$ be an asymmetric bilinear group. The (2,1)-discrete logarithm assumption holds with respect to \mathcal{BG} if for all PPT adversaries \mathcal{A} , there exists a negligible function ν such that

$$\Pr [z \leftarrow \mathbb{Z}_p^*; (Z, Z', \hat{Z}) \leftarrow (g^z, g^{z^2}, \hat{g}^z); z' \leftarrow \mathcal{A}(\text{pp}, Z, Z', \hat{Z}) : z' = z] < \nu(\kappa) .$$

Definition 38 (PS Assumption [PS16]). Let the advantage of an adversary \mathcal{A} against the PS game \mathbf{G}^{PS} , as defined in Figure 6.1, be as follows:

$$\text{Adv}_{\mathcal{A}}^{\text{PS}}(\kappa) = \Pr [\mathbf{G}_{\mathcal{A}}^{\text{PS}} = 1] .$$

The PS assumption holds if for all PPT adversaries \mathcal{A} , there exists a negligible function ν such that $\text{Adv}_{\mathcal{A}}^{\text{PS}}(\kappa) < \nu(\kappa)$.

The validity of the tuple (m^*, h^*, s^*) is decidable by checking $e(s^*, \hat{g}) = e(h^*, \hat{g}^x (\hat{g}^y)^{m^*})$. The PS assumption is an interactive assumption defined by Pointcheval and Sanders [PS16] to construct an efficient randomizable signature and has been shown to hold in the GGM.

Kim et al. [Kim+20] introduced a generalized version of the PS assumption (GPS) that splits the PS oracle $\mathcal{O}^{\text{PS}}(\cdot)$ into two oracles $\mathcal{O}_0^{\text{GPS}}(), \mathcal{O}_1^{\text{GPS}}(\cdot)$: the first samples $h \leftarrow \mathbb{G}_1$, and the second takes h and m as input and generates the PS value h^{x+my} . Recently, Kim et al. [Kim+22] extended the GPS assumption (GPS₂), replacing field element inputs, such as m , with group element inputs. The GPS₂ assumption holds under the (2,1)-DL assumption (Definition 37) in the AGM. Both the GPS and GPS₂ assumptions can be found in the full version [Cri+22].

$\mathbf{G}^{\text{PS}}(\kappa)$	$\mathcal{O}^{\text{PS}}(m) \text{ // } m \in \mathbb{Z}_p$
1 : $\text{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}) \leftarrow \mathcal{BG}(\kappa)$	1 : $h \leftarrow \$_{\mathbb{G}_1}$
2 : $x, y \leftarrow \$_{\mathbb{Z}_p}$	2 : $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
3 : $(m^*, h^*, s^*) \leftarrow \mathcal{A}^{\text{O}^{\text{PS}}}(\text{pp}, \hat{g}^x, \hat{g}^y)$	3 : return (h, h^{x+my})
4 : return $((1) h^* \neq 1_{\mathbb{G}_1} \wedge m^* \neq 0 \wedge$	
5 : $(2) s^* = h^{*x+m^*y} \wedge$	
6 : $(3) m^* \notin \mathcal{Q})$	

Figure 6.1: Game Defining the PS Assumption.

Owing to the fact that our SPS and TSPS constructions rely on a different message space, we introduce an analogous generalized PS assumption (GPS_3), defined as follows.

Definition 39 (GPS_3 Assumption). Let the advantage of an adversary \mathcal{A} against the GPS_3 game $\mathbf{G}^{\text{GPS}_3}$, as defined in Figure 6.2, be as follows:

$$\text{Adv}_{\mathcal{A}}^{\text{GPS}_3}(\kappa) = \Pr \left[\mathbf{G}_{\mathcal{A}}^{\text{GPS}_3} = 1 \right] .$$

The GPS_3 assumption holds if for all PPT adversaries \mathcal{A} , there exists a negligible function ν such that $\text{Adv}_{\mathcal{A}}^{\text{GPS}_3}(\kappa) < \nu(\kappa)$.

We prove that this assumption holds in the AGM if the $(2, 1)$ -DL problem is hard (Theorem 1).

6.4 Indexed Message Structure-Preserving Signatures

In this section, we introduce the notion of structure-preserving signatures (SPS) on indexed messages as well as a corresponding notion of security: unforgeability against chosen indexed message attack (EUF-CiMA). We provide an indexed message SPS construction, called IM-SPS, and prove its EUF-CiMA security under the GPS_3 assumption (Definition 39) in the random oracle model (ROM) (Theorem 2). We also propose an indexed *multi*-message SPS construction,

$\mathbf{G}^{\text{GPS}_3}(\kappa)$	
1 : $\mathbf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}) \leftarrow \mathcal{BG}(\kappa)$ 2 : $x, y \leftarrow \mathbb{Z}_p^*$ 3 : $(M_1^*, M_2^*, h^*, s^*) \leftarrow \mathcal{A}^{\mathcal{O}_0^{\text{GPS}_3}, \mathcal{O}_1^{\text{GPS}_3}}(\mathbf{pp}, \hat{g}^x, \hat{g}^y, \boxed{g^y})$ 4 : return ((1) $M_1^* \neq 1_{\mathbb{G}_1} \wedge h^* \neq 1_{\mathbb{G}_1} \wedge$ 5 : (2) $s^* = h^{*x} M_1^{*y} \wedge$ 6 : (3) $\text{dlog}_{h^*}(M_1^*) = \text{dlog}_{\hat{g}}(M_2^*) \wedge$ 7 : (4) $(\star, M_2^*) \notin \mathcal{Q}_1$)	
$\mathcal{O}_0^{\text{GPS}_3}()$	$\mathcal{O}_1^{\text{GPS}_3}(h, M_1, M_2)$
1 : $h \leftarrow \mathbb{G}_1$	1 : if $(h \notin \mathcal{Q}_0 \vee \text{dlog}_h(M_1) \neq \text{dlog}_{\hat{g}}(M_2)) :$
2 : $\mathcal{Q}_0 \leftarrow \mathcal{Q}_0 \cup \{h\}$	2 : return \perp
3 : return h	3 : if $(h, \star) \in \mathcal{Q}_1 :$
	4 : return \perp
	5 : $\mathcal{Q}_1 \leftarrow \mathcal{Q}_1 \cup \{(h, M_2)\}$
	6 : return $h^x M_1^y$

Figure 6.2: Game defining our GPS_3 assumption. The additional element in the solid box is required for blind signing only (cf. Section 6.6.1).

called IMM-SPS, which allows vectors of indexed messages to be signed, and prove its EUF-CiMA security under the same assumptions (Theorem 3). IMM-SPS are useful for applications where additional elements, such as attributes, are signed.

Indexing can be understood as requiring the existence of an injective function f that maps each message to an index. We model this by requiring that for all index/message pairs in an indexed message space \mathcal{M} , the following uniqueness property holds: $(id, \tilde{M}) \in \mathcal{M}, (id', \tilde{M}') \in \mathcal{M}, id = id' \Rightarrow \tilde{M} = \tilde{M}'$. That is, no two messages use the same index. We refer to index/message pairs as

$$M = (id, \tilde{M}).$$

Indexing is useful, as signatures can depend on the index; for example, in our schemes, signing involves evaluating a hash-to-curve function H on the index to obtain a base element $h \leftarrow H(id)$. Verifying a message/signature pair does not require availability of the index, making it structure preserving. Consequently, the verification message space $\tilde{\mathcal{M}}$ is obtained from \mathcal{M} by omitting the index.

For our schemes, we need to consider that in verification one can provide a base element h^r obtained by randomizing the original base element h . This is due to the partial randomizability of the signatures. Thus, different messages \tilde{M}, \tilde{M}' may be valid representations for the same scalar message m . Consequently, similar to SPS on equivalence classes (SPS-EQ) [FHS19], the verification message space $\tilde{\mathcal{M}}$ is expanded to consider equivalent (randomized) messages: $\tilde{\mathcal{M}} = \{\tilde{M} \mid \exists (\cdot, \tilde{M}') \in \mathcal{M}, \tilde{M} \in \text{EQ}(\tilde{M}')\}$. The function EQ depends on the concrete message space and determines the respective set of equivalent messages.

Next, we define the indexed Diffie-Hellman message space used by our IM-SPS scheme (cf. Figure 6.3 for its encoding function).

Definition 40 (Indexed Diffie-Hellman Message Space). Given an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}) \leftarrow \mathcal{BG}(\kappa)$, an index set \mathcal{I} , and a random oracle $H : \mathcal{I} \rightarrow \mathbb{G}_1$, $\mathcal{M}_{\text{iDH}}^H$ is an indexed Diffie-Hellman (DH) message space if $\mathcal{M}_{\text{iDH}}^H \subset \{(id, \tilde{M}) \mid id \in \mathcal{I}, m \in \mathbb{Z}_p, \tilde{M} = (H(id)^m, \hat{g}^m) \in \mathbb{G}_1 \times \mathbb{G}_2\}$ and the following index uniqueness property holds: for all $(id, \tilde{M}) \in \mathcal{M}_{\text{iDH}}^H$, $(id', \tilde{M}') \in \mathcal{M}_{\text{iDH}}^H$, $id = id' \Rightarrow \tilde{M} = \tilde{M}'$.

We define the equivalence class for each message $\tilde{M} = (M_1, M_2) \in \tilde{\mathcal{M}}_{\text{iDH}}^H$ as $\text{EQ}_{\text{iDH}}(M_1, M_2) = \{(M_1^r, M_2) \mid \exists r \in \mathbb{Z}_p\}$.

$\text{iDH}^H(id, m)$	$H(id)$
1 : $h \leftarrow H(id)$	1 : if $\mathcal{Q}_H[id] = \perp$:
2 : $\tilde{M} \leftarrow (h^m, \hat{g}^m)$	2 : $\mathcal{Q}_H[id] \leftarrow \$ \mathbb{G}_1$
3 : return (id, \tilde{M})	3 : return $\mathcal{Q}_H[id]$

Figure 6.3: Encoding Function of Indexed Diffie-Hellman Message Space in the ROM.

The subset membership is efficiently decidable by checking $e(M_1, \hat{g}) = e(h, M_2)$ for $h \leftarrow H(id)$. Note that, in addition, one needs to guarantee that no two

messages use the same index. This is the responsibility of the signer.² As mentioned above, messages \tilde{M} lie in a different verification message space $\tilde{\mathcal{M}}_{\text{iDH}}^H$ that is uniquely determined by $\mathcal{M}_{\text{iDH}}^H$ and EQ_{iDH} . Note that most $\tilde{M} \in \tilde{\mathcal{M}}_{\text{iDH}}^H$ are not indexed Diffie-Hellman messages. In particular, when expanding the definition of EQ_{iDH} , the verification message space is $\tilde{\mathcal{M}}_{\text{iDH}}^H = \{(M_1^r, M_2) \mid \exists r \in \mathbb{Z}_p, \exists (\cdot, M_1, M_2) \in \mathcal{M}_{\text{iDH}}^H\}$.

$$\underbrace{m \in \mathbb{Z}_p \xrightarrow{f} id}_{\text{Message Indexing}} \xrightarrow[\text{Indexed DH Message Space in ROM}]{\text{iDH}^H(id, m)} (id, M_1, M_2) \in \mathcal{M}_{\text{iDH}}^H \xrightarrow[\text{Randomization of } M_1]{} (\hat{M}_1, M_2)$$

Figure 6.4: From \tilde{M} to M and back again: The first message component is randomizable; the second fixes the index.

Does \tilde{M} depend on id or does id depend on \tilde{M} ?

One might observe the above apparent circularity with respect to the indexing technique. On the one hand, we require existence of an injective function f that maps (M_1, M_2) to id . On the other hand, M_1 is computed as $M_1 = H(id)^{\text{dlog}_g(M_2)}$. This circularity is avoided by computing id from the partial message M_2 , or more commonly its discrete logarithm m .

As illustrated in Figure 6.4, the indexing function f assigns an index id to each scalar message $m \in \mathbb{Z}_p$. Then, a hash-to-curve function $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$ (modeled as a random oracle) is used to generate a unique base element h . A source group message (M_1, M_2) can then be obtained using h . In an indexed message SPS, the signing algorithm takes as input the source group message together with an index and generates the underlying signature with access to H . Note that the index does not destroy the structure since the verifier does not need to know id to verify a signature on message $\tilde{M} = (M_1, M_2)$.

Indexing function instantiations. Depending on the application, the indexing function f can be instantiated in different ways. For example, if messages and signatures are allowed to be public, the indexing function can be instantiated by using the scalar message m itself as the index: $f(m) \mapsto m = id$.

²To highlight this responsibility, we enforce uniqueness both in the message space and later on in Line 1 of the $\mathcal{O}_{\text{Sign}}(\cdot)$ oracle of Figure 6.5.

If message and signatures must be hidden, as in the case of applications to anonymous credentials, one can take the approach of committing to the scalar message and providing a proof of well-formedness of the commitment, as done by Sonnino et al. [Son+19]. As it is infeasible to open a well-formed commitment to two different messages, this guarantees uniqueness of the index. Camenisch et al. [Cam+20] take yet another approach for indexing messages: they assume the existence of a pre-defined and publicly available indexing function. That is, there is a unique index value for each message that is known to all signers. The corresponding base element can be obtained by evaluating the hash-to-curve function at the given index. As the authors note, if the size of the message space is polynomial and known in advance, then this approach is secure, since it is equivalent to including the base element in the public parameters. However, this is impractical for large message spaces.

6.4.1 Definition of Unforgeability for Indexed Message SPS

We adapt the notion of EUF-CMA security for digital signatures (cf. Definition 13) to existential unforgeability against chosen *indexed* message attack (EUF-CiMA). There are two adjustments: (1) the adversary makes queries to the signing oracle by providing *index*/message pairs, and (2) we expand the set of signed messages $\mathcal{Q}_S = \{(id_i, \bar{M}_i)\}_i$ to the set of trivially forgeable messages $\mathcal{Q}_{EQ} = \{\text{EQ}(\bar{M}_i)\}_i$, i.e., all equivalent messages in the verification message space, and use it in the winning condition of the adversary.

Definition 41 (Existential Unforgeability under Chosen Indexed Message Attack (EUF-CiMA)). A digital signature scheme over indexed message space \mathcal{M} is EUF-CiMA secure if for all PPT adversaries \mathcal{A} playing game $\mathbf{G}_{\mathcal{A}}^{\text{EUF-CiMA}}$ (Figure 6.5), there exists a negligible function ν such that

$$\text{Adv}_{\mathcal{A}}^{\text{EUF-CiMA}}(\kappa) = \Pr [\mathbf{G}_{\mathcal{A}}^{\text{EUF-CiMA}}(\kappa) = 1] \leq \nu(\kappa) .$$

6.4.2 Our Indexed Message SPS

In Figure 6.6, we present our indexed message SPS construction IM-SPS over the indexed Diffie-Hellman message space $\mathcal{M}_{\text{IDH}}^{\text{H}}$.

$\mathbf{G}_{\mathcal{A}}^{\text{EUF-CiMA}}(\kappa)$	$\mathcal{O}_{\text{Sign}}(id, \tilde{M})$
1 : $\text{pp} \leftarrow \text{Setup}(\kappa)$	1 : if $(id, \star) \in \mathcal{Q}_S$:
2 : $(\text{sk}, \text{vk}) \leftarrow \mathcal{K}\text{Gen}(\text{pp})$	2 : return \perp
3 : $(\tilde{M}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sign}}}(\text{pp}, \text{vk})$	3 : else : $\sigma \leftarrow \text{Sign}(\text{pp}, \text{sk}, (id, \tilde{M}))$
4 : return $(\tilde{M}^* \notin \mathcal{Q}_{\text{EQ}} \wedge$	4 : $\mathcal{Q}_S \leftarrow \mathcal{Q}_S \cup \{(id, \tilde{M})\}$
5 : $\text{Verify}(\text{pp}, \text{vk}, \tilde{M}^*, \sigma^*))$	5 : $\mathcal{Q}_{\text{EQ}} \leftarrow \mathcal{Q}_{\text{EQ}} \cup \{\text{EQ}(\tilde{M})\}$
	6 : return σ

Figure 6.5: Game Defining $\mathbf{G}_{\mathcal{A}}^{\text{EUF-CiMA}}(\kappa)$.

$\text{Setup}(\kappa)$	$\text{KGen}(\text{pp})$
1 : $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}) \leftarrow \mathcal{BG}(\kappa)$	1 : $x, y \leftarrow \mathbb{Z}_p^*$
2 : $\text{H} : \{0, 1\}^* \rightarrow \mathbb{G}_1$	2 : $\text{sk} \leftarrow (\text{sk}_1, \text{sk}_2) = (x, y)$
3 : $\quad \quad \quad \text{// select hash function}$	3 : $\text{vk} \leftarrow (\text{vk}_1, \text{vk}_2, \boxed{\text{vk}_2^*})$
4 : $\text{pp} \leftarrow ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}), \text{H})$	4 : $\quad \quad \quad = (\hat{g}^x, \hat{g}^y, \boxed{g^y})$
5 : return pp	5 : return (sk, vk)
$\text{Sign}(\text{pp}, \text{sk}, (id, M_1, M_2))$	$\text{Verify}(\text{pp}, \text{vk}, (M_1, M_2), \sigma)$
1 : $h \leftarrow \text{H}(id)$	1 : $\quad \quad \quad \text{// does not invoke H}$
2 : if $e(h, M_2) = e(M_1, \hat{g})$:	2 : parse $\sigma = (h, s)$
3 : $\quad \quad \quad (h, s) \leftarrow (h, h^{\text{sk}_1} M_1^{\text{sk}_2})$	3 : return $(h \neq 1_{\mathbb{G}_1} \wedge M_1 \neq 1_{\mathbb{G}_1} \wedge$
4 : return $\sigma \leftarrow (h, s)$	4 : $e(h, M_2) = e(M_1, \hat{g}) \wedge$
5 : else : return \perp	5 : $e(h, \text{vk}_1) e(M_1, \text{vk}_2) = e(s, \hat{g}))$

Figure 6.6: Our Indexed Message SPS Construction IM-SPS. The additional elements in $\boxed{\text{solid boxes}}$ are required for blind signing only (cf. Section 6.6.1).

6.4.3 Security of IM-SPS

We prove that our proposed IM-SPS construction (Figure 6.6) is EUF-CiMA secure under the GPS_3 assumption (Definition 39) in the random oracle model.

The GPS_3 assumption underpins both the security of IM-SPS as well as our indexed *multi*-message SPS construction IMM-SPS (Section 6.4.4). Our security reductions from IM-SPS and IMM-SPS to GPS_3 are tight. Furthermore, we show the tight security of our TSPS (Section 6.5) under the security of IMM-SPS. Figure 6.7 defines a roadmap for our IM-SPS, IMM-SPS, and TSPS constructions and their underlying assumptions. Thus, as a starting point, we reduce the GPS_3 assumption to the hardness of the $(2, 1)$ -DL problem (Definition 37) in the algebraic group model.

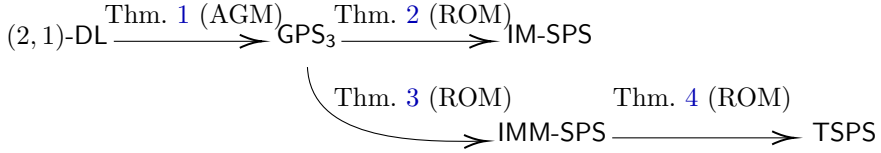


Figure 6.7: The proposed constructions and underlying assumptions.

Theorem 1. The GPS_3 assumption (Definition 39) holds in the asymmetric algebraic bilinear group model under the hardness of the $(2, 1)$ -DL problem (Definition 37).

The proof is provided in the full version [Cri+22].

Theorem 2. The indexed message SPS scheme IM-SPS (Figure 6.6) is correct and EUF-CiMA secure (Definition 41) under the GPS_3 assumption (Definition 39) in the random oracle model.

We first present an attack to motivate the need for uniqueness in the indexed message space. Assume there were no uniqueness requirement, and suppose the redundant check in Line 1 of the $\mathcal{O}_{\text{Sign}}(\cdot)$ oracle of Figure 6.5 were not present. Then, a forger could obtain two signatures $s = h^x M_1^y$, $s' = h^x M_1'^y$ and compute a forgery $s^* = s^2/s' = h^x (M_1^2/M_1')^y$.

Proof Outline. Let \mathcal{A} be a PPT adversary against the EUF-CiMA security of IM-SPS. We construct a PPT reduction \mathcal{B} against the GPS_3 assumption as follows. When \mathcal{A} queries the random oracle H on a fresh id , \mathcal{B} queries its oracle $\mathcal{O}_0^{\text{GPS}_3}()$ to obtain a random base element h , which it stores and returns to \mathcal{A} . When \mathcal{A} queries its signing oracle $\mathcal{O}_\sigma(\cdot)$ on (id, M_1, M_2) , \mathcal{B} looks up $h = H(id)$

and queries its oracle $\mathcal{O}_1^{\text{GPS}_3}(\cdot)$ on (h, M_1, M_2) to receive $h^x M_1^y$. Finally, \mathcal{B} returns the signature $\sigma = (h, h^x M_1^y)$ to \mathcal{A} . \mathcal{B} correctly simulates the EUF-CiMA game, and the success probability of \mathcal{A} and \mathcal{B} is the same.

The attack above would violate the condition $(h, \star) \notin \mathcal{Q}_1$ in Line 3 of the $\mathcal{O}_1^{\text{GPS}_3}(\cdot)$ oracle in Figure 6.2. The full proof is provided in the full version [Cri+22].

6.4.4 Our Indexed Multi-Message SPS

We extend our IM-SPS construction to an indexed *multi*-message SPS construction IMM-SPS, which allows vectors of indexed messages to be signed, and prove its EUF-CiMA security. Extending the message space to allow vectors of any length is desirable for applications in which several attributes may be signed. The number of pairings required for verification scales linearly with the length of the message vectors, but signatures remain constant sized (2 group elements).

$\text{MiDH}^H(id, \vec{m})$	$H(id)$
1 : $h \leftarrow H(id)$	1 : if $\mathcal{Q}_H[id] = \perp$:
2 : for $j \in [1, \ell]$:	2 : $\mathcal{Q}_H[id] \leftarrow \mathbb{G}_1$
3 : $M_{1j} \leftarrow h^{m_j}; M_{2j} \leftarrow \hat{g}^{m_j}$	3 : return $\mathcal{Q}_H[id]$
4 : return $(id, (\vec{M}_1, \vec{M}_2))$	

Figure 6.8: Encoding function of iDH multi-message space in the ROM.

We first generalize the notion of an indexed message space to the multi-message setting. In Figure 6.8, we present the encoding function $\text{MiDH}^H(id, \vec{m})$ of a multi-message variant of the indexed Diffie-Hellman message space that maps, for any $\ell > 1$, ℓ -scalar message vectors $\vec{m} = (m_1, \dots, m_\ell) \in \mathbb{Z}_p^\ell$ to 2ℓ -source group message vectors $(\vec{M}_1, \vec{M}_2) = ((M_{11}, \dots, M_{1\ell}), (M_{21}, \dots, M_{2\ell})) \in \mathbb{G}_1^\ell \times \mathbb{G}_2^\ell$ based on a given index id .

Definition 42 (Indexed Diffie-Hellman Multi-Message Space). Given an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}) \leftarrow \mathcal{BG}(\kappa)$, an index set \mathcal{I} , and a random oracle $H : \mathcal{I} \rightarrow \mathbb{G}_1$, $\mathcal{M}_{\text{MiDH}}^H$ is an indexed Diffie-Hellman (DH) message space if $\mathcal{M}_{\text{MiDH}}^H \subset \{(id, \vec{M}) \mid id \in \mathcal{I}, \vec{m} \in \mathbb{Z}_p^\ell, \vec{M} = \text{MiDH}^H(id, \vec{m})\}$

Setup(κ)	KGen(pp, ℓ)
1 : $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}) \leftarrow \mathcal{BG}(\kappa)$	1 : $x, y_1, \dots, y_\ell \leftarrow \$_{\mathbb{Z}_p^*}$
2 : $\mathbf{H} : \{0, 1\}^* \rightarrow \mathbb{G}_1$	2 : $\vec{\mathbf{sk}} \leftarrow (\mathbf{sk}_0, \dots, \mathbf{sk}_\ell) = (x, y_1, \dots, y_\ell)$
3 : $\quad \quad \quad // \text{ select hash function}$	3 : $\vec{\mathbf{vk}} \leftarrow (\mathbf{vk}_0, \mathbf{vk}_1, \boxed{\mathbf{vk}_1^*}, \dots, \mathbf{vk}_\ell, \boxed{\mathbf{vk}_\ell^*})$
4 : $\mathbf{pp} \leftarrow ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}), \mathbf{H})$	4 : $\quad \quad \quad = (\hat{g}^x, \hat{g}^{y_1}, \boxed{g^{y_1}}, \dots, \hat{g}^{y_\ell}, \boxed{g^{y_\ell}})$
5 : return pp	5 : return $(\vec{\mathbf{sk}}, \vec{\mathbf{vk}})$
Sign(pp, $\vec{\mathbf{sk}}, (id, \vec{M}_1, \vec{M}_2)$)	
1 : $h \leftarrow \mathbf{H}(id)$	
2 : if $\exists j \in [1, \ell] \mid e(h, M_{2j}) \neq e(M_{1j}, \hat{g})$:	
3 : return \perp	
4 : else : return $\sigma \leftarrow (h, s) = (h, h^{\mathbf{sk}_0} \prod_{j=1}^{\ell} M_{1j}^{\mathbf{sk}_j})$	
Verify(pp, $\vec{\mathbf{vk}}, (\vec{M}_1, \vec{M}_2), \sigma)$	
1 : $\quad \quad \quad // \text{ does not invoke } \mathbf{H}$	
2 : parse $\sigma = (h, s)$	
3 : return $(h \neq 1_{\mathbb{G}_1} \wedge \{M_{1j}\}_{j \in [1, \ell]} \neq 1_{\mathbb{G}_1} \wedge \{e(h, M_{2j}) = e(M_{1j}, \hat{g})\}_{j \in [1, \ell]} \wedge$	
4 : $\quad \quad \quad e(h, \mathbf{vk}_0) \prod_{j=1}^{\ell} e(M_{1j}, \mathbf{vk}_j) = e(s, \hat{g}))$	

Figure 6.9: Our Indexed Multi-Message SPS Construction IMM-SPS. The additional elements in solid boxes are required for blind signing only (cf. Section 6.6.1).

and the following index uniqueness property holds: for all $(id, \tilde{M}) \in \mathcal{M}_{\text{MiDH}}^{\mathbf{H}}$, $(id', \tilde{M}') \in \mathcal{M}_{\text{MiDH}}^{\mathbf{H}}$, $id = id' \Rightarrow \tilde{M} = \tilde{M}'$.

We define the equivalence class for each multi-message $\tilde{M} = (\vec{M}_1, \vec{M}_2) \in \mathcal{M}_{\text{MiDH}}^{\mathbf{H}}$ as $\text{EQ}_{\text{MiDH}}(\vec{M}_1, \vec{M}_2) = \{(\vec{M}_1^r, \vec{M}_2^r) \mid \exists r \in \mathbb{Z}_p\}$.

This generalization of the indexed Diffie-Hellman message space leads us to an indexed multi-message SPS, described in Figure 6.9.

Theorem 3. The indexed multi-message SPS scheme IMM-SPS (Figure 6.9) is correct and EUF-CiMA secure (Definition 41) under the GPS_3 assumption (Definition 39) in the random oracle model.

The proof is provided in the full version [Cri+22].

6.5 Threshold Structure-Preserving Signatures

We now define the syntax and security notions for non-interactive (n, t) -Threshold Structure-Preserving Signatures (TSPS) for indexed message spaces. We then propose an efficient instantiation for an indexed Diffie-Hellman multi-message space. In an (n, t) -TSPS, the signing key is distributed among n parties, and the generation of any signature requires the cooperation of a subset of at least t parties. We assume a centralized key generation algorithm for distributing the signing key, but a decentralized key generation protocol (DKG), such as Pedersen's DKG [Ped92], may be used instead.

Definition 43 (Threshold Structure-Preserving Signature). For a given security parameter κ and bilinear group \mathcal{BG} , an (n, t) -TSPS over indexed message space \mathcal{M} consists of a tuple ($\text{Setup}, \text{KGen}, \text{ParSign}, \text{ParVerify}, \text{Reconst}, \text{Verify}$) of PPT algorithms defined as follows:

- $\text{pp} \leftarrow \text{Setup}(\kappa)$: The setup algorithm takes the security parameter κ as input and returns the public parameters pp .
- $(\vec{\text{sk}}, \vec{\text{vk}}, \text{vk}) \leftarrow \text{KGen}(\text{pp}, \ell, n, t)$: The key generation algorithm takes the public parameters pp and length ℓ along with two integers $t, n \in \text{poly}(\kappa)$ such that $1 \leq t \leq n$ as inputs. It returns two vectors of size n of signing/verification keys $\vec{\text{sk}} = (\text{sk}_1, \dots, \text{sk}_n)$ and $\vec{\text{vk}} = (\text{vk}_1, \dots, \text{vk}_n)$ such that each party P_i for $i \in \{1, \dots, n\}$ receives a pair $(\text{sk}_i, \text{vk}_i)$ along with the global verification key vk .
- $\sigma_i \leftarrow \text{ParSign}(\text{pp}, \text{sk}_i, M)$: The partial signing algorithm takes the public parameters pp , a secret signing key sk_i , and a message $M \in \mathcal{M}$ as inputs and returns a partial signature σ_i .
- $0/1 \leftarrow \text{ParVerify}(\text{pp}, \text{vk}_i, \tilde{M}, \sigma_i)$: The partial verification algorithm is a deterministic algorithm that takes the public parameters pp , a verification key vk_i , message $\tilde{M} \in \mathcal{M}$, and a purported partial signature σ_i as inputs.

If σ_i is a valid partial signature, it returns 1 (accept); else, it returns 0 (reject).

- $(\sigma, \perp) \leftarrow \text{Reconst}(\text{pp}, \{i, \sigma_i\}_{i \in \mathcal{T}})$: The reconstruction algorithm is a deterministic algorithm that takes public parameters pp and a set \mathcal{T} of t partial signatures $\{i, \sigma_i\}$ with corresponding indices as inputs and returns an aggregated signature σ or \perp .
- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, \tilde{M}, \sigma)$: The verification algorithm is a deterministic algorithm that takes the public parameters pp , the global verification key vk , a message $\tilde{M} \in \tilde{\mathcal{M}}$, and a purported signature σ as inputs. If σ is a valid signature, it returns 1 (accept); else, it returns 0 (reject).

Three main security properties for TSPS defined over indexed message spaces are *partial verification correctness*, *evaluation correctness*, and *threshold existential unforgeability against chosen indexed message attack* (Threshold EUF-CiMA). Intuitively, partial verification correctness means that any correctly generated partial signature via the ParSign algorithm passes the ParVerify verification checks, and evaluation correctness means that the Reconst algorithm for a set of well-formed partial signatures $\{i, \sigma_i\}_{i \in \mathcal{T}}$ (meaning all with the same index, on a message M) results in a valid aggregated signature σ .

Definition 44 (Partial Verification Correctness). An (n, t) -TSPS scheme satisfies partial verification correctness if for all correctly indexed messages $M \in \mathcal{M}$, $\text{pp} \leftarrow \text{Setup}(\kappa)$, $(\tilde{\text{sk}}, \tilde{\text{vk}}, \text{vk}) \leftarrow \text{KGen}(\text{pp}, \ell, n, t)$ and $i \in [1, n]$ that

$$\Pr [\text{ParVerify}(\text{pp}, \text{vk}_i, \tilde{M}, \text{ParSign}(\text{pp}, \text{sk}_i, M)) = 1] = 1 .$$

Definition 45 (Evaluation Correctness). An (n, t) -TSPS scheme satisfies evaluation correctness if for all correctly indexed messages $M \in \mathcal{M}$, $\text{pp} \leftarrow \text{Setup}(\kappa)$, $(\tilde{\text{sk}}, \tilde{\text{vk}}, \text{vk}) \leftarrow \text{KGen}(\text{pp}, \ell, n, t)$ and $\mathcal{T} \subseteq [1, n]$, $|\mathcal{T}| = t$ that

$$\Pr [\sigma \leftarrow \text{Reconst}(\text{pp}, \{i, \text{ParSign}(\text{pp}, \text{sk}_i, M)\}_{i \in \mathcal{T}}) : \text{Verify}(\text{pp}, \text{vk}, \tilde{M}, \sigma) = 1] = 1 .$$

Threshold Unforgeability. We next define the notion of threshold unforgeability for non-interactive (n, t) -TSPS schemes. The Threshold EUF-CiMA game is defined formally in Figure 6.10. Given a set of party indices $\mathcal{P} = \{1, \dots, n\}$, we assume that the adversary can corrupt up to $t - 1$ parties and that there is at least one honest party. We denote the set of corrupt parties by \mathcal{C} and the set of honest parties by $\mathcal{H} = \mathcal{P} \setminus \mathcal{C}$.

In the unforgeability game, the challenger generates public parameters pp and returns them to the adversary. The adversary chooses the set of corrupted

$\mathbf{G}_{\mathcal{A}}^{\text{T-EUF-CiMA}}(\kappa)$	
1 :	$\text{pp} \leftarrow \text{Setup}(\kappa)$
2 :	$\mathcal{C} \leftarrow \$ \mathcal{A}(\text{pp}) \quad // \text{ set of corrupt signers}$
3 :	if $\mathcal{C} \notin [1, n] \vee \mathcal{C} > t - 1 :$
4 :	return \perp
5 :	else : $\mathcal{H} \leftarrow [1, n] \setminus \mathcal{C} \quad // \text{ set of honest signers}$
6 :	$(\vec{\text{sk}}, \vec{\text{vk}}, \text{vk}) \leftarrow \text{KGen}(\text{pp}, \ell, n, t)$
7 :	$(\tilde{M}^*, \sigma^*) \leftarrow \$ \mathcal{A}^{\mathcal{O}_{\text{PSign}}}(\{\text{sk}_i\}_{i \in \mathcal{C}}, \vec{\text{vk}}, \text{vk})$
8 :	return $(\tilde{M}^* \notin \mathcal{Q}_{\text{EQ}} \wedge \text{Verify}(\text{pp}, \text{vk}, \tilde{M}^*, \sigma^*))$
$\mathcal{O}_{\text{PSign}}(k, id, \tilde{M}) \quad // \quad M = (id, \tilde{M})$	
1 :	if $(k \notin \mathcal{H} \vee (k, id, \star) \in \mathcal{Q}_S \vee (\star, id, \tilde{M}') \in \mathcal{Q}_S, \tilde{M}' \neq \tilde{M}) :$
2 :	return \perp
3 :	else : $\sigma_k \leftarrow \text{ParSign}(\text{pp}, \text{sk}_k, (id, \tilde{M}))$
4 :	$\mathcal{Q}_S \leftarrow \mathcal{Q}_S \cup \{(k, id, \tilde{M})\}$
5 :	$\mathcal{Q}_{\text{EQ}} \leftarrow \mathcal{Q}_{\text{EQ}} \cup \{\text{EQ}(\tilde{M})\}$
6 :	return σ_k

Figure 6.10: Game $\mathbf{G}_{\mathcal{A}}^{\text{T-EUF-CiMA}}(\kappa)$.

participants \mathcal{C} . The challenger then runs KGen to derive the global verification key vk , the individual verification keys $\{\text{vk}_i\}_{i=1}^n$, and the secret signing shares $\{\text{sk}_i\}_{i=1}^n$. It returns vk , $\{\text{vk}_i\}_{i=1}^n$, and the set of corrupt signing shares $\{\text{sk}_j\}_{j \in \mathcal{C}}$ to the adversary. We assume the adversary maintains state before and after KGen .

After key generation, the adversary can request partial signatures on messages of its choosing from honest signers by querying oracle $\mathcal{O}_{\text{PSign}}(\cdot)$.

The adversary wins if it can produce a valid forgery (\tilde{M}^*, σ^*) with respect to the global verification key vk representing the set of n signers, on a message \tilde{M}^* for which no equivalent \tilde{M}^* has been previously queried to $\mathcal{O}_{\text{PSign}}(\cdot)$.

Setup (κ)
1 : $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \mathbf{G}_2) \leftarrow \mathcal{BG}(\kappa); \mathbf{H} : \{0, 1\}^* \rightarrow \mathbb{G}_1$ // select hash function
2 : return $\text{pp} \leftarrow ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \mathbf{G}_2), \mathbf{H})$
KGen (pp, ℓ, n, t)
1 : $x, y_1, \dots, y_\ell \leftarrow \$_{\mathbb{Z}_p^*}$
2 : $\vec{x} \leftarrow \$_{\text{Share}(x, p, n, t)}, \{\vec{y}_j \leftarrow \$_{\text{Share}(y_j, p, n, t)}\}_{j \in [1, \ell]}$
3 : for $i \in [1, n]$:
4 : $\text{sk}_i \leftarrow (\text{sk}_{i0}, \text{sk}_{i1}, \dots, \text{sk}_{i\ell}) = (x_i, y_{i1}, \dots, y_{i\ell})$
5 : $\text{vk}_i \leftarrow (\text{vk}_{i0}, \text{vk}_{i1}, \boxed{\text{vk}_{i1}^*}, \dots, \text{vk}_{i\ell}, \boxed{\text{vk}_{i\ell}^*}) = (\mathbf{G}_2^{x_i}, \mathbf{G}_2^{y_{i1}}, \boxed{g^{y_{i1}}}, \dots, \mathbf{G}_2^{y_{i\ell}}, \boxed{g^{y_{i\ell}}})$
6 : $\vec{\text{sk}} \leftarrow (\text{sk}_1, \dots, \text{sk}_n)$
7 : $\vec{\text{vk}} \leftarrow (\text{vk}_1, \dots, \text{vk}_n)$
8 : $\text{vk} \leftarrow (\text{vk}_{00}, \text{vk}_{01}, \boxed{\text{vk}_{01}^*}, \dots, \text{vk}_{0\ell}, \boxed{\text{vk}_{0\ell}^*}) = (\mathbf{G}_2^x, \mathbf{G}_2^{y_1}, \boxed{g^{y_1}}, \dots, \mathbf{G}_2^{y_\ell}, \boxed{g^{y_\ell}})$
9 : return $(\vec{\text{sk}}, \vec{\text{vk}}, \text{vk})$

Figure 6.11: Our Threshold SPS Construction TSPS. The additional elements in $\boxed{\text{solid boxes}}$ are required for blind signing only (cf. Section 6.6.1).

Definition 46 (Threshold EUF-CiMA). A non-interactive (n, t) -TSPS scheme over indexed message space \mathcal{M} is Threshold EUF-CiMA secure if for all PPT adversaries \mathcal{A} playing game $\mathbf{G}_{\mathcal{A}}^{\text{T-EUF-CiMA}}$ (Figure 6.10), there exists a negligible function ν such that

$$\text{Adv}_{\mathcal{A}}^{\text{T-EUF-CiMA}}(\kappa) = \Pr [\mathbf{G}_{\mathcal{A}}^{\text{T-EUF-CiMA}}(\kappa) = 1] \leq \nu(\kappa) .$$

6.5.1 Our Indexed Multi-Message TSPS

In Figure 6.11, we present our (n, t) -TSPS scheme TSPS over an indexed Diffie-Hellman multi-message space $\mathcal{M}_{\text{MiDH}}^{\text{H}}$, as defined in Figure 6.8.

$\text{ParSign}(\text{pp}, \text{sk}_i, (id, \vec{M}_1, \vec{M}_2))$	$\text{ParVerify}(\text{pp}, \text{vk}_i, (\vec{M}_1, \vec{M}_2), \sigma_i)$
1 : $h \leftarrow H(id)$	1 : $\text{// does not invoke H}$
2 : if $\exists j \in [1, \ell] \mid$	2 : parse $\sigma_i = (h_i, s_i)$
3 : $e(h, M_{2j}) \neq e(M_{1j}, G_2) :$	3 : return $(h_i \neq 1_{G_1} \wedge$
4 : return \perp	4 : $\{M_{1j}\}_{j \in [1, \ell]} \neq 1_{G_1} \wedge$
5 : else : $s_i \leftarrow h^{\text{sk}_{i0}} \prod_{j=1}^{\ell} M_{1j}^{\text{sk}_{ij}}$	5 : $\{e(h_i, M_{2j}) = e(M_{1j}, G_2)\}_{j \in [1, \ell]} \wedge$
6 : return $\sigma_i \leftarrow (h, s_i)$	6 : $e(h_i, \text{vk}_{i0}) \prod_{j=1}^{\ell} e(M_{1j}, \text{vk}_{ij}) = e(s_i, G_2)$
$\text{Reconst}(\text{pp}, \text{vk}, (\vec{M}_1, \vec{M}_2), \{i, \sigma_i\}_{i \in \mathcal{T}})$	$\text{Verify}(\text{pp}, \text{vk}, (\vec{M}_1, \vec{M}_2), \sigma)$
1 : parse $\sigma_i = (h_i, s_i)$	1 : $\text{// does not invoke H}$
2 : if $\exists i, j \in \mathcal{T}, i \neq j \mid h_i \neq h_j$	2 : parse $\sigma = (h, s)$
3 : $\vee \exists i \in \mathcal{T} \mid$	3 : return $(h \neq 1_{G_1} \wedge$
4 : $\text{ParVerify}(\text{pp}, \text{vk}_i, (\vec{M}_1, \vec{M}_2), \sigma_i) = 0$	4 : $\{M_{1j}\}_{j \in [1, \ell]} \neq 1_{G_1} \wedge$
5 : return \perp	5 : $\{e(h, M_{2j}) = e(M_{1j}, G_2)\}_{j \in [1, \ell]} \wedge$
6 : else : $h \leftarrow h_i$	6 : $e(h, \text{vk}_{00}) \prod_{j=1}^{\ell} e(M_{1j}, \text{vk}_{0j}) = e(s, G_2)$
7 : return $\sigma \leftarrow (h, s) = (h, \prod_{i \in \mathcal{T}} s_i^{\lambda_i})$	

Figure 6.11: Our Threshold SPS Construction TSPS. The additional elements in solid boxes are required for blind signing only (cf. Section 6.6.1).

6.5.2 Security of TSPS

Theorem 4. The indexed multi-message (n, t) -Threshold SPS scheme TSPS is correct and Threshold EUF-CiMA secure (Definition 46) in the random oracle model under the EUF-CiMA security of IMM-SPS (Theorem 3).

Proof. **Correctness.** We first show that the proposed TSPS satisfies partial

verification correctness (Definition 44), i.e., any correctly generated partial signature via the ParSign algorithm passes the ParVerify verification checks. Indeed, for all $i \in [1, n]$ and correctly indexed messages $M = (id, \vec{M}_1, \vec{M}_2) \in \mathcal{M}_{\text{MiDH}}^H$, we have:

$$e(h, \text{vk}_{i0}) \prod_{j=1}^{\ell} e(M_{1j}, \text{vk}_{ij}) = e(h, \hat{g}^{x_i}) \prod_{j=1}^{\ell} e(M_{1j}, \hat{g}^{y_{ij}}) e(h^{x_i} \prod_{j=1}^{\ell} M_{1j}^{y_{ij}}, \hat{g}) = e(s_i, \hat{g}) .$$

Next, we show that TSPS satisfies evaluation correctness (Definition 45); that is, the Reconst algorithm for a set of partial signatures $\{i, \sigma_i\}_{i \in \mathcal{T}}$, $\mathcal{T} \subseteq [1, n]$, $|\mathcal{T}| = t$, on a message $M = (id, \vec{M}_1, \vec{M}_2)$ with the same $h \leftarrow H(id)$ results in a valid aggregated signature $\sigma = (h, s)$. Indeed,

$$\begin{aligned} s &= \prod_{i \in \mathcal{T}} s_i^{\lambda_i} = \prod_{i \in \mathcal{T}} (h^{\text{sk}_{i0}} \prod_{j=1}^{\ell} M_{1j}^{\text{sk}_{ij}})^{\lambda_i} = h^{\sum_{i \in \mathcal{T}} \text{sk}_{i0} \lambda_i} \prod_{j=1}^{\ell} M_{1j}^{\sum_{i \in \mathcal{T}} \text{sk}_{ij} \lambda_i} \\ &= h^{\text{sk}_0} \prod_{j=1}^{\ell} M_{1j}^{\text{sk}_j} \end{aligned}$$

where λ_i is the Lagrange coefficient for party P_i with respect to the signing set \mathcal{T} . Next, we show that verification holds for the above aggregated signature σ on message $\vec{M} = (\vec{M}_1, \vec{M}_2)$. Indeed, $\forall j \in [1, \ell]$ we have that $e(h, M_{2j}) = e(h, \hat{g}^{m_j}) = e(h^{m_j}, \hat{g}) = e(M_{1j}, \hat{g})$ and

$$e(h, \text{vk}_0) \prod_{j=1}^{\ell} e(M_{1j}, \text{vk}_j) = e(h, \hat{g}^x) \prod_{j=1}^{\ell} e(M_{1j}, \hat{g}^{y_j}) = e(h^x \prod_{j=1}^{\ell} M_{1j}^{y_j}, \hat{g}) = e(s, \hat{g}) .$$

Note that successful partial signature verification using ParVerify and consistency of h guarantee successful reconstruction.

Need for Uniqueness. The hypothetical attack described after Theorem 2 also works with a partial signing oracle $\mathcal{O}_{\text{PSign}}(\cdot)$. Assume an (n, t) -TSPS with $n > 2t$, and suppose there were no uniqueness requirement for the message space and that the redundant check in Line 2 of the $\mathcal{O}_{\text{PSign}}(\cdot)$ oracle of Figure 6.10 were not present. Then, a forger could obtain $2t$ partial signatures to reconstruct signatures $s = h^x M_1^y$, $s' = h^x M_1'^y$ and compute a forgery $s^* = s^2/s' = h^x (M_1^2/M_1')^y$ that is a valid signature on fresh message M_1^2/M_1' .

Threshold EUF-CiMA. Our proof of security for TSPS resembles that of

threshold BLS in [Bol03]. We wish to show that if there exists a PPT adversary \mathcal{A} that breaks the Threshold EUF-CiMA security (Figure 6.10) of TSPS with non-negligible probability, then we can construct a PPT adversary \mathcal{B} that breaks the EUF-CiMA security (Figure 6.5) of the underlying IMM-SPS scheme (Figure 6.6) with non-negligible probability.

Suppose there exists such a PPT adversary \mathcal{A} . Then, running \mathcal{A} as a subroutine, we construct a reduction \mathcal{B} breaking the EUF-CiMA security of IMM-SPS as follows.

The reduction \mathcal{B} is responsible for simulating oracle responses for queries to $\mathcal{O}_{\text{PSign}}(\cdot)$ and \mathcal{H} . Let \mathcal{Q}_H be the set of \mathcal{H} queries id and their responses. \mathcal{B} may program the random oracle \mathcal{H} . Let \mathcal{Q}_S be the set of $\mathcal{O}_{\text{PSign}}(\cdot)$ queries (k, id, \tilde{M}) and \mathcal{Q}_{EQ} the set of equivalence classes of messages \tilde{M} . \mathcal{B} initializes $\mathcal{Q}_H, \mathcal{Q}_S, \mathcal{Q}_{\text{EQ}}$ to the empty set.

Initialization. \mathcal{B} takes as input public parameters $\text{pp} \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g})$ and an IMM-SPS verification key vk' . In the EUF-CiMA game, \mathcal{B} has access to oracles $\mathcal{O}'_{\text{Sign}}(\cdot)$ and \mathcal{H}' . \mathcal{B} uses $\text{vk}' = (\text{vk}'_{00}, \text{vk}'_{01}, \text{vk}'_{01}^*, \dots, \text{vk}'_{0\ell}, \text{vk}'_{0\ell}^*)$ as the TSPS verification key $\text{vk} = (\text{vk}_{00}, \text{vk}_{01}, \text{vk}_{01}^*, \dots, \text{vk}_{0\ell}, \text{vk}_{0\ell}^*)$.

Simulating Key Generation. \mathcal{B} simulates the key generation algorithm as follows.

- \mathcal{B} defines the pair of secret/verification keys of the corrupted parties $P_i, i \in \mathcal{C}$, as follows. Assume without loss of generality that $|\mathcal{C}| = t - 1$. For all $i \in \mathcal{C}$, \mathcal{B} samples random values $x_{i0}, y_{i1}, \dots, y_{i\ell} \leftarrow \$_{(\mathbb{Z}_p^*)^{\ell+1}}$ and defines party P_i 's secret key as $\text{sk}_i \leftarrow (\text{sk}_{i0}, \text{sk}_{i1}, \dots, \text{sk}_{i\ell}) = (x_{i0}, y_{i1}, \dots, y_{i\ell})$ and the corresponding verification key as $\text{vk}_i \leftarrow (\text{vk}_{i0}, \text{vk}_{i1}, \text{vk}_{i1}^*, \dots, \text{vk}_{i\ell}, \text{vk}_{i\ell}^*) = (\hat{g}^{x_{i0}}, \hat{g}^{y_{i1}}, g^{y_{i1}}, \dots, \hat{g}^{y_{i\ell}}, g^{y_{i\ell}})$.
- To generate the verification key of the honest parties $P_k, k \in \mathcal{H}, \mathcal{H} = [1, n] \setminus \mathcal{C}$, \mathcal{B} proceeds as follows:

1. For all $i \in \tilde{\mathcal{T}} := \mathcal{C} \cup \{0\}$, it computes the Lagrange polynomials evaluated at point k :

$$\tilde{\lambda}_{ki} = L_i^{\tilde{\mathcal{T}}}(k) = \prod_{j \in \tilde{\mathcal{T}}, j \neq i} \frac{(j - k)}{(j - i)}. \quad (6.3)$$

2. It takes the verification keys of corrupted parties $\{\text{vk}_i\}_{i \in \mathcal{C}}$ and the global verification key vk and then computes

$$\begin{aligned}
\mathbf{vk}_k &= (\mathbf{vk}_{k0}, \mathbf{vk}_{k1}, \mathbf{vk}_{k1}^*, \dots, \mathbf{vk}_{k\ell}, \mathbf{vk}_{k\ell}^*) \\
&= \left(\mathbf{vk}_{00}^{\tilde{\lambda}_{k0}} \prod_{i \in \mathcal{C}} \mathbf{vk}_{i0}^{\tilde{\lambda}_{ki}}, \mathbf{vk}_{01}^{\tilde{\lambda}_{k0}} \prod_{i \in \mathcal{C}} \mathbf{vk}_{i1}^{\tilde{\lambda}_{ki}}, \mathbf{vk}_{01}^* \prod_{i \in \mathcal{C}} \mathbf{vk}_{i1}^{\tilde{\lambda}_{ki}}, \dots, \right. \\
&\quad \left. \mathbf{vk}_{0\ell}^{\tilde{\lambda}_{k0}} \prod_{i \in \mathcal{C}} \mathbf{vk}_{i\ell}^{\tilde{\lambda}_{ki}}, \mathbf{vk}_{0\ell}^* \prod_{i \in \mathcal{C}} \mathbf{vk}_{i\ell}^{\tilde{\lambda}_{ki}} \right).
\end{aligned}$$

\mathcal{B} returns the global verification key \mathbf{vk} , $\vec{\mathbf{vk}} = (\mathbf{vk}_1, \dots, \mathbf{vk}_n)$, and secret keys $\{\mathbf{sk}_j\}_{j \in \mathcal{C}}$ to \mathcal{A} .

Simulating Random Oracle $H(id)$: When \mathcal{A} queries H on index id , if $\mathcal{Q}_H[id] = \perp$, then \mathcal{B} queries $H'(id)$, receives a base element h , and sets $\mathcal{Q}_H[id] \leftarrow h$. \mathcal{B} returns $\mathcal{Q}_H[id]$ to \mathcal{A} .

Simulating Signing Oracle $\mathcal{O}_{\text{PSign}}(k, id, \tilde{M})$: When \mathcal{A} queries $\mathcal{O}_{\text{PSign}}(\cdot)$ on (k, id, \tilde{M}) for honest party identifier $k \in \mathcal{H}$ and message $M = (id, \tilde{M}) = (id, \tilde{M}_1, \tilde{M}_2)$, if $k \notin \mathcal{H}$ or $(k, id, \star) \in \mathcal{Q}_S$ or $(\star, id, \tilde{M}') \in \mathcal{Q}_S, \tilde{M}' \neq \tilde{M}$, \mathcal{B} returns \perp . Otherwise, \mathcal{B} does the following:

1. \mathcal{B} looks up $h = \mathcal{Q}_H[id]$, queries $\mathcal{O}'_\sigma(id, \vec{M}_1, \vec{M}_2)$, and receives the signature $\sigma_0 = (h, s_0)$.
2. For all $i \in \mathcal{C}$, \mathcal{B} computes the partial signatures $\sigma_i = (h, s_i) = (h, h^{\mathbf{sk}_{i0}} \prod_{j=1}^{\ell} M_{1j}^{\mathbf{sk}_{ij}})$, as it knows the secret keys of corrupted parties.
3. For all $i \in \tilde{\mathcal{T}} = \mathcal{C} \cup \{0\}$, \mathcal{B} computes Lagrange coefficients $\tilde{\lambda}_{ki}$ as in Equation (6.3).
4. \mathcal{B} updates $\mathcal{Q}_S \leftarrow \mathcal{Q}_S \cup \{(k, id, \tilde{M})\}$ and $\mathcal{Q}_{\text{EQ}} \leftarrow \mathcal{Q}_{\text{EQ}} \cup \{\text{EQ}(\tilde{M})\}$.
5. \mathcal{B} computes $(h, s_k) = (h, s_0^{\tilde{\lambda}_{k0}} \prod_{i \in \mathcal{C}} s_i^{\tilde{\lambda}_{ki}})$ and returns $\sigma_k = (h, s_k)$ to \mathcal{A} .

Output. At the end of the game, \mathcal{A} produces a valid forgery $\sigma^* = (h^*, s^*)$ on message $\tilde{M}^* = (\tilde{M}_1^*, \tilde{M}_2^*)$, and \mathcal{B} returns (\tilde{M}^*, σ^*) as its forgery.

\mathcal{B} correctly simulates key generation and \mathcal{A} 's hash and signing queries. Since \mathcal{A} 's forgery satisfies $\tilde{M}^* \notin \mathcal{Q}_{\text{EQ}}$ and $\text{Verify}(\text{pp}, \mathbf{vk}, \tilde{M}^*, \sigma^*) = 1$, \mathcal{B} 's winning conditions are also satisfied and $\text{Adv}_{\text{TSPS}, \mathcal{A}}^{\text{T-EUF-CiMA}}(\kappa) \leq \text{Adv}_{\text{IMM-SPS}, \mathcal{B}}^{\text{EUF-CiMA}}(\kappa)$. \square

6.6 Applications to Threshold-Issuance Anonymous Credentials

Threshold-Issuance Anonymous Credential (TIAC) systems are a prime use-case of threshold SPS. TIAC systems, defined by Sonnino et al. [Son+19], are used in various applications [KKS22; Tom+22]. A TIAC is an anonymous credential scheme that enables a group of signers (or issuers) to jointly sign a blind message, i.e., issue a credential, without learning the original message. The core ingredient is a blind signing protocol for the used threshold signature scheme. Besides the threshold signature, this protocol relies on two main cryptographic primitives: NIZKs and commitment schemes.

The TIAC protocol of [Son+19], known as Coconut, lacks a rigorous security proof. Recently, Rial and Piotrowska [RP22] conducted a security analysis that required some modifications to the original Coconut scheme, resulting in Coconut⁺⁺. Coconut and Coconut⁺⁺ are based on a threshold Pointcheval-Sanders signature scheme that supports an efficient blind signing protocol.

6.6.1 Blind Signing for TSPS

In Figure 6.11, we show that our TSPS construction also supports threshold blind signing. In addition to the TSPS parameters, the public parameters \mathbf{pp} now contain the common reference string (CRS) of a NIZK and the public parameters of a commitment scheme.

For intuition, we note that in `PrepareBlindSign`, the index is computed as a commitment to \vec{m} , using the generalized Pedersen commitment scheme. The single messages are also committed in a Pedersen commitment, where one commitment parameter is computed on the fly via a random oracle as $h = H(id)$. The hiding property of commitments and the zero-knowledge property of NIZK ensure the blindness.

We note that the construction in Figure 6.11 follows the blind signing protocol for Coconut⁺⁺ closely, with only minor syntactical changes due to the indexed DH message space (highlighted in the figure). Consequently, the validity of the blinding operations readily follows from that of Coconut⁺⁺. The key generation phase is the same as in Figure 6.11.

```

PrepareBlindSign(pp,  $\vec{m}$ ) // pp = (ppc, CRS, H)
1 : parse  $\vec{m} = (m_1, \dots, m_\ell)$ 

2 :  $\omega \leftarrow \mathbb{Z}_p^*$ ,  $id \leftarrow \text{Com}(\text{pp}_c, \vec{m}, \omega) = G_0^\omega \prod_{i=1}^{\ell} G_i^{m_i}$ 

3 :  $(id, (\vec{M}_1, \vec{M}_2)) \leftarrow \text{MiDH}^H(id, \vec{m})$ 

4 : for  $j \in [1, \ell]$  :

5 :      $\omega_{1j}, \omega_{2j} \leftarrow \mathbb{Z}_p^*$ 

6 :      $(\text{cm}_{1j}, \text{cm}_{2j}) \leftarrow (g^{\omega_{1j}} M_{1j}, G_2^{\omega_{2j}} M_{2j})$ 

7 :  $\text{cm} = \{(\text{cm}_{1j}, \text{cm}_{2j})\}_{j=1}^{\ell}$ 

8 :  $\Omega \leftarrow (\omega, \omega_{11}, \omega_{21}, \dots, \omega_{1\ell}, \omega_{2\ell})$ 

9 :  $\pi_s \leftarrow \text{NIZK.Prove} \left\{ \Omega, \vec{m} \mid id = G_0^\omega \prod_{i=1}^{\ell} G_i^{m_i} \wedge \right.$ 

10 :      $\left. \{ \text{cm}_{1j} = g^{\omega_{1j}} H(id)^{m_j} \}_{j=1}^{\ell} \wedge \{ \text{cm}_{2j} = g^{\omega_{2j}} G_2^{m_j} \}_{j=1}^{\ell} \right\}$ 

11 : return  $(\Omega, id, \text{cm}, \pi_s)$ 

```

Figure 6.11: A Threshold Blind Signature with straight-line extraction. Grey boxes mark the changes from Coconut⁺⁺.

$\text{BlindSign}(\text{pp}, \text{sk}_i, \text{id}, \text{cm}, \pi_s)$
<pre> 1 : parse $\text{sk}_i = (\text{sk}_1, \dots, \text{sk}_n)$ 2 : $h \leftarrow \text{H}(\text{id})$ 3 : if $\text{NIZK.Verify}(\text{CRS}, (\text{id}, \text{cm}, h), \pi_s) = 0$: 4 : return \perp 5 : else : $\bar{s}_i \leftarrow h^{\text{sk}_{i0}} \prod_{j=1}^{\ell} \text{cm}_{1j}^{\text{sk}_{ij}}$ 6 : return $\bar{\sigma}_i \leftarrow (h, \bar{s}_i)$ </pre>
$\text{AggCred}(\text{pp}, \{i, \bar{\sigma}_i\}_{i \in \mathcal{T}})$
<pre> 1 : parse $\bar{\sigma}_i = (h_i, \bar{s}_i)$ 2 : if $\exists i, j \in \mathcal{T}, i \neq j \mid h_i \neq h_j$: return \perp 3 : else : $h \leftarrow h_i$ 4 : return $\bar{\sigma} \leftarrow (h, \bar{s}) = (h, \prod_{i \in \mathcal{T}} s_i^{\lambda_i})$ </pre>
$\text{UnBlind}(\text{pp}, \text{vk}, \bar{\sigma}, \Omega)$
<pre> 1 : parse $\bar{\sigma} = (h, \bar{s})$ 2 : return $\sigma := (h, s) \leftarrow (h, \bar{s} \prod_{j=1}^{\ell} (g^{y_j})^{-\omega_j})$ </pre>

Figure 6.11: A Threshold Blind Signature with straight-line extraction.

6.6.2 Removing Rewinding Extractors in TIAC

The TIAC constructions Coconut and Coconut⁺⁺ combine threshold signatures (with blind signing) with generalized Schnorr proofs [Sch90] turned into extractable (knowledge-sound) NIZK proofs via the Fiat-Shamir (FS) heuristic [FS87] in the random oracle model. This, however, is problematic if used within the universal composability (UC) framework [Can01], as extractability for such NIZK proofs requires rewinding. For instance, Coconut⁺⁺ is modeled in the UC framework but requires rewinding to prove that it realizes \mathcal{F}_{AC} [RP22, Theorem 3]. This, in turn, makes the formal security guarantees in the UC framework questionable.

Fischlin’s framework [Fis06], also in the random oracle model, is a well-known technique to avoid rewinding. However, this adds significant overhead that negatively affects its practical applicability. Groth-Sahai (GS) NIZK proofs [GS08] are an efficient alternative NIZK proof system. GS proofs are secure in the standard model and support straight-line extraction of the witnesses, i.e., avoid the rewinding required by the Fischlin transform. This makes them particularly attractive if one is interested in achieving composable security, e.g., UC security. We note that there are known transformations like [Gro06; GOS06; Cha+12] to make GS proofs UC secure despite their malleability. However, GS proofs can only extract group elements.

Towards achieving efficient straight-line extraction without the need of rewinding, we propose to replace the blind issuance threshold Pointcheval-Sanders signature of Coconut⁺⁺ with our blind issuance TSPS. We make the reasonable assumption that the scalar messages (attributes in the TIAC) come from some polynomially bounded message space, e.g., in practice, attributes can be encoded in small scalar values. This modification enables us to provide a GS proof of a valid signature for the showing of a credential with non-revealed messages. Noticing that GS NIZKs are commit-and-proof NIZKs, we can use an additional Schnorr NIZK obtained via Fiat-Shamir to prove a predicate over the scalar messages in the GS commitments. The interesting point is that the latter NIZK only needs to be sound, but does not need to be extractable, as GS commitments can be perfectly binding. Thus, we can avoid rewinding and, due to the polynomially bounded message space, we can extract the scalar messages (attributes in TIAC) efficiently from the straight-line extracted messages from the commitments of the GS proof.

6.7 Conclusion and Open Problems

In this work, we introduce the notion of a threshold structure-preserving signature (TSPS) and present an efficient fully non-interactive TSPS construction. We prove that the proposed TSPS is secure under a new variant of the generalized Pointcheval-Sanders (PS) assumption in the random oracle model. We have shown that our TSPS can be used as a drop-in replacement in TIAC systems to remove the need for rewinding extractors.

While we use a message indexing method in order to construct a non-interactive scheme, a non-interactive TSPS without indexing is an interesting open problem. Moreover, it is interesting to construct schemes that rely on weaker assumptions and avoid the use of the random oracle model. When it comes to the security model, the following two challenging problems remain open: obtaining security under adaptive corruptions more tightly than via a guessing argument from static corruptions, and achieving the strongest notion possible for fully non-interactive schemes (TS-UF-1) [Bel+22]. In general, we believe this work can open a new line of research for structure-preserving multi-party protocols, such as threshold structure-preserving encryption. Moreover, we expect that TSPS will have further applications beyond TIAC systems.

Threshold Structure-Preserving Signatures: Strong and Adaptive Security under Standard Assumptions

Source. A. Mitrokotsa, S. Mukherjee, M. Sedaghat, D. Slamanig, and J. Tomy. “Threshold Structure-Preserving Signatures: Strong and Adaptive Security Under Standard Assumptions”. In: *Public-Key Cryptography – PKC 2024*. Ed. by Q. Tang and V. Teague. Cham: Springer Nature Switzerland, 2024, pp. 163–195. ISBN: 978-3-031-57718-5. DOI: [10.1007/978-3-031-57718-5_6](https://doi.org/10.1007/978-3-031-57718-5_6)

Abstract. Structure-preserving signatures (SPS) have emerged as an important cryptographic building block, as their compatibility with the Groth-Sahai (GS) NIZK framework allows to construct protocols under standard assumptions with reasonable efficiency. Over the last years there has been a significant interest in the design of threshold signature schemes. However, only very recently Crites et al. (ASIACRYPT 2023) have introduced threshold SPS (TSPS) along with a fully non-interactive construction. While this is an important step, their work comes with several limitations. With respect to the construction, they require the use of random oracles, interactive complexity assumptions and are restricted to so called indexed Diffie-Hellman message spaces. Latter limits the use of their construction as a drop-in replacement for SPS. When it comes to security, they only support static corruptions and do not allow partial signature queries for the forgery.

In this paper, we ask whether it is possible to construct TSPS without such restrictions. We start from an SPS from Kiltz, Pan and Wee (CRYPTO 2015) which has an interesting structure, but thresholdizing it requires some

modifications. Interestingly, we can prove it secure in the strongest model (TS-UF-1) for fully non-interactive threshold signatures (Bellare et al., CRYPTO 2022) and even under fully adaptive corruptions. Surprisingly, we can show the latter under a standard assumption without requiring any idealized model. All known constructions of efficient threshold signatures in the discrete logarithm setting require interactive assumptions and idealized models.

Concretely, our scheme in type III bilinear groups under the SXDH assumption has signatures consisting of 7 group elements. Compared to the TSPS from Crites et al. (2 group elements), this comes at the cost of efficiency. However, our scheme is secure under standard assumptions, achieves strong and adaptive security guarantees and supports general message spaces, i.e., represents a drop-in replacement for many SPS applications. Given these features, the increase in the size of the signature seems acceptable even for practical applications.

7.1 Introduction

STRUCTURE-PRESERVING SIGNATURES. Structure-preserving signature schemes (SPS for short) introduced by Abe et al. [Abe+10] are signatures defined over bilinear groups where the messages, public keys and signatures are required to be source group elements. Moreover, signature verification just consists of group membership testing and evaluating pairing product equations (PPE). SPS are very attractive as they can be combined with efficient pairing-based non-interactive zero-knowledge (NIZK) proofs due to Groth and Sahai (GS) [GS08]. This allows to construct many privacy-preserving cryptographic primitives and protocols under standard assumptions with reasonable practical efficiency.

SPS have been used in the literature to construct numerous cryptographic primitives and building blocks. Among them are many variants of signatures such as blind signatures [Abe+10; FHS15], group signatures [Abe+10; LPY15], traceable signatures [Abe+11a], policy-compliant signatures [BMW21; BSW24], homomorphic and network coding signatures [Lib+13; ALP12] and protocols such as anonymous credentials [Cam+15], delegatable anonymous credentials [Fuc11], compact verifiable shuffles [Cha+12] or anonymous e-cash [Bla+11]. Due to their wide range of applications, SPS have attracted significant research interest. Looking ahead to the threshold setting (i.e., TSPS), we note that typical applications of SPS in privacy-preserving applications are as follows: a user obtains a signature from some entity and then prove possession of a valid signature without revealing it using GS NIZK. Consequently, thresholdizing the SPS signing process does not have any impact on the

remaining protocol and thus, TSPS can be considered a drop-in replacement for SPS.

The first SPS scheme presented by Abe et al. in [Abe+10] was followed by a line of research to obtain SPS with short signatures in the generic group model (GGM) [Abe+11b; Abe+14; Gha16; Gha17b], lower bounds [Abe+11b; AGO11; Abe+18a], security under standard assumptions [Abe+12; CDH12; HJ12; KPW15; LPY15; JR17] as well as tight security reductions [Abe+17; JOR18; Gay+18; Abe+18b; Abe+19; CH20].

THRESHOLD SIGNATURES. Motivated by real-world deployments in decentralized systems such as distributed ledger technologies, cryptocurrencies, and decentralized identity management, the use of threshold cryptography [DF90] and in particular threshold signatures has become a very active field of research in the last years with a main focus on ECDSA [GG18; Can+20; Dal+20; Abr+22; Den+21; BS23; Won+23], Schnorr [KG20; CKM23] and BLS [BL22] signatures. We recall that an (n, t) threshold signature allows a set of n potential signers to jointly compute a signature for a message m , which verifies under a single verification key, as long as at least a threshold t many signers participate.

There are different types of constructions in the literature; ones that require multiple rounds of interaction (e.g., ECDSA [GG18; Can+20]), ones that require a pre-processing round that does not depend on the message (often called non-interactive schemes), e.g., FROST [KG20] and finally, ones that are fully non-interactive. The latter are schemes where all the participating signers can simply send a partial signature and the final signatures can then be combined from threshold many valid partial signatures, e.g., BLS [Bol03].

SECURITY OF THRESHOLD SIGNATURES. Although many works on threshold signatures were known in the literature, the rigorous study of security notions was done only very recently. In particular, Bellare et al. in [Bel+22] studied a hierarchy of different notions of security for non-interactive schemes. As our work focuses on fully non-interactive schemes, we do not recall the entire hierarchy but only the ones relevant for this setting. In particular, the TS-UF-0 notion is the weaker one and prohibits adversaries from querying the signing oracle for partial signatures on the challenge message, i.e., the message corresponding to the forged signature. The stronger TS-UF-1 notion, which will be our main focus, allows adversaries to query the signing oracle up to $t - |\text{CS}|$ times for partial signatures, even on the challenge message. Here CS with $|\text{CS}| < t$ denotes the set of (statically corrupted) signers. Surprisingly, the majority of works on threshold signatures in the literature relied on weaker TS-UF-0-style notions instead of the much more realistic TS-UF-1 notion.

Another dimension in the security of threshold signatures is whether they support static or adaptive corruptions. In the case of static corruptions, the adversary has to declare the set of corrupted signers, CS , before seeing any parameters of the system apart from (n, t) . In contrast, an adaptive adversary can choose the set of corrupted signers within a security game based on its view of the execution, which is a realistic assumption in the decentralized setting. All the notions in [Bel+22] consider only a static setting and refer to a complexity leveraging argument for adaptive security. Precisely, it suggests that for small number of parties, a guessing argument can yield adaptive security for any statically secure scheme with a loss of $\binom{n}{t-1}$, i.e., guessing the set of corrupted parties and aborting if the guess is wrong. However, this exponential loss of security can become significant as the number of parties increases, e.g., supporting $n \geq 1024$ (cf. [CKM23]). While there are known generic techniques to lift statically secure schemes to adaptively secure ones [Can+99; JL00; LP01], they all have undesirable side-effects such as relying on additional heavy tools, e.g., non-committing encryption [Can+96], or relying on strong assumptions such as reliable erasure of secret states (cf. [CKM23]).

Apart from the adaptively secure threshold RSA signatures [ADN06], until recently there were no results on adaptively secure threshold signatures based on popular signature schemes in the discrete logarithm or pairing setting. Only very recently Bacho and Loss [BL22] as well as Crites et al. [CKM23] have shown tight adaptive security for threshold versions of the popular BLS [BLS01] and Schnorr schemes [Sch91], respectively. Interestingly, all these adaptive security proofs need to rely on interactive assumptions and in particular variants of the One-More Discrete Logarithm Assumption [Bel+03], which is known as a strong assumption. Only very recently and concurrent to this work, Bacho et al. [Bac+24] as well as Das and Ren [DR23] present schemes from standard and non-interactive assumptions in the pairing-free discrete logarithm setting and pairing setting, respectively. It is interesting that only few of the existing works achieve adaptive security under the TS-UF-1 notion, e.g., [LJY16; BL22; DR23], with [LJY16] being the only one from standard assumptions and without requiring idealized models.

THRESHOLD SPS. Recently, Crites et al. [Cri+23] have extended the concept of threshold signatures to threshold SPS (TSPS). They introduce a definitional framework for fully non-interactive TSPS and provide a construction that is proven secure in the Random Oracle Model (ROM) [BR93] under the hardness of a new interactive assumption, called the GPS_3 assumption, which is analyzed in the Algebraic Group Model (AGM) [FKL18]. The authors start from an SPS proposed by Ghadafi [Gha16], that is secure in the Generic Group Model (GGM), and introduce a message indexing technique to avoid non-linear operations in the signature components and thus to obtain a fully non-interactive threshold

version. While the TSPS proposed in [Cri+23] is highly efficient and compact (only 2 group elements), the defined message space is restricted to a so called indexed Diffie-Hellman message space. This prevents its use as a drop-in-replacement for SPS in arbitrary applications of SPS that are desired to be thresholdized. Additionally, the security of their proposed TSPS is only shown in the TS-UF-0 model, i.e., under static corruptions.

7.1.1 Our Contributions

In this paper, we ask if it is possible to construct TSPS without the aforementioned restrictions and we answer this question affirmatively. We start with an observation that the SPS from Kiltz, Pan and Wee [KPW15] has an interesting structure that makes it amenable for thresholdizing although this process requires some modifications of the original scheme. While Crites et al. [Cri+23] prove security in the TS-UF-0 model, i.e., under static corruptions, we are able to prove our construction is secure in the strongest model (TS-UF-1) for non-interactive threshold signatures [Bel+22] and even under fully adaptive corruptions (which we denote as **adp-TS-UF-1** security). We provide a brief overview in Table 7.1 about our results.

Table 7.1: Overview of security notions and our results. t denotes the threshold, M^* the message corresponding to the forgery, S_1 the set recording signer indices of issued partial signatures and CS the set of corrupted signers.

Security Notion	Corruption Model	Winning Condition	Our Scheme (proof)
TS-UF-0	static corruptions	$S_1(M^*) = \emptyset$	Theorem 5
TS-UF-1	static corruptions	$ S_1(M^*) < t - \text{CS} $	Theorem 6
adp-TS-UF-1	adaptive corruptions	$ S_1(M^*) < t - \text{CS} $	Theorem 7

Interestingly, we can do so by relying on standard assumptions, i.e., the Matrix Diffie-Hellman (MDDH) assumption family [Esc+17; MRV16]. While this comes at some cost in concrete efficiency, as shown in Table 7.2, the overhead is still not significant. For instance, when instantiated in type III bilinear groups under the SXDH assumption ($k = 1$), then signatures consist of 7 group elements. When taking the popular BLS12-381 curve giving around 110 bit of security, this amounts to signatures of size around 380 bytes. Compared to 256 bytes for an RSA signature with comparable security (2048 bit modulus), this gives

an increase of around 50%. This seems perfectly tolerable for most practical applications.

As can be seen from Table 7.2, an important benefit of our TSPS over the one by Crites et al. [Cri+23] is that it is not limited to an indexed Diffie-Hellman message space, but works for arbitrary group message vectors. Thus, it represents a drop-in replacement for SPS when aiming to thresholdize its applications (such as anonymous credentials, e-cash, etc). Moreover, we prove the unforgeability of the proposed TSPS scheme against an adaptive adversary under a stronger TS-UF-1 notion of security. We recall that in contrast, the TSPS proposed by Crites et al. in [Cri+23] only achieves TS-UF-0 security against a static adversary based on an interactive assumption, called GPS₃, in the AGM and ROM.

Table 7.2: Comparison with the existing threshold structure-preserving signature by Crites et al. [Cri+23]. iDH refers to the indexed Diffie-Hellman message spaces. ℓ is the length of the message vector to be signed. $|\mathbb{G}_i|$ denote the bit-length of elements in groups \mathbb{G}_i for $i \in \{1, 2\}$. NI stands for Non-Interactive.

Scheme	Message Space	Signature Size	Number of Pairings	Security Notion	Security Model	Underlying Assumption
[Cri+23]	iDH	$2 \mathbb{G}_1 $	$\ell + 2$	TS-UF-0 (Static)	AGM+ ROM	GPS ₃ (Interactive)
[Mit+24]	\mathbb{G}_1	$(3k + 3) \mathbb{G}_1 + \mathbb{G}_2 $	$5k + \ell + 6$	TS-UF-1 (Adaptive)	Standard Model	\mathcal{D}_k -MDDH (NI)

7.1.2 Technical Overview

Considering the insights discussed in [Cri+23, Section 1], it can be deduced that a fully non-interactive TSPS scheme does not involve any non-linear operations during the partial signing phase. The use of non-linear operations prevents the reconstruction of the final signature from the partial signatures via Lagrange interpolation. These non-linear operations include the inversion of secret share keys (i.e., $[1/\text{sk}_i]$), performing multiplication of distinct randomness and secret shares (i.e., $[r_i \text{sk}_i]$), as well as raising either secret shares or distinct randomness to a power (e.g., $[\text{sk}_i^\zeta]$ or $[r_i^\zeta]$ for any $\zeta > 1$). By employing an indexing approach, the authors in [Cri+23] were able to circumvent the need for multiplying randomness and secret keys, as required by Ghadafi's SPS [Gha16].

In contrast, in our proposed TSPS scheme, we adopt a distinct perspective for avoiding the non-linear operations.

We start from an observation regarding the SPS construction of Kiltz et al. [KPW15] which computes the first and second components of signature on a message $[\mathbf{m}]_1 \in \mathbb{G}_1^\ell$ as:

$$\text{KPW15} : (\sigma_1, \sigma_2) := \left(\underbrace{[(1 \quad \mathbf{m}^\top)]_1 \mathbf{K}}_{\text{SP-OTS}} + \overbrace{\mathbf{r}^\top [\mathbf{B}^\top (\mathbf{U} + \tau \cdot \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1}^{\text{randomized PRF}} \right),$$

where τ is a fresh random integer and \mathbf{r} is a fresh random vector of proper size.¹ Additionally, the secret signing and verification keys are defined as follows:

$$\begin{aligned} \text{KPW15} : \text{sk} &:= (\mathbf{K}, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1, [\mathbf{B}]_1), \\ \text{vk} &:= ([\mathbf{K}\mathbf{A}]_2, [\mathbf{U}\mathbf{A}]_2, [\mathbf{V}\mathbf{A}]_2, [\mathbf{A}]_2), \end{aligned}$$

where \mathbf{K} , \mathbf{A} , \mathbf{B} , \mathbf{U} and \mathbf{V} are random matrices of appropriate dimensions.

As noted by Kiltz et al. in their work [KPW15], their SPS is build based on two fundamental primitives: (i) a structure-preserving one-time signature (SP-OTS), $([(1 \quad \mathbf{m}^\top)]_1 \mathbf{K})$, and (ii) a randomized pseudorandom function (PRF), $(\mathbf{r}^\top [\mathbf{B}^\top (\mathbf{U} + \tau \cdot \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1)$. In their proof of security, we observe that both the building blocks are involved in a loose manner. In particular, in most of their proofs, the reduction samples the SP-OTS signing key \mathbf{K} . It is easy to verify that this observation still holds even when they are arguing about the security of the randomized PRF. Our approach in this work is motivated by this fact which further inspires us to modify Kiltz et al.'s SPS. This adjustment involves defining the secret key as $\text{sk} := \mathbf{K}$ and transferring the remaining parameters to the set of public parameters, i.e., $\text{pp} := ([\mathbf{A}]_2, [\mathbf{U}\mathbf{A}]_2, [\mathbf{V}\mathbf{A}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1)$ and the verification is defined as $\text{vk} := [\mathbf{K}\mathbf{A}]_2$. This rather simple structure allows to obtain the first TSPS for general message spaces in the standard model that can withhold adaptive corruptions without the exponential degradation [Bel+22] and can be proven secure in the TS-UF-1 model.

Consider the following setting. Imagine there are n signers, each equipped with their own signing key, either obtained through the involvement of a trusted

¹Here we follow the group notation by Escala et al. [Esc+17]. See Definition 48 for more details.

dealer or by conducting a Distributed Key Generation (DKG). Their collective objective is to generate a signature for a given message $[\mathbf{m}]_1 \in \mathbb{G}_1^\ell$. It is clear that the linear structure of the SP-OTS $\{[(1 \quad \mathbf{m}^\top)]_1 \mathbf{K}_i\}_{i \in S}$ allows for effortless aggregation when dealing with a collection of them over any subset $S \subseteq [1, n]$. Since the random quantities τ_i and \mathbf{r}_i are independently sampled from a uniform distribution by each signer $i \in [1, n]$, aggregating the PRF elements is still challenging. Consequently, we must explore potential modifications needed to enable the aggregation of these components in comparison to Kiltz et al.’s SPS. We choose to make the tag τ dependent on the message. Thus, the randomized PRF computed by every signer, while still being a random element in the respective space, now allows aggregation. Moreover, by establishing an injective mapping between $[\mathbf{m}]_1$ and τ , we can observe that the randomized PRF structure still guarantees the unforgeability in [KPW15] when attempting to forge a signature on a distinct message. We employ a collision-resistant hash function (CRHF), $\mathcal{H}(\cdot)$, to derive τ from $[\mathbf{m}]_1$. This gives the basis of our construction, where each signer $i \in [1, n]$ computes a partial signature on $[\mathbf{m}]_1$ as

$$(\sigma_1, \sigma_2) = ([(1 \quad \mathbf{m}^\top)]_1 \mathbf{K}_i + \mathbf{r}_i^\top [\mathbf{B}^\top (\mathbf{U} + \tau \cdot \mathbf{V})]_1, [\mathbf{r}_i^\top \mathbf{B}^\top]_1) .$$

Here the signer i is holding the secret share \mathbf{K}_i and chooses a random quantity \mathbf{r}_i of appropriate size and uses $\tau = \mathcal{H}([\mathbf{m}]_1)$. It is easy to verify that this signature can be aggregated in a non-interactive manner. Looking ahead, as a first step we prove that this construction achieves TS-UF-0 security, relying on the well-established and non-interactive standard assumption, i.e., the MDDH assumption.

In case of a TS-UF-1 adversary, we need to deal with the fact that the adversary is allowed to obtain partial signatures on the forged message $[\mathbf{m}^*]_1$. Let us first consider the case of static corruptions. We cannot apply the unforgeability of [KPW15] here as it did not consider strong Uf-CMA security.² To overcome this problem, we introduce an information theoretic step to argue that given a number of partial signatures on the forged message $[\mathbf{m}^*]_1$ below the threshold, the adversary does not gather extra information. In particular, we use Shamir’s secret reconstruction security to ensure that partial signatures do not really leak much information. In this argument, we implicitly use the “selective security” of Shamir’s secret sharing where all the parties in the corrupted set are fixed at the start of the game.

²A signature is called strongly unforgeable when the adversary is not only incapable of producing a valid signature for a fresh message but also, it cannot generate a new signature for a challenge message M^* , by observing a valid signature for the same message M^* .

In the case of adaptive corruptions, an **adp-TS-UF-1** adversary not only is allowed to obtain partial signatures on the forged message $[\mathbf{m}^*]_1$, but also it can corrupt different users to get the corresponding secret keys within the security game, adaptively. We obviously could follow a standard guessing argument to achieve **adp-TS-UF-1** security based on **TS-UF-1** security. However, that direction unfortunately induces a significant security loss. We critically look at our proof of **TS-UF-1** security we have briefly discussed above. To make our construction **adp-TS-UF-1** secure, we show that it is sufficient to argue that the underlying secret sharing achieves “adaptive security”. In this work, we indeed form an argument that Shamir’s secret sharing achieves “adaptive security” which in turn makes our construction **adp-TS-UF-1** secure.

Next, we provide a brief intuition of the formal argument for the “adaptive security” of Shamir’s secret sharing. Informally speaking, we produce a reduction \mathcal{B} to break the “selective security” of Shamir’s secret sharing given an adaptive adversary \mathcal{A} of the secret sharing. Being an information theoretic reduction, \mathcal{B} basically runs the adaptive adversary \mathcal{A} an exponential number of times. Since \mathcal{B} chooses the target set S independently of \mathcal{A} ’s run, the expected number of parallel runs of \mathcal{A} required to ensure all the parties whose secrets \mathcal{A} queried are indeed from S is upper bounded by exponential. Being an information theoretically secure secret sharing scheme, Shamir’s secret sharing basically achieves “adaptive security” due to complexity leveraging but without any degradation in the advantage of the adversary. While we use Shamir secret sharing as our canonical choice, we believe that all information-theoretically secure Linear Secret Sharing schemes can be used instead.

7.2 Preliminaries

Notation. Throughout the paper, we let $\kappa \in \mathbb{N}$ denote the security parameter and 1^κ as its unary representation. Given a polynomial $p(\cdot)$, an efficient randomized algorithm, \mathcal{A} , is called *probabilistic polynomial time*, PPT in short, if its running time is bounded by a polynomial $p(|x|)$ for every input x . A function $\nu(\kappa) : \mathbb{N} \rightarrow \mathbb{R}^+$ is called *negligible* if for every positive polynomial $f(x)$, there exists x_0 such that for all $x > x_0$: $\nu(\kappa) < 1/f(x)$. If clear from the context, we sometimes omit κ for improved readability. The set $\{1, \dots, n\}$ is denoted as $[1, n]$ for a positive integer n . For the equality check of two elements, we use “ $=$ ”. The assign operator is denoted with “ $:=$ ”, whereas the randomized assignment is denoted by $a \leftarrow \$ A$, with a randomized algorithm A and where the randomness is not explicit. We use $\mathcal{D}_1 \approx_c \mathcal{D}_2$ to show two distributions like \mathcal{D}_1 and \mathcal{D}_2 are computationally indistinguishable.

Definition 47 (Secret Sharing). For any two positive integers $n, t < n$, an $(n, t)_{\mathbb{Z}_p^{a \times b}}$ -secret-sharing scheme over $\mathbb{Z}_p^{a \times b}$ for $a, b \in \mathbb{N}$ consists of two functions **Share** and **Rec**. **Share** is a randomized function that takes a secret $\vec{M} \in \mathbb{Z}_p^{a \times b}$ and outputs $(\vec{M}_1, \dots, \vec{M}_n) \leftarrow \text{Share}(\vec{M}, \mathbb{Z}_p^{a \times b}, n, t)$ where $\vec{M}_i \in \mathbb{Z}_p^{a \times b} \forall i \in [1, n]$. The pair of functions **(Share, Rec)** satisfy the following requirements.

- **Correctness:** For any secret $\vec{M} \in \mathbb{Z}_p^{a \times b}$ and a set of parties $\{i_1, i_2, \dots, i_k\} \subseteq [1, n]$ such that $k \geq t$, we have

$$\Pr[\text{Rec}(\vec{M}_{i_1}, \dots, \vec{M}_{i_k}) : (\vec{M}_1, \dots, \vec{M}_n) \leftarrow \text{Share}(\vec{M}, \mathbb{Z}_p^{a \times b}, n, t)] = 1.$$

- **Security:** For any secret $\vec{M} \in \mathbb{Z}_p^{a \times b}$ and a set of parties $S \subseteq [1, n]$ such that $|S| = k < t$, for all information-theoretic adversary \mathcal{A} we have

$$\Pr \left[S = \{i_i\}_{i \in [1, k]} \wedge \vec{M}^* = \vec{M} \left| \begin{array}{l} (\vec{M}_1, \dots, \vec{M}_n) \leftarrow \text{Share}(\vec{M}, \mathbb{Z}_p^{a \times b}, n, t) \\ S \leftarrow \mathbf{rA}() \\ \vec{M}^* \leftarrow \mathbf{rA}(\vec{M}_{i_1}, \dots, \vec{M}_{i_k}) \end{array} \right. \right] = 1/p.$$

We follow standard nomenclature to call this “selective security”. In case of “adaptive security”, \mathcal{A} adaptively chooses $i_j \in [1, n]$ to get \vec{M}_{i_j} one at a time.

We briefly recall the well-known secret sharing scheme due to Shamir [Sha79]. In (n, t) -Shamir Secret Sharing, a secret s is shared to n parties via n evaluations of a polynomial of degree $(t - 1)$. Reconstruction of the secret is essentially Lagrange interpolation where one computes Lagrange polynomials $\{\lambda_{i_j}(x)\}_{j \in S}$ and linearly combine them with the given polynomial evaluations. The degree of the original polynomial confirms that one needs at least $|S| = t$ many polynomial evaluations. In this work, we use Shamir Secret Sharing to secret share a matrix of size $a \times b$, i.e., we use ab -many parallel instances of Shamir Secret Sharing. To keep our exposition simpler, we however assume that we have an (n, t) -Shamir Secret Sharing scheme **(Share, Rec)** which operates on matrices. Since, our work here uses Shamir Secret Sharing quite generically, it is convenient to make such abstraction without going into the details.

Definition 48 (Bilinear Groups). Let an asymmetric bilinear group generator, $\text{ABSGen}(1^\kappa)$, that returns a tuple $\mathcal{BG} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{G}_1, \mathbf{G}_2, e)$, such that \mathbb{G}_1 , \mathbb{G}_2 and \mathbb{G}_T are cyclic groups of the same prime order p such that there is no known homomorphism between \mathbb{G}_1 and \mathbb{G}_2 . \mathbf{G}_1 and \mathbf{G}_2 are the generators of \mathbb{G}_1 and \mathbb{G}_2 , respectively, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficiently computable (non-degenerate) bilinear map with the following properties:

- $\forall a, b \in \mathbb{Z}_p, e([a]_1, [b]_2) = [ab]_T = e([b]_1, [a]_2) ,$
- $\forall a, b \in \mathbb{Z}_p, e([a + b]_1, [1]_2) = e([a]_1, [1]_2)e([b]_1, [1]_2) ,$

where we use an implicit representation of group elements, in which for $\zeta \in \{1, 2, T\}$ and an integer $\alpha \in \mathbb{Z}_p$, the implicit representation of integer α in group \mathbb{G}_ζ is defined by $[\alpha]_\zeta = \alpha \mathbf{G}_\zeta \in \mathbb{G}_\zeta$, where $\mathbf{G}_T = e(\mathbf{G}_1, \mathbf{G}_2)$. To be more general, the implicit representation of a matrix $\mathbf{A} = (\alpha_{ij}) \in \mathbb{Z}_p^{m \times n}$ in \mathbb{G}_ζ is defined by $[\mathbf{A}]_\zeta$ and we have:

$$[\mathbf{A}]_\zeta = \begin{pmatrix} \alpha_{1,1} \mathbf{G}_\zeta & \cdots & \alpha_{1,n} \mathbf{G}_\zeta \\ \alpha_{2,1} \mathbf{G}_\zeta & \cdots & \alpha_{2,n} \mathbf{G}_\zeta \\ \vdots & \ddots & \vdots \\ \alpha_{m,1} \mathbf{G}_\zeta & \cdots & \alpha_{m,n} \mathbf{G}_\zeta \end{pmatrix} .$$

For two matrices \mathbf{A} and \mathbf{B} with matching dimensions we define $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T$.

Definition 49 (Matrix Distribution). Let $k, \ell \in \mathbb{N}^*$ s.t. $k < \ell$. We call $\mathcal{D}_{\ell,k}$ a matrix distribution if it outputs matrices over $\mathbb{Z}_p^{\ell \times k}$ of full rank k in polynomial time. W.l.o.g, we assume the first k rows of matrix $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}$ form an invertible matrix. For $\ell = k + 1$, we write \mathcal{D}_k in short.

Next, we recall the Matrix Decisional Diffie-Hellman assumption, which defines over \mathbb{G}_ζ for any $\zeta = \{1, 2\}$ and states two distributions $([\mathbf{A}]_\zeta, [\mathbf{Ar}]_\zeta)$ and $([\mathbf{A}]_\zeta, [\mathbf{u}]_\zeta)$, where $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}, \mathbf{r} \leftarrow \mathbb{Z}_p^k, \mathbf{u} \leftarrow \mathbb{Z}_p^\ell$ are computationally indistinguishable.

Definition 50 ($\mathcal{D}_{\ell,k}$ -Matrix Decisional Diffie-Hellman ($\mathcal{D}_{\ell,k}$ -MDDH) Assumption [Esc+17]). For a given security parameter κ , let $k, \ell \in \mathbb{N}^*$ s.t. $k < \ell$ and $\mathcal{D}_{\ell,k}$ be a matrix distribution, defined in Definition 49. We say $\mathcal{D}_{\ell,k}$ -MDDH assumption over \mathbb{G}_ζ for $\zeta = \{1, 2\}$ holds, if for all PPT adversaries \mathcal{A} we have:

$$\text{Adv}_{\mathcal{D}_{\ell,k}, \mathbb{G}_\zeta, \mathcal{A}}^{\text{MDDH}}(\kappa) = \left| \Pr [\mathcal{A}(\mathcal{BG}, [\mathbf{A}]_\zeta, [\mathbf{Ar}]_\zeta) = 1] - \Pr [\mathcal{A}(\mathcal{BG}, [\mathbf{A}]_\zeta, [\mathbf{u}]_\zeta) = 1] \right| \leq \nu(\kappa) ,$$

where $\mathcal{BG} \leftarrow \text{ABSGen}(1^\kappa)$, $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}, \mathbf{r} \leftarrow \mathbb{Z}_p^k$ and $\mathbf{u} \leftarrow \mathbb{Z}_p^\ell$.

Definition 51 (\mathcal{D}_k -Kernel Matrix Diffie-Hellman (\mathcal{D}_k -KerMDH) Assumption [MRV16]). For a given security parameter κ , let $k \in \mathbb{N}^*$ and \mathcal{D}_k is a matrix distribution, defined in Definition 49. We say \mathcal{D}_k -KerMDH assumption over \mathbb{G}_ζ for $\zeta = \{1, 2\}$ holds, if for all PPT adversaries \mathcal{A} we have:

$$\text{Adv}_{\mathcal{D}_k, \mathbb{G}_\zeta, \mathcal{A}}^{\text{KerMDH}}(\kappa) = \Pr [\mathbf{c} \in \text{orth}(\mathbf{A}) \mid [\mathbf{c}]_{3-\zeta} \leftarrow \mathcal{A}(\mathcal{BG}, [\mathbf{A}]_\zeta)] \leq \nu(\kappa) .$$

The Kernel Matrix Diffie-Hellman assumption is a natural computational analog of the MDDH assumption. It is well-known that for all $k \geq 1$, $\mathcal{D}_k\text{-MDDH} \Rightarrow \mathcal{D}_k\text{-KerMDH}$ [KPW15; MRV16].

7.3 Threshold Structure-Preserving Signatures

In this section, we first present our security model for Threshold Structure-Preserving Signatures (TSPS) and then present our construction and prove its security.

7.3.1 TSPS: Syntax and Security Definitions

First, we recall the definition of the Threshold Structure-Preserving Signatures (TSPS) from [Cri+23] and their main security properties: correctness and threshold unforgeability. Informally, a threshold signature scheme enables a group of servers S of size n to collaboratively sign a message. In this paper, we assume the existence of a trusted dealer who shares the secret key among the signers. However, there are straightforward and well-known techniques in particular distributed key generation (DKG) protocols (e.g., [Ped92]) that eliminate this needed trust.

Definition 52 (Threshold Structure-Preserving Signatures [Cri+23]). Over a security parameter κ and a bilinear group, an (n, t) -TSPS contains the following PPT algorithms:

- $\text{pp} \leftarrow \text{Setup}(1^\kappa)$: The setup algorithm takes the security parameter κ as input and returns the set of public parameters pp as output.
- $(\{\text{sk}_i, \text{vk}_i\}_{i \in [1, n]}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp}, n, t)$: The key generation algorithm takes the public parameters pp along with two integers n, t s.t. $1 \leq t \leq n$ as inputs. It then returns secret/verification keys $(\text{sk}_i, \text{vk}_i)$ for $i \in [1, n]$ along with a global verification key vk as output.
- $\Sigma_i \leftarrow \text{ParSign}(\text{pp}, \text{sk}_i, [\mathbf{m}])$: The partial signing algorithm takes pp , the i^{th} party's secret key, sk_i , and a message $[\mathbf{m}] \in \mathcal{M}$ as inputs. It then returns a partial signature Σ_i as output.

- $0/1 \leftarrow \text{ParVerify}(\text{pp}, \text{vk}_i, [\mathbf{m}], \Sigma_i)$: The partial verification algorithm as a deterministic algorithm, takes pp , the i^{th} verification key, vk_i , and a message $[\mathbf{m}] \in \mathcal{M}$ along with partial signature Σ_i as inputs. It then returns 1 (accept), if the partial signature is valid and 0 (reject), otherwise.
- $\Sigma \leftarrow \text{CombineSign}(\text{pp}, T, \{\Sigma_i\}_{i \in T})$: The combine algorithm takes a set of partial signatures Σ_i for $i \in T$ along with $T \subseteq [1, n]$ and then returns an aggregated signature Σ as output.
- $0/1 \leftarrow \text{Verify}(\text{pp}, \text{vk}, [\mathbf{m}], \Sigma)$: The verification algorithm as a deterministic algorithm, takes pp , the global verification key, vk , and message $[\mathbf{m}] \in \mathcal{M}$ along with an aggregated signature Σ as inputs. It then returns 1 (accept), if the aggregated signature is valid and 0 (reject), otherwise.

Correctness. Correctness guarantees that a signature obtained from a set $T \subseteq [1, n]$ s.t. $|T| \geq t$ of honest signers always verifies.

Definition 53 (Correctness). An (n, t) -TSPS scheme is called correct if we have:

$$\Pr \left[\begin{array}{l} \forall \text{pp} \leftarrow \text{Setup}(1^\kappa), (\{\text{sk}_i, \text{vk}_i\}_{i \in [1, n]}, \text{vk}) \leftarrow \text{KeyGen}(\text{pp}, n, t), [\mathbf{m}] \in \mathcal{M}, \\ \Sigma_i \leftarrow \text{ParSign}(\text{pp}, \text{sk}_i, [\mathbf{m}]) \text{ for } i \in [1, n], \forall T \subseteq [1, n], |T| \geq t, \\ \Sigma \leftarrow \text{CombineSign}(\text{pp}, T, \{\Sigma_i\}_{i \in T}) : \text{Verify}(\text{pp}, \text{vk}, [\mathbf{m}], \Sigma) = 1 \end{array} \right] = 1.$$

Unforgeability. Our security model for threshold unforgeability extends the one from Crites et al. [Cri+23]. Therefore, we need to recall a recent work by Bellare et al. [Bel+22], which investigates existing security notions and proposes stronger and more realistic security notions for threshold signatures under static corruptions. In particular, the authors in [Bel+22] present a hierarchy of different notions of security for non-interactive schemes. We focus on fully non-interactive schemes, i.e., ones that do not require one round of pre-processing, and thus in this paper only the TS-UF-0 and TS-UF-1 notions are relevant. The TS-UF-0 notion is a less stringent notion of unforgeability. In this context, if the adversary has previously seen a partial signature on a challenge message $[\mathbf{m}^*]$, the act of forging a signature for that specific message is considered as a trivial forgery. The security of the original TSPS is proved under this notion of unforgeability.

The stronger TS-UF-1 notion, which is our main focus, allows adversaries to query the signing oracle up to $t - |\text{CS}|$ times for partial signatures, even on the challenge message. Here CS with $|\text{CS}| < t$ denotes the set of (statically

corrupted) signers. Moreover, the model in [Bel+22] as well as the TSPS construction in [Cri+23] only considers static corruptions. But we also integrate the core elements of the model introduced in the recent work by Crites et al. [CKM23], adapted to fully non-interactive schemes, to support fully adaptive corruptions. Our model is depicted in Figure 7.1. The dashed box as well as the solid white box in the winning condition apply to the TS-UF-0 and TS-UF-1 notions, respectively. Grey boxes are only present in the adaptive version of the game, i.e., adp-TS-UF-0 and adp-TS-UF-1 .

Definition 54 (Threshold Unforgeability). Let $\text{TSPS} = (\text{Setup}, \text{KeyGen}, \text{ParSign}, \text{ParVerify}, \text{CombineSign}, \text{Verify})$ be an (n, t) -TSPS scheme over message space \mathcal{M} and let $\text{prop} \in \{\text{TS-UF-b}, \text{adp-TS-UF-b}\}_{b \in \{0,1\}}$. The advantage of a PPT adversary \mathcal{A} playing described security games in Figure 7.1, is defined as,

$$\text{Adv}_{\text{TSPS}, \mathcal{A}}^{\text{prop}}(\kappa) = \Pr [\mathbf{G}_{\text{TS}, \mathcal{A}}^{\text{prop}}(\kappa) = 1] \quad .$$

A TSPS achieves prop -security if we have, $\text{Adv}_{\text{TSPS}, \mathcal{A}}^{\text{prop}}(\kappa) \leq \nu(\kappa)$.

7.3.2 Core Lemma

Prior to introducing our construction, we first present the core lemma that forms a basis in the proofs of our proposed TSPS. It extends the core lemmas from [KW15; KPW15], however it is important to note that both of these schemes are standard SPS, where there was no need to simulate signatures on forged messages. In contrast, both the TS-UF-1 and adp-TS-UF-1 security models necessitate the simulation of partial signature queries on forged messages. Thus we define our core lemma with a key difference being the introduction of a new oracle, denoted as $\mathcal{O}^{**}(\cdot)$.

Lemma 1 (Core Lemma). Let the game $\mathbf{G}_{\mathcal{D}_k, \text{ABSGen}}^{\text{Core}}(\kappa)$ be defined as Figure 7.2. For any adversary \mathcal{A} with the advantage of $\text{Adv}_{\mathcal{D}_k, \text{ABSGen}, \mathcal{A}}^{\text{Core}}(\kappa) := |\Pr[\mathbf{G}_{\mathcal{D}_k, \text{ABSGen}}^{\text{Core}}(\kappa)] - 1/2|$, there exists an adversary \mathcal{B} against the \mathcal{D}_k -MDDH assumption such that with the running time $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ it holds that

$$\text{Adv}_{\mathcal{D}_k, \text{ABSGen}, \mathcal{A}}^{\text{Core}}(\kappa) \leq 2q \text{Adv}_{\mathcal{D}_k, \mathcal{G}_1, \mathcal{B}}^{\text{MDDH}}(\kappa) + q/p \quad ,$$

where q is a bound on the number of queries requested by adversary \mathcal{A} for oracle $\mathcal{O}_b(\cdot)$. Note that \mathcal{A} can only query the other oracles only once.

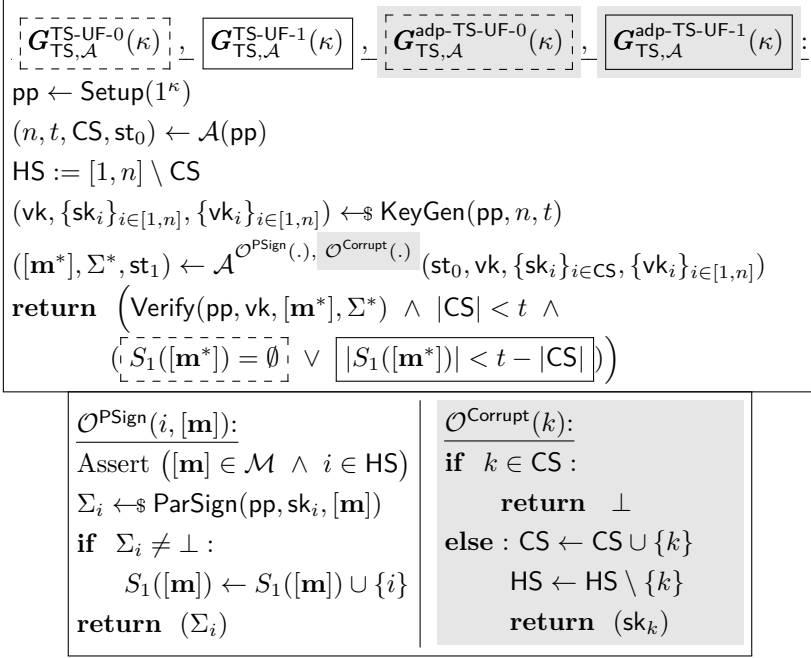


Figure 7.1: Games defining the $\boxed{\text{TS-UF-0}}$, $\boxed{\text{TS-UF-1}}$, $\boxed{\text{adp-TS-UF-0}}$, and $\boxed{\text{adp-TS-UF-1}}$ unforgeability notions of threshold signatures.

Proof Sketch. The proof of this lemma uses the proof of core lemma in [KW15; KPW15]. The fundamental concept of these proofs is primarily an information-theoretic argument that $(\mathbf{t}^\top(\mathbf{U} + \tau\mathbf{V}), \mathbf{U} + \tau^*\mathbf{V})$ is identically distributed to $(\mu\mathbf{a}^{\perp\top} + \mathbf{t}^\top(\mathbf{U} + \tau\mathbf{V}), \mathbf{U} + \tau^*\mathbf{V})$ for $\mu \leftarrow \mathbb{Z}_p$, $\mathbf{a}^\perp, \mathbf{t} \leftarrow \mathbb{Z}_p^{k+1}$ and $\tau \neq \tau^*$. We use $[b\mu\mathbf{a}^{\perp\top} + \mathbf{t}^\top(\mathbf{U} + \tau\mathbf{V})]_1$ to simulate $\mathcal{O}_b([\tau]_1)$, $[\mathbf{U} + \tau^*\mathbf{V}]_2$ to simulate $\mathcal{O}^*([\tau^*]_2)$ and $[\mathbf{B}^\top(\mathbf{U} + \tau^*\mathbf{V})]_1$ to simulate $\mathcal{O}^{**}([\tau^*]_1)$. The detailed proof can be found in Section 7.3.5. \square

7.3.3 Our Threshold SPS Construction

Given a collision resistant hash function, $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$, and message space $\mathcal{M} := \mathbb{G}_1^\ell$, we present our (n, t) -TSPS construction in Figure 7.3. This consists of six main PPT algorithms – Setup, KeyGen, ParSign, ParVerify, CombineSign

$\text{Init}():$ $\mathbf{A}, \mathbf{B} \leftarrow \mathcal{D}_k, \mathbf{U}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)}$ $\text{vk} := (\mathbf{A}, \mathbf{U}\mathbf{A}, \mathbf{V}\mathbf{A}, [\mathbf{B}]_1, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1)$ $b \leftarrow \{0, 1\}$ Let $\mathbf{a}^\perp \leftarrow \mathbb{Z}_p^{1 \times (k+1)}$ such that $\mathbf{a}^\perp \mathbf{A} = \mathbf{0}$ $q := 0, \mathcal{Q}_{\text{tag}} := \emptyset$ return vk	$\mathcal{O}^*([\tau^*]_2):$ return $[\mathbf{U} + \tau^* \mathbf{V}]_2$ $\mathcal{O}^{**}([\tau^*]_1):$ return $[\mathbf{B}^\top (\mathbf{U} + \tau^* \mathbf{V})]_1$
$\mathcal{O}_b([\tau]_1):$ $\mu \leftarrow \mathbb{Z}_p, \mathbf{r} \leftarrow \mathbb{Z}_p^k, q := q + 1$ $\mathcal{Q}_{\text{tag}} := \mathcal{Q}_{\text{tag}} \cup \{\tau\}$ return $([b\mu \mathbf{a}^\perp + \mathbf{r}^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1)$	

Figure 7.2: Game defining the core lemma, $\mathbf{G}_{\mathcal{D}_k, \text{ABSGen}}^{\text{Core}}(\kappa)$.

and Verify, as defined in Definition 52. Similar to the settings of Bellare et al. [Bel+22], we also assume there is a dealer who is responsible for generating key pairs for all signers and a general verification key.

7.3.4 Security

Theorem 5. Under the \mathcal{D}_k -MDDH Assumption in \mathbb{G}_1 and \mathcal{D}_k -KerMDH Assumption in \mathbb{G}_2 , the proposed Threshold Structure-Preserving Signature construction in Figure 7.3 achieves TS-UF-0 security against an efficient adversary making at most q partial signature queries.

Proof. We prove the above theorem through a series of games and we use \mathbf{Adv}_i to denote the advantage of the adversary \mathcal{A} in winning the Game i . The games are described below.

Game 0. This is the TS-UF-0 security game described in Definition 54. As shown in Figure 7.4, an adversary \mathcal{A} after receiving the set of public parameters, pp , returns (n, t, CS) , where n , t and CS represents the total number of signers, the threshold, and the set of corrupted signers, respectively. The adversary can query the partial signing oracle $\mathcal{O}^{\text{PSign}}(\cdot)$ to receive partial signatures and q represents the total number of these queries. In the end, the adversary outputs a message $[\mathbf{m}^*]_1$ and a forged signature Σ^* .

Setup(1^κ):

- 1: $\mathcal{BG} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e) \leftarrow \mathcal{ABSGen}(1^\kappa)$.
- 2: $\mathbf{A}, \mathbf{B} \leftarrow \mathcal{D}_k$, $\mathbf{U}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)}$.
- 3: $\text{pp} := ([\mathbf{A}]_2, [\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1)$.

KeyGen(pp, n, t):

- 1: $\mathbf{K} \leftarrow \mathbb{Z}_p^{(\ell+1) \times (k+1)}$.
- 2: $\mathbf{K}_1, \dots, \mathbf{K}_n \leftarrow \text{Share}(\mathbf{K}, \mathbb{Z}_p^{(\ell+1) \times (k+1)}, n, t)$.
- 3: Set $\text{vk} := [\mathbf{KA}]_2$ and $(\text{sk}_i, \text{vk}_i) := (\mathbf{K}_i, [\mathbf{K}_i \mathbf{A}]_2)$.

ParSign(pp, $\text{sk}_i, [\mathbf{m}]_1$):

- 1: $\mathbf{r}_i \leftarrow \mathbb{Z}_p^k$.
- 2: $\tau := \mathcal{H}([\mathbf{m}]_1)$.
- 3: Output $\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ s.t.
- 4: $\sigma_1 := \left[\begin{pmatrix} 1 & \mathbf{m}^\top \end{pmatrix} \right]_1 \mathbf{K}_i + \mathbf{r}_i^\top [\mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1$,
 $\sigma_2 := [\mathbf{r}_i^\top \mathbf{B}^\top]_1$,
 $\sigma_3 := [\tau \mathbf{r}_i^\top \mathbf{B}^\top]_1$,
 $\sigma_4 := [\tau]_2$.

ParVerify(pp, $\text{vk}_i, [\mathbf{m}]_1, \Sigma_i$): Output 1 if the following checks hold; else output 0.

- 1: $e(\sigma_1, [\mathbf{A}]_2) = e\left(\left[\begin{pmatrix} 1 & \mathbf{m}^\top \end{pmatrix}\right]_1, \text{vk}_i\right) \cdot e(\sigma_2, [\mathbf{UA}]_2) \cdot e(\sigma_3, [\mathbf{VA}]_2)$.
- 2: $e(\sigma_2, \sigma_4) = e(\sigma_3, [1]_2)$.

CombineSign(pp, $S, \{\Sigma_i\}_{i \in S}$):

- 1: Parse $\Sigma_i = (\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}, \sigma_{i,4})$ for all $i \in S$.
- 2: Compute Lagrange polynomials λ_i for $i \in S$.
- 3: Output $\Sigma := (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4)$ s.t.
- 4: $\hat{\sigma}_1 := \prod_{i \in S} \sigma_{i,1}^{\lambda_i} = \left[\begin{pmatrix} 1 & \mathbf{m}^\top \end{pmatrix} \sum_{i \in S} \lambda_i \mathbf{K}_i \right]_1 + \sum_{i \in S} \lambda_i \mathbf{r}_i^\top [\mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1 =$
 $\left[\begin{pmatrix} 1 & \mathbf{m}^\top \end{pmatrix} \mathbf{K} \right]_1 + \mathbf{r}^\top [\mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1$,
 $\hat{\sigma}_2 := \prod_{i \in S} \sigma_{i,2}^{\lambda_i} = \left[\sum_{i \in S} \lambda_i \mathbf{r}_i^\top \mathbf{B}^\top \right]_1 = [\mathbf{r}^\top \mathbf{B}^\top]_1$,
 $\hat{\sigma}_3 := \prod_{i \in S} \sigma_{i,3}^{\lambda_i} = \left[\sum_{i \in S} \tau \lambda_i \mathbf{r}_i^\top \mathbf{B}^\top \right]_1 = [\tau \mathbf{r}^\top \mathbf{B}^\top]_1$,
 $\hat{\sigma}_4 := \sigma_{i,4}$.

Verify(pp, vk, $[\mathbf{m}]_1, \Sigma$): Output 1 if the following checks satisfy; else output 0.

- 1: $e(\hat{\sigma}_1, [\mathbf{A}]_2) = e\left(\left[\begin{pmatrix} 1 & \mathbf{m}^\top \end{pmatrix}\right]_1, \text{vk}\right) \cdot e(\hat{\sigma}_2, [\mathbf{UA}]_2) \cdot e(\hat{\sigma}_3, [\mathbf{VA}]_2)$.
- 2: $e(\hat{\sigma}_2, \hat{\sigma}_4) = e(\hat{\sigma}_3, [1]_2)$.

Figure 7.3: Our proposed TSPS construction.

$G_0(\kappa)$:

- 1: $\mathcal{BG} \leftarrow \text{ABSGen}(1^\kappa)$,
- 2: $\mathbf{A}, \mathbf{B} \leftarrow \mathcal{D}_k$,
- 3: $\mathbf{U}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1) \times (k+1)}$.
- 4: $\text{pp} := ([\mathbf{A}]_2, [\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1)$.
- 5: $(n, t, \text{CS}, \text{st}_0) \leftarrow \mathcal{A}(\text{pp})$.
- 6: Assert $\text{CS} \subset [1, n]$.
- 7: Sample $\mathbf{K} \leftarrow \mathbb{Z}_p^{(\ell+1) \times (k+1)}$.
- 8: $(\mathbf{K}_1, \dots, \mathbf{K}_n) \leftarrow \text{Share}(\mathbf{K}, \mathbb{Z}_p^{(\ell+1) \times (k+1)}, n, t)$.
- 9: $\text{vk} := [\mathbf{KA}]_2$.
- 10: **for** $i \in [1, n]$:
- 11: $\text{sk}_i := \mathbf{K}_i, \text{vk}_i := [\mathbf{K}_i \mathbf{A}]_2$.
- 12: $([\mathbf{m}^*]_1, \Sigma^*, \text{st}_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\text{PSign}(\cdot)}}(\text{st}_0, \text{vk}, \{\text{sk}_i\}_{i \in \text{CS}}, \{\text{vk}_i\}_{i \in [1, n]})$.
- 13: **return** $(\text{Verify}(\text{pp}, \text{vk}, [\mathbf{m}^*]_1, \Sigma^*) \wedge |\text{CS}| < t \wedge S_1([\mathbf{m}^*]_1) = \emptyset)$

$\mathcal{O}^{\text{PSign}}(i, [\mathbf{m}]_1)$:

- 1: Assert $([\mathbf{m}]_1 \in \mathcal{M} \wedge i \in \text{HS})$.
- 2: $\mathbf{r}_i \leftarrow \mathbb{Z}_p^k$.
- 3: $\tau := \mathcal{H}([\mathbf{m}]_1)$.
- 4: $\sigma_1 := \left[\begin{pmatrix} 1 & \mathbf{m}^\top \end{pmatrix} \mathbf{K}_i + \mathbf{r}_i^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V}) \right]_1$,
- $\sigma_2 := [\mathbf{r}_i^\top \mathbf{B}^\top]_1$,
- $\sigma_3 := [\tau \mathbf{r}_i^\top \mathbf{B}^\top]_1$,
- $\sigma_4 := [\tau]_2$.
- 5: $\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$.
- 6: **if** $\Sigma_i \neq \perp$:
- 7: $S_1([\mathbf{m}]_1) := S_1([\mathbf{m}]_1) \cup \{i\}$.
- 8: **return** Σ_i

$\text{Verify}(\text{pp}, \text{vk}, [\mathbf{m}^*]_1, \Sigma^*)$:

- 1: Parse Σ^* as $(\widehat{\sigma}_1, \widehat{\sigma}_2, \widehat{\sigma}_3, \widehat{\sigma}_4)$.
- 2: **return** $\left(e(\widehat{\sigma}_1, [\mathbf{A}]_2) = e \left(\left[\begin{pmatrix} 1 & \mathbf{m}^{*\top} \end{pmatrix} \right]_1, [\mathbf{KA}]_2 \right) \cdot e(\widehat{\sigma}_2, [\mathbf{UA}]_2) \cdot \right.$
 $\left. e(\widehat{\sigma}_3, [\mathbf{VA}]_2) \wedge e(\widehat{\sigma}_2, \widehat{\sigma}_4) = e(\widehat{\sigma}_3, [1]_2) \right)$

Figure 7.4: Game_0 .

Game 1. We modify the verification procedure to the one described in Figure 7.5. Consider any forged message/signature pair $([\mathbf{m}^*]_1, \Sigma^* = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4))$, where $e(\hat{\sigma}_2, \hat{\sigma}_4) = e(\hat{\sigma}_3, [1]_2)$, $|\mathbf{CS}| < t$ and $S_1([\mathbf{m}^*]_1) = \emptyset$. It is easy to observe that if the pair $([\mathbf{m}^*]_1, \Sigma^*)$ meets the $\text{Verify}^*(\cdot)$ criteria, outlined in Figure 7.5, it also satisfies $\text{Verify}(\cdot)$ procedure, described in Figure 7.4. This is primarily due to the fact that:

$$\begin{aligned} e(\hat{\sigma}_1, [\mathbf{A}]_2) &= e([(1 \quad \mathbf{m}^{*\top})]_1, [\mathbf{KA}]_2) \cdot e(\hat{\sigma}_2, [\mathbf{UA}]_2) \cdot e(\hat{\sigma}_3, [\mathbf{VA}]_2) \\ &\Longleftarrow e(\hat{\sigma}_1, [1]_2) = e([(1 \quad \mathbf{m}^{*\top})]_1, [\mathbf{K}]_2) \cdot e(\hat{\sigma}_2, [\mathbf{U}]_2) \cdot e(\hat{\sigma}_3, [\mathbf{V}]_2) \\ &\Longleftrightarrow e(\hat{\sigma}_1, [1]_2) = e([(1 \quad \mathbf{m}^{*\top}) \mathbf{K}]_1, [1]_2) \cdot e(\hat{\sigma}_2, [\mathbf{U} + \tau^* \mathbf{V}]_2) . \end{aligned}$$

Assume there exists a message/signature pair like $([\mathbf{m}^*]_1, \Sigma^* = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4))$ that satisfies $\text{Verify}(\cdot)$ and not $\text{Verify}^*(\cdot)$, then we can compute a non-zero vector \mathbf{c} in the kernel of \mathbf{A} as follows:

$$\mathbf{c} := \hat{\sigma}_1 - ([(1 \quad \mathbf{m}^{*\top}) \mathbf{K}]_1 + \hat{\sigma}_2 \mathbf{U} + \hat{\sigma}_3 \mathbf{V}) \in \mathbb{G}_1^{1 \times (k+1)} .$$

According to $\mathcal{D}_k\text{-KerMDH}$ assumption over \mathbb{G}_2 described in Definition 51, computing such a vector \mathbf{c} is considered computationally hard. Thus,

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \text{Adv}_{\mathcal{D}_k, \mathbb{G}_2, \mathcal{B}_0}^{\text{KerMDH}}(\kappa) .$$

$\text{Verify}^*(\text{pp}, \text{vk}, [\mathbf{m}^*]_1, \Sigma^*):$ 1: Parse Σ^* as $(\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4 = [\tau^*]_2)$. 2: return $\left(e(\hat{\sigma}_1, [1]_2) = e([(1 \quad \mathbf{m}^{*\top}) \mathbf{K}]_1, [1]_2) \cdot e(\hat{\sigma}_2, [\mathbf{U} + \tau^* \mathbf{V}]_2) \wedge \right.$ $\left. e(\hat{\sigma}_2, \hat{\sigma}_4) = e(\hat{\sigma}_3, [1]_2) \right)$
--

Figure 7.5: Modifications in **Game**₁.

Game 2. On receiving a partial signature query on a message $[\mathbf{m}_i]_1$, the query list is updated to include the message $[\mathbf{m}_i]_1$ along with its corresponding tag, $\tau_i := \mathcal{H}([\mathbf{m}_i]_1)$. The challenger aborts if an adversary can generate two tuples $([\mathbf{m}_i]_1, \tau_i)$, $([\mathbf{m}_j]_1, \tau_j)$ with $[\mathbf{m}_i]_1 \neq [\mathbf{m}_j]_1$ and $\tau_i = \tau_j$. By the collision resistance property of the underlying hash function we have,

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \text{Adv}_{\mathcal{H}}^{\text{CRHF}}(\kappa) .$$

Game 3. In this game, we introduce randomness to the partial signatures by adding $\mu \mathbf{a}^\perp$ to each partial signature, where μ is chosen uniformly

```

 $\mathcal{O}^{\text{PSign}^*}(i, [\mathbf{m}]_1):$ 
1: Assert  $([\mathbf{m}]_1 \in \mathcal{M} \wedge i \in \text{HS})$ .
2:  $\mathbf{r}_i \leftarrow \mathbb{Z}_p^k, \tau := \mathcal{H}([\mathbf{m}]_1), \mu \leftarrow \mathbb{Z}_p$ .
3:  $\sigma_1 := [(1 \quad \mathbf{m}^\top) \mathbf{K}_i + \mu \mathbf{a}^\perp + \mathbf{r}_i^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1$ ,
    $\sigma_2 := [\mathbf{r}_i^\top \mathbf{B}^\top]_1$ ,
    $\sigma_3 := [\tau \mathbf{r}_i^\top \mathbf{B}^\top]_1$ ,
    $\sigma_4 := [\tau]_2$ .
4:  $\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ .
5: if  $\Sigma_i \neq \perp$  :
6:    $S_1([\mathbf{m}]_1) := S_1([\mathbf{m}]_1) \cup \{i\}$ .
7: return  $\Sigma_i$ 

```

Figure 7.6: Modifications in Game_3 .

at random and the vector \mathbf{a}^\perp is a non-zero vector in the kernel of \mathbf{A} . The new partial signatures satisfy the verification procedure as $\mathbf{a}^\perp \mathbf{A} = \mathbf{0}$. Figure 7.6 describes the new partial signing oracle, $\mathcal{O}^{\text{PSign}^*}(\cdot)$.

Lemma 2. $|\text{Adv}_2 - \text{Adv}_3| \leq 2q \text{Adv}_{\mathcal{D}_k, \mathbb{G}_1, \mathcal{B}_1}^{\text{MDDH}}(\kappa) + q/p$.

Proof. We prove this lemma through a reduction to the core lemma, Lemma 1. Let us assume there exists an adversary \mathcal{A} that can distinguish the games Game_2 and Game_3 , we can use it to build an adversary \mathcal{B}_1 , defined in Figure 7.7, which breaks the core lemma, Lemma 1. The adversary \mathcal{B}_1 has access to four oracles, $\text{Init}(\cdot), \mathcal{O}_b(\cdot), \mathcal{O}^*(\cdot), \mathcal{O}^{**}(\cdot)$, however in this reduction, we only use the first three oracles, defined as follows:

Oracle $\text{Init}(\cdot)$: The oracle Init provides the set of public parameters pp .

Oracle $\mathcal{O}_b(\cdot)$: On the i -th query to this oracle on $[\tau]_1$, it outputs $([b\mu \mathbf{a}^\perp + \mathbf{r}_i^\top \mathbf{B}^\top (\mathbf{U} + \tau \cdot \mathbf{V})]_1, [\mathbf{r}_i^\top \mathbf{B}^\top]_1)$ depending on a random bit b .

Oracle $\mathcal{O}^*(\cdot)$: On input $[\tau^*]_2$, it returns $[\mathbf{U} + \tau^* \mathbf{V}]_2$.

When the lemma challenger selects the challenge bit as $b = 0$, it leads to the game Game_2 , and when $b = 1$, it results in the game Game_3 . All the other values are simulated perfectly. Thus, $|\text{Adv}_2 - \text{Adv}_3| \leq \text{Adv}_{\mathcal{D}_k, \text{ABSGen}, \mathcal{B}_1}^{\text{Core}}(\kappa)$ holds and therefore we have,

$$|\text{Adv}_2 - \text{Adv}_3| \leq 2q \text{Adv}_{\mathcal{D}_k, \mathbb{G}_1, \mathcal{B}}^{\text{MDDH}}(\kappa) + q/p.$$

□

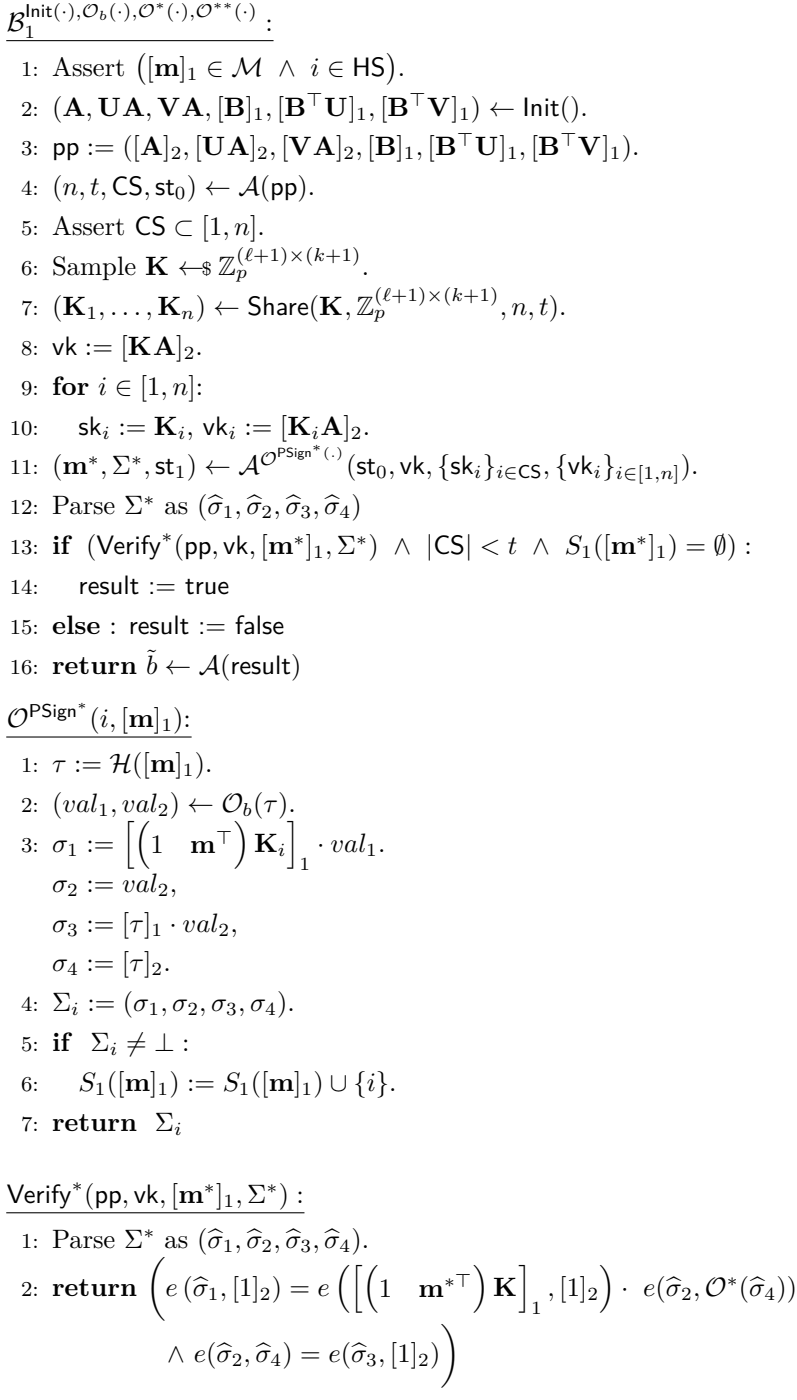
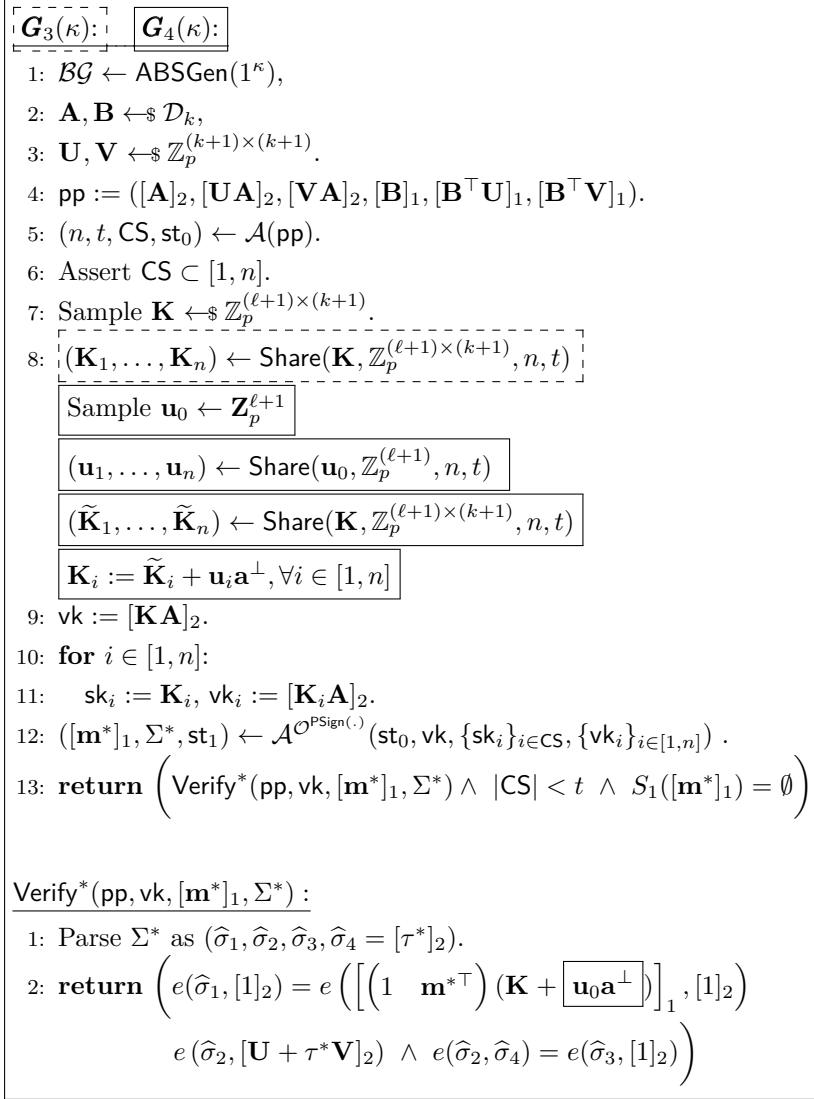


Figure 7.7: Reduction to the core lemma in Lemma 1.

Figure 7.8: Modification from Game_3 to Game_4 .

Game 4. In this game, we apply the modifications described in Figure 7.8. Shamir secret sharing (see Definition 47) ensures that $(\mathbf{K}_1, \dots, \mathbf{K}_n)$ in Game₃ and $(\tilde{\mathbf{K}}_1, \dots, \tilde{\mathbf{K}}_n)$ in Game₄ have identical distributions. W.l.o.g, \mathbf{K}_i in Game₃ and $\tilde{\mathbf{K}}_i$ in Game₄ are identically distributed. In Game₄, on the other hand, $\tilde{\mathbf{K}}_i$ and $\mathbf{K}_i = \tilde{\mathbf{K}}_i - \mathbf{u}_i \mathbf{a}^\perp$ are identically distributed. Combining these observations, it follows that \mathbf{K}_i in Game₃ and \mathbf{K}_i in Game₄ are identically distributed for all $i \in [1, n]$. Consequently, it can be deduced that \mathbf{K} in Game₃ and $\mathbf{K} + \mathbf{u}_0 \mathbf{a}^\perp$ in Game₄ are identically distributed. Therefore, this change is just a conceptual change and we have,

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| = 0.$$

Now, we give a bound on \mathbf{Adv}_4 via an information-theoretic argument. We first consider the information about \mathbf{u}_0 (and subsequently $\{\mathbf{u}_i\}_{i \in [1, n] \setminus \text{CS}}$) leaked from \mathbf{vk} (and subsequently $\{\mathbf{vk}_i\}_{i \in [1, n]}$) and partial signing queries:

- $\mathbf{vk} := [\mathbf{KA}]_2 = [\tilde{\mathbf{K}}\mathbf{A}]_2$ and $\mathbf{vk}_i := [\mathbf{K}_i\mathbf{A}]_2 = [\tilde{\mathbf{K}}_i\mathbf{A}]_2$ for all $i \in [1, n]$.
- The output of the j^{th} partial signature query on $(i, [\mathbf{m}]_1)$ for $[\mathbf{m}]_1 \neq [\mathbf{m}^*]_1$ completely hides $\{\mathbf{u}_i\}_{i \in [1, n] \setminus \text{CS}}$ (and subsequently \mathbf{u}_0 as the adversary has only $|\text{CS}|$ many \mathbf{u}_i with $|\text{CS}| < t$), since

$$(1 \quad \mathbf{m}^\top) \mathbf{K}_i + \mu_j \mathbf{a}^\perp = (1 \quad \mathbf{m}^\top) \tilde{\mathbf{K}}_i + (1 \quad \mathbf{m}^\top) \mathbf{u}_i \mathbf{a}^\perp + \mu_j \mathbf{a}^\perp.$$

distributed identically to $(1 \quad \mathbf{m}^\top) \tilde{\mathbf{K}}_i + \mu_j \mathbf{a}^\perp$. This is because $\mu_j \mathbf{a}^\perp$ already hides $(1 \quad \mathbf{m}^\top) \mathbf{u}_i \mathbf{a}^\perp$ for uniformly random $\mu_j \leftarrow \mathbb{Z}_p$.

The only way to successfully convince the verification to accept a signature Σ^* on \mathbf{m}^* , the adversary must correctly compute $(1 \quad \mathbf{m}^{*\top}) (\mathbf{K} + \mathbf{u}_0 \mathbf{a}^\perp)$ and thus $(1 \quad \mathbf{m}^{*\top}) \mathbf{u}_0$. Observe that, $\{\mathbf{u}_i\}_{i \in [1, n] \setminus \text{CS}}$ (and thereby \mathbf{u}_0) are completely hidden to the adversary, $(1 \quad \mathbf{m}^{*\top}) \mathbf{u}_0$ is uniformly random from \mathbb{Z}_p from the adversary's viewpoint. Therefore, $\mathbf{Adv}_4 = 1/p$.

□

Theorem 6. Under the \mathcal{D}_k -MDDH Assumption in \mathbb{G}_1 and \mathcal{D}_k -KerMDH Assumption in \mathbb{G}_2 , our Threshold Structure-Preserving Signature construction achieves TS-UF-1 security against an efficient adversary making at most q partial signature queries.

$\mathcal{O}^{\text{PSign}^*}(i, [\mathbf{m}]_1):$ 1: assert $([\mathbf{m}]_1 \in \mathcal{M} \wedge i \in \text{HS})$ 2: $\mathbf{r}_i \leftarrow \$_{\mathbb{Z}_p^k}, \tau := \mathcal{H}([\mathbf{m}]_1), \mu \leftarrow \$_{\mathbb{Z}_p}$ If $[\mathbf{m}]_1 = [\mathbf{m}^*]_1$, set $\mu := 0$ 3: $\sigma_1 := [(1 \quad \mathbf{m}^\top) \mathbf{K}_i + \mu \mathbf{a}^\perp + \mathbf{r}_i^\top \mathbf{B}^\top (\mathbf{U} + \tau \cdot \mathbf{V})]_1,$ $\sigma_2 := [\mathbf{r}_i^\top \mathbf{B}^\top]_1,$ $\sigma_3 := [\tau \mathbf{r}_i^\top \mathbf{B}^\top]_1,$ $\sigma_4 := [\tau]_2$ 4: $\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ 5: if $\Sigma_i \neq \perp$: 6: $S_1([\mathbf{m}]_1) := S_1([\mathbf{m}]_1) \cup \{i\}$ 7: return Σ_i
--

Figure 7.9: Game'_3 in the proof of Theorem 6.

Proof Sketch. The difference between TS-UF-0 and TS-UF-1 lies in the fact that, in the latter model, an adversary can request $\mathcal{O}^{\text{PSign}}(\cdot)$ queries on $[\mathbf{m}^*]_1$ for which it aims to forge a signature. The natural restriction in Figure 7.1 is expressed as $|S_1([\mathbf{m}^*]_1)| < t - |\text{CS}|$, where t is the threshold value and the corrupted parties CS are fixed at the beginning of the game. As this security model allows partial signature oracle queries on $[\mathbf{m}]_1$, we next explore the changes we need to make on the proof of Theorem 5.

Game_0 , Game_1 and Game_2 stay the same. To handle TS-UF-1 adversaries, we introduce an additional game Game'_2 to handle partial signature queries on the forged message. In Game'_2 , the challenger makes a list of all the partial signature queries and guesses the message on which forgery will be done. However, the guess will be made on the list of partial signature queries. More precisely, let \mathcal{A} make partial signature queries on $[\mathbf{m}_1]_1, \dots, [\mathbf{m}_Q]_1$ s.t. $Q \leq q$, the challenger of Game'_2 rightly guesses the forged message with $1/Q$ probability which introduces a degradation in the advantage. This small yet powerful modification allows the challenger in Game_3 to add a uniformly random quantity μ to partial signature oracle queries on $[\mathbf{m}]_1 \neq [\mathbf{m}^*]_1$. This concept is formulated by adding an additional line between lines number 2 and 3 in Figure 7.6. In particular, the new Game'_3 (See Figure 7.9) would set $\mu = 0$ if $[\mathbf{m}]_1 = [\mathbf{m}^*]_1$. Next, we give an intuitive explanation of the indistinguishability of Game'_2 and Game'_3 which basically is a modification of the proof of Lemma 2.

The novelty of this research lies in the need to simulate partial signature queries on the forged message $[\mathbf{m}^*]_1$, a challenge not addressed in previous works like [KW15; KPW15] upon which this study is based. It's important to mention that an extra oracle, termed $\mathcal{O}^{**}(\cdot)$, is sufficient for our objectives. On

any partial signature query on the forged message $[\mathbf{m}^*]_1$, the reduction calls $\mathcal{O}^{**}([\tau^*]_1)$ for $\tau^* \leftarrow \mathcal{H}([\mathbf{m}^*]_1)$. Next we see that a single query to $\mathcal{O}^{**}([\tau^*]_1)$ is sufficient to handle multiple partial signature queries on $[\mathbf{m}^*]_1$. In particular, given a partial signature oracle query on $(i, [\mathbf{m}^*]_1)$, the reduction uses $\mathcal{O}^{**}(\cdot)$ of the so-called core-lemma (in Lemma 1) to get $\mathbf{X} = [\mathbf{B}^\top(\mathbf{U} + \tau^*\mathbf{V})]_1$, where $\tau^* = \mathcal{H}([\mathbf{m}^*]_1)$. The reduction then replies with $([(1 \ \mathbf{m}^{*\top})]_1 \mathbf{K}_i + \mathbf{r}^\top \cdot \mathbf{X}, [\mathbf{r}^\top \mathbf{B}^\top]_1, [\tau^* \mathbf{r}^\top \mathbf{B}^\top]_1, [\tau^*]_2)$ as a partial signature response to \mathcal{A} . Thus, a single call to $\mathcal{O}^{**}(\cdot)$ suffices to handle all partial signature queries on $[\mathbf{m}^*]_1$.

We define Game_4 as being identical to the proof of Theorem 5. In fact, the argument for the indistinguishability of Game_3 and Game_4 from the proof of Theorem 5 applies here as well. The argument that Adv_4 is negligible however requires a small modification. Similar to the proof of Theorem 5, we can show that all verification keys vk and $\{\text{vk}_i\}_{i \in [1, n]}$ stay the same. Furthermore, all partial signature queries on $[\mathbf{m}]_1 \neq [\mathbf{m}^*]_1$ do not leak any information about $\{\mathbf{u}_i\}_{i \in [1, n] \setminus \text{CS}}$. Since, partial signature oracle queries are allowed on $[\mathbf{m}^*]_1$, observe that at most $\{\mathbf{u}_i\}_{i \in S_1([\mathbf{m}^*]_1)}$ are leaked to the adversary. To summarise, an adversary in TS-UF-1 gets at most $\{\mathbf{u}_i\}_{i \in S_1([\mathbf{m}^*]_1) \cup \text{CS}}$ even when it is unbounded. Due to the natural restriction, $|S_1([\mathbf{m}^*]_1)| + |\text{CS}| < t$ ensures that \mathbf{u}_0 stays completely hidden to the adversary. Thus, $(1 \ \mathbf{m}^{*\top}) \mathbf{u}_0$ is uniformly random from \mathbb{Z}_p from the adversary's viewpoint. Therefore, $\text{Adv}_4 \leq 1/p$. \square

Theorem 7. Under the \mathcal{D}_k -MDDH Assumption in \mathbb{G}_1 and \mathcal{D}_k -KerMDH Assumption in \mathbb{G}_2 , the proposed Threshold Structure-Preserving Signature construction in Figure 7.3 achieves adp-TS-UF-1 security against an efficient adversary making at most q partial signature queries.

Proof. The difference between TS-UF-1 and adp-TS-UF-1 is that an adversary of the later model has access to $\mathcal{O}^{\text{Corrupt}}(\cdot)$ oracle and can corrupt the honest signers, adaptively. As per Figure 7.1, an adp-TS-UF-1 adversary proposes a corrupted set CS at the start of the game which it updates incrementally as the game progresses. At the time of forgery, the natural restriction in Figure 7.1 formulates as $|S_1([\mathbf{m}^*]_1)| < t - |\text{CS}|$, where t is the threshold value and CS contains the list of corrupted signers at the forgery phase. Given that this security model permits an adversary to obtain the secret keys of users it may have queried using the $\mathcal{O}^{\text{PSign}}(\cdot)$ oracle in the past, our next step involves investigating the main modifications required for the proof in Theorem 6.

Game_0 , Game_1 , Game_2 , and Game'_2 stay the same. In the proof of Theorem 6, we also have showed that Game'_2 and Game'_3 to be indistinguishable due to the so-called core lemma, Lemma 1. We reuse the reduction in Figure 7.7 towards this purpose. The reduction in Figure 7.7 samples $\mathbf{K} \leftarrow_{\$} \mathbb{Z}_p^{(\ell+1) \times (k+1)}$ and generates $(\mathbf{K}_1, \dots, \mathbf{K}_n) \leftarrow \text{Share}(\mathbf{K}, \mathbb{Z}_p^{(\ell+1) \times (k+1)}, n, t)$. Recall that, the

adp-TS-UF-1 adversary \mathcal{A} of Lemma 2 corrupts a party $i \in [1, n]$ adaptively. Since the reduction of Lemma 2 already knows \mathbf{K}_i in plain, it can handle the $\mathcal{O}^{\text{Corrupt}}(\cdot)$ oracle queries quite naturally.

The indistinguishability of Game_3 and Game_4 are argued exactly the same as in Theorem 6. We now focus on Adv_4 . In Game_4 , the adversary gets to update CS adaptively. Intuitively, all \mathbf{K}_i are independently sampled. Giving out a few of them to the adversary does not change the adversary's view. In the proof of Theorem 6, we already have managed to address partial signature queries on forged message. Except a few details, this ensures our proof will work out. We next give a formal argument.

We prove this theorem through a series of games and we use Adv_i to denote the advantage of the adversary \mathcal{A} in winning the Game i . The games are described below.

Game 0. This is the adp-TS-UF-1 security game described in Definition 54.

As shown in Figure 7.10, an adversary \mathcal{A} after receiving the set of public parameters, pp , returns (n, t, CS) , where n , t and CS represents the total number of signers, the threshold, and the set of corrupted signers, respectively. The adversary can query the partial signing oracle $\mathcal{O}^{\text{PSign}}(\cdot)$ to receive partial signatures. Let \mathcal{Q} represent the number of distinct messages where partial signing queries are made. In the end, the adversary outputs a message $[\mathbf{m}^*]_1$ and a forged signature Σ^* .

Game 1. We modify the verification procedure to the one described in Figure 7.11. Consider any forged message/signature pair $([\mathbf{m}^*]_1, \Sigma^* = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4))$ where $e(\hat{\sigma}_2, \hat{\sigma}_4) = e(\hat{\sigma}_3, [1]_2)$, $|\text{CS}| < t$ and $S_1([\mathbf{m}^*]_1) = \emptyset$. Note that if the pair $([\mathbf{m}^*]_1, \Sigma^*)$ meets the $\text{Verify}^*(\cdot)$ conditions, outlined in Figure 7.11, it also satisfies $\text{Verify}(\cdot)$ procedure, described in Figure 7.10. This is primarily due to the fact that:

$$\begin{aligned} e(\hat{\sigma}_1, [\mathbf{A}]_2) &= e([(1 \quad \mathbf{m}^{*\top})]_1, [\mathbf{KA}]_2) \cdot e(\hat{\sigma}_2, [\mathbf{UA}]_2) \cdot e(\hat{\sigma}_3, [\mathbf{VA}]_2) \\ &\iff e(\hat{\sigma}_1, [1]_2) = e([(1 \quad \mathbf{m}^{*\top})]_1, [\mathbf{K}]_2) \cdot e(\hat{\sigma}_2, [\mathbf{U}]_2) \cdot e(\hat{\sigma}_3, [\mathbf{V}]_2) \\ &\iff e(\hat{\sigma}_1, [1]_2) = e([(1 \quad \mathbf{m}^{*\top}) \mathbf{K}]_1, [1]_2) \cdot e(\hat{\sigma}_2, [\mathbf{U} + \tau^* \mathbf{V}]_2) \cdot \end{aligned}$$

Assume there exists a message/signature pair $([\mathbf{m}^*]_1, \Sigma^* = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4))$ that satisfies $\text{Verify}(\cdot)$ and not $\text{Verify}^*(\cdot)$, then we can compute a non-zero vector \mathbf{c} in the kernel of \mathbf{A} as follows:

$$\mathbf{c} := \hat{\sigma}_1 - ([(1 \quad \mathbf{m}^{*\top}) \mathbf{K}]_1 + \hat{\sigma}_2 \mathbf{U} + \hat{\sigma}_3 \mathbf{V}) \in \mathbb{G}_1^{1 \times (k+1)}.$$

$G_0(\kappa)$:

- 1: $\mathcal{BG} \leftarrow \text{ABSGen}(1^\kappa)$,
- 2: $\mathbf{A}, \mathbf{B} \leftarrow \$ \mathcal{D}_k$,
- 3: $\mathbf{U}, \mathbf{V} \leftarrow \$ \mathbb{Z}_p^{(k+1) \times (k+1)}$.
- 4: $\text{pp} := ([\mathbf{A}]_2, [\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1)$.
- 5: $(n, t, \text{CS}, \text{st}_0) \leftarrow \mathcal{A}(\text{pp})$.
- 6: Assert $\text{CS} \subset [1, n]$.
- 7: Sample $\mathbf{K} \leftarrow \$ \mathbb{Z}_p^{(\ell+1) \times (k+1)}$.
- 8: $(\mathbf{K}_1, \dots, \mathbf{K}_n) \leftarrow \text{Share}(\mathbf{K}, \mathbb{Z}_p^{(\ell+1) \times (k+1)}, n, t)$.
- 9: $\text{vk} := [\mathbf{KA}]_2$.
- 10: $\text{sk}_i := \mathbf{K}_i, \text{vk}_i := [\mathbf{K}_i \mathbf{A}]_2$ for $i \in [1, n]$.
- 11: $([\mathbf{m}^*]_1, \Sigma^*, \text{st}_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\text{PSign}(\cdot)}, \mathcal{O}^{\text{Corrupt}(\cdot)}}(\text{st}_0, \text{vk}, \{\text{sk}_i\}_{i \in \text{CS}}, \{\text{vk}_i\}_{i \in [1, n]})$.
- 12: **return** $\left(\text{Verify}(\text{pp}, \text{vk}, [\mathbf{m}^*]_1, \Sigma^*) \wedge |\text{CS}| < t \wedge S_1([\mathbf{m}^*]_1) = \emptyset \right)$

$\mathcal{O}^{\text{PSign}}(i, [\mathbf{m}]_1)$:

- 1: Assert $([\mathbf{m}]_1 \in \mathcal{M} \wedge i \in \text{HS})$.
- 2: $\mathbf{r}_i \leftarrow \$ \mathbb{Z}_p^k$.
- 3: $\tau := \mathcal{H}([\mathbf{m}]_1)$.
- 4: $\sigma_1 := \left[\begin{pmatrix} 1 & \mathbf{m}^\top \end{pmatrix} \mathbf{K}_i + \mathbf{r}_i^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V}) \right]_1$.
- 5: $\sigma_2 := [\mathbf{r}_i^\top \mathbf{B}^\top]_1$,
- 6: $\sigma_3 := [\tau \mathbf{r}_i^\top \mathbf{B}^\top]_1$,
- 7: $\sigma_4 := [\tau]_2$.
- 8: $\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$.
- 9: **if** $\Sigma_i \neq \perp$:
- 10: $S_1([\mathbf{m}]_1) := S_1([\mathbf{m}]_1) \cup \{i\}$.
- 11: **return** Σ_i

$\mathcal{O}^{\text{Corrupt}}(j)$:

- 1: $\text{CS} \leftarrow \text{CS} \cup \{j\}$
- 2: $\text{HS} \leftarrow \text{CS} \setminus \{j\}$
- 3: **return** sk_j

$\text{Verify}(\text{pp}, \text{vk}, [\mathbf{m}^*]_1, \Sigma^*)$:

- 1: Parse Σ^* as $(\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4)$.
- 2: **return** $\left(e(\hat{\sigma}_1, [\mathbf{A}]_2) = e\left(\left[\begin{pmatrix} 1 & \mathbf{m}^{*\top} \end{pmatrix}\right]_1, [\mathbf{KA}]_2\right) \cdot e(\hat{\sigma}_2, [\mathbf{UA}]_2) \cdot e(\hat{\sigma}_3, [\mathbf{VA}]_2) \wedge e(\hat{\sigma}_2, \hat{\sigma}_4) = e(\hat{\sigma}_3, [1]_2) \right)$

Figure 7.10: Game_0 .

According to \mathcal{D}_k -KerMDH assumption over \mathbb{G}_2 described in Definition 51, such a vector \mathbf{c} is hard to compute. Thus,

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \text{Adv}_{\mathcal{D}_k, \mathbb{G}_2, \mathcal{B}_0}^{\text{KerMDH}}(\kappa).$$

Game 2. On receiving a partial signature query on a message $[\mathbf{m}_i]_1$, a list is updated with the message $[\mathbf{m}_i]_1$ and the corresponding tag $\tau_i := \mathcal{H}([\mathbf{m}_i]_1)$. The challenger aborts if an adversary can generate two tuples $([\mathbf{m}_i]_1, \tau_i)$, $([\mathbf{m}_j]_1, \tau_j)$ with $[\mathbf{m}_i]_1 \neq [\mathbf{m}_j]_1$ and $\tau_i = \tau_j$. By the collision resistance property of the underlying hash function we have:

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq \text{Adv}_{\mathcal{H}}^{\text{CRHF}}(\kappa).$$

$\begin{aligned} &\text{Verify}^*(\text{pp}, \text{vk}, [\mathbf{m}^*]_1, \Sigma^*): \\ &1: \text{Parse } \Sigma^* \text{ as } (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4 = [\tau^*]_2). \\ &2: \text{return } \left(e(\hat{\sigma}_1, [1]_2) = e\left([(1 \quad \mathbf{m}^{*\top}) \mathbf{K}]_1, [1]_2\right) \cdot e(\hat{\sigma}_2, [\mathbf{U} + \tau^* \mathbf{V}]_2) \wedge \right. \\ &\quad \left. e(\hat{\sigma}_2, \hat{\sigma}_4) = e(\hat{\sigma}_3, [1]_2) \right) \end{aligned}$
--

Figure 7.11: Modifications in Game_1 .

Game 2'. In Game'_2 , the challenger randomly chooses an index $j^* \leftarrow_{\$} [1, Q]$ as its guess of the message on which the forgery will be done. This game is the same as Game 2 except that the challenger aborts the game immediately if forged message $[\mathbf{m}^*]_1 \neq [\mathbf{m}_{j^*}]_1$.

The challenger of Game'_2 rightly guesses the forged message $[\mathbf{m}^*]_1$ with $1/Q$ probability which introduces a degradation in the advantage of Game'_2 : $\mathbf{Adv}_{2'} = \frac{1}{Q} \mathbf{Adv}_2$.

Game 3'. This game is same as Game'_2 except we introduce randomness to the partial signatures by adding $\mu \mathbf{a}^\perp$ to each partial signature query on all messages $[\mathbf{m}]_1$ except $[\mathbf{m}]_1^*$ on which the forgery is done.

We show that, we can make a reduction algorithm \mathcal{B} for the so-called core-lemma (in Lemma 1) using \mathcal{A} . At the start of the game, \mathcal{B} randomly chooses an index $j^* \leftarrow_{\$} [1, Q]$ as its guess of the message on which forgery will be done. If $[\mathbf{m}^*]_1 \neq [\mathbf{m}_{j^*}]_1 = [\mathbf{m}^*]_1$, \mathcal{B} aborts. Otherwise, \mathcal{B} outputs \mathcal{A} 's output as it is. In particular, \mathcal{B} does the following:

1. \mathcal{B} receives pp from the challenger.
2. \mathcal{B} samples $\mathbf{K} \leftarrow_{\$} \mathbb{Z}_p^{(\ell+1) \times (k+1)}$.

$\mathcal{O}^{\text{PSign}^*}(i, [\mathbf{m}]_1):$

- 1: assert $([\mathbf{m}]_1 \in \mathcal{M} \wedge i \in \text{HS})$
- 2: $\mathbf{r}_i \leftarrow \mathbb{Z}_p^k, \tau := \mathcal{H}([\mathbf{m}]_1), \mu \leftarrow \mathbb{Z}_p$ If $[\mathbf{m}]_1 = [\mathbf{m}^*]_1$, set $\mu := 0$
- 3: $\sigma_1 := [(1 \quad \mathbf{m}^\top) \mathbf{K}_i + \mu \mathbf{a}^\perp + \mathbf{r}_i^\top \mathbf{B}^\top (\mathbf{U} + \tau \cdot \mathbf{V})]_1,$
 $\sigma_2 := [\mathbf{r}_i^\top \mathbf{B}^\top]_1,$
 $\sigma_3 := [\tau \mathbf{r}_i^\top \mathbf{B}^\top]_1,$
 $\sigma_4 := [\tau]_2$
- 4: $\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$
- 5: **if** $\Sigma_i \neq \perp$:
- 6: $S_1([\mathbf{m}]_1) := S_1([\mathbf{m}]_1) \cup \{i\}$
- 7: **return** Σ_i

Figure 7.12: Game'_3 in the proof of Theorem 7.

3. \mathcal{B} then secret shares \mathbf{K} into $(\mathbf{K}_1, \dots, \mathbf{K}_n) \leftarrow \text{Share}(\mathbf{K}, \mathbb{Z}_p^{(\ell+1) \times (k+1)}, n, t).$
4. On a $\mathcal{O}^{\text{Corrupt}}(\cdot)$ oracle query on $j \in [1, n]$, \mathcal{B} returns \mathbf{K}_j .
5. \mathcal{B} simulates the partial signature query on $(i, [\mathbf{m}]_1)$ as following:
 - If $[\mathbf{m}]_1 = [\mathbf{m}^*]_1$, it makes a query (i, τ^*) on $\mathcal{O}^{**}(\cdot)$ where $\tau^* \leftarrow \mathcal{H}([\mathbf{m}^*]_1)$.
 - Let \mathcal{B} receives val as the response of the above queries.
 - \mathcal{B} samples $\mathbf{r}_i \leftarrow \mathbb{Z}_p^k$ and returns $\Sigma_i := ([(1 \quad \mathbf{m}^\top) \mathbf{K}_i]_1 \cdot \mathbf{r}_i^\top \cdot val, \mathbf{r}_i^\top \cdot val, \tau \cdot \mathbf{r}_i^\top \cdot val, [\tau]_2)$ to \mathcal{A} as the partial signature.
 - If $[\mathbf{m}]_1 \neq [\mathbf{m}^*]_1$, it makes a query (i, τ) on $\mathcal{O}^b(\cdot)$, where $\tau \leftarrow \mathcal{H}([\mathbf{m}]_1)$.
 - Let \mathcal{B} receives (val_1, val_2) as the response of the above queries.
 - It returns $\Sigma_i := ([(1 \quad \mathbf{m}^\top) \mathbf{K}_i]_1 \cdot val_1, val_2, \tau \cdot val_2, [\tau]_2)$ to \mathcal{A} as the partial signature.
6. On $\text{Verify}^*(\cdot)$ on $(vk, [\mathbf{m}^*]_1, \Sigma^*)$, \mathcal{B} queries on $\mathcal{O}^*(\cdot)$ on $[\tau^*]_2$ where $\tau^* \leftarrow \mathcal{H}([\mathbf{m}^*]_1)$.
 - Let Σ^* is $(\sigma_1, \sigma_2, \sigma_3, \sigma_4 = [\tau^*]_2)$.
 - Let \mathcal{B} receives val as the response of the above query.
 - \mathcal{B} verifies the signature: $e(\sigma_1, [1]_2) = e([(1 \quad \mathbf{m}^{*\top}) \mathbf{K}]_1, [1]_2) \cdot e(\sigma_2, val) \wedge e(\sigma_2, \sigma_4) = e(\sigma_3, [1]_2)$.

Game'_2 and Game'_3 are indistinguishable due to the so-called core-lemma (in Lemma 1), then we have:

$$|\mathbf{Adv}_{2'} - \mathbf{Adv}_{3'}| \leq 2Q \mathbf{Adv}_{\mathcal{D}_k, \mathbb{G}_1, \mathcal{B}_1}^{\text{MDDH}}(\kappa) + Q/p.$$

Game 4. This game is same as Game_3' except that $\{\mathbf{K}_i\}_{i \in [n]}$ are sampled. In particular, we sample $\mathbf{K}_i = \tilde{\mathbf{K}}_i + \mathbf{u}_i \mathbf{a}^\perp$ for $i \in [1, n]$.

Shamir secret sharing (see Definition 47) ensures that $(\mathbf{K}_1, \dots, \mathbf{K}_n)$ in Game_3 and $(\tilde{\mathbf{K}}_1, \dots, \tilde{\mathbf{K}}_n)$ in Game_4 are identically distributed. W.l.o.g, \mathbf{K}_i in Game_3' and $\tilde{\mathbf{K}}_i$ in Game_4 are identically distributed. In Game_4 , on the other hand, $\tilde{\mathbf{K}}_i$ and $\mathbf{K}_i = \tilde{\mathbf{K}}_i - \mathbf{u}_i \mathbf{a}^\perp$ are identically distributed. Considering both together, \mathbf{K}_i is Game_3' and \mathbf{K}_i in Game_4 are identically distributed for all $i \in [1, n]$. Thus further ensures that \mathbf{K} in Game_3' and $\mathbf{K} + \mathbf{u}_0 \mathbf{a}^\perp$ in Game_4 are identically distributed. Therefore, this change is just a conceptual change and $\mathbf{Adv}_{3'} - \mathbf{Adv}_4 = 0$.

Finally, we argue that $\mathbf{Adv}_4 = 1/p$. Notice that, the adversary gets to update CS adaptively. To complete the argument, we have to ensure that even after getting $\mathbf{K}_i = \tilde{\mathbf{K}}_i + \mathbf{u}_i \mathbf{a}^\perp$ for $i \in [\text{CS}]$ chosen adaptively and even after having several partial signatures (possibly on the corrupted keys too), \mathbf{u}_0 is still hidden to the adversary.

- Firstly, vk and $\{\text{vk}_i\}_{i \in [1, n]}$ do not leak anything about \mathbf{u}_0 and $\{\mathbf{u}_i\}_{i \in [1, n]}$ respectively. Note that, \mathcal{A} gets $\text{sk}_i = \mathbf{K}_i = \tilde{\mathbf{K}}_i + \mathbf{u}_i \mathbf{a}^\perp$ for $i \in [\text{CS}]$ as a part of Input .
- The output of j^{th} partial signature query on $(i, [\mathbf{m}]_1)$ for $[\mathbf{m}]_1 \neq [\mathbf{m}^*]_1$ completely hides $\{\mathbf{u}_i\}_{i \in [1, n] \setminus \text{CS}}$ (and subsequently \mathbf{u}_0 as the adversary has only $|\text{CS}|$ many \mathbf{u}_i where $|\text{CS}| < t$), since

$$(1 \quad \mathbf{m}^\top) \mathbf{K}_i + \mu_j \mathbf{a}^\perp = (1 \quad \mathbf{m}^\top) \tilde{\mathbf{K}}_i + (1 \quad \mathbf{m}^\top) \mathbf{u}_i \mathbf{a}^\perp + \mu_j \mathbf{a}^\perp.$$

distributed identically to $(1 \quad \mathbf{m}^\top) \tilde{\mathbf{K}}_i + \mu_j \mathbf{a}^\perp$. This is because $\mu_j \mathbf{a}^\perp$ already hides $(1 \quad \mathbf{m}^\top) \mathbf{u}_i \mathbf{a}^\perp$ for uniformly random $\mu_j \leftarrow \mathbb{Z}_p$.

- In case of the j^{th} partial signature query on $(i, [\mathbf{m}^*]_1)$, observe that at most $\{\mathbf{u}_i\}_{i \in S_1([\mathbf{m}^*]_1)}$ are leaked to the adversary. To summarise, an adp-TS-UF-1 adversary gets at most $\{\mathbf{u}_i\}_{i \in S_1([\mathbf{m}^*]_1)}$ even when it is unbounded.
- Finally, we take a look at the corrupted set CS. We emphasize that this set was updated through out the game adaptively.

From the above discussion, it is clear that the information theoretically adversary can at most gets hold of $\{\mathbf{u}_i\}_{i \in S_1([\mathbf{m}^*]_1) \sqcup \text{CS}}$ adaptively. Note that, the only way to successfully convince the verification to accept a signature Σ^* on \mathbf{m}^* , the adversary must correctly compute $(1 \quad \mathbf{m}^{*\top})(\mathbf{K} +$

$\mathbf{u}_0 \mathbf{a}^\perp$) and thus $(1 - \mathbf{m}^{*\top}) \mathbf{u}_0$. So the question now reduces to if the adversary can compute \mathbf{u}_0 from $\{\mathbf{u}_i\}_{i \in S_1([\mathbf{m}^*]_1) \sqcup \text{CS}}$ which it got adaptively. Since Shamir secret sharing is information theoretically secure, the advantage of an adversary in case of selective corruption of users is same as the advantage of an adversary in case of adaptive corruption of users. Thus, \mathbf{u}_0 is completely hidden to the adaptive adversary, $(1 - \mathbf{m}^{*\top}) \mathbf{u}_0$ is uniformly random from \mathbb{Z}_p from its viewpoint. Therefore, $\mathbf{Adv}_4 = 1/p$.

□

7.3.5 Proof of Core Lemma

Proof of Lemma 1. We proceed through a series of games from Game_0 to Game_q . Note that, Init outputs the same in all the games. In Game_i , the first i queries to the oracle $\mathcal{O}_b(\cdot)$ are responded with $([\mu \mathbf{a}^\perp + \mathbf{r}^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1)$ and the next $q - i$ queries are responded with $([\mathbf{r}^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1)$. The intermediate games Game_i and Game_{i+1} respond differently to the $i + 1$ -th query to $\mathcal{O}_b(\cdot)$. The Game_i responds with $([\mathbf{r}^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1)$ whereas Game_{i+1} responds with $([\mu \mathbf{a}^\perp + \mathbf{r}^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1)$. We compute the advantage of the adversary in differentiating the two games below. The advantage of the adversary in Game_i is denoted by \mathbf{Adv}_i for $i = 0, \dots, q$. On querying $\mathcal{O}_b(\cdot)$, Game_i responds to $i + 1$ -th query with

$$([\mathbf{r}^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1),$$

where $\mathbf{r} \leftarrow \mathbb{Z}_p^k$.

We define a sub-game $\text{Game}_{i,1}$ where $[\mathbf{Br}]_1$ is replaced with $[\mathbf{w}]_1$, $[\mathbf{w}]_1 \leftarrow \mathbb{G}_1^{k+1}$. From the MDDH assumption, a MDDH adversary cannot distinguish between the distributions $([\mathbf{B}]_1, [\mathbf{Br}]_1)$ and $([\mathbf{B}]_1, [\mathbf{w}]_1)$. Thus,

$$([\mathbf{r}^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1) \approx_c ([\mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{w}]_1).$$

All the other values can be perfectly simulated in the reduction by choosing \mathbf{U} and \mathbf{V} from the appropriate distributions. In the next sub-game $\text{Game}_{i,2}$, we introduce the randomness $\mu \mathbf{a}^\perp$ to $[\mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1$ and proceed to use an information-theoretic argument to bound the advantage in this experiment. As shown in [KW15], for every $\mathbf{A}, \mathbf{B} \leftarrow \mathcal{D}_k$, $\tau \neq \tau^*$, the following distributions are identically distributed

$$(\text{vk}, [\mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1, \mathbf{U} + \tau^* \mathbf{V}) \text{ and } (\text{vk}, [\mu \mathbf{a}^\perp + \mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1, \mathbf{U} + \tau^* \mathbf{V}).$$

with probability $1 - 1/p$ over \mathbf{w} . The values $[\mathbf{B}^\top \mathbf{U}]_1$ and $[\mathbf{B}^\top \mathbf{V}]_1$ are part of the public values $\mathbf{vk} := (\mathbf{A}, \mathbf{U}\mathbf{A}, \mathbf{V}\mathbf{A}, [\mathbf{B}]_1, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1)$ and anyone can compute $[\mathbf{B}^\top (\mathbf{U} + \tau^* \mathbf{V})]_1$ corresponding to a τ^* . Thus, for $\tau \neq \tau^*$, we have the two following identical distributions:

$$\begin{aligned} &(\mathbf{vk}, [\mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{U} + \tau^* \mathbf{V}]_2, [\mathbf{B}^\top (\mathbf{U} + \tau^* \mathbf{V})]_1) \text{ and} \\ &(\mathbf{vk}, [\mu \mathbf{a}^\perp + \mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{U} + \tau^* \mathbf{V}]_2, [\mathbf{B}^\top (\mathbf{U} + \tau^* \mathbf{V})]_1). \end{aligned} \quad (7.1)$$

From Equation (7.1), the subgames $\text{Game}_{i,1}$ and $\text{Game}_{i,2}$ are statistically close. We use the MDDH assumption again in the next sub-game $\text{Game}_{i,3}$ and replace $[\mathbf{w}]_1$ with $[\mathbf{Br}]_1$. The resulting distribution is

$$(\mathbf{vk}, [\mu \mathbf{a}^\perp + \mathbf{r}^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{U} + \tau^* \mathbf{V}]_2, [\mathbf{B}^\top (\mathbf{U} + \tau^* \mathbf{V})]_1),$$

which is same as Game_{i+1} . Thus, from the two MDDH instances as well as the information-theoretic argument,

$$|\text{Adv}_i - \text{Adv}_{i+1}| \leq 2\text{Adv}_{\mathcal{D}_{k,\mathbb{G}_1,\mathcal{B}}}^{\text{MDDH}}(\kappa) + 1/p.$$

□

7.4 Conclusion

In this paper, we give the first construction of a non-interactive threshold structure-preserving signature (TSPS) scheme from standard assumptions. We prove our construction secure in the adp-TS-UF-1 security model where the adversary is allowed to obtain partial signatures on the forged message and additionally allow the adversary to adaptively corrupt parties. Although the signatures are constant-size (and in fact quite small), we consider improving the efficiency of TSPS under standard assumptions as an interesting future work.

Benchmarking the Setup of Updatable zk-SNARKs

Source. K. Bagheri, A. Mertens, and M. Sedaghat. “Benchmarking the Setup of Updatable zk-SNARKs”. In: *8th International Conference on Cryptology and Information Security in Latin America, LATINCRYPT*. ed. by A. Aly and M. Tibouchi. Vol. 14168. Lecture Notes in Computer Science. Springer, 2023, pp. 375–396. DOI: [10.1007/978-3-031-44469-2_19](https://doi.org/10.1007/978-3-031-44469-2_19)

Abstract. Subversion-resistant zk-SNARKs allow the provers to verify the Structured Reference String (SRS), via an SRS Verification (SV) algorithm and bypass the need for a Trusted Third Party (TTP). Pairing-based zk-SNARKs with *updatable* and *universal* SRS are an extension of subversion-resistant ones which additionally allow the verifiers to update the SRS, via an SRS Updating (SU) algorithm, and similarly bypass the need for a TTP. In this paper, we examine the setup of these zk-SNARKs by benchmarking the efficiency of the SV and SU algorithms within the Arkworks library. The benchmarking covers a range of updatable zk-SNARKs, including Sonic, Plonk, Marlin, Lunar, and Basilisk. Our analysis reveals that relying solely on the standard Algebraic Group Model (AGM) may not be sufficient in practice, and we may need a model with weaker assumptions. Specifically, we find that while Marlin is secure in the AGM, additional elements need to be added to its SRS to formally prove certain security properties in the updatable CRS model. We demonstrate that the SV algorithms become inefficient for mid-sized circuits with over 20,000 multiplication gates and 100 updates. To address this, we introduce Batched SV algorithms (BSV) that leverage standard batching techniques and offer significantly improved performance. As a tool, we propose an efficient verification approach that allows the parties to identify a malicious SRS updater

with logarithmic verification in the number of updates. In the case of Basilisk, for a circuit with 2^{20} multiplication gates, a 1000-time updated SRS can be verified in less than 30 sec, a malicious updater can be identified in less than 4 min (improvable by pre-computation), and each update takes less than 6 min.

8.1 Introduction

Let \mathbf{R} be an NP relation which defines the language \mathbf{L} of all statements, x , for which there exists a witness, w , s.t. $(x, w) \in \mathbf{R}$. A Non-Interactive Zero-Knowledge (NIZK) argument [GMR89; BFM88] for \mathbf{R} allows an untrusted prover P , knowing w , to non-interactively convince a sceptical verifier V about the truth of a statement x , without leaking extra information about the witness w . Due to a wide range of applications, there has been a growing interest in recent years to develop NIZK proof systems, particularly those allowing for *succinct* proofs and efficient verifications, so-called zk-SNARKs (zero-knowledge Succinct Non-interactive Arguments of Knowledge) [Par+13; Gro16].

A zk-SNARK is expected to satisfy Zero-Knowledge (ZK) and Knowledge Soundness (KS). ZK ensures that V learns nothing beyond the truth of statement, x , from the proof. KS ensures that no malicious P can convince honest V of a false statement, unless he knows the witness. To achieve ZK and KS at the same time, zk-SNARKs rely on a Structured Reference String (SRS), which is supposed to be sampled by a Trusted Third Party (TTP), using the SRS generation algorithm SG [BFM88]. Therefore, in the SRS model a zk-SNARK consists of three algorithms (SG, P, V) . In practice, finding a mutually TTP for executing the SG algorithm to generate the SRS can be challenging.

Mitigating the Trust on the Setup of zk-SNARKs. To relax the imposed trust on the setup of zk-SNARK, a line of research distributes the SG algorithm and constructed Multi-Party Computation (MPC) protocols to sample the SRS [Ben+15; BGM17; Koh+21]. In such protocols, both P and V need to trust only 1 out of $i > 1$ participants.

In a different research direction, in 2016, Bellare et al. [BFS16] built the first NIZK argument that can achieve ZK, even if its SRS was subverted, so-called Subversion ZK (Sub-ZK). In a Sub-ZK NIZK argument, the prover does not need to trust the SRS generator, instead, it needs to run an algorithm, so-called SRS Verification (SV), and verify the validity of SRS before using it. The SV algorithm uses some pairing equations to verify the well-formedness of SRS elements. Two subsequent works of [Abd+17; Fuc18] presented subversion-resistant zk-SNARKs that similarly come with an SV algorithm and can achieve

Sub-ZK. In a Sub-ZK SNARK, consisting of four algorithms (SG, SV, P, V), the provers can verify the validity of SRS, by one-time executing the SV algorithm, and then bypass the need for a TTP. On the other side, the verifiers either need a TTP to generate the SRS, or they need to run an MPC protocol (e.g. [Ben+15; BGM17]) to sample the SRS elements, which will relax the level of trust to 1 out of i (participants).

As an extension to the MPC approach and subversion-resistant zk-SNARKs, in 2018, Groth et al. [Gro+18] proposed a new model, so-called updatable SRS model, which allows the verifiers to also bypass the trust on a TTP. To this end, a V needs to update the SRS one time, using an SRS Updating (SU) algorithm, and also verify the validity of previous updates and the final SRS, using the SV algorithm. Roughly speaking, in a zk-SNARK with updatable SRS, which consists of five algorithms (SG, SU, SV, P, V), to bypass the trust on a third party, a P needs to run the SV algorithm, and a V needs to run both SU and SV. In this model, the SRS is universal and can be used for various circuits within a bounded size. Then, Groth et al. [Gro+18] built the first zk-SNARK with universal and updatable SRS, but comes with $O(n^2)$ SRS size, where n is the number of multiplication gates in the circuit. In practice, this results in a huge SRS size, and impractical SU and SV algorithms.

Recently, there has been an impressive progress on designing Random Oracle-based zk-SNARKs with linear-size updatable SRS, shorter proofs, and more efficient provers and verifiers. Some of the known schemes that consecutively improve the initial scheme of [Gro+18] and the subsequent works are called, Sonic [Mal+19], Plonk [GWC19], Marlin [Chi+19], Lunar [Cam+21], Basilisk [RZ21], and Counting Vampires [LSZ22]. Currently, Counting Vampires [LSZ22] has the shortest proofs, i.e., two group elements less than Basilisk, but its SRS is $17\times$ larger than the SRS of Basilisk, and this can result in a considerably slower setup phase. The SU and SV algorithms are two essential algorithms for achieving Sub-ZK and Updatable Knowledge Soundness (Upd-KS, KS in the updatable SRS model) and the employment of updatable zk-SNARKs. In order to achieve Sub-ZK and Upd-KS in the updatable SRS model, the underlying SRS *must be publicly verifiable and trapdoor extractable* [BFS16; Abd+17; Fuc18; Gro+18]. Meaning that, the consistency of SRS elements should be publicly verifiable, and one should be able to extract the SRS trapdoors from the setup phase (e.g., by relying on a knowledge assumption). The initial scheme [Gro+18], and some follow-up generic constructions [ARS20; BS21; BB22] come with SU and SV algorithms, under Bilinear Diffie-Hellman Knowledge of Exponent (BDH-KE) assumption. But their SV algorithm is identical for both P and V, which in case of verifying an i -time updated SRS, it brings $O(i)$ pairing operations as an overload for the P. In [LSZ22], authors have proposed an SV algorithm to achieve Sub-ZK in their construction. However,

their SV algorithm can only be used by P (to achieve Sub-ZK), and it does not consider the verification of an i -time updated SRS, needed by V .

Our Contributions. The main objective of the current paper is to examine the efficiency of the setup phase in updatable zk-SNARKs, and evaluate their empirical performance, particularly in large-scale applications.

To this end, we first present a pair of (SU, SV) algorithms for each of the updatable zk-SNARKs including: Sonic [Mal+19], Plonk [GWC19], Marlin [Chi+19], LunarLite [Cam+21] and Basilisk [RZ21]. Similar to the earlier works [BFS16; Abd+17; Fuc18; Gro+18], the proposed algorithms use pairing products and are tailored to each specific updatable zk-SNARK. As all the aforementioned zk-SNARKs can be instantiated in various ways, we focus on the pairing-based version of them with the shortest proof, which is commonly used for comparison in the literature. During the construction of the SU and SV algorithms, we noticed that relying only on the standard Algebraic Group Model (AGM) may not be enough in practice. In some cases, we may require a model with weaker assumptions, such as the AGM with *hashing* [Lip22]. In fact, there might be a case that a zk-SNARK with monomial SRS is proven to achieve ZK and KS in the AGM model, but their SRS needs to be modified to achieve Sub-ZK and U-KS. The reason is that, to achieve Sub-ZK and Upd-KS the SRS needs to be publicly verifiable and trapdoor extractable [Abd+17; Gro+18]. In the rest, we show that the SRS of Marlin [Chi+19] is not *trapdoor extractable* as it is, but it can be made trapdoor extractable under the BDH-KE assumption, by adding a single group element to its SRS.

In the rest, we show that using the presented SU and SV algorithms, Sonic, Plonk, LunarLite and Basilisk also can achieve trapdoor extractability, under a subverted/maliciously updated SRS. Since all of them already are proven that satisfy ZK and KS, this implies that they also satisfy Sub-ZK and Upd-KS. Similar to the earlier works [BFS16; Abd+17; Fuc18; Chi+19], our SV algorithms use pairing product equations to verify the SRS. But, differently our SV algorithms get an additional input, denoted by party , which allows us to determine whether a P or V runs the algorithm. Due to achieving Sub-ZK and Upd-KS in the updatable zk-SNARKs, P only needs to verify the final $(\vec{c}\vec{r}_i, \Pi_i)$, while V additionally needs to verify the intermediate proofs $\{\Pi_j\}_{j=0}^{i-1}$. Fig. 8.1 depicts a graphical representation of the setup phase in the pairing-based updatable zk-SNARKs, and highlights the parts that need to be verified by P or V . By running an SV algorithm, P needs to compute $O(n)$ pairings, where n is the number of multiplication gates in the circuit, and V requires to compute at least $O(n + i)$ pairings, where i is the number of updates done on the SRS. In practice, even for mid-size circuits (e.g. $n \geq 10^4$) with 100 updates,

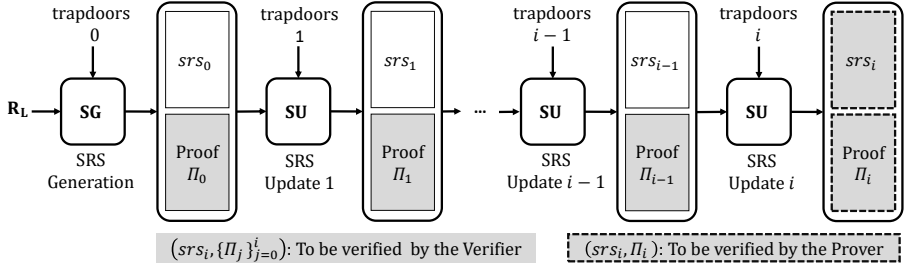


Figure 8.1: Setup in the updatable zk-SNARKs: SG, SU, and SV by P or V.

the SV algorithms can be very slow, consequently impractical.

Next, we use the standard batching techniques from [BGR98] and propose a batched version of the SV algorithms, so-called BSV, for each of the studied updatable zk-SNARKs. Using the BSV algorithms, to verify an i -time updated SRS, P needs $O(n)$ exponentiations (with short exponents) and constant number of pairings, which is independent of the number of updates. A V needs to compute $O(n + i)$ exponentiations (with short exponents) and $O(i)$ pairings. Table 8.1, compares the efficiency of our proposed SU, SV and BSV algorithms for both P and V.

The schemes built in the updatable SRS model [Gro+18] can achieve security only with abort, if the parties do not verify the updated SRS after each update. Namely, by verifying the final SRS $c\tilde{r}s_i$ and the intermediate proofs $\{\Pi_j\}_{j=0}^i$ [Gro+18] the parties will abort the final SRS $c\tilde{r}s_i$ and would not be able to identify a malicious SRS generator/updater. To identify a malicious SRS generator/updater, if the parties (or a third party) verify each updated

Table 8.1: An efficiency comparison of our proposed SU, SV and BSV algorithms. SV_P : SV run by P, BSV_V : BSV run by V, E_l : Exponentiations in \mathbb{G}_l , \bullet : Pairing, m : #total (multiplication and addition) gates, n : #multiplication gates, k : #matrix elements with non-zero values describing the circuit, i : # SRS updates

	SG/SU		SV _P	SV _V	BSV _P			BSV _V		
Scheme	E ₁	E ₂	•	•	E ₁	E ₂	•	E ₁	E ₂	•
Sonic	4n	4n	12n	12n + 10i	8n	4n	7	8n + 8i	6n + 2i	4i + 14
Marlin	k	log k	2k + 12	2k + 9i + 12	2k	log k	4	2k + 5i	2i + log k	2i + 9
Plonk	3m	2	6m	6m + 4i	6m	—	2	6m + 3i	i	i + 3
LunarLite	n	n	3n	3n + 4i + 2	2n	n	3	2n + 3i	n + i	i + 3
Basilisk	n	2	2n	2n + 4i	2n	—	2	2n + 3i	i	i + 3

SRS $\{\text{c}\vec{r}s_j\}_{j=0}^i$ (instead of only $\text{c}\vec{r}s_i$), then the verification of whole setup phase will be impractical. To deal with that, we introduce an efficient verification approach for identifying the malicious updater. For an i -time updated SRS, it allows the parties to identify the (first) malicious SRS updater with $\log i$ times running the BSV (or SV) algorithm. We discuss different optimizations that can speed up the proposed recursive SRS verification considerably, at the cost of some pre-computations and storage.

Finally, we present a comprehensive benchmark on the efficiency of our proposed SU, SV and BSV algorithms in the **Arkworks** library, which is written in Rust and currently is one of the most popular libraries programming zk-SNARKs. Full details of the benchmarking are reported in Sec. 8.5. In summary, for a particular circuit, by comparing the performance of BSV and SV algorithms, we observed that BSV can achieve up to $110 - 150\times$ better efficiency. In the case of Basilisk which has the most efficient setup phase, for a circuit with $n = 2^{20}$ multiplication gates, a 1000-time updated SRS can be verified in less than 30 sec. In the case that the verification of final SRS fails, using our proposed recursive verification approach, a malicious SRS updater can be identified in less than 4 min (or in less than 1 min by some pre-computations), and each party equipped with a multi-core CPU can update the SRS in less than 6 min. Our BSV_P algorithms are considerably faster than BSV_V ones, in case of a short SRS (e.g. $n \leq 30K$) and a large number of updates (e.g. $i \geq 200$).

Related Works. To mitigate the trust in the setup phase of zk-SNARKs, there are two key research directions. Either, by using an MPC protocol to sample the SRS [Ben+15; BGM17; Koh+21] or by directly constructing subversion-resistant [BFS16; Abd+17; Fuc18; Bag19b] and updatable zk-SNARKs [Gro+18; Mal+19; ARS20; BS21; BB22]. Our work is focused on the latter approach.

A bottleneck with the initial MPC protocols [Ben+15], is that the number of parties has to be known in advance. Bowe et al. [BGM17] presented an MPC protocol for Groth16 [Gro16] setup, which has two phases. The first phase is known as “Powers of Tau”, which can be used to sample a universal SRS for all circuits up to a given size. In the second phase, given the universal SRS generated in the previous phase, parties generate a circuit-dependent SRS. In the Powers of Tau protocol, a coordinator is used to manage messages between the participants, however the output of the protocol is verifiable. Compared with the case one uses the Powers of Tau protocol [BGM17], 1) our proposed algorithms do not need a random beacon, 2) our SV and BSV algorithms are constructed in the updatable SRS model which allows one to verify an i -time updated SRS considerably more efficient than i -time running their SRS verification algorithm. For verifying even one-time updated SRS, our proposed BSV algorithms can be

more than $100\times$ faster than their verification algorithm, 3) our SV and BSV algorithms for the provers and verifiers are different, which allows the provers to verify a large-time updated SRS more efficient than verifiers. 4) our protocols can achieve identifiable security more efficiently (using a new recursive SRS verification approach).

In [Koh+21], Kohlweiss et al. presented a more efficient version of the Powers of Tau [BGM17]. Their ceremony protocol [Koh+21] uses an RO-based proof system, and comes with a BSV algorithm. Similar to previous SG, SU, SV and BSV algorithms, our algorithms do not use a random beacon or a random oracle. Similar to the earlier works on subversion-resistant or updatable NIZK arguments [Abd+17; Fuc18; Gro+18; Bag19b; ARS20; BS21; BB22], we rely on particular knowledge assumptions. In comparison with the case that one uses the protocol proposed in [Koh+21], 1) our proposed algorithms (i.e., SG, SU, SV, and BSV) do not rely on RO, 2) we have different SV (and BSV) algorithms for the provers and verifiers, which allow the provers to verify an updated SRS more efficient than the verifiers, 3) our constructions can achieve identifiable security.

In another related research direction, some studies have defined subversion-resistant and updatable commitments [Bag20; DRZ20; Gan+22], and have proposed SV and SU algorithms for their studied (knowledge, vector, and polynomial) commitment schemes. Our proposed SV algorithm for Sonic can be considered as an extension of the one proposed in [Bag20], which checks some extra terms and also allows the verifiers to verify an i -time updated SRS. Our SV algorithm for the verifiers in Basilisk is similar to the one proposed in [Gan+22], but our SV algorithm for the provers is more efficient. We also propose a batched version of SV algorithms that make them considerably more efficient in practice.

8.2 Preliminaries

Throughout, we suppose the security parameter of the scheme and its unary representation to be denoted by κ and 1^κ , respectively. For all positive functions $\varepsilon(\lambda)$, a mapping function $\text{negl} : \mathbb{N} \rightarrow \mathbb{R}^+$ is called negligible function if there exists $\lambda_0 \in \mathbb{N}$ such that for all $\lambda > \lambda_0$ we have, $\nu(\kappa) < 1/\varepsilon(\lambda)$. We use $x \leftarrow \$ X$ to denote x sampled uniformly according to the distribution X .

Let PPT and NUPPT denote probabilistic polynomial-time and non-uniform probabilistic polynomial-time, respectively. For an algorithm \mathcal{A} , let $\text{im}(\mathcal{A})$ be the image of \mathcal{A} , i.e., the set of valid outputs of \mathcal{A} . Moreover, assume $\text{RND}(\mathcal{A})$ denotes the random tape of \mathcal{A} , and $r \leftarrow \$ \text{RND}(\mathcal{A})$ denotes sampling

of a randomizer r of sufficient length for \mathcal{A} 's needs. By $y \leftarrow \mathcal{A}(x; r)$ we mean given an input x and a randomizer r , \mathcal{A} outputs y . For algorithms \mathcal{A} and $\text{Ext}_{\mathcal{A}}$, we write $(y \parallel y') \leftarrow (\mathcal{A} \parallel \text{Ext}_{\mathcal{A}})(x; r)$ as a shorthand for “ $y \leftarrow \mathcal{A}(x; r)$, $y' \leftarrow \text{Ext}_{\mathcal{A}}(x; r)$ ”.

We use additive and the bracket notation, i.e., in group \mathbb{G}_{ζ} , $[a]_{\zeta} = a[1]_{\zeta}$, where $[1]_{\zeta}$ is the generator of \mathbb{G}_{ζ} for $\zeta \in \{1, 2, T\}$. A *bilinear group generator* $\text{BGgen}(1^{\kappa})$ returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2)$, where p (a large prime) is the order of cyclic abelian groups \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T . Finally, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficient non-degenerate bilinear pairing, s.t. $\hat{e}([a]_1, [b]_2) = [ab]_T$. Denote $[a]_1 \bullet [b]_2 = \mathbf{E}([a]_1, [b]_2)$.

8.2.1 Updatable, Universal and Subversion-Resistant zk-SNARKs

We adopt the definition of subversion-resistant and updatable zk-SNARKs from [Abd+17; Gro+18]. Let \mathcal{R} be a relation generator, such that $\mathcal{R}(1^{\kappa})$ returns a polynomial-time decidable binary relation $\mathbf{R} = \{(x, w)\}$, where x is the statement and w is the witness. We assume one can deduce κ from the description of \mathbf{R} . Let $\mathbf{L} = \{x : \exists w \mid (x, w) \in \mathbf{R}\}$ be an NP-language including all the statements which there exist corresponding witnesses in relation \mathbf{R} . A NIZK argument Ψ_{NIZK} in the updatable SRS model for \mathcal{R} consists of the following PPT algorithms:

- $(\text{crs}_0, \Pi_0) \leftarrow \text{SG}(\mathbf{R})$: Given \mathbf{R} , the SRS generator SG first deduces the upper bound N on the relation size. Next, sample the trapdoor $\vec{\text{ts}}$ and then use it to generate crs_0 along with Π_0 as a proof of its well-formedness. Finally, return (crs_0, Π_0) as the output.
- $(\text{crs}_i, \Pi_i) \leftarrow \text{SU}(\text{crs}_{i-1}, \{\Pi_j\}_{j=0}^{i-1})$: Given $(\text{crs}_{i-1}, \{\Pi_j\}_{j=0}^{i-1})$, an SRS updater SU returns the pair of (crs_i, Π_i) , where crs_i is the updated SRS and Π_i is a proof for correct updating.
- $(\perp/1) \leftarrow \text{SV}(\text{crs}_i, \{\Pi_j\}_{j=0}^i, \text{party})$: Given a potentially updated crs_i , $\{\Pi_j\}_{j=0}^i$, SV , and $\text{party} \in \{\text{P}, \text{V}\}$, return either \perp (if crs_i is incorrectly formed or updated) or 1 (if crs_i is correctly formed or updated).
- $(\pi/\perp) \leftarrow \text{P}(\mathbf{R}, \text{crs}_i, x, w)$: Given the tuple of $(\mathbf{R}, \text{crs}_i, x, w)$, such that $(x, w) \in \mathbf{R}$, P output an argument π . Otherwise, it returns \perp .
- $(0/1) \leftarrow \text{V}(\mathbf{R}, \text{crs}_i, x, \pi)$: Given $(\mathbf{R}, \text{crs}_i, x, \pi)$, V verify the proof π and return either 0 (reject) or 1 (accept).

In the standard SRS model, a zk-SNARK for \mathcal{R} has a tuple of algorithms $(\text{SG}, \text{P}, \text{V})$ (and SG does not return the Π_0), while subversion-resistant constructions [BFS16; Abd+17] additionally have an SV algorithm which is used to verify the well-formedness of the SRS elements to achieve Sub-ZK [BFS16]. But as listed above, in the *updatable* SRS model, a NIZK argument additionally has an SU algorithm that allows the parties (more precisely, the verifiers) to update the SRS and add their own private shares to the SRS generation. Note that in the latest case, the algorithm SG does not necessarily need \mathbf{R} , and it only deduces security parameter 1^κ and the upper bound N from it. We highlight that, in comparison with previous definitions [Gro+18], our SV algorithm gets an additional input $\text{party} \in \{\text{P}, \text{V}\}$. We later show that this allows us to build a more efficient SV algorithm for the prover. It is worth mentioning that in the updatable SRS model, there also exists a publicly computable deterministic algorithm Derive which given $(\mathbf{R}, \text{c}\vec{\text{r}}\text{s}_i)$ outputs a specialized SRS for relation \mathbf{R} . The output elements of Derive all are in the span of the universal SRS, but they allow to build more efficient proof generation and verification algorithms.

In the subversion-resistant and updatable SRS model, a zk-SNARK is expected to satisfy *updatable completeness*, Subversion Zero-Knowledge (Sub-ZK) and Updatable Knowledge Soundness (Upd-KS), of which the definitions are summarized below. In the definition of Sub-ZK, one requires the existence of a PPT simulator Sim consisting of algorithms $(\text{Sim}_{\text{SG}}, \text{Sim}_{\text{P}})$ that share state with each other. The idea is that it can be used to simulate the SRS and proofs without knowing the corresponding trapdoors.

The algorithm of proof simulation. $\pi \leftarrow \text{Sim}_{\text{P}}(\mathbf{R}, \text{c}\vec{\text{r}}\text{s}_i, \vec{\text{t}}\text{s}_i, \mathbf{x})$: For $\text{SV}(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i) = 1$, given the tuple $(\mathbf{R}, \text{c}\vec{\text{r}}\text{s}_i, \vec{\text{t}}\text{s}_i, \mathbf{x})$, where $\vec{\text{t}}\text{s}_i$ is the simulation trapdoor associated with the latest SRS, namely $\text{c}\vec{\text{r}}\text{s}_i$, outputs a simulated argument π .

Definition 55 (Perfect Updatable Completeness). A non-interactive argument Ψ_{NIZK} is *perfectly updatable complete* for \mathcal{R} , if for all $(\mathbf{R}) \in \text{im}(\mathcal{R}(1^\kappa))$, and $(\mathbf{x}, \mathbf{w}) \in \mathbf{R}$, the following probability is 1 on security parameter κ ,

$$\Pr \left[\begin{array}{l} (\mathbf{R}) \leftarrow \mathcal{R}(1^\kappa), (\text{c}\vec{\text{r}}\text{s}_0, \Pi_0) \leftarrow \text{SG}(\mathbf{R}), (\{\text{c}\vec{\text{r}}\text{s}_j, \Pi_j\}_{j=1}^i) \leftarrow \mathcal{A}(\mathbf{R}, \text{c}\vec{\text{r}}\text{s}_0), \\ \{\text{SV}(\text{c}\vec{\text{r}}\text{s}_j, \Pi_j, \text{party}) = 1\}_{j=0}^i : (\mathbf{x}, \pi) \leftarrow \text{P}(\mathbf{R}, \text{c}\vec{\text{r}}\text{s}_i, \mathbf{x}, \mathbf{w}) \wedge \text{V}(\mathbf{R}, \text{c}\vec{\text{r}}\text{s}_i, \mathbf{x}, \pi) = 1 \end{array} \right],$$

where Π_i is a proof for the correctness of the initial SRS generation or SRS updating. Note that in the above definition and all the following one, i is the index of final update, and without loss of generality, \mathcal{A} can also first generate $\{\text{c}\vec{\text{r}}\text{s}_j\}_{j=0}^{i-1}$ and then an honest updater updates $\text{c}\vec{\text{r}}\text{s}_{i-1}$ to $\text{c}\vec{\text{r}}\text{s}_i$.

Definition 56 (Sub-ZK). A NI argument Ψ is *computationally Sub-ZK* for \mathcal{R} , if for any PPT subverter Sub there exists a PPT extractor Ext_{Sub} , s.t. for all κ , all $\mathbf{R} \in \text{im}(\mathcal{R}(1^\kappa))$, and for any PPT \mathcal{A} , one has $\varepsilon_0 \approx_\kappa \varepsilon_1$, where

$$\varepsilon_b = \Pr \left[\begin{array}{l} r \leftarrow \$ \text{RND}(\text{Sub}), ((\vec{\text{crs}}, \Pi_{\vec{\text{crs}}}, \xi_{\text{Sub}}) \parallel \vec{\text{ts}}) \leftarrow (\text{Sub} \parallel \text{Ext}_{\text{Sub}})(\mathbf{R}; r) : \\ \text{SV}(\vec{\text{crs}}, \Pi_{\vec{\text{crs}}}, \text{party}) = 1 \wedge \mathcal{A}^{\text{O}_b(\cdot, \cdot)}(\mathbf{R}, \vec{\text{crs}}, \vec{\text{ts}}, \xi_{\text{Sub}}) = 1 \end{array} \right].$$

Here, ξ_{Sub} is auxiliary information generated by subverter Sub , the **party** is set to be the prover, and the oracle $\text{O}_0(x, w)$ returns \perp (reject) if $(x, w) \notin \mathbf{R}$, and otherwise it returns $\text{P}(\mathbf{R}, \vec{\text{crs}}, x, w)$. Similarly, $\text{O}_1(x, w)$ returns \perp (reject) if $(x, w) \notin \mathbf{R}$, and otherwise it returns $\text{Sim}(\mathbf{R}, \vec{\text{crs}}, \vec{\text{ts}}, x)$. Ψ is *perfectly Sub-ZK* for \mathcal{R} if one requires that $\varepsilon_0 = \varepsilon_1$.

Definition 57 (Updatable nBB Knowledge Soundness). A non-interactive argument Ψ_{NIZK} is *updatable non-black-box knowledge sound* for \mathcal{R} , if for every PPT adversary \mathcal{A} and any subverter Sub , there exists a PPT extractor $\text{Ext}_{\mathcal{A}}$, and the following probability is $\nu(\kappa)$,

$$\Pr \left[\begin{array}{l} \mathbf{R} \leftarrow \mathcal{R}(1^\kappa), (\vec{\text{crs}}_0, \Pi_0) \leftarrow \text{SG}(\mathbf{R}), r_s \leftarrow \$ \text{RND}(\text{Sub}), \\ \{(\vec{\text{crs}}_j, \Pi_j)_{j=1}^i, \xi_{\text{Sub}}\} \leftarrow \text{Sub}(\vec{\text{crs}}_0, \Pi_0, r_s), \{\text{SV}(\vec{\text{crs}}_j, \Pi_j, \text{party}) = 1\}_{j=0}^i, \\ r_{\mathcal{A}} \leftarrow \$ \text{RND}(\mathcal{A}), ((x, \pi) \parallel w) \leftarrow (\mathcal{A} \parallel \text{Ext}_{\mathcal{A}})(\mathbf{R}, \vec{\text{crs}}_i, \xi_{\text{Sub}}; r_{\mathcal{A}}) : \\ (x, w) \notin \mathbf{R} \wedge \text{V}(\mathbf{R}, \vec{\text{crs}}_i, x, \pi) = 1 \end{array} \right],$$

Here $\text{RND}(\mathcal{A}) = \text{RND}(\text{Sub})$, $\Pi_{\vec{\text{crs}}}$ is a proof for correctness of SRS generation or updating process, and the **party** is set to be the verifier.

8.2.2 Assumptions

Definition 58 (Bilinear Diffie-Hellman Knowledge of Exponent (BDH-KE) Assumption [Abd+17]). We say BGgen is *BDH-KE secure* for relation set \mathcal{R} if for any κ , $\mathbf{R} \in \text{im}(\mathcal{R}(1^\kappa))$, and PPT adversary \mathcal{A} , there exists a PPT extractor $\text{Ext}_{\mathcal{A}}$, such that, the following probability is $\nu(\kappa)$,

$$\Pr \left[\begin{array}{l} (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2) \leftarrow \text{BGgen}(1^\kappa), r \leftarrow \$ \text{RND}(\mathcal{A}), \\ ([\alpha_1]_1, [\alpha_2]_2 \parallel a) \leftarrow (\mathcal{A} \parallel \text{Ext}_{\mathcal{A}})(\mathbf{R}, r) : [\alpha_1]_1 \bullet [1]_2 = [1]_1 \bullet [\alpha_2]_2 \wedge a \neq \alpha_1 \end{array} \right].$$

The BDH-KE assumption [Abd+17] is an asymmetric-pairing version of the original knowledge assumption [Dam92]. We refer to full version [BS20] for some preliminaries on polynomial commitments that are used in the rest of paper.

8.3 SU and SV Algorithms for Updatable zk-SNARKs

In this section, we present a pair of SRS updating and SRS verification algorithms for each of the studied updatable zk-SNARKs, Sonic [Mal+19], Plonk [GWC19], Marlin [Chi+19], LunarLite [Cam+21] and Basilisk [RZ21].

The General Strategy. Our proposed SV and SU algorithms use pairing checks for SRS verification and the SRS elements are updated in a round-robin multiplicative manner. In comparison with the earlier works, we have a subtle change in the construction of SV algorithms, which allows the provers to verify an updated SRS more efficiently, especially in case of small circuits with a large number of updates. Recall that, a pairing-based zk-SNARK satisfies Sub-ZK if it can achieve ZK, even if its SRS is subverted (i.e., is generated by the adversary). In Sub-ZK zk-SNARKs [BFS16; Abd+17; Fuc18; Bag19b], this is formalized and achieved by building an SV algorithm that verifies the *well-formedness* and *trapdoor extractability* of the SRS. The former guarantees that the whole SRS elements are consistent with each other, and the latter ensures that the (simulation) trapdoors of SRS can be extracted from an SRS subverter. Given the simulation trapdoors of SRS, the proofs are simulated as in the standard ZK. On the other side, a universal zk-SNARK is updatable [Gro+18] if its SRS can be sequentially updated by the parties, such that Upd-KS holds if at least one of the updates with SU or the initial SRS generation with SG is done honestly. To ensure that SRS generation/updating is done correctly, parties should return a knowledge assumption-based proof Π when running SG or SU algorithms. This proof is also known as the well-formedness proof of the SRS. In the presented SV algorithms, we use the fact that to achieve Sub-ZK, a P only needs to verify the final SRS. Without loss of generality, one can assume that the initial SRS generation and all the follow-up updates are done with a single adversary who can control all the updaters who run SU and the initial party who runs SG. However, to achieve Upd-KS without a TTP, a V needs to one-time run the SU and update the SRS, and also verify the final SRS and the correctness of all intermediate proofs, generated by all the updaters (See Fig. 8.1).

Next, in each subsection, we present an overview of a particular updatable zk-SNARK, and then describe its SRS Generation (SG) algorithm. Different from the original papers, in the description of SG algorithms, we also determine what constitutes a well-formedness proof that can be used to extract individual shares from the SRS generator/updaters, and more importantly, can be used to verify the final SRS. The well-formedness proof is shown with Π which

consists of two sets of elements $(\Pi^{\text{Agg}}, \Pi^{\text{Ind}})$, where Π^{Agg} can be interpreted as the aggregated elements necessary for verifying the well-formedness of final SRS, and Π^{Ind} can be interpreted as an individual proof for the correctness of updating using the secret shares, e.g. \bar{x} . The latter, also enables extracting the individual shares from a malicious SRS generator/updater in the proof of Upd-KS. Finally, we present SU and SV algorithms.

8.3.1 SU and SV Algorithms for Sonic

Sonic and its SG algorithm. The first proposed updatable zk-SNARK, presented by Groth et al. [Gro+18], came with explicit SU and SV algorithms, but its SRS size scales quadratically in the number of multiplication gates in the circuit that encodes the relation, which made the algorithms very slow. In a follow-up work, Maller et al. [Mal+19] proposed Sonic as the first updatable zk-SNARK with linear size SRS. The authors mostly focused on achieving a linear size SRS and more efficient P and V algorithms, and omitted the descriptions of SU and SV algorithms (and even SG which should determine the well-formedness proof) and mentioned that they can built as in [Gro+18]. For further details, we refer to the main paper [Mal+19]. We describe the SG algorithm of Sonic in Figure 8.2.

SU and SV Algorithms and Their Efficiency. Fig. 8.3 describes the SU and SV algorithms for Sonic. As briefly mentioned before, the SRS update is done

SRS Generation, $(c\vec{r}s_0, \Pi_0) \leftarrow \text{SG}(\mathbf{R})$: Given \mathbf{R} , first deduce the security parameter 1^κ and k , then obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2) \leftarrow \text{BGgen}(1^\kappa)$; after that act as follows:

- Sample $\bar{x}_0, \bar{a}_0 \leftarrow \mathbb{Z}_p^*$, and set $x_0 := \bar{x}_0$ and $a_0 := \bar{a}_0$ which are the simulation trapdoor associated with $c\vec{r}s_0$;
- For $k = -n, \dots, n$: compute $[x_0^k]_1, [x_0^k]_2, [a_0 x_0^k]_2$;
- For $k = -n, \dots, -1, 1, \dots, n$: compute $[a_0 x_0^k]_1$; Compute $[a_0]_T$;
- Set $c\vec{r}s_0 := (([x_0^k]_1, [x_0^k]_2, [a_0 x_0^k]_2)_{k=-n}^n, ([a_0 x_0^k]_1)_{k=-n, k \neq 0}^n, [a_0]_T)$, and the well-formedness proof $\Pi_0 := (\Pi_0^{\text{Agg}}, \Pi_0^{\text{Ind}}) := (([x_0]_1, [a_0 x_0]_1, [a_0]_2), ([x_0]_1, [x_0]_2, [a_0 x_0]_1, [a_0 x_0]_2, [a_0]_2))$;
- Return $(c\vec{r}s_0, \Pi_0)$;

Figure 8.2: SG algorithm for SONIC.

in a multiplicative manner, such that the updater multiplies a proper power of its secret shares \bar{x}_i and \bar{a}_i to the SRS elements. Similar to the **SG** algorithm, we also determine the elements of the well-formedness proof separately. Note that $[a]_T$ is omitted from updating, as due to the fact that $[a]_T := [1]_1 \bullet [a]_2$, it can finally be computed from the other SRS elements. The pairing checks inside **SV** chase two main goals. First, they check if all the individual proofs generated by the SRS generator and by all the follow-up SRS updaters are correct. If so, then it uses the elements of Π_i and verifies the final SRS, $\text{cr}\bar{s}_i$.

SRS Update, $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i) \leftarrow \text{SU}(\text{c}\vec{\text{r}}\text{s}_{i-1}, \{\Pi_{j-1}\}_{j=0}^{i-1})$: Given $(\text{c}\vec{\text{r}}\text{s}_{i-1}, \{\Pi_{j-1}\}_{j=0}^{i-1})$,

- Parse $\text{c}\vec{\text{r}}\text{s}_{i-1} := (([x_{i-1}^k]_1, [x_{i-1}^k]_2, [a_{i-1}x_{i-1}^k]_2)_{k=-n}^n, ([a_{i-1}x_{i-1}^k]_1)_{k=-n, k \neq 0}^n)$;
- Sample $\bar{x}_i, \bar{a}_i \leftarrow \mathbb{Z}_p^*$, as the secret shares to be used for updating $\text{c}\vec{\text{r}}\text{s}_{i-1}$.
- For $k = -n, \dots, n$: set $[x_i^k]_1 := \bar{x}_i \cdot [x_{i-1}^k]_1$; set $[x_i^k]_2 := \bar{x}_i \cdot [x_{i-1}^k]_2$; set $[a_i x_i^k]_2 := \bar{a}_i \bar{x}_i \cdot [a_{i-1} x_{i-1}^k]_2$;
- For $k = -n, \dots, -1, 1, \dots, n$: set $[a_i x_i^k]_1 := \bar{a}_i \bar{x}_i \cdot [a_{i-1} x_{i-1}^k]_1$;
- Set $\text{c}\vec{\text{r}}\text{s}_i := (([x_i^k]_1, [x_i^k]_2, [a_i x_i^k]_2)_{k=-n}^n, ([a_i x_i^k]_1)_{k=-n, k \neq 0}^n, [a_i]_T)$, and the well-formedness proof $\Pi_i := (\Pi_i^{\text{Agg}}, \Pi_i^{\text{Ind}}) := (([x_i]_1, [a_i x_i]_1, [a_i]_2), ([\bar{x}_i]_1, [\bar{x}_i]_2, [\bar{a}_i \bar{x}_i]_1, [\bar{a}_i \bar{x}_i]_2, [\bar{a}_i]_2))$;
- Return $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i)$;

SRS Verify, $(\perp/1) \leftarrow \text{SV}(\text{c}\vec{\text{r}}\text{s}_i, (\Pi_j)_{j=0}^i, \text{party})$: To verify (an i -time updated) $\text{c}\vec{\text{r}}\text{s}_i := (([x_i^k]_1, [x_i^k]_2, [a_i x_i^k]_2)_{k=-n}^n, ([a_i x_i^k]_1)_{k=-n, k \neq 0}^n, [a_i]_T)$, and $\Pi_j := (\Pi_j^{\text{Agg}}, \Pi_j^{\text{Ind}}) := (([x_j]_1, [a_j x_j]_1, [a_j]_2), ([\bar{x}_j]_1, [\bar{x}_j]_2, [\bar{a}_j \bar{x}_j]_1, [\bar{a}_j \bar{x}_j]_2, [\bar{a}_j]_2))$ for $j = 0, 1, \dots, i$:

If party = P:

1. For $k = -n, \dots, n$: check if $[x_i^k]_1 \bullet [1]_2 = [1]_1 \bullet [x_i^k]_2$;
2. For $k = -n+1, \dots, n$: check if $[x_i^k]_1 \bullet [1]_2 = [x_i^{k-1}]_1 \bullet [x_i]_2$;
3. For $k = -n, \dots, -1, 1, \dots, n$: check if $[a_i x_i^k]_1 \bullet [1]_2 = [1]_1 \bullet [a_i x_i^k]_2 = [x_i^k]_1 \bullet [a_i]_2$;

If party = V:

- If $i = 0$: $\text{c}\vec{\text{r}}\text{s}_0$ is sampled by verifier, and it does not need to be verified.
- If $i \geq 1$:
 1. Check that $[x_0]_1 = [\bar{x}_0]_1$, $[a_0 x_0]_1 = [\bar{a}_0 \bar{x}_0]_1$, and $[a_0]_2 = [\bar{a}_0]_2$;
 2. For $j = 0, 1, \dots, i$: check if $[\bar{x}_j]_1 \bullet [1]_2 = [1]_1 \bullet [\bar{x}_j]_2$
 3. For $j = 0, 1, \dots, i$: check if $[\bar{a}_j \bar{x}_j]_1 \bullet [1]_2 = [1]_1 \bullet [\bar{a}_j \bar{x}_j]_2 = [\bar{x}_j]_1 \bullet [\bar{a}_j]_2$;
 4. For $j = 1, 2, \dots, i$: check if $[x_j]_1 \bullet [1]_2 = [x_{j-1}]_1 \bullet [\bar{x}_j]_2$;
 5. For $j = 1, 2, \dots, i$: check if $[a_j x_j]_1 \bullet [1]_2 = [x_j]_1 \bullet [a_j]_2 = [a_{j-1} x_{j-1}]_1 \bullet [\bar{a}_j \bar{x}_j]_2$;
 6. For $k = -n, \dots, n$: check if $[x_i^k]_1 \bullet [1]_2 = [1]_1 \bullet [x_i^k]_2$;
 7. For $k = -n+1, \dots, n$: check if $[x_i^k]_1 \bullet [1]_2 = [x_i^{k-1}]_1 \bullet [x_i]_2$;
 8. For $k = -n, \dots, -1, 1, \dots, n$: check if $[a_i x_i^k]_1 \bullet [1]_2 = [1]_1 \bullet [a_i x_i^k]_2 = [x_i^k]_1 \bullet [a_i]_2$;

Return 1 if all the checks passed, otherwise return \perp .

Figure 8.3: SU and SV algorithms for SONIC.

Efficiency. As it can be seen in Figure 8.2 and 8.3, given the SU algorithm, similar to the SG algorithm, to update the SRS of size n in Sonic, one needs to compute $4n + 2$ exponentiations in \mathbb{G}_1 and $4n + 2$ exponentiations in \mathbb{G}_2 . Using the SV algorithm described in Figure 8.3, to verify an i -time updated SRS, $i \geq 1$, a prover needs to compute $12n - 1$ pairing operations (importantly, independent of the number of updates), while a verifier needs to compute $12n + 10i + 4$ pairings.

Security Proofs. In [Mal+19, Theorem 6.1, 6.2], authors proved that assuming the ability to extract a trapdoor for the subverted/updated SRS (without proving it), Sonic satisfies Sub-ZK and KS. The following lemmas prove that using the SG, SU and SV algorithms (given in Fig.8.2 and 8.3), under the BDH-KE assumption, one can extract the simulation trapdoors from a subverted/updated SRS.

Lemma 3 (Trapdoor Extraction from a Subverted SRS). Given the algorithm in Figure 8.2 and 8.3, suppose that there exists a PPT adversary \mathcal{A} that outputs a $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i)$ such that $\text{SV}(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i, \text{P}) = 1$ with non-negligible probability. Then, by the BDH-KE assumption (given in Definition 58) there exists a PPT extractor $\text{Ext}_{\mathcal{A}}$ given the random tape of \mathcal{A} as input, outputs (x_i, a_i) such that running SG with (x_i, a_i) results in $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i)$.

Proof. An SRS, $\text{c}\vec{\text{r}}\text{s}_i$, and proof, Π_i , that passes verification is structured as if it were computed by $\text{SG}(\mathbf{R})$; i.e., there exist values $(x_i, a_i) \in \mathbb{F}_p^2$ such that Π_i includes $([x_i]_1, [a_i x_i]_1, [a_i]_2)$ and $\text{c}\vec{\text{r}}\text{s}_i$ includes $(([x_i^k]_1, [x_i^k]_2, [a_i x_i^k]_2)_{k=-n}^n, ([a_i x_i^k]_1)_{k=-n, k \neq 0}^n)$. Note that, for $k = 1$, one can deduce $([x_i]_1, [a_i x_i]_1, [x_i]_2, [a_i]_2, [a_i x_i]_2)$ from $\text{c}\vec{\text{r}}\text{s}_i$.

Let \mathcal{A} be an adversary that outputs $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i)$. We then define algorithms \mathcal{A}_{x_i} and \mathcal{A}_{a_i} , that each run $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i) \leftarrow \mathcal{A}(\mathbf{R})$, parse Π as above, and returns $([x_i]_1, [x_i]_2)$ and $([a_i x_i]_1, [a_i]_2)$, respectively. According to the BDH-KE assumption (given in Definition 58) there exist PPT extractors $\text{Ext}_{\mathcal{A}_{x_i}}$ and $\text{Ext}_{\mathcal{A}_{a_i}}$ that, given the randomness of \mathcal{A}_{x_i} and \mathcal{A}_{a_i} , output some $x_i, a_i \in \mathbb{F}_p$ that can be used to generate $([x_i]_1, [a_i x_i]_1, [x_i]_2, [a_i]_2, [a_i x_i]_2)$. By combining $\text{Ext}_{\mathcal{A}_{x_i}}$ and $\text{Ext}_{\mathcal{A}_{a_i}}$, we obtain a full extractor for \mathcal{A} . From the rest of checks within the SV algorithm one concludes that all the SRS elements are consistent and the SRS is well-formed. \square

The following lemma shows that SRS trapdoors can be extracted from an updated SRS. To this end, we first recall a corollary from [Gar+18].

Corollary 1. In the updatable SRS model, single adversarial updates imply full updatable security [Gar+18, Lemma 6].

Lemma 4 (Trapdoor Extraction from an Updated SRS). Given the algorithm in Figure 8.2 and 8.3, suppose that there exists a PPT \mathcal{A} such that given $(\text{c}\vec{\text{rs}}_0, \pi_0) \leftarrow \text{SG}(\mathbf{R})$, \mathcal{A} returns an updated SRS $(\text{c}\vec{\text{rs}}_1, \pi_1)$, where $\text{SV}(\text{c}\vec{\text{rs}}_1, \Pi_1, \mathbf{V}) = 1$ with a non-negligible probability. Then, the BDH-KE assumption implies that there exists a PPT extractor $\text{Ext}_{\mathcal{A}}$ that, given the randomness of \mathcal{A} as input, outputs (\bar{x}_1, \bar{a}_1) that are used to update $\text{c}\vec{\text{rs}}_0$ and generate $(\text{c}\vec{\text{rs}}_1, \Pi_1)$.

Proof. According to the Corollary 1, we consider the case that \mathcal{A} updates the SRS only once, but similar to [Gar+18, Lemma 6], it can be generalized. Parse Π_0 as a tuple $(([x_0]_1, [a_0x_0]_1, [a_0]_2), ([x_0]_1, [x_0]_2, [a_0x_0]_1, [a_0x_0]_2, [a_0]_2))$ and $\text{c}\vec{\text{rs}}_0$ as $([x_0^k]_1, [x_0^k]_2, [a_0x_0^k]_2)_{k=-n}^n$ and $[a_0x_0^k]_1$ for $k = -n, \dots, -1, 1, \dots, n$.

We consider an adversary \mathcal{A} , that given $(\text{c}\vec{\text{rs}}_0, \Pi_0)$, returns an updated SRS, $\text{c}\vec{\text{rs}}_1$, which contains $([x_1^k]_1, [x_1^k]_2, [a_1x_1^k]_2)_{k=-n}^n$ and $[a_1x_1^k]_1$ for $k = -n, \dots, -1, 1, \dots, n$, and a proof π_1 for correct updating as containing $(([x_1]_1, [a_1x_1]_1, [a_1]_2), ([\bar{x}_1]_1, [\bar{x}_1]_2, [\bar{a}_0\bar{x}_0]_1, [\bar{a}_0\bar{x}_0]_2, [\bar{a}_0]_2))$. If the SRS verification accepts the updated SRS, namely if $\text{SV}(\text{c}\vec{\text{rs}}_1, \Pi_1, \mathbf{V}) = 1$, then the following equations hold, 1) $[\bar{x}_1]_1 \bullet [1]_2 = [1]_1 \bullet [\bar{x}_1]_2$, 2) $[\bar{a}_1\bar{x}_1]_1 \bullet [1]_2 = [1]_1 \bullet [\bar{a}_1\bar{x}_1]_2 = [\bar{x}_1]_1 \bullet [\bar{a}_1]_2$, 3) $[x_1]_1 \bullet [1]_2 = [x_0]_1 \bullet [\bar{x}_1]_2$, 4) $[a_1x_1]_1 \bullet [1]_2 = [x_1]_1 \bullet [a_1]_2 = [a_0x_0]_1 \bullet [\bar{a}_1\bar{x}_1]_2$. So from the equations 1) and 2), under the BDH-KE assumption, there exist extractors $\text{Ext}_{\bar{x}_1}$ and $\text{Ext}_{\bar{a}_1}$ that output \bar{a}_1 and \bar{x}_1 . If \bar{a}_1 and \bar{x}_1 are non-zero, then from the rest of verification equations within SV algorithm (e.g., $[x_1]_1 \bullet [1]_2 = [x_0]_1 \bullet [\bar{x}_1]_2$), one can conclude that $x_1 = \bar{x}_1x_0$, $a_1 = \bar{a}_1a_0$, and the SRS is well-formed. \square

8.3.2 SU and SV Algorithms for Marlin

Marlin. As a follow-up work to Sonic and a concurrent work to Plonk, Chiesa et al. proposed Marlin [Chi+19], which is comparable to Plonk in performance and outperforms Sonic. Compared to Sonic, Marlin reduces P’s computational cost by a factor of $10\times$ and improves V’s time by a factor of $4\times$ without compromising the constant-size property of proofs. To this end, the authors first propose an information-theoretic model called Algebraic Holographic Proof (AHP), which is an interactive protocol between algebraic P and V. The verifier performs a small number of queries on an encoding of the circuit instead of receiving the entire circuit description. At the end, the verifier makes a number of queries to the proofs provided by the prover and then performs low-degree tests to be convinced about the validity of proof and the encoding of the circuit. Then, they proposed

a transformation that uses PCs with Fiat-Shamir transformation [FS87] and compiles any public coin AHP for sparse Rank 1 Constraint System (R1CS) instances into a preprocessing zk-SNARK with universal and updatable SRS. To build Marlin, authors first proposed two PC schemes, which one is proven to be secure under a concrete knowledge assumption, and the other one is built in the Algebraic Group Model (AGM) [Chi+19, Appendix B]. The scheme built in the AGM model achieves a better efficiency and requires a single group element to commit to a polynomial (instead of two in the initial construction). Marlin is a zk-SNARK which is obtained by instantiating their transformation by the AGM-based PC scheme. Both their PC schemes are proven to be secure under a *trusted setup* [Chi+19, Lemmas B.5-B.15], and later, the AGM-based one is used to obtain updatable zk-SNARK Marlin.

Achieving Sub-ZK and Upd-KS in Marlin. Marlin uses a universal SRS and assuming that the simulation trapdoors are provided to the ZK simulator, it is proven to achieve ZK and KS in the AGM. In [Chi+19, Remark 7.1], authors argue that their constructions have updatable SRS because of using monomial terms in the SRS, and thus fall within the framework of [Gro+18]. The SRS of Marlin, which is equivalent to the SRS of their AGM-based PC scheme, consists of $\text{c}\vec{\text{r}}\text{s} := (([x^k]_1, [\gamma x^k]_1)_{k=0}^n, [1]_2, [x]_2)$ group elements. This SRS is shown to be sufficient for their PC scheme. Note that a standard PC scheme, is constructed under a trusted setup, and there is no guarantee that it will remain secure under a subverted SRS or a maliciously updated SRS. Therefore, once we use the SRS of a PC scheme (with a trusted setup) to build a Sub-ZK zk-SNARK with updatable SRS, we need to ensure that the SRS of resulting zk-SNARK is well-formed and trapdoor-extractable [Gar+18]. Since Marlin is proven to satisfy KS under the above SRS $\text{c}\vec{\text{r}}\text{s}$, therefore, to prove that it also achieves Upd-KS, we need to show that the SRS trapdoors can be extracted from a subverted or a (maliciously) updated SRS. However, one may notice that in practice an adversary, capable of hashing to an elliptic curve, can produce the SRS $([x]_1, [\gamma x]_1, [1]_2, [x]_2)$ without knowing γ . For instance, it can sample a group element from \mathbb{G}_1 , without knowing its exponent, and then use a known x to compute $([x]_1, [\gamma x]_1, [1]_2, [x]_2)$ for an unknown γ . A malicious SRS updater can perform a similar attack.

One may argue that Marlin (and some follow-up schemes) is proven in the original AGM [FKL18], which adversaries are purely algebraic and do not have the capability to create random group elements without knowing their discrete logarithms. This argument is valid, but the problem still exists in practice and such constructions may not achieve Sub-ZK by default, as an adversary can use elliptic curve hashing [Ica09] to sample random group elements without knowing the exponents. To deal with such concerns, earlier Sub-ZK SNARKs [Abd+17;

[Lip22] used and are proven in more realistic models, namely the Generic Group Model (GGM) with *hashing* [Abd+17] and the AGM with *hashing* [Lip22]. The “with hashing” parts mean that the adversary is allowed to sample random group elements without knowing the exponents, say using the elliptic curve hashing [Ica09]. Considering the discussed issue, one can see that to achieve Sub-ZK/Upd-KS in updatable zk-SNARKs, including Marlin, a more realistic option is to prove them in the more realistic variant of AGM, namely AGM with hashing [Lip22], and also explicitly construct the extraction algorithms required in the games of Sub-ZK/Upd-KS. It is worth to mention that, by chance, the SRS of Kate et al.’s polynomial commitment scheme [KZG10] is well-formed and without further modification, its SRS can achieve trapdoor extractability under BDH-KE assumption. This is the reason that the updatable zk-SNARKs that directly use Kate et al.’s PC scheme [KZG10], e.g., Lunar or Basilisk, do not face with the mentioned issue. In the rest, we focus on constructing a concrete extraction algorithm which is necessary to prove the Sub-ZK and Upd-KS of Marlin. As we argued above, γ cannot be extracted from the original SRS of Marlin, and we need to slightly modify its SRS to achieve trapdoor extractability and prove Sub-ZK and Upd-KS.

SRS Generation, $(\text{c}\tilde{\text{r}}\text{s}_0, \Pi_0) \leftarrow \text{SG}(\mathbf{R})$: Given \mathbf{R} , first deduce the security parameter 1^κ and obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2) \leftarrow \text{BGgen}(1^\kappa)$; then act as follows:

- Sample $\bar{x}_0, \bar{\gamma}_0 \leftarrow \mathbb{Z}_p^*$, and set $x_0 := \bar{x}_0$, and $\gamma_0 := \bar{\gamma}_0$ which are the trapdoors of $\text{c}\tilde{\text{r}}\text{s}_0$;
- For $k = 0, \dots, n$: compute $[x_0^k]_1, [\gamma_0 x_0^k]_1$;
- Compute $[x_0]_2$, and $[x_0 \gamma_0]_2$;
- Set $\text{c}\tilde{\text{r}}\text{s}_0 := (([x_0^k]_1, [\gamma_0 x_0^k]_1)_{k=0}^n, [x_0]_2, [x_0 \gamma_0]_2)$, and the well-formedness proof $\Pi_0 := (\Pi_0^{\text{agg}}, \Pi_0^{\text{ind}}) := (([\gamma_0]_1, [x_0 \gamma_0]_1, [x_0]_2), ([\bar{x}_0]_1, [\bar{\gamma}_0]_1, [\bar{x}_0 \bar{\gamma}_0]_2))$;
- Return $(\text{c}\tilde{\text{r}}\text{s}_0, \Pi_0)$;

Figure 8.4: Slightly modified SG algorithm of Marlin. The term $[x_0 \gamma_0]_2$ is added to SRS and proof to make the SRS well-formed and achieve trapdoor extractability.

Marlin with a Trapdoor Extractable SRS. To deal with the discussed issue, the solution is to force the adversary to add a proof of knowledge of γ to the SRS, such that the simulator would be able to extract γ from a maliciously generated SRS. In earlier works [BFS16; Abd+17; Gar+18], this is simply achieved by

forcing the SRS generator to return γ in two different groups. Then, relying on the BDH-KE assumption one can extract γ from a maliciously generated SRS. Consequently, we slightly modify the SRS of Marlin and add a single group element $[\gamma x]_2$ to it. Then, we show that in the modified version, the SRS trapdoors can be extracted from a subverted/updated SRS, which would allow to prove Sub-ZK/Upd-KS. We describe the modified SG algorithm of Marlin in Figure 8.4, and the new added element is shown with gray background.

SU and SV Algorithms and Their Efficiency. in Figure 8.5, we describe our constructed SU and SV algorithms for Marlin with the modified SRS. As the other cases, the SRS update is multiplicative, and at the end, the updater also gives a well-formedness proof which includes the new element $[\bar{x}_i \bar{\gamma}_i]_2$. The new element allows one to verify the well-formedness of the final SRS as well as the validity of intermediate proofs. The SV algorithm verifies if $\{\Pi_j\}_{j=0}^i$ are valid and the final SRS, $c\bar{r}s_i$, is well-formed.

Using the SU algorithm in Figure 8.5, similar to the SG algorithm (in Figure 8.4), to update the SRS of size n in Marlin, one needs to compute 2 exponentiations in \mathbb{G}_2 and $2n + 1$ exponentiations in \mathbb{G}_1 . Using the SV algorithm described in Figure 8.5, to verify an i -time updated SRS, $i \geq 1$, a prover needs to compute $4n + 2$ pairing operations, while a verifier needs to compute $4n + 2 + 9i + 4$ pairings.

Security Proofs. Relying on the fact that the underlying PC scheme is secure, Marlin, is proven to achieve ZK and KS in the AGM model [Chi+19, Theorem 8.1, 8.3 and 8.4]. Our evaluations show that our minimal modification to their PC scheme does not compromise the security of the original scheme. Moreover, in the rest, we show that using the presented SG, SU and SV algorithms (given in Figure 8.4 and 8.5), under the BDH-KE assumption (as in [Lip22]), it is also possible to extract the simulation trapdoors from a subverted/updated SRS and achieve Sub-ZK and Upd-KS in the AGM.

Lemma 5 (Trapdoor Extraction from a Subverted SRS). Given the algorithms in Figure 8.4 and 8.5, suppose that there exists a PPT adversary \mathcal{A} that outputs $(c\bar{r}s_i, \Pi_i)$ such that $\text{SV}(c\bar{r}s_i, \Pi_i, P) = 1$ with a non-negligible probability. Then, by the BDH-KE assumption (given in Definition 58) there exists a PPT extractor $\text{Ext}_{\mathcal{A}}$ that, given the random tape of \mathcal{A} as input, outputs (x_i, γ_i) such that running SG with (x_i, γ_i) results in $(c\bar{r}s_i, \Pi_i)$.

Proof. The proof is analogue to the proof of Lemma 3. □

SRS Update, $(\text{c}\tilde{\text{rs}}_i, \Pi_i) \leftarrow \text{SU}(\text{c}\tilde{\text{rs}}_{i-1}, \{\Pi_{j-1}\}_{j=0}^{i-1})$: Given $(\text{c}\tilde{\text{rs}}_{i-1}, \{\Pi_{j-1}\}_{j=0}^{i-1})$,

- Parse $\text{c}\tilde{\text{rs}}_{i-1} := (([x_{i-1}^k]_1, [\gamma_{i-1}x_{i-1}^k]_1)_{k=0}^n, [x_{i-1}]_2, [x_{i-1}\gamma_{i-1}]_2)$;
- Sample $\bar{x}_i, \bar{\gamma}_i \leftarrow \mathbb{Z}_p^*$ as the secret shares to use for updating $\text{c}\tilde{\text{rs}}_{i-1}$.
- For $k = 0, \dots, n$: set $[x_i^k]_1 := \bar{x}_i^k \cdot [x_{i-1}^k]_1$, $[\gamma_i x_i^k]_1 := \bar{\gamma}_i \bar{x}_i^k \cdot [\gamma_{i-1} x_{i-1}^k]_1$;
- set $[x_i]_2 := \bar{x}_i \cdot [x_{i-1}]_2$ and $[x_i \gamma_i]_2 := \bar{x}_i \bar{\gamma}_i \cdot [x_{i-1} \gamma_{i-1}]_2$;
- Set $\text{c}\tilde{\text{rs}}_i := (([x_i^k]_1, [\gamma_i x_i^k]_1)_{k=0}^n, [x_i]_2, [x_i \gamma_i]_2)$, and the well-formedness proof $\Pi_i := (\Pi_i^{\text{agg}}, \Pi_i^{\text{ind}}) := (([\gamma_i]_1, [x_i \gamma_i]_1, [x_i]_2), ([\bar{x}_i]_1, [\bar{\gamma}_i]_1, [\bar{x}_i]_2, [\bar{x}_i \bar{\gamma}_i]_2))$;
- Return $(\text{c}\tilde{\text{rs}}_i, \Pi_i)$;

SRS Verify, $(\perp/1) \leftarrow \text{SV}(\text{c}\tilde{\text{rs}}_i, (\Pi_j)_{j=0}^i, \text{party})$: To verify (an i -time updated) $\text{c}\tilde{\text{rs}}_i := (([x_i^k]_1, [\gamma_i x_i^k]_1)_{k=0}^n, [x_i]_2, [x_i \gamma_i]_2)$, and $\Pi_j := (\Pi_j^{\text{agg}}, \Pi_j^{\text{ind}}) := (([\gamma_j]_1, [x_j \gamma_j]_1, [x_j]_2), ([\bar{x}_j]_1, [\bar{\gamma}_j]_1, [\bar{x}_j]_2, [\bar{x}_j \bar{\gamma}_j]_2))$; for $j = 0, 1, \dots, i$:

If party = P:

1. For $k = 1, \dots, n$: check if $[x_i^k]_1 \bullet [1]_2 = [x_i^{k-1}]_1 \bullet [x_i]_2$;
2. For $k = 1, \dots, n$: check if $[\gamma_i x_i^k]_1 \bullet [1]_2 = [\gamma_i x_i^{k-1}]_1 \bullet [x_i]_2$;
3. Check if $[x_i \gamma_i]_1 \bullet [1]_2 = [1]_1 \bullet [\gamma_i x_i]_2$;

If party = V:

- If $i = 0$: $\text{c}\tilde{\text{rs}}_0$ is sampled by verifier, and it does not need to be verified.
- If $i \geq 1$:
 1. Check if $[\gamma_0]_1 = [\bar{\gamma}_0]_1$ and $[x_0]_2 = [\bar{x}_0]_2$;
 2. For $j = 0, 1, \dots, i$: check if $[\bar{x}_j]_1 \bullet [1]_2 = [1]_1 \bullet [\bar{x}_j]_2$ and $[1]_1 \bullet [\bar{x}_j \bar{\gamma}_j]_2 = [\bar{\gamma}_j]_1 \bullet [\bar{x}_j]_2$.
 3. For $j = 1, 2, \dots, i$: check if $[1]_1 \bullet [x_j]_2 = [\bar{x}_j]_1 \bullet [x_{j-1}]_2$, $[x_j \gamma_j]_1 \bullet [1]_2 = [x_{j-1} \gamma_{j-1}]_1 \bullet [\bar{x}_j \bar{\gamma}_j]_2 = [\gamma_j]_1 \bullet [x_j]_2$;
 4. For $k = 1, \dots, n$: check if $[x_i^k]_1 \bullet [1]_2 = [x_i^{k-1}]_1 \bullet [x_i]_2$;
 5. For $k = 1, \dots, n$: check if $[\gamma_i x_i^k]_1 \bullet [1]_2 = [\gamma_i x_i^{k-1}]_1 \bullet [x_i]_2$;
 6. Check if $[x_i \gamma_i]_1 \bullet [1]_2 = [1]_1 \bullet [\gamma_i x_i]_2$;

Return 1 if all the checks passed, otherwise return \perp .

Figure 8.5: SV and SU algorithms for Marlin with the slightly modified SRS.

Lemma 6 (Trapdoor Extraction from an Updated SRS). Given the algorithms in Figure 8.4 and 8.5, suppose that there exists a PPT \mathcal{A} such that given $(\text{c}\tilde{\text{rs}}_0, \Pi_0) \leftarrow \text{SG}(\mathbf{R})$, \mathcal{A} returns an updated SRS $(\text{c}\tilde{\text{rs}}_1, \Pi_1)$ s.t. $\text{SV}(\text{c}\tilde{\text{rs}}_1, \Pi_1, \mathbf{V}) =$

SRS Generation, $(\vec{crs}_0, \Pi_0) \leftarrow \text{SG}(\mathbf{R})$: Given \mathbf{R} , first deduce the security parameter 1^κ and obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2) \leftarrow \text{BGgen}(1^\kappa)$; then act as follows:

- Sample $x_0 := \bar{x}_0 \leftarrow \mathbb{Z}_p^*$, which is the trapdoor associated with \vec{crs}_0 ;
- For $k = 1, \dots, n$: compute $[x_0^k]_1, [x_0^k]_2$;
- Set $\vec{crs}_0 := ([x_0^k]_1, [x_0^k]_2)_{k=0}^n$, and the well-formedness proof $\Pi_0 := (\Pi_0^{\text{Agg}}, \Pi_0^{\text{Ind}}) := ([x_0]_1, ([\bar{x}_0]_1, [\bar{x}_0]_2))$;
- Return (\vec{crs}_0, Π_0) ;

Figure 8.6: SG algorithm for LunarLite.

1, with a non-negligible probability. Then, the BDH-KE assumption implies that there exists a PPT extractor $\text{Ext}_{\mathcal{A}}$ that, given the randomness of \mathcal{A} as input, outputs $(\bar{x}_1, \bar{\gamma}_1)$ that are used to update \vec{crs}_0 and generate (\vec{crs}_1, Π_1) .

Proof. The proof is analogue to the proof of Lemma 4. □

8.3.3 SU and SV Algorithms for LunarLite

LunarLite and its SG algorithm. In 2021, Campanelli et al. proposed Lunar [Cam+21] and compared to Marlin the authors describe several improvements. As we mentioned above, Marlin is built over sparse R1CS instances while Campanelli et al. define a new and simpler version of R1CS, known as R1CS-lite, with only two characterizing matrices instead of three s.t. one of the matrices can be the identity matrix. R1CS-lite with almost the same complexity as R1CS can be utilized to express the language of circuit satisfiability. Meanwhile, the property of two-matrix instances enables the authors to achieve more efficient and simpler zk-SNARKs. In addition, Campanelli et al. demonstrate an efficient method to prove PC soundness with partial opening rather than opening all the commitments. Note that in Marlin and Lunar like Plonk, the prover to commit to vectors utilizes the Lagrange interpolation basis. For further details, we refer to the main paper [Cam+21]. In Figure 8.6, we describe the SG algorithm of LunarLite, which is the most efficient instantiate of Lunar in term of proof size. Lunar has another variant, so called LunarLite2x, which has slightly shorter SRS, but results in slightly longer proofs.

SRS Update, $(\text{c}\bar{\text{r}}\text{s}_i, \Pi_i) \leftarrow \text{SU}(\text{c}\bar{\text{r}}\text{s}_{i-1}, \{\Pi_{j-1}\}_{j=0}^{i-1})$: Given $(\text{c}\bar{\text{r}}\text{s}_{i-1}, \{\Pi_{j-1}\}_{j=0}^{i-1})$,

- Parse $\text{c}\bar{\text{r}}\text{s}_{i-1} := ([x_{i-1}^k]_1, [x_{i-1}^k]_2)_{k=0}^n$;
- Sample $\bar{x}_i \leftarrow \mathbb{Z}_p^*$, as the secret share used for updating $\text{c}\bar{\text{r}}\text{s}_{i-1}$.
- For $k = 1, \dots, n$: set $[x_i^k]_1 := \bar{x}_i^k \cdot [x_{i-1}^k]_1$; and $[x_i^k]_2 := \bar{x}_i^k \cdot [x_{i-1}^k]_2$;
- Set $\text{c}\bar{\text{r}}\text{s}_i := ([x_i^k]_1, [x_i^k]_2)_{k=0}^n$, and $\Pi_i := (\Pi_i^{\text{Agg}}, \Pi_i^{\text{Ind}}) := ([x_i]_1, ([\bar{x}_i]_1, [\bar{x}_i]_2))$.
- Return $(\text{c}\bar{\text{r}}\text{s}_i, \Pi_i)$;

SRS Verify, $(\perp/1) \leftarrow \text{SV}(\text{c}\bar{\text{r}}\text{s}_i, (\Pi_j)_{j=0}^i, \text{party})$: To verify (an i -time updated) $\text{c}\bar{\text{r}}\text{s}_i := ([x_i^k]_1, [x_i^k]_2)_{k=0}^n$, and $\Pi_j := (\Pi_j^{\text{Agg}}, \Pi_j^{\text{Ind}}) := ([x_j]_1, ([\bar{x}_j]_1, [\bar{x}_j]_2))$ for $j = 0, 1, \dots, i$:

If party = P:

1. For $k = 1, 2, \dots, n$: check if $[x_i^k]_1 \bullet [1]_2 = [1]_1 \bullet [x_i^k]_2 = [x_i^{k-1}]_1 \bullet [x_i]_2$;

If party = V:

- If $i = 0$: $\text{c}\bar{\text{r}}\text{s}_0$ is sampled by verifier, and it does not need to be verified.
- If $i \geq 1$:
 1. Check that $[x_0]_1 = [\bar{x}_0]_1$;
 2. For $j = 0, 1, \dots, i$: check if $[\bar{x}_j]_1 \bullet [1]_2 = [1]_1 \bullet [\bar{x}_j]_2$;
 3. For $j = 1, 2, \dots, i$: check if $[x_j]_1 \bullet [1]_2 = [x_{j-1}]_1 \bullet [x_j]_2$;
 4. For $k = 1, \dots, n$: check if $[x_i^k]_1 \bullet [1]_2 = [1]_1 \bullet [x_i^k]_2 = [x_i^{k-1}]_1 \bullet [x_i]_2$;

Return 1 if all the checks passed, otherwise return \perp .

Figure 8.7: SU and SV algorithms for LunarLite.

SU and SV Algorithms and Their Efficiency. Fig. 8.7 illustrates our constructed SU and SV algorithms for updatable zk-SNARK LunarLite. In order to update the SRS of size n in LunarLite, one needs to execute the SU algorithm described in Figure 8.7 that similar to the SRS generation algorithm, it requires $n + 1$ exponentiations in \mathbb{G}_1 and $n + 1$ exponentiations in \mathbb{G}_2 . Using the SV algorithm (given in Fig 8.7), to verify an i -time updated SRS, $i \geq 1$, a prover needs to compute $3n$ pairing operations (importantly, independent of the value of i), while a verifier needs to compute $3n + 4i + 2$ pairings.

Security Proofs. In [Cam+21], authors proved that different versions of Lunar, including LunarLite can achieve ZK and KS. However, similar to other constructions, they did not explicitly prove Sub-ZK and Upd-KS. For example, to prove ZK, they assumed that the simulation trapdoor x is provided to

the simulator. The same as other constructions, next, we prove that using the SG, SU and SV algorithms (given in Fig.8.6 and 8.7), under the BDH-KE assumption, we can extract the simulation trapdoors for LunarLite from a subverted or updated SRS, which implies that LunarLite meets Sub-ZK and Upd-KS.

Lemma 7 (Trapdoor Extraction from a Subverted SRS). Given the algorithms in Figure 8.6 and 8.7, suppose that there exists a PPT adversary \mathcal{A} that outputs a $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i)$ such that $\text{SV}(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i, \text{P}) = 1$ with non-negligible probability. Then, by the BDH-KE assumption (given in Definition 58) there exists a PPT extractor $\text{Ext}_{\mathcal{A}}$ that, given the random tape of \mathcal{A} as input, outputs x_i such that running SG with x_i results in $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i)$.

Proof. An $\text{c}\vec{\text{r}}\text{s}_i$ and Π_i that passes verification, namely $\text{SV}(\text{c}\vec{\text{r}}\text{s}_1, \Pi_1, \text{P}) = 1$, is structured as if it were computed by $\text{SG}(\mathbf{R})$; i.e., there exist values $x_i \in \mathbb{F}_p$ s.t. Π_i includes $[x_i]_1$ and $\text{c}\vec{\text{r}}\text{s}_i$ includes $([x_i^k]_1, [x_i^k]_2)_{k=0}^n$. Therefore, for $k = 1$, one can obtain $([x_i]_1, [x_i]_2)$.

Let \mathcal{A} be an adversary (or subverter) that outputs $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i)$. We then define algorithm \mathcal{A}_{x_i} , that runs $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i) \leftarrow \mathcal{A}(\mathbf{R})$, parse Π_i as above, and returns $([x_i]_1, [x_i]_2)$. Under the BDH-KE assumption (given in Definition 58) there exists a PPT extractor $\text{Ext}_{\mathcal{A}_{x_i}}$ that, given the randomness of \mathcal{A}_{x_i} , outputs a $x_i \in \mathbb{F}_p$ that can be used to generate $([x_i]_1, [x_i]_2)$. This gives an extractor for \mathcal{A} . From the rest of pairing checks within the SV algorithm, one concludes that all SRS elements are consistent and the SRS is well-formed. \square

Lemma 8 (Trapdoor Extraction from an Updated SRS). Given the algorithms in Figure 8.6 and 8.7, suppose that there exists a PPT adversary \mathcal{A} such that given $(\text{c}\vec{\text{r}}\text{s}_0, \pi_0) \leftarrow \text{SG}(\mathbf{R})$, \mathcal{A} returns an updated SRS, $(\text{c}\vec{\text{r}}\text{s}_1, \pi_1)$, where $\text{SV}(\text{c}\vec{\text{r}}\text{s}_1, \Pi_1, \text{V}) = 1$ with a non-negligible probability. Then, the BDH-KE assumption implies that there exists a PPT extractor $\text{Ext}_{\mathcal{A}}$, given the randomness of \mathcal{A} as input, outputs \bar{x}_1 that are used to update $\text{c}\vec{\text{r}}\text{s}_0$ and generate $(\text{c}\vec{\text{r}}\text{s}_1, \Pi_1)$.

Proof. In Corollary 1, we consider the case where \mathcal{A} updates the SRS only once. However, as in [Gar+18, Lemma 6], this case is generalizable. Parse Π_0 as containing $([x_0]_1, ([x_0]_1, [x_0]_2))$ and $\text{c}\vec{\text{r}}\text{s}_0$ includes $([x_0^k]_1, [x_0^k]_2)_{k=0}^n$.

We consider an adversary \mathcal{A} , that given $(\text{c}\vec{\text{r}}\text{s}_0, \Pi_0)$, returns an updated SRS, $\text{c}\vec{\text{r}}\text{s}_1$, which contains $([x_1^k]_1, [x_1^k]_2)_{k=0}^n$, and a proof $\pi_1 = ([x_1]_1, ([\bar{x}_1]_1, [\bar{x}_1]_2))$ for correct updating. If the SV algorithm accepts the updated SRS, say $\text{SV}(\text{c}\vec{\text{r}}\text{s}_1, \Pi_1, \text{V}) = 1$, then the following equations hold, 1) $[\bar{x}_1]_1 \bullet [1]_2 = [1]_1 \bullet [\bar{x}_1]_2$, 2) $[x_1]_1 \bullet [1]_2 = [x_0]_1 \bullet [\bar{x}_1]_2$, 3) $[a_1]_1 \bullet [1]_2 = [1]_1 \bullet [x_1]_2$. If the equation 1) holds, under the BDH-KE assumption, there exists an extractor

$\text{Ext}_{\bar{x}_1}$ that outputs \bar{x}_1 . If \bar{x}_1 is non-zero, then from the other one concludes that $x_1 = \bar{x}_1 x_0$, and the SRS is well-formed. \square

8.3.4 SU and SV Algorithms for Plonk and Basilisk

Plonk and Basilisk and their SG algorithms. As a subsequent work on Sonic [Mal+19], in 2019, Gabizon et al. [GWC19] designed Plonk as an updatable and universal zk-SNARK that can be run in two modes: either with a small proof (large SRS) or with a long proof (short SRS). Although Plonk relies neither on bivariate polynomials nor sparse matrices that leads to a more general type of constraints, its SRS depends both on addition and multiplication gates for any given circuit. While Sonic commits to vectors using standard interpolation basis, Plonk uses Lagrange interpolation basis. As a subsequent work on Sonic, Plonk, Marlin and LunarLite, in 2021, Rafols and Zapico [RZ21] presented Basilisk updatable zk-SNARK. They, first defined a novel information theoretical interactive technique called Checkable Subspace Sampling (CSS) arguments in which P shows that a vector is sampled from a subspace based on V 's coin. To be more precise, for a given matrix M , both P and V agree on a polynomial $F(x)$ which encodes a row v within M 's rows space. This method is efficient because, in spite of the fact that the coefficients of the linear combination defining v are sampled according to V 's coin, there is no need to perform a linear number of operations in order to check that $F(x)$ is well-formed. There are a number of trade-offs associated with universal and updatable zk-SNARK resulting from the CSS proof systems constructed. The most efficient instantiation is called Basilisk that is built for a limited constraint system in which R1CS instances' matrices have a small constant number of elements per row (it is equivalent to arithmetic circuits of bounded fan-out). For further details about Plonk and Basilisk, we refer to their main papers [GWC19; RZ21]. Plonk and Basilisk have almost the same SG algorithms with a similar SRS elements, except that in the case of Basilisk, there exists an extra element u_i in the SRS and the SRS is generated with a smaller upper bound on the size of relation. The reason is that the constraint system used in Plonk, encodes both the addition and multiplication gates that leads to a longer SRS. We investigate and construct the SG, SU and SV algorithms for Basilisk, but with minimal changes they can be adapted and be used for Plonk. We start by describing the SG algorithm of Basilisk in Figure 8.8.

SU and SV Algorithms and Their Efficiency. Fig. 8.9 describes our constructed SU and SV algorithms for updatable zk-SNARK Basilisk. By removing the parts related to the element u_i , the algorithms can also be used for Plonk. To one time updating the SRS of Basilisk, one would need to execute the SU algorithm

SRS Generation, $(\text{c}\vec{\text{r}}\text{s}_0, \Pi_0) \leftarrow \text{SG}(\mathbf{R})$: Given \mathbf{R} , first deduce the security parameter 1^κ and obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2) \leftarrow \text{BGgen}(1^\kappa)$; then act as follows:

- Sample $x_0 \leftarrow \mathbb{Z}_p$, which is the simulation trapdoor associated with $\text{c}\vec{\text{r}}\text{s}_0$; $\text{chooses an arbitrary } u_0 \in \mathbb{Z}_p^*$;
- Compute $[x_0]_2$ and $[x_0^k]_1$ for $k = 1, 2, \dots, n$;
- Set $\text{c}\vec{\text{r}}\text{s}_0 := \left(([x_0^k]_1)_{k=1}^n, [x_0]_2, \text{the elements } u_0 \right)$, and $\Pi_0 := (\Pi_0^{\text{Agg}}, \Pi_0^{\text{Ind}}) := ([x_0]_1, ([x_0]_1, [x_0]_2))$;
- Return $(\text{c}\vec{\text{r}}\text{s}_0, \Pi_0)$;

Figure 8.8: SG algorithm for Basilisk (and Plonk without the elements u_0).

that requires to compute $n + 1$ exponentiations in \mathbb{G}_1 and 2 exponentiations in \mathbb{G}_2 . On the other side, to verify an i -time updated SRS, a prover would need to compute $2n$ pairing operations (independent of the number of updates), while a verifier would need to compute $4i + 2n + 2$ pairings.

Using a minimally modified versions of the algorithms given in Figure 8.8 and 8.9, to update the SRS of Plonk, one would need to compute $3m + 1$ exponentiations in \mathbb{G}_1 and 2 exponentiations in \mathbb{G}_2 , where m is the number of total (addition and multiplication) gates in the circuit. To verify an i -time updated SRS, a prover would need to compute $6m$ pairing operations (independent of the number of updates), while a verifier would need to compute $4i + 6m + 2$ pairings.

Security Proofs. Similar to Lunar [Cam+21], the authors of Plonk and Basilisk have proved that their schemes achieve ZK and KS. Similarly, by assuming that the simulation trapdoor x is provided to the simulator. Next, we prove that using the SG, SU and SV algorithms (given in Fig.8.8 and 8.9), under the BDH-KE assumption, one can extract the trapdoor x from a subverted or updated SRS of Basilisk and Plonk, that shows that they both satisfy Sub-ZK and Upd-KS.

Lemma 9 (Trapdoor Extraction from a Subverted SRS). Given the algorithms in Figure 8.8 and 8.9, suppose that there exists a PPT adversary \mathcal{A} that outputs a $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i)$ such that $\text{SV}(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i, \mathbf{P}) = 1$ with a non-negligible probability. Then, by the BDH-KE assumption (given in Definition 58) there exists a PPT extractor $\text{Ext}_{\mathcal{A}}$ that, given the random tape of \mathcal{A} as input, outputs x_i such that running SG with x_i results in $(\text{c}\vec{\text{r}}\text{s}_i, \Pi_i)$.

SRS Update, $(\text{crs}_i, \Pi_i) \leftarrow \text{SU}(\text{crs}_{i-1}, \{\Pi_{j-1}\}_{j=0}^{i-1})$: Given $(\text{crs}_{i-1}, \{\Pi_{j-1}\}_{j=0}^{i-1})$,

- Parse $\text{crs}_{i-1} := \left(([x_{i-1}^k]_1)_{k=1}^n, [x_{i-1}]_2, u_{i-1} \right)$;
- Sample $\bar{x}_i \leftarrow \mathbb{Z}_p$; Chooses an arbitrary $\bar{u}_i \in \mathbb{Z}_p^*$;
- Set $\bar{u}_i := \bar{u}_i \cdot u_{i-1}$; $[x_i]_2 = \bar{x}_i \cdot [x_{i-1}]_2$; and $[x_i^k]_1 := \bar{x}_i^k \cdot [x_{i-1}^k]_1$ for $k = 1, 2, \dots, n$;
- Set $\text{crs}_i := \left(([x_i^k]_1)_{k=1}^n, [x_i]_2, \bar{u}_i \right)$, and $\Pi_i := (\Pi_i^{\text{Agg}}, \Pi_i^{\text{Ind}}) := ([x_i]_1, ([\bar{x}_i]_1, [x_i]_2))$;
- Return (crs_i, Π_i) ;

SRS Verify, $(\perp/1) \leftarrow \text{SV}(\text{crs}_i, (\Pi_j)_{j=0}^i, \text{party})$: To verify (an i -time updated) $\text{crs}_i := \left(([x_i^k]_1)_{k=1}^n, [x_i]_2, \bar{u}_i \right)$, and $\Pi_j := (\Pi_j^{\text{Agg}}, \Pi_j^{\text{Ind}}) := ([x_j]_1, ([\bar{x}_j]_1, [x_j]_2))$ for $j = 0, 1, \dots, i$:

If $\text{party} = \text{P}$:

1. For $k = 1, 2, \dots, n$: check if $[x_i^k]_1 \bullet [1]_2 = [x_i^{k-1}]_1 \bullet [x_i^1]_2$;
2. Check if $u_i \in \mathbb{Z}_p^*$;

If $\text{party} = \text{V}$:

- If $i = 0$: crs_0 is sampled by verifier, and it does not need to be verified.
- If $i \geq 1$:
 1. Check that $[x_0]_1 = [\bar{x}_0]_1$;
 2. For $j = 0, 1, \dots, i$: check if $[\bar{x}_j]_1 \bullet [1]_2 = [1]_1 \bullet [\bar{x}_j]_2$;
 3. For $j = 1, 2, \dots, i$: check if $[x_j]_1 \bullet [1]_2 = [x_{j-1}]_1 \bullet [x_j]_2$;
 4. For $k = 1, 2, \dots, n$: check if $[x_i^k]_1 \bullet [1]_2 = [x_i^{k-1}]_1 \bullet [x_i^1]_2$;
 5. Check if $u_i \in \mathbb{Z}_p^*$

Return 1 if all the checks passed, otherwise return \perp .

Figure 8.9: SU and SV algorithms for Basilisk (and Plonk without the elements u_i).

Proof. The proof is analogue to the proof of Lemma 7. □

Lemma 10 (Trapdoor Extraction from an Updated SRS). Given the algorithms in Figure 8.8 and 8.9, suppose that there exists a PPT adversary \mathcal{A} such that given $(\text{crs}_0, \pi_0) \leftarrow \text{SG}(\mathbf{R})$, \mathcal{A} returns an updated SRS (crs_1, π_1) , where $\text{SV}(\text{crs}_1, \Pi_1, \text{V}) = 1$ with a non-negligible probability. Then, the BDH-KE assumption implies that there exists a PPT extractor $\text{Ext}_{\mathcal{A}}$ that, given the

randomness of \mathcal{A} as input, outputs \bar{x}_1 that are used to update $\text{cr}\bar{s}_0$ and generate $(\text{cr}\bar{s}_1, \Pi_1)$.

Proof. The proof is analogue to the proof of Lemma 8. □

8.4 Batched SRS Verification Algorithms

By now, we presented SU and SV algorithms for Sonic, Marlin, Plonk, LunarLite and Basilisk, that allow the parties to update/verify the SRS and bypass the need for a TTP. However, when running an SV algorithm, the prover needs to compute at least $O(n)$ pairing operations, where n denotes the number of multiplication gates in the circuit. On the other hand, the verifier needs to compute $O(n + i)$ pairings, where i represents the number of updates done on the SRS. Consequently, even for circuits of moderate size (e.g., $n \geq 10^4$) with a considerable number of updates (e.g., $i = 100$), the efficiency of these algorithms can be severely impacted. In Section 8.5, we will provide concrete numerical examples to further illustrate this inefficiency.

To make them practical, we use batching techniques from [BGR98] and construct a Batched version of the SV algorithms, BSV in short, which allow the provers to verify the SRS by $O(n)$ exponentiations (mostly, short-exponent) and constant pairings, and the verifiers by $O(n + i)$ exponentiations (mostly, short-exponent) and $O(i)$ pairings. To build the BSV algorithms, we use a corollary of the Schwartz-Zippel lemma stating that if $\sum_{i=1}^{s-1} t_i X_i + X_s = 0$ is a polynomial in $\mathbb{Z}_q[t_i]$ with coefficients X_1, \dots, X_s , $t_i \leftarrow_r \{1, \dots, 2^\kappa\}$ for $i < s$, then $X_i = 0$ for each i , with probability $1 - 1/2^\kappa$. Namely, if $\sum_{i=1}^{s-1} t_i ([a_i]_1 \bullet [b_i]_2) = \sum_{i=1}^{s-1} t_i [c]_T$ for uniformly random t_i , then w.h.p., $[a_i]_1 \bullet [b_i]_2 = [c]_T$ for each individual $i = 1, 2, \dots, s - 1$. In Sec. 8.5, we show that the BSV algorithms can be considerably faster than SV algorithms (at the soundness error rate 2^{-80} , where 80 is a statistical security parameter) and even faster at the soundness error rate 2^{-40} .

It is worth to mention that, using the batching technique comes at the cost of a small loss of soundness: even if the batched equation verifies, there is a probability of at most $2^{-\kappa}$ that one of the non-batched (original) equations was false. In other words, the BSV algorithms will become probabilistic, and they will accept incorrect SRSs with negligible probability. Therefore, once using the BSV algorithms, one needs to modify some of our security results from Sec. 8.3 to achieve statistical SRS trapdoor extractability. Next, we describe a BSV algorithm for each of the studied schemes, and then discuss their efficiency.

Batched SRS Verification, $(\perp/1) \leftarrow \text{BSV}(\text{c}\vec{r}s_i, (\Pi_j)_{j=0}^i, \text{party})$:

To verify (an i -time updated) $\text{c}\vec{r}s_i := \left(([x_i^k]_1, [x_i^k]_2, [a_i x_i^k]_2)_{k=-n}^n, ([a_i x_i^k]_1)_{k=-n, k \neq 0}^n, [a_i]_T \right)$, and $\Pi_j := (\Pi_j^{\text{Agg}}, \Pi_j^{\text{Ind}}) := \left(([x_j]_1, [a_j x_j]_1, [a_j]_2), ([\bar{x}_j]_1, [\bar{x}_j]_2, [\bar{a}_j \bar{x}_j]_1, [\bar{a}_j \bar{x}_j]_2, [\bar{a}_j]_2) \right)$ for $j = 0, 1, \dots, i$:

If party = P:

1. Sample $\{t_k, \hat{t}_k \leftarrow \mathbb{Z}_p^*\}_{k=-n}^n$;
2. Check if $(\sum_{k=-n}^n t_k \cdot [x_i^k]_1) \bullet [1]_2 = [1]_1 \bullet (\sum_{k=-n}^n t_k \cdot [x_i^k]_2)$;
3. Check if $(\sum_{k=-n+1}^n t_k \cdot [x_i^k]_1) \bullet [1]_2 = (\sum_{k=-n+1}^n t_k \cdot [x_i^{k-1}]_1) \bullet [x_i]_2$;
4. Check if $(\sum_{k=-n, k \neq 0}^n \hat{t}_k \cdot [a_i x_i^k]_1) \bullet [1]_2 = [1]_1 \bullet (\sum_{k=-n, k \neq 0}^n \hat{t}_k \cdot [a_i x_i^k]_2) = (\sum_{k=-n, k \neq 0}^n \hat{t}_k \cdot [x_i^k]_1) \bullet [a_i]_2$;

If party = V:

1. Sample $\{r_{1,j}, r_{2,j}, r_{3,j}, r_{4,j} \leftarrow \mathbb{Z}_p^*\}_{j=0}^i$; and $\{t_k, \hat{t}_k \leftarrow \mathbb{Z}_p^*\}_{k=-n}^n$;
2. Check that $[x_0]_1 = [\bar{x}_0]_1$, $[a_0 x_0]_1 = [\bar{a}_0 \bar{x}_0]_1$, and $[a_0]_2 = [\bar{a}_0]_2$;
3. Check if $(\sum_{j=0}^i r_{1,j} \cdot [\bar{x}_j]_1) \bullet [1]_2 = [1]_1 \bullet (\sum_{j=0}^i r_{1,j} \cdot [\bar{x}_j]_2)$;
4. Check if $(\sum_{j=0}^i r_{2,j} \cdot [\bar{a}_j \bar{x}_j]_1) \bullet [1]_2 = [1]_1 \bullet (\sum_{j=0}^i r_{2,j} \cdot [\bar{a}_j \bar{x}_j]_2) = \sum_{j=0}^i (r_{2,j} \cdot [\bar{x}_j]_1 \bullet [\bar{a}_j]_2)$;
5. Check if $(\sum_{j=1}^i r_{3,j} \cdot [x_j]_1) \bullet [1]_2 = \sum_{j=1}^i (r_{3,j} \cdot [x_{j-1}]_1 \bullet [\bar{x}_j]_2)$;
6. Check if $(\sum_{j=1}^i r_{4,j} \cdot [a_j x_j]_1) \bullet [1]_2 = \sum_{j=1}^i (r_{4,j} \cdot [x_j]_1 \bullet [a_j]_2) = \sum_{j=1}^i (r_{4,j} \cdot [a_{j-1} x_{j-1}]_1 \bullet [\bar{a}_j \bar{x}_j]_2)$;
7. Check if $(\sum_{k=-n}^n t_k \cdot [x_i^k]_1) \bullet [1]_2 = [1]_1 \bullet (\sum_{k=-n}^n t_k \cdot [x_i^k]_2)$;
8. Check if $(\sum_{k=-n+1}^n t_k \cdot [x_i^k]_1) \bullet [1]_2 = (\sum_{k=-n+1}^n t_k \cdot [x_i^{k-1}]_1) \bullet [x_i]_2$;
9. Check if $(\sum_{k=-n, k \neq 0}^n \hat{t}_k \cdot [a_i x_i^k]_1) \bullet [1]_2 = [1]_1 \bullet (\sum_{k=-n, k \neq 0}^n \hat{t}_k \cdot [a_i x_i^k]_2) = (\sum_{k=-n, k \neq 0}^n \hat{t}_k \cdot [x_i^k]_1) \bullet [a_i]_2$;

Return 1 if all the checks passed, otherwise return \perp .

Figure 8.10: BSV: The Batched version of SV algorithm for Sonic.

Batched SV Algorithm for Sonic

Fig. 8.10, describes the batched version of Sonic's SV algorithm (given in Figure 8.3). Using the proposed BSV algorithm, to verify an i -time updated SRS of size n : 1) a prover will compute 7 pairings, $4n$ exponentiations in \mathbb{G}_2 , and $8n - 1$ exponentiations in \mathbb{G}_1 , 2) and a verifier would need to calculate $4i + 14$ pairings, $6n + 2i + 1$ exponentiations in \mathbb{G}_2 , and $8n + 8i + 2$ exponentiations in \mathbb{G}_1 .

Batched SV Algorithm for Marlin.

Our proposed BSV algorithm for Marlin is described in Fig. 8.11. Using the BSV algorithm, to verify an i -time updated SRS of size n : 1) a prover will compute 3 pairings and $4n$ exponentiations (mostly, short exponent) in \mathbb{G}_1 , 2) and a verifier would need to calculate $4i + 9$ pairings, $3i + 2$ exponentiations in \mathbb{G}_2 , and $4n + 6i + 2$ exponentiations in \mathbb{G}_1 .

Batched SV Algorithm for LunarLite.

The batched SV algorithm for LunarLite is shown in Figure 8.12. Using the proposed BSV algorithm, to verify an i -time updated SRS of size n , 1) a P will compute 3 pairings, $n - 1$ exponentiations in \mathbb{G}_2 , and $2n - 2$ exponentiations in \mathbb{G}_1 , 2) and a V would need to compute $i + 3$ pairings, $n + i$ exponentiations in \mathbb{G}_2 , and $2n + 3i$ exponentiations in \mathbb{G}_1 .

Batched SV Algorithm for Basilisk and Plonk.

Similar to the previous cases, in Figure 8.13, we propose a probabilistic batched variant of the Basilisk's SV algorithm (given in Figure 8.9). A slightly modified version of this algorithm can be used for Plonk, just by removing the checks related to u_i . However, one should pay attention that Plonk and Basilisk are using two different constraint systems, and the value of n will be different once one uses the same BSV for both. In summary, by running the BSV algorithm given in Figure 8.13, to verify an i -time updated SRS of size n : 1) a prover will compute 2 pairings and $2n - 2$ exponentiations in \mathbb{G}_1 , 2) and a verifier would need to compute $i + 3$ pairings, i exponentiations in \mathbb{G}_2 , and $2n + 3i$ exponentiations in \mathbb{G}_1 .

Batched SRS Verification, $(\perp/1) \leftarrow \text{BSV}(\text{c}\tilde{\text{r}}s_i, (\Pi_j)_{j=0}^i, \text{party})$: To verify (an i -time updated) $\text{c}\tilde{\text{r}}s_i := (([x_i^k]_1, [\gamma_i x_i^k]_1)_{k=0}^n, [x_i]_2, [x_i \gamma_i]_2)$, and $\Pi_j := (\Pi_j^{\text{Agg}}, \Pi_j^{\text{Ind}}) := (([\gamma_i]_1, [x_i \gamma_i]_1, [x_i]_2), ([\tilde{x}_i]_1, [\tilde{\gamma}_i]_1, [\tilde{x}_i]_2, [\tilde{x}_i \tilde{\gamma}_i]_2))$; for $j = 0, 1, \dots, i$:

If party = P:

1. Sample $\{t_{1,k}, t_{2,k}\} \leftarrow \mathbb{Z}_p^*$; $_{k=1}^n$;
2. Check if $([\gamma_i x_i]_1 + \sum_{k=1}^n (t_{1,k} \cdot [x_i^k]_1 + t_{2,k} \cdot [\gamma_i x_i^k]_1)) \bullet [1]_2 = (\sum_{k=1}^n (t_{1,k} \cdot [x_i^{k-1}]_1 + t_{2,k} \cdot [\gamma_i x_i^{k-1}]_1)) \bullet [x_i]_2 + [1]_1 \bullet [\gamma_i x_i]_2$;

If party = V:

- If $i = 0$: $\text{c}\tilde{\text{r}}s_0$ is sampled by verifier, and it does not need to be verified.
- If $i \geq 1$, act as follows,
 1. Sample $\{r_{1,j}, r_{2,j}, r_{3,j}, r_{4,j}\} \leftarrow \mathbb{Z}_p^*$; $_{j=0}^i$; and $\{t_k, \hat{t}_k\} \leftarrow \mathbb{Z}_p^*$; $_{k=1}^n$;
 2. Check if $[\gamma_0]_1 = [\tilde{\gamma}_0]_1$ and $[x_0]_2 = [\tilde{x}_0]_2$;
 3. Check if $(\sum_{j=0}^i r_{1,j} \cdot [\tilde{x}_j]_1) \bullet [1]_2 = [1]_1 \bullet (\sum_{j=0}^i r_{1,j} [\tilde{x}_j]_2)$;
 4. Check if $[1]_1 \bullet (\sum_{j=0}^i r_{2,j} \cdot [\tilde{\gamma}_j \tilde{x}_j]_2) = \sum_{j=0}^i (r_{2,j} [\tilde{\gamma}_j]_1 \bullet [\tilde{x}_j]_2)$;
 5. Check if $[1]_1 \bullet (\sum_{j=1}^i r_{3,j} \cdot [x_j]_2) = \sum_{j=1}^i (r_{3,j} \cdot [\tilde{x}_j]_1 \bullet [x_{j-1}]_2)$;
 6. Check if $(\sum_{j=1}^i r_{4,j} \cdot [\gamma_j x_j]_1) \bullet [1]_2 = \sum_{j=1}^i (r_{4,j} \cdot [\gamma_{j-1} x_{j-1}]_1 \bullet [\tilde{\gamma}_j \tilde{x}_j]_2) = \sum_{j=1}^i (r_{4,j} \cdot [\gamma_j]_1 \bullet [x_j]_2)$;
 7. Check if $([\gamma_i x_i]_1 + \sum_{k=1}^n (t_k \cdot [x_i^k]_1 + \hat{t}_k \cdot [\gamma_i x_i^k]_1)) \bullet [1]_2 = (\sum_{k=1}^n (t_k \cdot [x_i^{k-1}]_1 + \hat{t}_k \cdot [\gamma_i x_i^{k-1}]_1)) \bullet [x_i]_2 + [1]_1 \bullet [\gamma_i x_i]_2$;

Return 1 if all the checks passed, otherwise return \perp .

Figure 8.11: BSV: The Batched version of SV algorithm for Marlin.

8.5 Performance Analysis and Identifiable Security

8.5.1 Implementation Results

Next, we evaluate the efficiency of the presented algorithms using a prototype implementation in Arkworks library¹, which is a Rust library for developing and programming with zk-SNARKs. We have made the source code of our benchmarks publicly available to the research community for reproducibility

¹Available on <https://github.com/arkworks-rs>.

Batched SRS Verification, $(\perp/1) \leftarrow \text{BSV}(\text{c}\vec{r}s_i, (\Pi_j)_{j=0}^i, \text{party})$: To verify (an i -time updated) $\text{c}\vec{r}s_i := ([x_i^k]_1, [x_i^k]_2)_{k=1}^n$, and $\Pi_j := (\Pi_j^{\text{Agg}}, \Pi_j^{\text{Ind}}) := ([x_j]_1, ([\bar{x}_j]_1, [\bar{x}_j]_2))$ for $j = 0, 1, \dots, i$:

If party = P:

1. Sample $t_k \leftarrow \mathbb{Z}_p^*$ for $k = 2, \dots, n$;
2. Check if $([x_i]_1 + \sum_{k=2}^n t_k \cdot [x_i^k]_1) \bullet [1]_2 = [1]_1 \bullet ([x_i]_1 + \sum_{k=2}^n t_k \cdot [x_i^k]_2) = ([1]_1 + \sum_{k=2}^n t_k [x_i^{k-1}]_1) \bullet [x_i]_2$;

If party = V:

- If $i = 0$: $\text{c}\vec{r}s_0$ is sampled by verifier, and it does not need to be verified.
- If $i \geq 1$, act as follows,
 1. Sample $t_j, s_j \leftarrow \mathbb{Z}_p^*$ for $j = 1, \dots, i$; and $h_k \leftarrow \mathbb{Z}_p^*$ for $k = 1, \dots, n$;
 2. Check if $[x_0]_1 = [\bar{x}_0]_1$;
 3. Check if $([\bar{x}_0]_1 + \sum_{j=1}^i (t_j [\bar{x}_j]_1 + s_j [x_j]_1) + 2 \sum_{k=1}^n h_k [x_i^k]_1) \bullet [1]_2 = [1]_1 \bullet ([\bar{x}_0]_2 + \sum_{j=1}^i t_j [\bar{x}_j]_2 + \sum_{k=1}^n h_k [x_i^k]_2) + \sum_{j=1}^i (s_j [x_{j-1}]_1 \bullet [\bar{x}_j]_2) + (\sum_{k=1}^n h_k [x_i^{k-1}]_1) \bullet [x_i]_2$

Return 1 if all the checks passed, otherwise return \perp .

Figure 8.12: BSV: The Batched version of SV algorithm for LunarLite.

and further experimentation ². For benchmarking Sonic, Plonk, LunarLite, and Basilisk we use the algorithms constructed in Sec. 8.3 and Sec. 8.4. But in case of Marlin, we use a variant of it, which is implemented in Arkworks ³. The original paper does not explain this variant, which uses a different PC scheme to reduce proof size, which is a variant of the scheme proposed in [Gab19]. We built the associated (SG, SU, SV, BSV) algorithms for that version in the full version [BMS23].

Our empirical analysis are done with the elliptic curves BLS12-381 that is estimated to achieve between 117 and 120 bits security [NCC19]. All experiments are done on a desktop machine with Ubuntu 20.4.2 LTS, an Intel Core i9-9900 processor at base frequency 3.1 GHz, and 128GB of memory. All algorithms first are executed in the single-thread mode, while later we show that they all can be parallelized and executed in the multi-thread mode. We also report the benchmarks for Basilisk in the multi-thread mode, with 16 threads. For the

²Our open-source implementations can be accessed on the Git page at <https://github.com/Baghery/BMS23>.

³Available on <https://github.com/arkworks-rs/marlin>.

Batched SRS Verification, $(\perp/1) \leftarrow \text{BSV}(\text{c}\vec{r}s_i, (\Pi_j)_{j=0}^i, \text{party})$: To verify (an i -time updated) $\text{c}\vec{r}s_i := (([x_i^k]_1)_{k=1}^n, [x_i]_2, u_i)$, and $\Pi_j := (\Pi_j^{\text{Agg}}, \Pi_j^{\text{Ind}}) := ([x_j]_1, ([\bar{x}_j]_1, [\bar{x}_j]_2))$ for $j = 0, 1, \dots, i$:

If party = P:

1. Sample $t_k \leftarrow \mathbb{Z}_p^*$ for $k = 2, \dots, n$;
2. Check if $([x_i]_1 + \sum_{k=2}^n t_k \cdot [x_i^k]_1) \bullet [1]_2 = ([1]_1 + \sum_{k=2}^n t_k [x_i^{k-1}]_1) \bullet [x_i]_2$;
3. Check if $u_i \in \mathbb{Z}_p^*$;

If party = V:

- If $i = 0$: $\text{c}\vec{r}s_0$ is sampled by verifier, and it does not need to be verified.
- If $i \geq 1$, act as follows,
 1. Sample $t_j, s_j \leftarrow \mathbb{Z}_p^*$ for $j = 1, \dots, i$; and $h_k \leftarrow \mathbb{Z}_p^*$ for $k = 1, \dots, n$;
 2. Check that $[x_0]_1 = [\bar{x}_0]_1$;
 3. Check if $([\bar{x}_0]_1 + \sum_{j=1}^i (t_j [\bar{x}_j]_1 + s_j [x_j]_1) + \sum_{k=1}^n h_k [x_i^k]_1) \bullet [1]_2 = [1]_1 \bullet ([\bar{x}_0]_2 + \sum_{j=1}^i t_j [\bar{x}_j]_2) + \sum_{j=1}^i (s_j [x_{j-1}]_1 \bullet [\bar{x}_j]_2) + (\sum_{k=1}^n h_k [x_i^{k-1}]_1) \bullet [x_i]_2$;
 4. Check if $u_i \in \mathbb{Z}_p^*$;

Return 1 if all the checks passed, otherwise return \perp .

Figure 8.13: The BSV algorithm for Basilisk (and Plonk without checking u_i).

benchmarks, we report the running times of all the proposed algorithms, for an arithmetic circuit with different circuit sizes, and by circuit size we mean sum of the multiplication and the addition gates. For Plonk, whose constraint system encodes both multiplication and addition gates, we set the number of addition gates $2 \times$ the number of multiplication gates. This choice was based on the evaluation done in the original paper [GWC19]. Motivated by the blockchains and large-scale applications, we also report the SRS verification/updating times for a big number of users and large circuits. All times are expressed in seconds or minutes. In the execution of the BSV algorithms, we first sample some vectors \vec{t}_i of random numbers from the range $[1..2^{80}]$, the time of sampling randomnesses are not included in the run times of BSV algorithm, as they can be pre-computed. Based on earlier results, one can re-use the same randomness for different verification equations, and zk-SNARKs.

The graphs in Figure 8.14 summarize our implementation results based on different criteria for all our studied zk-SNARKs. In the rest, we go through them sequentially and explain the key points. The plot A compares the run

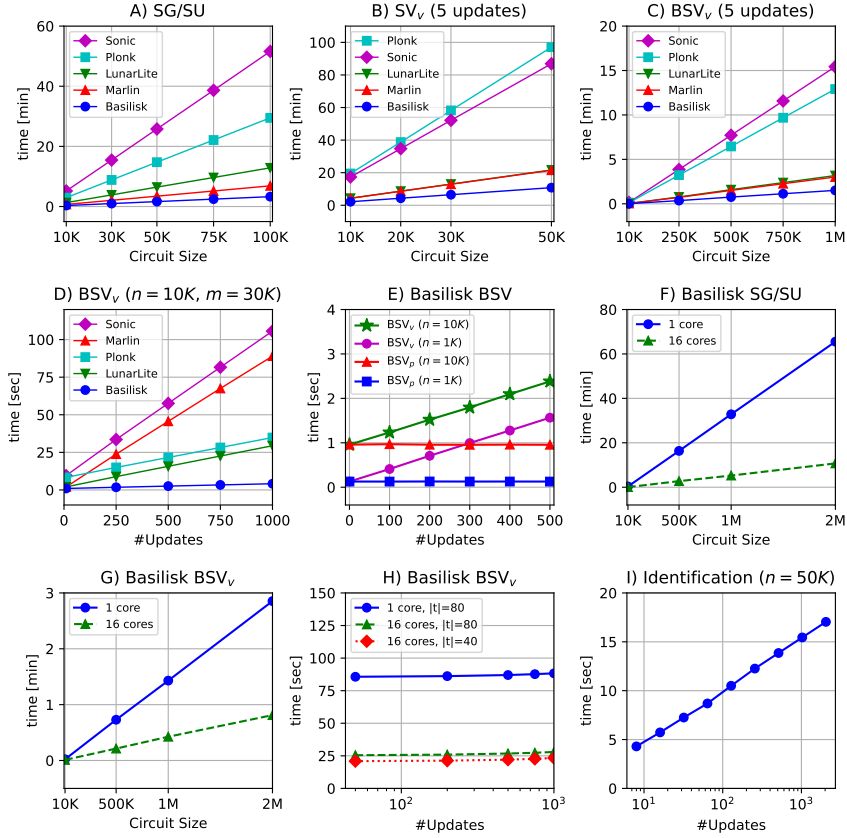


Figure 8.14: A) SG or SU, B) SV_V for a fixed $i = 5$, C) BSV_V for a fixed $i = 5$, D) BSV_V for a fixed circuit size ($n = 10K, m = 30K$), E) Comparison of Basilisk's BSV_P and BSV_V for a fixed $n = 10K$, F) Basilisk's SG/SU with multi-threading, G) Basilisk's BSV_V with multi-threading, H) Basilisk's BSV_V with $n = 10^6$, and different security parameters in batching, I) Identifying a malicious SRS updater with recursive verification in Basilisk.

times of SG and SU in the single-thread mode, for all the studied zk-SNARKs. Naturally, the shorter SRS, the faster SG and SU algorithms. The plot B presents the run times of SV algorithm executed by V, for a 5-time updated SRS and various circuit sizes. As it can be seen, standard SV algorithms can be very slow for even small circuits, e.g. circuits of sizes $< 50K$ (this is why we are not giving its timings for $n > 50K$). In this case, since the size of SRS

($n = 50K$), is considerably larger than the number of updates ($i = 5$), then the run times of SV_P and SV_V are almost the same, therefore SV_P is omitted from the plot.

The plot C illustrates the efficiency of BSV algorithm run by V, for $i = 5$ (5-time updated SRS) and different circuit sizes. One can see that they are considerably faster than standard SV algorithms, and in some cases they are very efficient even for large circuits. e.g. circuits of sizes $> 1M$. Similar to the last plot, in this setting again the run times of SV_P and SV_V are very close. In plot D, we set the circuit size fixed ($n = 10K$ multiplication gates, $m = 30K$ total gates) and plot the run times of BSV_V algorithms for different number of updates. Similar to the previous plots, we observe that the setup phase of Basilisk can be considerably faster than other schemes. Therefore, in the rest of benchmarks, we mainly used Basilisk's algorithms. In plot E, we compare the run times of Basilisk's BSV algorithm executed by the prover (BSV_P) and verifier (BSV_V), for a circuit with $n = 1K$ or $10K$ multiplication gates, and different numbers of updates. As it can be seen, for the cases that a small circuit is updated many times, BSV_P can be significantly faster, independent of the number of updates. The plot shows that BSV_P for $n = 10K$, is as efficient as BSV_V for $n = 1K$ and $i \approx 300$.

By now all evaluations are done in the single-thread mode. In the rest, in both plots F and G, we execute the algorithms of Basilisk in the multi-thread mode and re-evaluate the efficiency of SU (or SG) and BSV_V , for various circuits and different number of updates. We observed that, the SRS of Basilisk for a particular circuit with 2M multiplication gates, can be generated/updated in about 11 min, and verified in less than 1 minute. As mentioned before, within the BSV algorithms, the randomness vector \vec{t}_i are sampled from $[1 .. 2^{80}]$ which assures that the batching causes security gap not bigger than 2^{-80} . This is a conservative approach. In plot H, we compare the run times of BSV_V for Basilisk in the case that the coordinates of \vec{t}_i were chosen from $[1 .. 2^{40}]$. This makes the SRS verification even faster, but at the cost of a bigger error rate, i.e., 2^{-40} .

8.5.2 Identifiable Security in the Updatable SRS Model

In the updatable SRS model [Gro+18], the initial SRS generator and the follow-up SRS updaters attach a proof to each updated SRS, and the parties do not store every updated SRS but only update proofs. At the end, each party runs the SV (or BSV) algorithm once to verify the validity of proofs in a chain and then uses the final proof to check the well-formedness of the final SRS (see Fig. 8.1). In lemmas 3-10, we also observed that after the final update on SRS,

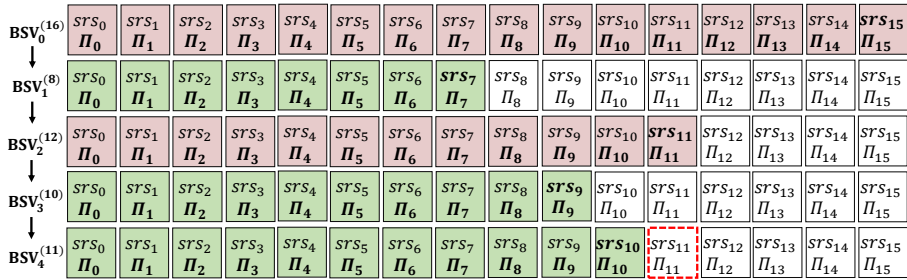


Figure 8.15: Recursive execution of BSV to identify a malicious SRS updater.

it is sufficient that all the participants in the SRS generation/updating phases run the SV (or BSV) algorithm only once. In the rest, this case is referred to as the *optimistic case* or *optimistic verification*. As we observed in Figure 8.14, in this case the setup phase of updatable zk-SNARKs can be significantly fast, and can easily be scaled for a large number of users (e.g. thousands of parties), without the need for a third party. However, then the parties would only abort a maliciously updated SRS at the end, without identifying a malicious party. This can lead to repeat the SRS generation and updates all over again. Note that, the SV (and BSV) algorithm verifies the proofs Π_0 till Π_i , and the final SRS $c\dot{r}s_i$. If a malicious SRS generator/updater generates a valid proof but an invalid SRS, it cannot be detected by just verifying the proofs. To deal with this concern, a naive solution is to verify the SRS after each update (by either all the participants or a TTP) and identify the malicious party. In practice, the above approach would be impractical for large scale applications.

Identifying a Malicious Updater with Logarithmic Verification. Next, we describe an efficient approach to identify a malicious party in the updatable SRS model. To this end, the parties need to store all the transcripts, as in current ceremonies, and then recursively run the BSV (or SV) algorithm for one SRS and a smaller set of proofs. More precisely, parties would run the BSV (or SV) algorithm of the target zk-SNARK, with a single SRS and $\frac{i}{2^1}, \frac{i}{2^2}, \dots, \frac{i}{2^{\log i}}$ proofs, respectively. Note that with this approach, only $\lceil \log i \rceil + 1$ of SRSs are verified (e.g., boldface SRSs $c\dot{r}s_{15}, c\dot{r}s_7, c\dot{r}s_{11}, c\dot{r}s_9, c\dot{r}s_{10}$ in Figure 8.15), instead of i . As in practice, the circuit size is considerably higher than the number of SRS updates, e.g. 2^{22} vs. 100 in current ceremonies, therefore the run time of SV (and BSV) is dominated by the size of SRS, rather than the number of updates. Due to this fact, in practice, the proposed verification approach can be considerably faster than the naive solution. Fig. 8.15, presents an example of such recursive execution of BSV algorithms for $i = 15$. We also evaluate

the performance of this approach with a sample implementation. The plot in Figure 8.14, illustrates the required time to identify a malicious updater in Basilisk's setup for different number of updates with the SRS of a circuit with $n = 50K$ multiplication gates. As it can be seen, for 2000-time updated SRS of length $50K$, the first malicious updater can be identified in less than 20 sec. In similar settings, where $n \gg i$, the identification time would be independent of the precise position of the malicious updater, and it will take an approximate run time of $\log i$ times that of a single BSV.

As an optimization, one may notice that once a verifier runs the BSV algorithm on the final SRS, e.g. $\text{BSV}_0^{(16)}$ in the mentioned example, we already compute the batched form of the proof elements required in all the follow-up steps of the recursive search, as e.g. $\sum_{i=0}^{15} t_i [x_i]_1 = \sum_{i=0}^7 t_i [x_i]_1 + \sum_{i=8}^{15} t_i [x_i]_1 = \sum_{i=0}^3 t_i [x_i]_1 + \sum_{i=4}^7 t_i [x_i]_1 + \sum_{i=8}^{11} t_i [x_i]_1 + \sum_{i=12}^{15} t_i [x_i]_1$. By storing a proper set of batched proofs, one can speed up the follow-up executions of BSV. This optimization is more effective in cases that the circuit size is small but the SRS is updated many times. As another optimization, one can precompute the batched version of the checks on some intermediate SRSs, e.g. $\text{c}\vec{\text{r}}\text{s}_{11}, \text{c}\vec{\text{r}}\text{s}_7, \text{c}\vec{\text{r}}\text{s}_3$, and speed-up the run times of BSV algorithms in the follow-up steps. Note that our BSV and SV algorithms, by default verifies all the proofs for $j = 0$ till the final SRS $\text{c}\vec{\text{r}}\text{s}_i$, i.e. $j = i$. In the recursive execution, we need to run the BSV (or SV) algorithm for a particular set of SRSs and proofs. In those cases, one can feed proper starting and finishing indexes to the BSV (or SV) algorithms. For instance, to check the SRS $\text{c}\vec{\text{r}}\text{s}_{11}$ and the set of proofs $\{\Pi_8, \Pi_9, \Pi_{10}, \Pi_{11}\}$ one needs to run the algorithms for $j = 8$ till $j = 11$, which will verify a batched variant of $(\Pi_8, \Pi_9, \Pi_{10}, \Pi_{11})$ and the final SRS $\text{c}\vec{\text{r}}\text{s}_{11}$.

In practice, if the values of i and n will be huge, it might happen that the setup phase would take a long time, especially if a malicious update occurs during the earlier updates. To minimize the run time, as well as to gain the benefits of the optimistic verification, an effective solution would be to verify the updated SRS after a particular number of updates, i.e. one would need to verify the updated SRS every k updates, where $1 < k < i$. Basically, the idea is rather than verifying every update (the slowest case), or all i updates once (the fastest case), the parties will verify the SRS after each k updates. If the verification of $\text{c}\vec{\text{r}}\text{s}_k$ was successful, then the parties will continue with updating the SRS. If not, they would use the recursive search approach (given in Figure 8.15) to find the first malicious updater and then will continue the SRS update from there (without the malicious updater).

Since the entire described procedure is accountable, in practice one can minimize the risk of a malicious SRS update significantly by enforcing a high penalty for a malicious SRS updater.

8.6 Conclusion

In this study, we examined the setup phase of updatable zk-SNARKs. We constructed the necessary algorithms, namely (SG, SU, SV), for the setup phase of various updatable zk-SNARKs, including Sonic, Plonk, Marlin, Lunar, and Basilisk. To make SV algorithms practical, we also presented a batched version of them, called BSV. We constructed the algorithms for the most efficient version of each zk-SNARK, in terms of proof size. However, the proposed algorithm can be adapted to their different versions. Our results show that in a few cases, to achieve better efficiency in the setup phase, one option would be to use a version of the studied schemes, with a shorter SRS but slightly larger proofs and slower provers. For instance, Lunar [Cam+21] has a version, so-called LunarLite2x, which has the same SRS as Basilisk, therefore can be as efficient as Basilisk in the setup phase, but in cost of slightly longer proofs and slower prover. In another example, we observed that Counting Vampires [LSZ22] has only two fewer group elements than Basilisk in the proof, but its SRS size is $17\times$ larger and such an SRS can result in a prolonged setup.

Meanwhile, we observed that to achieve Sub-ZK/Upd-KS in updatable zk-SNARKs, a more realistic model for security proofs could be the AGM with hashing [Lip22], rather than the original AGM [FKL18].

Moreover, we showed that pairing-based updatable zk-SNARKs, or other primitives constructed in the updatable SRS model, by default achieve security with abort, and the parties cannot identify a malicious SRS generator/updater. A naive solution to deal with this concern is verifying the SRS after each update (either by the parties or a third party), but it can be impractical in a large-scale application. To make it practical, we proposed an efficient recursive verification approach, that allows the parties to identify a malicious SRS updater by a logarithmic number of SRS verification (instead of linear) in the number of updates. We believe our proposed approach to achieve identifiable security, can also be used in the MPC SRS generation protocols [Koh+21], as well as in other cryptographic primitives (like commitments, signatures, encryptions) constructed in the updatable SRS model [DRZ20; ARS20; BS21; Gan+22; BB22].

Finally, our empirical analysis showed that the algorithms are practical for large-scale applications, and among the current updatable zk-SNARKs, Basilisk (and the LunarLite2x variant of Lunar) can have the fastest setup phase. Counting Vampires, Sonic and Plonk can have a very slow setup phase, which is mainly because of having a very long SRS or using a specific constraint system (i.e., Plonk) that encodes both addition and multiplication gates.

Tiramisu: Black-Box Simulation Extractable NIZKs in the Updatable CRS Model

Source. K. Bagheri and M. Sedaghat. “Tiramisu: Black-Box Simulation Extractable NIZKs in the Updatable CRS Model”. In: *Cryptology and Network Security (CANS)*. ed. by M. Conti, M. Stevens, and S. Krenn. Cham: Springer International Publishing, 2021, pp. 531–551. ISBN: 978-3-030-92548-2. DOI: [10.1007/978-3-030-92548-2_28](https://doi.org/10.1007/978-3-030-92548-2_28)

Abstract. Zk-SNARKs, as the most efficient NIZK arguments in terms of proof size and verification, are ubiquitously deployed in practice. In applications like Hawk [S&P’16], Gyges [CCS’16], Ouroboros Cryptsinous [S&P’19], the underlying zk-SNARK is lifted to achieve Black-Box Simulation Extractability (BB-SE) under a trusted setup phase. To mitigate the trust in such systems, we propose TIRAMISU¹, as a construction to build NIZK arguments that can achieve *updatable BB-SE*, which we define as a new variant of BB-SE. This new variant allows *updating* the public parameters, therefore eliminating the need for a trusted third party, while unavoidably relies on a *non-black-box* extraction algorithm in the setup phase. In the cost of one-time individual CRS update by the parties, this gets around a known impossibility result by Bellare et al. from ASIACRYPT’16, which shows that BB extractability cannot be achieved with subversion ZK (ZK without trusting a third party). TIRAMISU uses an efficient public-key encryption with updatable keys which may be of independent interest. We instantiate TIRAMISU, implement the overhead and present efficient BB-SE zk-SNARKs with updatable parameters that can be used in various applications while allowing the end-users to update the parameters and eliminate the needed trust.

¹In Italian, TIRAMISU literally means “lift me up”.

9.1 Introduction

Zero-Knowledge (ZK) [GMR85] proof systems, particularly Non-Interactive Zero-Knowledge (NIZK) arguments [BFM88] are one of the elegant tools in modern cryptography that due to their impressive advantages and practical efficiency, they are ubiquitously deployed in practical applications [Ben+14; Kos+16; JKS16; Ker+19]. A NIZK proof system allows a party P (called prover) to non-interactively prove the truth of a statement to another party V (called verifier) without leaking any information about his/her secret inputs. For instance, they allow a prover P to convince a verifier V that for a (public) statement x , he/she knows a (secret) witness w that satisfies a relation R , $(x, w) \in R$, without leaking any information about w .

Typically, a NIZK argument is expected to satisfy, (i) Completeness, which implies that an honest prover always convinces an honest verifier (ii) Soundness, which ensures that an adversarial prover cannot convince an honest verifier except with negligible probability. (iii) Zero-Knowledge (ZK), which guarantees that an honestly generated proof does not reveal any information about the (secret) witness w . In practice, it is shown that bare *soundness* is not sufficient and it needs either to be amplified [Kos+16] or the protocol needs to be supported by other cryptographic primitives [Ben+14]. To deal with such concerns, different constructions are proposed that either satisfy one of the following notions, one of which is an amplified variation of soundness. (iv) Simulation Soundness (SS), which ensures that an adversarial prover cannot convince an honest verifier, even if he has seen polynomially many simulated proofs (generated by Sim), except with negligible probability. (v) Knowledge Soundness (KS), which guarantees that an adversarial prover cannot convince an honest verifier, unless he *knows* a witness w for statement x such that $(x, w) \in R$. (vi) Simulation Extractability (SE) (a.k.a. *Simulation Knowledge Soundness*), which guarantees that an adversarial prover cannot convince an honest verifier, even if he has seen polynomially time simulated proofs, unless he *knows* a witness w for statement x .

The term “*knowledge*” in KS (in item [v](#)) and SE (in item [vi](#)) means that a successful prover should *know* a w . *knowing* is formalized by showing that there exists an algorithm Ext , which can extract the witness w (from the prover or proof) in either *non-Black-Box* (nBB) or *Black-Box* (BB) manner. Typically, nBB extraction can result in more efficient constructions, as it allows $\text{Ext}_{\mathcal{A}}$ to get access to the source-code and random coins of the adversary \mathcal{A} . Although the constructions that obtain BB extractability are less efficient, they provide stronger security guarantees, as it allows us to have a universal extractor Ext for *any* \mathcal{A} . The term “*simulation*” in notions SS (in item [iv](#)) and SE (in item [vi](#))

guarantees that the proofs are non-malleable and an adversary cannot change an old (simulated) proof to a new one such that V accepts it. The notion SE provides the strongest security and also implies non-malleability of proofs as defined in [De +01]. Moreover, it is shown [Gro06] that SE is a sufficient requirement for a NIZK argument to be deployed in a Universally Composable (UC) protocol [Can01].

zk-SNARKs. In the Common Reference String (CRS) model [BFM88], NIZK arguments require a trusted setup phase. Based on the underlying assumptions, they are constructed either using falsifiable or non-falsifiable assumptions [Nao03]. At the beginning of the last decade, a line of research initiated that focused on constructing NIZK arguments with succinct proofs, which finally led to an efficient family of NIZK arguments, called zero-knowledge Succinct Non-interactive ARGument of Knowledge (zk-SNARK) [Gro10; Lip12; Par+13; Ben+13; Gro16; GM17; BG18], [Lip19; BPR20]. zk-SNARKs are constructed based on knowledge assumptions [Dam92] that allow *succinct* proofs and nBB extractability. Gentry and Wichs’s impossibility result [GW11] confirmed that *succinct* proofs cannot be built based on falsifiable assumptions. Beside *succinct* proofs, all initial zk-SNARKs were designed to achieve completeness, ZK and KS (in item v) [Gro10; Lip12; Par+13; Ben+13; Gro16]. KS proofs are malleable, thus in practice users needed to make extra efforts to guarantee the non-malleability of proofs [Ben+14]. Following this concern, in 2017, Groth and Maller [GM17] presented a zk-SNARK that can achieve SE (in item vi) with nBB extractability, consequently generates non-malleable proofs. Recent works in this direction have led to more efficient schemes with the same security guarantees [BG18; Lip19; Bag+21; BPR20].

Mitigating the trust in the setup phase of zk-SNARKs. In 2016, Bellare et al. [BFS16] studied the security of NIZK arguments in the face of subverted CRS. They defined (vii) *Subversion-Soundness* (Sub-SND), which ensures that the protocol guarantees soundness even if \mathcal{A} has generated the CRS, and (viii) *Subversion-ZK* (Sub-ZK), which ensures that the scheme achieves ZK even if \mathcal{A} has generated the CRS. Then, they showed that Sub-SND is not achievable with (standard) ZK, and also we cannot achieve Sub-ZK along with BB extractability. Two follow-up works [Abd+17; Fuc18] showed that most of zk-SNARKs can be lifted to achieve Sub-ZK (in item viii) and KS with nBB extraction (nBB-KS). Bagheri [Bag19b] showed that using the folklore OR technique [BG90] any Sub-ZK SNARK can be lifted to achieve Sub-ZK and SE (in item vi) with nBB extraction (nBB-SE). Meanwhile, as an extension to the MPC approach [Ben+15] and subversion security, in 2018 Groth et al. [Gro+18] introduced a new variation of the CRS model, called *updatable* CRS model which allows both prover and verifier to update the CRS and bypass the needed trust in a third party. Groth et al. first defined, (ix) *Updatable KS* (U-KS),

which ensures that the protocol guarantees KS (in item **v**) as long as the initial CRS generation or one of CRS updates is executed honestly, and (x) *Updatable ZK (U-ZK)*, which ensures that the protocol guarantees ZK as long as the initial CRS generation or one of CRS updates is done by an honest party². Then, they presented a zk-SNARK that can achieve Sub-ZK and U-KS with nBB extraction (U-nBB-KS). Namely, the prover achieves ZK without trusting the CRS generator and the verifier achieves nBB-KS without trusting the CRS generator but by one-time CRS updating. Recent constructions in this direction have better efficiency [Mal+19; GWC19]. Recently, Abdolmaleki, Ramacher, and Slamanig [ARS20] presented a construction, called LAMASSU, and showed that using a similar folklore OR technique [BG90; DS16; Bag19b] any zk-SNARK that satisfies Sub-ZK and U-nBB-KS can be lifted to achieve Sub-ZK and U-nBB-SE. (xi) *Updatable nBB-SE (U-nBB-SE)*, which ensures that the protocol achieves SE with nBB extraction as long as the initial CRS generation or one of CRS updates is done honestly. Recently, it is shown that two efficient updatable universal zk-SNARKs Plonk [GWC19] and Sonic [Mal+19] can also achieve U-nBB-SE [Gan+22]. Considering the impossibility of achieving Sub-ZK along with BB extraction [BFS16], such schemes [ARS20; GWC19; Mal+19] achieve the strongest notion with nBB extraction.

Using zk-SNARKs in UC-Protocols. A UC protocol [Can01] does not interfere with other protocols and can be arbitrarily composed with other protocols. In 2006, Groth [Gro06] showed that a NIZK argument that can achieve BB-SE can realize the ideal NIZK-functionality $\mathcal{F}_{\text{NIZK}}$ [GOS06]. In 2015 Kosba et al. [Kos+15] proposed a framework called $\mathcal{C}\mathcal{O}\mathcal{C}\mathcal{O}$ along with several constructions that allows lifting a sound NIZK argument to a BB-SE NIZK argument, such that the lifted version can be deployed in UC-protocols. In summary, given a sound NIZK argument for language \mathbf{L} , the $\mathcal{C}\mathcal{O}\mathcal{C}\mathcal{O}$ defines a new extended language $\hat{\mathbf{L}}$ appended with some primitives and returns a NIZK argument that can achieve BB-SE. We review the strongest construction of the $\mathcal{C}\mathcal{O}\mathcal{C}\mathcal{O}$ in full version [BS20].

Unfortunately, the default security of zk-SNARKs is insufficient to be directly deployed in UC protocols. The reason is that zk-SNARK achieves nBB extraction and the extractor $\text{Ext}_{\mathcal{A}}$ requires access to the source code and random coins of \mathcal{A} , while in UC-secure NIZK arguments, the simulator of *ideal-world* should be able to simulate corrupted parties. To do so, the simulator should be able to extract witnesses without getting access to the source code of the environment's algorithm. Due to this fact, all those UC-secure applications that use zk-SNARKs [Kos+16; JKS16; Ker+19], use $\mathcal{C}\mathcal{O}\mathcal{C}\mathcal{O}$ to lift the underlying zk-SNARK to achieve BB-SE, equivalently UC-security [Gro06]. Note that the

²Sub-ZK is a stronger notion than U-ZK, as in Sub-ZK \mathcal{A} has generated the CRS, while the later achieves ZK if at least one of CRS updates is done honestly.

lifted zk-SNARKs that achieve BB-SE are not *witness* succinct any more, but they still are *circuit* succinct.

9.1.1 Our Contributions

Tiramisu. The core of our results is presenting TIRAMISU as an alternative to the $\mathsf{C}\mathsf{C}\mathsf{C}$ framework but in the *updatable* CRS model. Technically speaking, TIRAMISU allows one to build simulation extractable NIZK arguments with updatable parameters that satisfies a variant of black-box extractability which we define in this work. In the NIZK arguments built with TIRAMISU the parties can update the CRS themselves instead of trusting a third party. The construction is suitable for modular use in larger cryptographic protocols, which aim to build SE NIZK arguments with BB extractability, while avoiding to trust a third party.

To construct TIRAMISU, we start with the $\mathsf{C}\mathsf{C}\mathsf{C}$'s strongest construction and lift it to a construction that works in the updatable CRS model. Meanwhile, to attain fast practical performance, we consider the state-of-the-art constructions proposed in the updatable CRS model and show that we can simplify the construction of $\mathsf{C}\mathsf{C}\mathsf{C}$ and still achieve the same goal, particularly in the updatable CRS model. Technically speaking, the strongest construction of the $\mathsf{C}\mathsf{C}\mathsf{C}$ framework, gets a sound NIZK argument for the language \mathbf{L} and lifts it to a new NIZK argument for the extended language $\hat{\mathbf{L}}$, that can achieve BB-SE. The language $\hat{\mathbf{L}}$ is an extension of \mathbf{L} appended with some necessary and sufficient primitives, including an encryption scheme to encrypt the witness and a Pseudo-Random Function (PRF) along with a commitment scheme that commits to the secret key of the PRF (more details in full version [BS20] and Sec. 9.4). In composing TIRAMISU, we show that considering recent developments in building NIZK arguments with updatable CRS, namely due to the existence of nBB-SE NIZK arguments with updatable CRS (either with a two-phase updatable CRS [Gro16; BGM17; BG18; Bag+21; BPR20] or with a universal updatable string [Gro+18; ARS20; GWC19; Mal+19]) we can simplify the definition of $\hat{\mathbf{L}}$ by removing the commitment and PRF and construct more efficient SE NIZK arguments with (a variant of) BB extractability that also have *updatable* CRS. We show that, TIRAMISU also can be added as a layer on top of the construction proposed in [ARS20], called LAMASSU, and together act as a generic compiler to lift any sound NIZK argument to a SE NIZK argument with a variant of black-box extractability in the updatable CRS model. However, we show that the arguments built with this approach are inefficient in comparison with the ones built with only TIRAMISU. Fig. 9.1 illustrates how one can use $\mathsf{C}\mathsf{C}\mathsf{C}$ and TIRAMISU to build BB-SE NIZK arguments in the *standard* and *updatable* CRS models, respectively. Similar to $\mathsf{C}\mathsf{C}\mathsf{C}$ framework, TIRAMISU results in

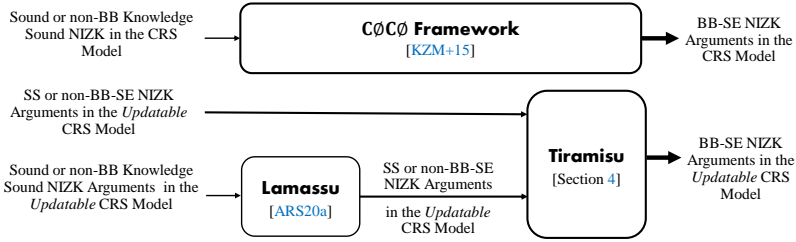


Figure 9.1: Using C0C0 and TIRAMISU to build BB-SE NIZK arguments in the *standard* and *updatable* CRS models.

NIZK arguments whose proof size and verification time are (quasi-)linear in the *witness* size, that is an unavoidable requirement for UC security [Can01], but still are independent of the size of the circuit, which encodes $\hat{\mathbf{L}}$.

Bellare et al.’s Negative Result. In [BFS16], Bellare et al. observed that achieving Sub-ZK and BB extractability is impossible at the same time. As BB extractability requires the simulator create a CRS with a trapdoor it withholds, then it can extract the witness from a valid proof. But Sub-ZK requires that even if \mathcal{A} generates the CRS, it should not be able to learn about the witnesses from the proof. However, if a NIZK argument achieves BB extractability, an adversary (CRS subverter) can generate the CRS like the simulator. So it has the trapdoor and can also extract the witness and break Sub-ZK. Considering the above negative result, TIRAMISU achieves the best possible combination with downgrading Sub-ZK (in item viii) to U-ZK (in item x) while achieving *updatable* BB extractability, either U-BB-SE or U-BB-KS. U-BB-SE and U-BB-KS does not need a trusted third party, therefore from the trust point of view, they are stronger definitions than *standard* BB-SE and BB-KS, respectively, which require a trusted setup phase. But, in definitions of U-BB-SE and U-BB-KS, to bypass the needed trust, we rely on the existence of a nBB extraction algorithm in the setup phase that can extract the trapdoors from the (malicious) parameter generator or updaters. This seems to be unavoidable fact to achieve updatability and BB extractability at the same time.

Key-Updatable Public-key Cryptosystems. TIRAMISU uses a semantically secure cryptosystem with *updatable keys* that we define here. We show that such cryptosystems can be built either in a generic manner from key-homomorphic encryption schemes [AHI11], or via an ad-hoc approach. Using both generic and ad-hoc approaches, we present two variations of El-Gamal cryptosystem [ELG84] instantiated in the pairing-based groups which fulfil the requirements of a

Table 9.1: A comparison of TIRAMISU with related works. ZK: Zero-knowledge, SE: Simulation Extractable, U: Updatable, S: Subversion, nBB: non-Black-Box, BB: Black-Box. ✓: Achieves, ×: Does not achieve.

	Zero-Knowledge			Simulation Extractability			
	ZK	U-ZK	S-ZK	nBB-SE	BB-SE	U-nBB-SE	U-BB-SE
TIRAMISU	✓	✓	×	✓	✓	✓	✓
C0C0 [Kos+15; Bag19a]	✓	×	×	✓	✓	×	×
[GM17; BG18; AB19]	✓	×	×	✓	×	×	×
[Bag19b; Lip19; BPR20]	✓	✓	✓	✓	×	×	×
[BGM17; BG18; ARS20]	✓	✓	✓*	✓	×	✓	×

*Theorem 4 in [ARS20] states LAMASSU, can achieve U-ZK and U-nBB-SE, but it can be shown that it can achieve Sub-ZK along with U-nBB-SE which is a stronger combination.

cryptosystem with updatable keys. Efficiency of both constructions are evaluated with a prototype implementation in the Charm-Crypto framework [Aki+13], and seem to be practical. The new syntax and constructions can be interesting in their own right, particularly for building other primitives in the updatable CRS model [CFQ19; Daz+19]. There are some related definitions for encryption schemes that support updating the keys [CHK03; Fau+19], however their definitions do not fit our requirements for distributing trust across multiple updaters in the updatable CRS model.

Tab. 9.1 compares NIZK arguments built with TIRAMISU with existing schemes that can achieve a flavour of SE and ZK. Schemes built with C0C0 achieve BB extractability, thus they cannot achieve S-ZK, and the constructions that achieve Sub-ZK [Bag19b; Lip19; ARS20] can achieve (U-)nBB-SE in the best case.

Road-map. The rest of the paper is organized as follows; Sec. 9.2 presents necessary preliminaries. Sec. 9.3 defines the syntax of a key-updatable cryptosystems and presents efficient variations of the El-Gamal cryptosystem as an instantiation. Our construction, TIRAMISU, and its security proofs are described in Sec. 9.4. In Sec. 9.5, we present U-BB-SE NIZK arguments built with TIRAMISU.

9.2 Notations

Throughout, we suppose the security parameter of the scheme be κ and $\nu(\kappa)$ denotes a negligible function. We use $x \leftarrow \$ X$ to denote x sampled uniformly according to the distribution X . Also, we use $[1..n]$ to denote the set of integers

in range of 1 to n . Let PPT and NUPPT denote probabilistic polynomial-time and non-uniform probabilistic polynomial-time, respectively. For an algorithm \mathcal{A} , let $\text{im}(\mathcal{A})$ be the image of \mathcal{A} , i.e., the set of valid outputs of \mathcal{A} . Moreover, assume $\text{RND}(\mathcal{A})$ denotes the random tape of \mathcal{A} , and $r \leftarrow \$ \text{RND}(\mathcal{A})$ denotes sampling of a randomizer r of sufficient length for \mathcal{A} 's needs. By $y \leftarrow \mathcal{A}(x; r)$ we mean given an input x and a randomizer r , \mathcal{A} outputs y . For algorithms \mathcal{A} and $\text{Ext}_{\mathcal{A}}$, we write $(y \parallel y') \leftarrow (\mathcal{A} \parallel \text{Ext}_{\mathcal{A}})(x; r)$ as a shorthand for “ $y \leftarrow \mathcal{A}(x; r)$, $y' \leftarrow \text{Ext}_{\mathcal{A}}(x; r)$ ”. Two computationally IND distributions A and B are shown with $A \approx_c B$.

We use additive and the bracket notation, i.e., in group \mathbb{G}_{μ} , $[a]_{\mu} = a[1]_{\mu}$, where $[1]_{\mu}$ is a generator of \mathbb{G}_{μ} . A *bilinear group generator* $\text{BGgen}(1^{\kappa})$ returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2)$, where p (a large prime) is the order of cyclic abelian groups \mathbb{G}_1 , \mathbb{G}_2 , and \mathbb{G}_T . Finally, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficient non-degenerate bilinear pairing, s.t. $\hat{e}([a]_1, [b]_2) = [ab]_T$. Denote $[a]_1 \bullet [b]_2 = \mathbf{E}([a]_1, [b]_2)$.

9.3 Public-Key Cryptosystems with Updatable Keys

As briefly discussed in Sec. 9.1, one of the key building blocks used in TIRAMISU is the cryptosystem schemes with updatable keys that we define next. Similar definitions are recently proposed for zk-SNARKs [Gro+18], and signatures [ARS20], but considering previous definitions in [CHK03; Fau+19], to the best of our knowledge this is the first time that this notion is defined for the public-key cryptosystems. In contrast to subversion-resilient encryption schemes [ABK18] that the key-generation phase might be subverted, here we consider the case that the output of the key-generation phase is updatable and parties can update the keys. We aim to achieve the standard security requirements of a cryptosystem as long as either the original key generation or at least one of the updates was done honestly. Similar to the case on pairing-based subversion resistant NIZK arguments [BFS16], we assume that the group generator is a deterministic polynomial time algorithm, which given the security parameter, it can be run by every entity without the need for a trusted third party.

9.3.1 Definition and Security Requirements

Definition 59 (Cryptosystems with Updatable Keys). A public-key cryptosystem Ψ_{Enc} with updatable keys over the message space \mathcal{M} and ciphertext space \mathcal{C} , consists of five PPT algorithms (KG, KU, KV, Enc, Dec), defined as follows,

- $(pk_0, \Pi_{pk_0}, sk_0) \leftarrow KG(1^\kappa)$: Given the security parameter 1^κ returns the corresponding key pair (pk_0, sk_0) and Π_{pk_0} as a proof of correctness.
- $(pk_i, \Pi_{pk_i}) \leftarrow KU(pk_{i-1})$: Given a valid (possibly updated) public key pk_{i-1} outputs (pk_i, Π_{pk_i}) , where pk_i denotes the updated public-key and Π_{pk_i} is a proof for the correctness of the updating process.
- $(1, \perp) \leftarrow KV(pk_i, \Pi_{pk_i})$: Given a potentially updated pk_i and Π_{pk_i} , checks the validity of the updated key. It returns either \perp if pk_i is incorrectly formed (and updated), otherwise it outputs 1.
- $(c) \leftarrow \text{Enc}(pk_i, m)$: Given a (potentially updated) public key pk_i and a message $m \in \mathcal{M}$, it outputs a ciphertext $c \in \mathcal{C}$.
- $(\perp, m') \leftarrow \text{Dec}(sk_i, c)$: Given $c \in \mathcal{C}$ and the secret key sk_i , returns either \perp (reject) or $m' \in \mathcal{M}$ (successful). Note that in the standard public-key cryptosystems (and in this definition before any updating) $sk_i = sk_0$.

Primary requirements for a public-key cryptosystem with updatable keys, $\Psi_{\text{Enc}} := (KG, KU, KV, \text{Enc}, \text{Dec})$, can be summarized as follows,

Definition 60 (Perfect Updatable Correctness). A cryptosystem Ψ_{Enc} with updatable keys is perfect updatable correct, if we have,

$$\Pr \left[\begin{array}{l} (pk_0, \Pi_{pk_0}, sk_0 := sk'_0) \leftarrow KG(1^\kappa), r_s \leftarrow \$ \text{RND}(\text{Sub}), \\ ((\{pk_j, \Pi_{pk_j}\}_{j=1}^i, \xi_{\text{Sub}}) \parallel \{sk'_j\}_{j=1}^i) \leftarrow (\text{Sub} \parallel \text{Ext}_{\text{Sub}})(pk_0, \Pi_{pk_0}, r_s), \\ \{KV(pk_j, \Pi_{pk_j}) = 1\}_{j=0}^i : \text{Dec}(sk_i := \{sk'_j\}_{j=0}^i, \text{Enc}(pk_i, m)) = m \end{array} \right] = 1 .$$

where sk'_j is the individual secret-key of each party and pk_i is the final public-key.

Definition 61 (Updatable Key Hiding). In a cryptosystem Ψ_{Enc} with updatable keys, for $(pk_0, \Pi_{pk_0}, sk_0 := sk'_0) \leftarrow KG(1^\kappa)$ and $(pk_i, \Pi_{pk_i}) \leftarrow KU(pk_{i-1})$, we say that Π_{Enc} is updatable key hiding, if one of the following cases holds,

- the original pk_0 was honestly generated and KV algorithm returns 1, namely $(pk_0, \Pi_{pk_0}, sk_0) \leftarrow KG(1^\kappa)$ and $KV(pk_0, \Pi_{pk_0}) = 1$,
- the original pk_0 verifies successfully with KV and the key-update was generated honestly once, namely $KV(pk_0, \Pi_{pk_0}) = 1$ and $(\{pk_j, \Pi_{pk_j}\}_{j=1}^i) \leftarrow KU(pk_0)$ such that $\{KV(pk_j, \Pi_{pk_j}) = 1\}_{j=1}^i$.

Definition 62 (Updatable IND-CPA). A public-key cryptosystem Ψ_{Enc} with updatable keys satisfies updatable IND-CPA, if for all PPT subverter Sub , for all κ , and for all PPT adversaries \mathcal{A} ,

$$\Pr \left[\begin{array}{l} (\text{pk}_0, \Pi_{\text{pk}_0}, \text{sk}_0 := \text{sk}'_0) \leftarrow \text{KG}(1^\kappa), r_s \leftarrow \$ \text{RND}(\text{Sub}), \\ (\{\text{pk}_j, \Pi_{\text{pk}_j}\}_{j=1}^i, \xi_{\text{Sub}}) \leftarrow \text{Sub}(\text{pk}_0, \Pi_{\text{pk}_0}, r_s), b \leftarrow \$ \{0, 1\}, (m_0, m_1) \leftarrow \\ \mathcal{A}(\text{pk}_i, \xi_{\text{Sub}}), b' \leftarrow \mathcal{A}(\text{Enc}(\text{pk}_i, m_b)) : \{\text{KV}(\text{pk}_j, \Pi_{\text{pk}_j}) = 1\}_{j=0}^i \wedge b' = b \end{array} \right] \approx_\kappa \frac{1}{2}.$$

where ξ_{Sub} is the auxiliary information which is returned by the subverter Sub . Note that Sub can also generate the initial pk_0 and then an honest key updater KU updates it and outputs pk_i and the proof Π_{pk_i} .

9.3.2 Building Key-Updatable Cryptosystems

We first prove a theorem that gives a generic approach for building a cryptosystem with updatable keys using the key-homomorphic cryptosystems. Then, we use the generic approach and present the first key-updatable cryptosystem.

Theorem 8 (Cryptosystems with Updatable Keys). Every correct, IND-CPA secure, and key-homomorphic cryptosystem Ψ_{Enc} with an efficient extractor Ext_{Sub} , satisfies updatable correctness, updatable key hiding and updatable IND-CPA security. The proof is provided in full version [BS20].

A Key-Updatable Cryptosystem from Key-Homomorphic Cryptosystems.

Next, we show that the El-Gamal cryptosystem [ElG84] instantiated in a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2)$ can be represented as a key-updatable encryption scheme constructed from key-homomorphic encryption schemes. In bilinear group based instantiation, in contrast to the standard El-Gamal encryption (reviewed in full version [BS20]), the public key consists of a pair $([x]_1, [x]_2)$. Consequently, the algorithms of new variation can be expressed as follows,

- $(\text{pk}_0, \Pi_{\text{pk}_0}, \text{sk}_0 := \text{sk}'_0) \leftarrow \text{KG}(1^\kappa)$: Given 1^κ , obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2) \leftarrow \text{BGgen}(1^\kappa)$; sample $\text{sk}'_0 \leftarrow \$ \mathbb{Z}_p^*$ and return the key pair $(\text{pk}_0, \text{sk}_0) := ((\text{pk}_0^1, \text{pk}_0^2), \text{sk}_0) := (([\text{sk}'_0]_1, [\text{sk}'_0]_2), \text{sk}'_0)$ and $\Pi_{\text{pk}_0} := (\Pi_{\text{pk}_0}^1, \Pi_{\text{pk}_0}^2) := ([\text{sk}'_0]_1, [\text{sk}'_0]_2)$ as a proof of correctness (a.k.a. well-formedness).

- $(pk_i, \Pi_{pk_i}) \leftarrow KU(pk_{i-1})$: Obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2) \leftarrow BGgen(1^\kappa)$; then for a given $pk_{i-1} := (pk_{i-1}^1, pk_{i-1}^2) := ([sk_{i-1}]_1, [sk_{i-1}]_2)$, for $i \geq 1$, sample $sk'_i \leftarrow \mathbb{Z}_p^*$ and output: $(pk_i, \Pi_{pk_i}) := \left(\left([sk_{i-1} + sk'_i]_1, [sk_{i-1} + sk'_i]_2 \right), \left([sk'_i]_1, [sk'_i]_2 \right) \right)$, where $pk_i := (pk_i^1, pk_i^2)$ denotes the updated public-key associated with the secret key $sk_i := sk_{i-1} + sk'_i$ and $\Pi_{pk_i} := (\Pi_{pk_i}^1, \Pi_{pk_i}^2) := ([sk'_i]_1, [sk'_i]_2)$ is the proof for correctness of the update.
- $(1, \perp) \leftarrow KV(\{pk_j\}_{j=0}^i, \Pi_{pk_i})$: Obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2) \leftarrow BGgen(1^\kappa)$, and then,
 - for $i = j = 0$, given $pk_0 := (pk_0^1, pk_0^2) := ([sk_0]_1, [sk_0]_2)$, and the proof $\Pi_{pk_0} := (\Pi_{pk_0}^1, \Pi_{pk_0}^2) := ([sk'_0]_1, [sk'_0]_2)$, checks $\Pi_{pk_0}^1 \bullet [1]_2 \stackrel{?}{=} [1]_1 \bullet pk_0^2$, $[1]_1 \bullet \Pi_{pk_0}^2 \stackrel{?}{=} pk_0^1 \bullet [1]_2$, $[1]_1 \bullet \Pi_{pk_0}^2 \stackrel{?}{=} \Pi_{pk_0}^1 \bullet [1]_2$.
 - for $i \geq 1$, given $pk_{i-1} := (pk_{i-1}^1, pk_{i-1}^2) := ([sk_{i-1}]_1, [sk_{i-1}]_2)$, a potentially updated $pk_i := (pk_i^1, pk_i^2) := ([sk_{i-1} + sk'_i]_1, [sk_{i-1} + sk'_i]_2)$, and $\Pi_{pk_i} := (\Pi_{pk_i}^1, \Pi_{pk_i}^2) := ([sk'_i]_1, [sk'_i]_2)$, checks $(pk_{i-1}^1 + \Pi_{pk_i}^1) \bullet [1]_2 \stackrel{?}{=} [1]_1 \bullet pk_i^2$, $[1]_1 \bullet (pk_{i-1}^2 + \Pi_{pk_i}^2) \stackrel{?}{=} pk_i^1 \bullet [1]_2$ and $[1]_1 \bullet \Pi_{pk_i}^2 \stackrel{?}{=} \Pi_{pk_i}^1 \bullet [1]_2$.

in each case, if all the checks pass, it returns 1, otherwise \perp .

- $(c) \leftarrow Enc(pk_i, m)$: Obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2) \leftarrow BGgen(1^\kappa)$ and then given a (potentially updated) public key $pk_i := ([sk_i]_1, [sk_i]_2)$, such that $sk_i := sk_{i-1} + sk'_i$, and a message $m \in \mathcal{M}$, samples a randomness $r \leftarrow \mathbb{Z}_p^*$ and outputs $c := (c_1, c_2) := (m \cdot [rsk_i]_T, [r]_T)$.
- $(\perp, m) \leftarrow Dec(sk_i, c)$: Obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{E}, [1]_1, [1]_2) \leftarrow BGgen(1^\kappa)$ and then given a ciphertext $c \in \mathcal{C}$ and a potentially updated secret key $sk_i = sk_{i-1} + sk'_i$ it returns, $\frac{c_1}{c_2^{sk}} = \frac{m \cdot [rsk_i]_T}{[rsk_i]_T} = m$.

In the proposed construction, for the case that $\{KV(\{pk_j\}_{j=0}^i, \Pi_{pk_i}) = 1\}_{j=0}^i$, under the BDH-KE knowledge assumption (See full version [BS20]) with checking $[1]_1 \bullet \Pi_{pk_j}^2 \stackrel{?}{=} \Pi_{pk_j}^1 \bullet [1]_2$ for $0 \leq j \leq i$, there exists an efficient nBB extractor Ext_{Sub} that can extract all sk'_j from the subvector Sub_j . Note that here we considered the standard version of the El-Gamal cryptosystem, but we could also take its *lifted* version, which encrypts g^m instead of m .

A More Efficient Key-Updatable Cryptosystem.

The technique proposed in The. 8, acts as a generic approach but might lead to inefficient constructions. We present a more efficient key-updatable variant of El-Gamal cryptosystem.

Hash-based El-Gamal Cryptosystem in Bilinear Groups. The hash-based variation of El-Gamal cryptosystem [ELG84], is proven to achieve IND-CPA in the random oracle model. In the rest, we present a new variation of it, instantiated with bilinear groups, and show that the proposed variation can be represented as a secure key-updatable encryption scheme. The PPT algorithms (KG, KU, KV) in the new variation are identical to those in the first variation, while the encryption and decryption algorithms (Enc, Dec) behave as follows:

- $(c) \leftarrow \text{Enc}(\mathbf{H}, \mathbf{pk}_i, m)$: Given the one-way hash function \mathbf{H} , a public key $\mathbf{pk}_i := (\mathbf{pk}_i^1, \mathbf{pk}_i^2)$ and a message $m \in \{0, 1\}^n$ as inputs. It samples $r \leftarrow \mathbb{Z}_p^*$ and returns $c := (c_1, c_2) := (m \oplus \mathbf{H}((\mathbf{pk}_i^1)^r), [r]_1)$.
- $(\perp, m) \leftarrow \text{Dec}(\mathbf{H}, \mathbf{sk}_i, c)$: Given the hash function \mathbf{H} , the secret key \mathbf{sk}_i , corresponding to \mathbf{pk}_i , and a ciphertext $c := (c_1, c_2)$, decrypts c by calculating $m := c_1 \oplus \mathbf{H}(c_2^{\mathbf{sk}_i})$.

Theorem 9 (Hashed El-Gamal Cryptosystem with Updatable Keys). The proposed variation of Hashed El-Gamal encryption satisfies updatable correctness, updatable key hiding and updatable IND-CPA if BDH-KE and Extended asymmetric Computational Diffie–Hellman assumptions hold in $(\mathbb{G}_1, \mathbb{G}_2)$, and the hash function \mathbf{H} is a random oracle. The proof is provided in full version [BS20].

9.3.3 Performance of the Proposed Key-Updatable Cryptosystems

We evaluate practical efficiency of both the proposed key-updatable cryptosystems using the Charm-Crypto framework [Aki+13], a Python library for pairing-based cryptography³. We apply Barreto-Naehrig (BN254) curve, $y^2 = x^3 + b$ with embedding curve degree 12 [BN05] as an SNARK-friendly curve. Benchmarks are done on a laptop with Ubuntu 20.04.2 LTS equipped with an Intel Core i7-9850H CPU @2.60 GHz and 16 GB of memory. As we observed in Sec. 9.3.2, both the pairing-based and hash-based constructions

³The source code is publicly available on <https://github.com/Bagheri/Tiramisu>.

have the same (KG, KU, KV) algorithms. In Fig. 9.2, we plot the running time of key-updating, KU, key-verification, KV, and the transcript size versus the number of key updates, where *transcript* refers to all the keys as well as the proofs generated with all updaters.

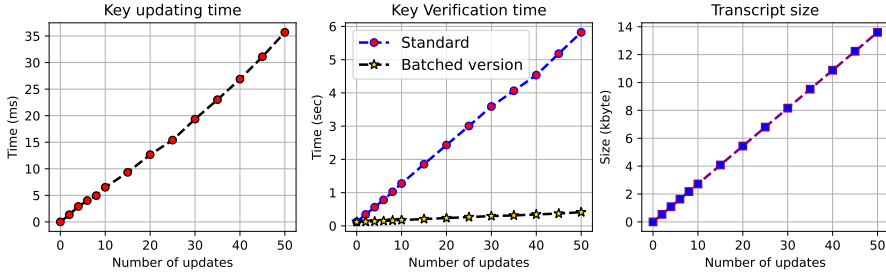


Figure 9.2: Key Updating, Key Verification (standard & Batched Versions) and Transcript Size for both the Proposed Key-Updatable Cryptosystems.

As it is illustrated in Fig. 9.2, in both constructions, the key updating, key verification times and the transcript size are practical and grow linearly with the number of updates. One time key updating along with generating the underlying proof requires ≈ 1 millisecond (ms), while to update a key 50 times and provide proof of correctness only takes ≈ 36 ms. To verify the validity of a key that is updated 50 times, a verifier requires ≈ 6 seconds in the standard form of KV algorithm, however, using the standard batching techniques [Abd+17] this can be done $12\times$ faster, in ≈ 0.5 second. In terms of the transcript size, for a key that is updated 10 times, the verifier requires to store ≈ 3 Kbytes.

Our experiments confirm that the time required for running the encryption algorithm is constant and takes about ≈ 32 ms and ≈ 1.2 ms in the pairing-based and hash-based constructions independent of the number of updates, respectively. While the running time for the decryption algorithm are equal to ≈ 4.5 ms and ≈ 1 ms, respectively. One may notice that the ciphertext size remains constant in our setting they are equal to 1028 and 46 bytes in the pairing-based and Hash-based encryption schemes, respectively.

9.4 Tiramisu: BB-SE NIZK in Updatable CRS Model

We present TIRAMISU, as a protocol that allows one to generically build NIZK arguments in the updatable CRS model, which achieve U-ZK (defined in full

version [BS20]) along with either Updatable Black-Box Simulation Extractability (U-BB-SE) or Updatable Black-Box Knowledge Soundness (U-BB-KS) which we define next. We first define Updatable Simulation Soundness (U-SS) that is used in TIRAMISU.

Definition 63 (Updatable Simulation Soundness). A non-interactive argument Ψ_{NIZK} is *updatable simulation soundness* for \mathcal{R} , if for any subverter Sub , and every PPT \mathcal{A} , the following probability is $\nu(\kappa)$,

$$\Pr \left[\begin{array}{l} (\mathbf{R}, \xi_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\kappa), ((\text{crs}_0, \Pi_{\text{crs}_0}) \parallel \vec{\text{ts}}_0 := \vec{\text{ts}}'_0) \leftarrow \text{Gen}_{\text{crs}}(\mathbf{R}, \xi_{\mathbf{R}}), r_s \leftarrow_{\$} \text{RND}(\text{Sub}), \\ ((\{\text{crs}_j, \Pi_{\text{crs}_j}\}_{j=1}^i, \xi_{\text{Sub}}) \parallel \{\vec{\text{ts}}'_j\}_{j=1}^i) \leftarrow (\text{Sub} \parallel \text{Ext}_{\text{Sub}})(\text{crs}_0, \Pi_{\text{crs}_0}, r_s), \\ \{\text{CV}(\text{crs}_j, \Pi_{\text{crs}_j}) = 1\}_{j=0}^i, (x, \pi) \leftarrow \mathcal{A}^{\mathcal{O}(\vec{\text{ts}}_i, \dots)}(\mathbf{R}, \xi_{\mathbf{R}}, \text{crs}_i, \xi_{\text{Sub}}) : \\ (x, \pi) \notin Q \wedge x \notin \mathbf{L} \wedge \mathbf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \text{crs}_i, x, \pi) = 1 \end{array} \right],$$

where Π_{crs} is a proof for correctness of CRS generation/updating, $\vec{\text{ts}}_i$ is the simulation trapdoor associated with the final CRS that can be computed using $\{\vec{\text{ts}}'_j\}_{j=0}^i$, and Q is the set of simulated statement-proof pairs returned by oracle $\mathcal{O}(\cdot)$.

Definition 64 (Updatable Black-Box Simulation Extractability). An argument Ψ_{NIZK} is *updatable black-box (strong) simulation-extractable* for \mathcal{R} , if for every PPT \mathcal{A} and subverter Sub , the following probability is $\nu(\kappa)$,

$$\Pr \left[\begin{array}{l} (\mathbf{R}, \xi_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\kappa), ((\text{crs}_0, \Pi_{\text{crs}_0}) \parallel \vec{\text{ts}}_0 := \vec{\text{ts}}'_0 \parallel \vec{\text{te}}_0 := \vec{\text{te}}'_0) \leftarrow \text{Gen}_{\text{crs}}(\mathbf{R}, \xi_{\mathbf{R}}), \\ r_s \leftarrow_{\$} \text{RND}(\text{Sub}), ((\{\text{crs}_j, \Pi_{\text{crs}_j}\}_{j=1}^i, \xi_{\text{Sub}}) \parallel \{\vec{\text{ts}}'_j\}_{j=1}^i \parallel \{\vec{\text{te}}'_j\}_{j=1}^i) \leftarrow \dots \\ \dots (\text{Sub} \parallel \text{Ext}_{\text{Sub}})(\text{crs}_0, \Pi_{\text{crs}_0}, r_s), \{\text{CV}(\text{crs}_j, \Pi_{\text{crs}_j}) = 1\}_{j=0}^i, r_{\mathcal{A}} \leftarrow_{\$} \text{RND}(\mathcal{A}), \\ (x, \pi) \leftarrow \mathcal{A}^{\mathcal{O}(\vec{\text{ts}}_i, \dots)}(\mathbf{R}, \xi_{\mathbf{R}}, \text{crs}_i, \xi_{\text{Sub}}; r_{\mathcal{A}}), w \leftarrow \text{Ext}(\mathbf{R}, \xi_{\mathbf{R}}, \text{crs}_i; \vec{\text{te}}_i) : \\ (x, \pi) \notin Q \wedge (x, w) \notin \mathbf{R} \wedge \mathbf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \text{crs}_i, x, \pi) = 1 \end{array} \right].$$

where Ext_{Sub} in a nBB PPT extractor (e.g. based of rewinding or knowledge assumption), Ext is a black-box PPT extractor (e.g. using a decryption algorithm), Π_{crs} is a proof for correctness of CRS generation/updating, and $\vec{\text{ts}}_i, \vec{\text{te}}_i$ are the simulation and extraction trapdoors associated with the final CRS that can be computed using $\{\vec{\text{ts}}'_j\}_{j=0}^i$ and $\{\vec{\text{te}}'_j\}_{j=0}^i$, respectively. Here, $\text{RND}(\mathcal{A}) = \text{RND}(\text{Sub})$ and Q is the set of the statement and simulated proofs returned by oracle $\mathcal{O}(\cdot)$.

Intuitively, the definition implies that under the existence of a nBB extractor in the *setup phase*, the protocol achieves SE with BB extraction, as long as the initial CRS generation or one of CRS updates is done by an honest party. Our definition of U-BB-SE is inspired from the standard definition (realized under a trusted setup) presented by Groth [Gro06], which considers two extractors, one for the setup phase and the other for the rest of argument. However, our definition uses a non-black-box extractor in the setup phase, which seems a *unavoidable* requirement for building U-BB-SE NIZK argument *without a trusted third party* [BFS16]. Indeed, using some arguments or assumptions with non-black box extraction techniques, e.g. by rewinding [Dam+12] or knowledge assumptions [BFS16; Abd+17; Gro+18], is a common and practical way to mitigate or eliminate the trust on the parameters of various cryptographic protocols. We also consider building NIZK arguments that can achieve U-BB-KS which is a weaker version of U-BB-SE, where in the former, \mathcal{A} would not have access to oracle $\mathcal{O}(\cdot)$. Note that in Def. 63 and Def. 64, it is equivalent for the adversary to batch all its updates and then think of one honest update. This requires that the trapdoor contributions of setup and update commute. This is true of known constructions in the updatable CRS model [Mal+19]. Therefore, in the underlying NIZK and key-updatable cryptosystem, we expect that they both satisfy the property that trapdoors combine and commute.

Our main goal is to construct an alternative to the C0C0 framework [Kos+15] but in the *updatable* CRS model, such that in new constructions the end-users can bypass the blind trust in the setup phase by one-time updating the shared parameters. Our starting point is the strongest construction of the C0C0 framework (reviewed in full version [BS20]) that gets a sound NIZK argument and lifts it to a BB-SE NIZK argument. To do so, given a language \mathbf{L} with the corresponding NP relation $\mathbf{R}_{\mathbf{L}}$, the C0C0 framework defines a new language $\hat{\mathbf{L}}$ such that $((x, c, \mu, \mathbf{pk}_s, \mathbf{pk}_e, \rho), (r, r_0, \mathbf{w}, s_0)) \in \mathbf{R}_{\hat{\mathbf{L}}}$ iff,

$$c = \text{Enc}(\mathbf{pk}_e, \mathbf{w}; r) \wedge ((x, \mathbf{w}) \in \mathbf{R}_{\mathbf{L}} \vee (\mu = f_{s_0}(\mathbf{pk}_s) \wedge \rho = \text{Com}(s_0; r_0))),$$

where $\{f_s : \{0, 1\}^* \rightarrow \{0, 1\}^\kappa\}_{s \in \{0, 1\}^\kappa}$ is a pseudo-random function family, $(\text{KG}_e, \text{Enc}, \text{Dec})$ is a set of algorithms for a semantically secure encryption scheme, $(\text{KG}_s, S_s, \text{Vfy}_s)$ is a one-time signature scheme and (Com, Vfy) is a perfectly binding commitment scheme.

As a result, given a sound NIZK argument Ψ_{NIZK} for \mathcal{R} constructed from PPT algorithms $(\text{Gen}_{\text{crs}}, \text{P}, \text{V}, \text{Sim}, \text{Ext})$, the C0C0 framework returns a BB-SE NIZK argument $\hat{\Psi}_{\text{NIZK}}$ with PPT algorithms $(\hat{\text{Gen}}_{\text{crs}}, \hat{\text{P}}, \hat{\text{V}}, \hat{\text{Sim}}, \hat{\text{Ext}})$, where $\hat{\text{Gen}}_{\text{crs}}$ is the CRS generator for new construction and acts as follows,

- $(\hat{\text{crs}} \parallel \hat{\text{ts}} \parallel \hat{\text{te}}) \leftarrow \hat{\text{Gen}}_{\text{crs}}(\mathbf{R}_{\mathbf{L}}, \xi_{\mathbf{R}_{\mathbf{L}}})$: Given $(\mathbf{R}_{\mathbf{L}}, \xi_{\mathbf{R}_{\mathbf{L}}})$, sample $(\text{crs} \parallel \text{ts}) \leftarrow \text{Gen}_{\text{crs}}(\mathbf{R}_{\hat{\mathbf{L}}}, \xi_{\mathbf{R}_{\hat{\mathbf{L}}}})$; $(\mathbf{pk}_e, \mathbf{sk}_e) \leftarrow \text{KG}_e(1^\kappa)$; $s_0, r_0 \leftarrow_{\$} \{0, 1\}^\kappa$; $\rho := \text{Com}$

$(s_0; r_0)$; and output $(\hat{c}\tilde{r}s \parallel \hat{t}\tilde{s} \parallel \hat{t}\tilde{e}) := ((\hat{c}\tilde{r}s, \text{pk}_e, \rho) \parallel (s_0, r_0) \parallel \text{sk}_e)$, where $\hat{c}\tilde{r}s$ is the CRS of $\hat{\Psi}_{\text{NIZK}}$ and $\hat{t}\tilde{s}$ and $\hat{t}\tilde{e}$, respectively, are the simulation trapdoor and extraction trapdoor associated with $\hat{c}\tilde{r}s$.

Considering the description of algorithm $\hat{\text{Gen}}_{\hat{c}\tilde{r}s}$, to construct an alternative to the $\mathcal{C}\mathcal{O}\mathcal{C}\mathcal{O}$ framework but in the *updatable* CRS model, a naive solution is to construct the three primitives above (with *gray* background) in the *updatable* CRS model, and then define a similar language but using the primitives constructed in the updatable CRS model. But, considering the state-of-the-art ad-hoc constructions and generic compilers to build NIZK arguments with updatable CRS model, a more efficient solution is to simplify the language $\hat{\mathbf{L}}$ and construct more efficient BB-SE NIZK arguments with updatable parameters.

Continuing the second solution, since currently there exist some ad-hoc constructions that allow two-phase updating (e.g. [BGM17; BG18; Bag+21; BPR20]) or even a lifting construction to build nBB-SE zk-SNARKs with universal CRS in the updatable CRS model (e.g. [ARS20]), therefore we simplify the original language $\hat{\mathbf{L}}$ defined in $\mathcal{C}\mathcal{O}\mathcal{C}\mathcal{O}$ and show that given a simulation sound NIZK argument with *updatable* CRS we can construct U-BB-SE NIZK arguments in a more efficient manner than the mentioned naive way. To this end, we use the key-updatable cryptosystems, defined and built in Sec. 9.3.

Let $\Psi_{\text{Enc}} := (\text{KG}, \text{KU}, \text{KV}, \text{Enc}, \text{Dec})$ be a set of algorithms for a semantically secure cryptosystem with updatable keys $(\text{pk}_i, \text{sk}_i)$. Similar to $\mathcal{C}\mathcal{O}\mathcal{C}\mathcal{O}$ framework, we define a new language $\hat{\mathbf{L}}$ based on the main language \mathbf{L} corresponding to the input updatable nBB-SE NIZK $\Psi_{\text{NIZK}} := (\text{Gen}_{\text{crs}}, \text{CU}, \text{CV}, \text{P}, \text{V}, \text{Sim}, \text{Ext})$. The language $\hat{\mathbf{L}}$ is embedded with the encryption of witness with the *potentially updated* public key pk_i given in the CRS. Namely, given a language \mathbf{L} with the corresponding NP relation $\mathbf{R}_{\mathbf{L}}$, we define $\hat{\mathbf{L}}$ for a given random element $r \leftarrow \$ \mathbb{F}_p$, such that $((x, c, \text{pk}_i), (w, r)) \in \mathbf{R}_{\hat{\mathbf{L}}}$ iff, $c = \text{Enc}(\text{pk}_i, w; r) \wedge (x, w) \in \mathbf{R}_{\mathbf{L}}$.

The intuition behind $\hat{\mathbf{L}}$ is to enforce the P to encrypt its witness with a potentially updated public key pk_i , given in the CRS, and send the ciphertext c along with a *simulation sound* proof. Consequently, in proving BB-SE, the updated sk_i of the defined cryptosystem Ψ_{Enc} is given to the Ext, which makes it possible to extract the witness in a *black-box* manner. By sending the encryption of witnesses, the proof will not be *witness* succinct anymore, but still, it is succinct in the size of the circuit that encodes $\hat{\mathbf{L}}$.

In security proofs, we show that due to updatable simulation soundness (in Def. 63) of the underlying NIZK argument Ψ_{NIZK} , the *updatable IND-CPA* security (in Def. 62) and perfect *updatable completeness* (in Def. 60) of Ψ_{Enc} is sufficient to achieve BB-SE in the updatable NIZK argument $\hat{\Psi}_{\text{NIZK}}$ for

the language $\hat{\mathbf{L}}$. By considering new language $\hat{\mathbf{L}}$, the modified construction $\hat{\Psi}_{\text{NIZK}} := (\hat{\text{Gen}}_{\text{crs}}, \hat{\text{CU}}, \hat{\text{CV}}, \hat{\text{P}}, \hat{\text{V}}, \text{Sim}, \text{Ext})$ for $\hat{\mathbf{L}}$ can be written as in Fig. 9.3.

Efficiency. Considering new language $\hat{\mathbf{L}}$, in new argument $\hat{\Psi}_{\text{NIZK}}$ the CRS generation (CRS updating and CRS verification) of the input argument Ψ_{NIZK} will be done for a larger instance, and one also needs to generate (update and verify) the key pairs of the updatable public-key cryptosystem. The corresponding circuit of the newly defined language $\hat{\mathbf{L}}$, expands by the number of constraints needed for the encryption function. Recall that the language $\hat{\mathbf{L}}$ is an appended form of language \mathbf{L} by encryption of witnesses. However, due to our simplifications in defining language $\hat{\mathbf{L}}$, the overhead in TIRAMISU will be less than the case one uses the $\mathcal{C}\emptyset\mathcal{C}\emptyset$ framework. Meanwhile, as we later show in Sec. 9.5 the efficiency of final constructions severely depends on the input NIZK argument.

The prover of the new construction $\hat{\Psi}_{\text{NIZK}}$ needs to generate a proof for new language $\hat{\mathbf{L}}$ that would require extra computations. The proofs will be the proof of input nBB-SE updatable NIZK argument Ψ_{NIZK} appended with the ciphertext c which leads to having proofs linear in *witness* size but still succinct in the *circuit* size. It is a known result that having proofs linear in witness size is an undeniable fact to achieve BB extraction and UC-security [Can01; GW11].

As the verifier is unchanged, so the verification of new constructions will be the same as NIZK Ψ_{NIZK} but for a larger statement.

The proof of Theorems 10-12, are provided in the full version of paper [BS20].

Theorem 10 (Perfect Updatable Completeness). If the input NIZK argument Ψ_{NIZK} guarantees perfect updatable completeness for the language \mathbf{L} , and the public-key cryptosystem Ψ_{Enc} be perfectly updatable correct, then the NIZK argument constructed in Fig. 9.3 for language $\hat{\mathbf{L}}$, is perfectly updatable complete.

Theorem 11 (Computationally Updatable Zero-Knowledge). If the input NIZK argument Ψ_{NIZK} guarantees ZK, and the public-key cryptosystem Ψ_{Enc} is updatable IND-CPA and satisfies updatable key hiding, then the NIZK argument constructed in Fig. 9.3 for $\hat{\mathbf{L}}$ satisfies computational updatable ZK.

Theorem 12 (Updatable Black-Box Simulation Extractability). If the input NIZK argument Ψ_{NIZK} guarantees updatable correctness, updatable simulation soundness and updatable zero-knowledge, and the public-key cryptosystem Ψ_{Enc} satisfies updatable perfect correctness, updatable key hiding, and updatable IND-CPA, then the NIZK argument constructed in Fig. 9.3 for language $\hat{\mathbf{L}}$ satisfies updatable BB simulation extractability.

Note that to bypass the impossibility of achieving Sub-ZK and BB extractability in NIZKs [BFS16], one-time honest key generation/updating on pk_i is a crucial requirement which does not allow an adversary to obtain the trapdoors associated with final updated CRS, particularly the extraction keys.

Building Updatable Black-Box Knowledge Sound NIZK Arguments with Tiramisu. The primary goal of TIRAMISU is constructing BB-SE NIZK arguments in the updatable CRS model. However, due to some efficiency reasons, in practice one might need to build an Updatable Black-Box Knowledge Sound (U-BB-KS) NIZK argument. In such cases, starting from either an updatable sound NIZK or an U-nBB-KS NIZK (e.g. Groth et al.’s updatable zk-SNARK [Gro+18]), the same language $\hat{\mathbf{L}}$ defined in TIRAMISU along with our constructed updatable public-key cryptosystem allows one to build an U-BB-KS NIZK argument. Namely, given an updatable cryptosystem $\Psi_{\text{Enc}} := (\text{KG}, \text{KU}, \text{KV}, \text{Enc}, \text{Dec})$ with updatable keys $(\text{pk}_i, \text{sk}_i)$, and an *updatable sound* NIZK $\Psi_{\text{NIZK}} := (\text{Gen}_{\text{crs}}, \text{CU}, \text{CV}, \text{P}, \text{V}, \text{Sim})$ for language \mathbf{L} with the corresponding NP relation $\mathbf{R}_{\mathbf{L}}$, we define the language $\hat{\mathbf{L}}$ for a given random element $r \leftarrow \$ \mathbb{F}_p$, such that $((x, c, \text{pk}_i), (w, r)) \in \mathbf{R}_{\hat{\mathbf{L}}}$ iff, $(c = \text{Enc}(\text{pk}_i, w; r)) \wedge ((x, w) \in \mathbf{R}_{\mathbf{L}})$.

Corollary 2. If the input Ψ_{NIZK} for $\mathbf{R}_{\mathbf{L}}$ guarantees updatable correctness, updatable soundness and updatable zero-knowledge, and the public-key cryptosystem Ψ_{Enc} satisfies updatable perfect correctness, updatable key hiding, and updatable IND-CPA, then the NIZK argument for language $\hat{\mathbf{L}}$ satisfies updatable correctness, updatable *knowledge* soundness and updatable zero-knowledge.

The proof can be done similar to the proof of Theorem 12, without providing the simulation oracle to the adversaries \mathcal{A} and \mathcal{B} .

9.5 Building U-BB-SE NIZK Arguments with Tiramisu

To build an U-BB-SE NIZK argument with TIRAMISU, one needs (1) a key-updatable cryptosystem Ψ_{Enc} that satisfies *perfect updatable correctness*, *updatable key hiding*, and *updatable IND-CPA*, and (2) a NIZK argument Ψ_{NIZK} that guarantees *updatable simulation soundness* or *U-nBB-SE*. Next, we instantiate Ψ_{Enc} and Ψ_{NIZK} , and obtain two U-BB-SE NIZK arguments. For Ψ_{Enc} , one can use either of the proposed variations of El-Gamal cryptosystem in Sec. 9.3. Whereas for Ψ_{NIZK} , one can either use an ad-hoc construction

Table 9.2: A comparison of BB-SE NIZK arguments built with the $C\emptyset C\emptyset$ and TIRAMISU. n' : Number of constraints used to encode language $\hat{\mathbf{L}}$, $|\mathbf{pk}|$: Size of the public key of Ψ_{Enc} , κ : Security parameter, E_i : Exponentiation in \mathbb{G}_i , P : Paring, l' : the size of statement in new language $\hat{\mathbf{L}}$, \mathbf{w} : the witness for $\mathbf{R}_{\hat{\mathbf{L}}}$.

	$C\emptyset C\emptyset$ (with [Gro16])	Tiramisu (with [Gro+18; ARS20])	Tiramisu (with [BGM17; BG18])
Trusted Setup	Yes	No	No
CRS Upd.	No	One-phase (Universal)	Two-phase
CRS Size	$\approx 3n'\mathbb{G}_1 + n'\mathbb{G}_2$	$\approx 30n'^2\mathbb{G}_1 + 9n'^2\mathbb{G}_2$	$\approx 3n'\mathbb{G}_1 + n'\mathbb{G}_2$
CRS Verifier	—	$\approx 78n'^2P$	$14n'P$ (batchable)
CRS Updater	—	$\approx 30n'^2E_1 + 9n'^2E_2$	$\approx 6n'E_1 + n'E_2$
Prover	$\approx 4n'E_1 + n'E_2$	$\approx 4n'E_1 + n'E_2$	$\approx 4n'E_1 + n'E_2$
Proof Size	$o(\mathbf{w}) + 3\mathbb{G}_1 + 2\mathbb{G}_2 + \kappa$	$o(\mathbf{w}) + 4\mathbb{G}_1 + 3\mathbb{G}_2$	$o(\mathbf{w}) + 3\mathbb{G}_1 + 2\mathbb{G}_2$
Verifier	$4P + l'E_1$	$6P + l'E_1$	$5P + l'E_1$

(e.g. [GWC19; Mal+19] with universal CRS, or [BG18; Bag+21; BPR20] when their CRS is generated with [BGM17], which will have a two-phase updating), or a construction lifted with LAMASSU [ARS20] (e.g. using [Gro+18]).

In BB-SE NIZK arguments built with TIRAMISU, the parties have to update the shared parameters individually once and check the validity of the previous updates. This is basically the computational cost that the end-users need to pay to bypass the trust in the standard CRS model. As an important practical optimization, it can be shown that the prover can only update the CRS $\hat{\mathbf{crs}}_i := (\mathbf{crs}_i, \mathbf{pk}_i)$ partially, namely only \mathbf{pk}_i . Tab. 9.2 summarizes the efficiency of two BB-SE NIZK arguments built with TIRAMISU and compares them with a construction lifted by the $C\emptyset C\emptyset$ framework in the standard CRS model. We instantiate $C\emptyset C\emptyset$ with the state-of-the-art zk-SNARK [Gro16] and instantiate TIRAMISU with 1) the lifted version of [Gro+18] with LAMASSU [ARS20], and 2) one of the constructions proposed in [BPR20] when their CRS is sampled using the two-phase protocol proposed in [BGM17]. As we observed in Section 9.3.3, in the resulting U-BB-SE zk-SNARKs, the overhead added by the key updateable encryption schemes add very little overhead to the CU and CV algorithms.

Both $C\emptyset C\emptyset$ and TIRAMISU constructions result a linear proof in the witness size, but they keep the asymptotic efficiency of other algorithms in the input NIZK. Consequently, instantiating TIRAMISU with a more efficient nBB-SE NIZK argument will result in a more efficient BB-SE NIZK argument. Therefore, as also is shown in Tab. 9.2, suitable ad-hoc constructions result in more efficient

U-BB-SE NIZK arguments. We found constructing more efficient updatable nBB-SE zk-SNARKs as an interesting future research direction. Following, the impossibility result of Gentry and Wichs [GW11], it is undeniable that achieving BB extraction will result in non-succinct proof. Consequently, in all the schemes in Tab. 9.2, the proof size is dominated with the size of c which is a ciphertext of IND-CPA cryptosystem and is $o(w)$.

CRS and trapdoor generation, $(\hat{c\vec{r}s}_0, \hat{\Pi}_{\hat{c\vec{r}s}_0}) \leftarrow \hat{\text{Gen}}_{\text{crs}}(\mathbf{R}_L, \xi_{\mathbf{R}_L})$: Given $(\mathbf{R}_L, \xi_{\mathbf{R}_L})$ acts as follows: execute key generation of Ψ_{Enc} as $(\text{pk}_0, \Pi_{\text{pk}_0}, \text{sk}_0 := \text{sk}'_0) \leftarrow \text{KG}(1^\kappa)$; run CRS generator of NIZK argument Ψ_{NIZK} and sample $(\text{c}\vec{r}s_0, \Pi_{\text{c}\vec{r}s_0}, \vec{t}s_0 := \vec{t}s'_0) \leftarrow \text{Gen}_{\text{crs}}(\mathbf{R}_L, \xi_{\mathbf{R}_L})$, where $\vec{t}s_0$ is the simulation trapdoor associated with $\text{c}\vec{r}s_0$; set $(\hat{c\vec{r}s}_0 \parallel \hat{\Pi}_{\hat{c\vec{r}s}_0} \parallel \hat{t\vec{e}}_0) := ((\text{c}\vec{r}s_0, \text{pk}_0) \parallel (\Pi_{\text{c}\vec{r}s_0}, \Pi_{\text{pk}_0}) \parallel \vec{t}s_0 \parallel \text{sk}_0)$; where $\hat{\Pi}_{\hat{c\vec{r}s}_0}$ is the proof of well-formedness of $\hat{c\vec{r}s}_0$, $\hat{t\vec{e}}_0$ is the simulation trapdoor associated with $\hat{c\vec{r}s}_0$, and $\hat{t\vec{e}}_0$ is the extraction trapdoor associated with $\hat{c\vec{r}s}_0$; Return $(\hat{c\vec{r}s}_0, \hat{\Pi}_{\hat{c\vec{r}s}_0})$.

CRS Updating, $(\hat{c\vec{r}s}_i, \hat{\Pi}_{\hat{c\vec{r}s}_i}) \leftarrow \hat{\text{CU}}(\mathbf{R}_L, \xi_{\mathbf{R}_L}, \hat{c\vec{r}s}_{i-1})$: Given $(\mathbf{R}_L, \xi_{\mathbf{R}_L}) \in \text{im}(\mathcal{R}(1^\kappa))$, and $\hat{c\vec{r}s}_{i-1}$ as an input CRS, act as follows: Parse $\hat{c\vec{r}s}_{i-1} := (\text{c}\vec{r}s_{i-1}, \text{pk}_{i-1})$; execute $(\text{c}\vec{r}s_i, \Pi_{\text{c}\vec{r}s_i}) \leftarrow \text{CU}(\mathbf{R}_L, \xi_{\mathbf{R}_L}, \text{c}\vec{r}s_{i-1})$; run $(\text{pk}_i, \Pi_{\text{pk}_i}) \leftarrow \text{KU}(\text{pk}_{i-1})$; set $(\hat{c\vec{r}s}_i \parallel \hat{\Pi}_{\hat{c\vec{r}s}_i}) := ((\text{c}\vec{r}s_i, \text{pk}_i) \parallel (\Pi_{\text{c}\vec{r}s_i}, \Pi_{\text{pk}_i}))$, where $\hat{\Pi}_{\hat{c\vec{r}s}_i}$ is the proof of well-formedness of $\hat{c\vec{r}s}_i$; Return $(\hat{c\vec{r}s}_i, \hat{\Pi}_{\hat{c\vec{r}s}_i})$. Note that after each update, the simulation and extraction trapdoors are updated, for instance $\vec{t}s_i := \vec{t}s_i = \vec{t}s_{i-1} + \vec{t}s'_i$, and $\vec{t\vec{e}}_i := \vec{t\vec{e}}_i = \vec{t\vec{e}}_{i-1} + \vec{t\vec{e}}'_i := \text{sk}_{i-1} + \text{sk}'_i$, where $\vec{t}s'_i$ and $\vec{t\vec{e}}'_i$ are individual (simulation and extraction) trapdoors of the updater i , and $\vec{t}s_i$ and $\vec{t\vec{e}}_i$ are the trapdoors of the CRS after updating by i -th updater.

CRS Verify, $(\perp, 1) \leftarrow \hat{\text{CV}}(\hat{c\vec{r}s}_i, \hat{\Pi}_{\hat{c\vec{r}s}_i})$: Given $\hat{c\vec{r}s}_i := (\text{c}\vec{r}s_i, \text{pk}_i)$, and $\hat{\Pi}_{\hat{c\vec{r}s}_i} := (\Pi_{\text{c}\vec{r}s_i}, \Pi_{\text{pk}_i})$ act as follows: if $\hat{\text{CV}}(\text{c}\vec{r}s_i, \Pi_{\text{c}\vec{r}s_i}) = 1$ and $\text{KV}(\text{pk}_i, \Pi_{\text{pk}_i}) = 1$ return 1 (i.e., the updated $\hat{c\vec{r}s}_i$ is correctly formed), otherwise \perp .

Prover, $(\hat{\pi}, \perp) \leftarrow \hat{\text{P}}(\mathbf{R}_L, \xi_{\mathbf{R}_L}, \hat{c\vec{r}s}_i, x, w)$: Parse $\hat{c\vec{r}s}_i := (\text{c}\vec{r}s_i, \text{pk}_i)$; Return \perp if $(x, w) \notin \mathbf{R}_L$; sample $r \leftarrow_{\$} \{0, 1\}^\kappa$; compute encryption of witnesses $c = \text{Enc}(\text{pk}_i, w; r)$. Then execute prover P of the input NIZK argument Ψ_{NIZK} and generate $\pi \leftarrow \text{P}(\mathbf{R}_L, \xi_{\mathbf{R}_L}, \text{c}\vec{r}s_i, (x, c, \text{pk}_i), (w, r))$; and output $\hat{\pi} := (c, \pi)$.

Verifier, $(0, 1) \leftarrow \hat{\text{V}}(\mathbf{R}_L, \xi_{\mathbf{R}_L}, \hat{c\vec{r}s}_i, x, \hat{\pi})$: Parse $\hat{c\vec{r}s}_i := (\text{c}\vec{r}s_i, \text{pk}_i)$ and $\hat{\pi} := (c, \pi)$; call verifier of the input NIZK argument Ψ_{NIZK} as $\text{V}(\mathbf{R}_L, \xi_{\mathbf{R}_L}, \text{c}\vec{r}s_i, (x, c, \text{pk}_i), \pi)$ and returns 1 if $((x, c, \text{pk}_i), (w, r)) \in \mathbf{R}_L$, otherwise it responses by 0.

Simulator, $(\hat{\pi}) \leftarrow \hat{\text{Sim}}(\mathbf{R}_L, \xi_{\mathbf{R}_L}, \hat{c\vec{r}s}_i, x, \hat{t\vec{s}}_i)$: Parse $\hat{c\vec{r}s}_i := (\text{c}\vec{r}s_i, \text{pk}_i)$ and $\hat{t\vec{s}}_i := \vec{t\vec{s}}_i$; sample $z, r \leftarrow_{\$} \{0, 1\}^\kappa$; compute $c = \text{Enc}(\text{pk}_i, z; r)$; execute simulator of the input NIZK argument Ψ_{NIZK} and generate $\pi \leftarrow \text{Sim}(\mathbf{R}_L, \xi_{\mathbf{R}_L}, \text{c}\vec{r}s_i, (x, c, \text{pk}_i), \vec{t\vec{s}}_i)$; and output $\hat{\pi} := (c, \pi)$.

Extractor, $(w) \leftarrow \hat{\text{Ext}}(\mathbf{R}_L, \xi_{\mathbf{R}_L}, \hat{c\vec{r}s}_i, \hat{t\vec{e}}_i, x, \hat{\pi})$: Parse $\hat{\pi} := (c, \pi)$ and $\hat{t\vec{e}}_i := \text{sk}_i$; extract $w \leftarrow \text{Dec}(\text{sk}_i, c)$; output w .

Figure 9.3: TIRAMISU, a construction for building BB-SE NIZK argument $\hat{\Psi}_{\text{NIZK}}$ with updatable CRS.

Publications

International Conferences

- [BSW24] C. Badertscher, M. Sedaghat, and H. Waldner. “Unlinkable Policy-Compliant Signatures for Compliant and Decentralized Anonymous Payments”. In: *Cryptology ePrint Archive, Paper 2023/1070 (to appear in PETS’2024)* (2024). URL: <https://eprint.iacr.org/2023/1070>
- [BMS23] K. Bagheri, A. Mertens, and M. Sedaghat. “Benchmarking the Setup of Updatable zk-SNARKs”. In: *8th International Conference on Cryptology and Information Security in Latin America, LATINCRYPT*. Ed. by A. Aly and M. Tibouchi. Vol. 14168. Lecture Notes in Computer Science. Springer, 2023, pp. 375–396. DOI: [10.1007/978-3-031-44469-2_19](https://doi.org/10.1007/978-3-031-44469-2_19)
- [BS21] K. Bagheri and M. Sedaghat. “Tiramisu: Black-Box Simulation Extractable NIZKs in the Updatable CRS Model”. In: *Cryptology and Network Security (CANS)*. Ed. by M. Conti, M. Stevens, and S. Krenn. Cham: Springer International Publishing, 2021, pp. 531–551. ISBN: 978-3-030-92548-2. DOI: [10.1007/978-3-030-92548-2_28](https://doi.org/10.1007/978-3-030-92548-2_28)
- [Bal+24] F. Baldimtsi, K. K. Chalkias, Y. Ji, J. Lindstrøm, D. Maram, B. Riva, A. Roy, M. Sedaghat, and J. Wang. “zkLogin: Privacy-Preserving Blockchain Authentication with Existing Credentials”. In: *The Science of Blockchain Conference (SBC) 2024 (To be presented)* (2024). <https://arxiv.org/pdf/2401.11735>
- [Cri+23] E. Crites, M. Kohlweiss, B. Preneel, M. Sedaghat, and D. Slamanig. “Threshold Structure-Preserving Signatures”. In: *Advances in Cryptology – ASIACRYPT 2023*. Ed. by J. Guo and R. Steinfeld. Singapore: Springer Nature Singapore, 2023, pp. 348–382. ISBN: 978-981-99-8724-5. DOI: [10.1007/978-981-99-8724-5_11](https://doi.org/10.1007/978-981-99-8724-5_11)

- [Mad+23] A. Madhusudan, M. Sedaghat, S. Tiwari, K. Cong, and B. Preneel. “Reusable, Instant and Private Payment Guarantees for Cryptocurrencies”. In: *Information Security and Privacy - 28th Australasian Conference, ACISP 2023, Brisbane, QLD, Australia, July 5-7, 2023, Proceedings*. Ed. by L. Simpson and M. A. R. Baee. Vol. 13915. Lecture Notes in Computer Science. Springer, 2023, pp. 580–605. DOI: [10.1007/978-3-031-35486-1_25](https://doi.org/10.1007/978-3-031-35486-1_25)
- [Mit+24] A. Mitrokotsa, S. Mukherjee, M. Sedaghat, D. Slamanig, and J. Tomy. “Threshold Structure-Preserving Signatures: Strong and Adaptive Security Under Standard Assumptions”. In: *Public-Key Cryptography – PKC 2024*. Ed. by Q. Tang and V. Teague. Cham: Springer Nature Switzerland, 2024, pp. 163–195. ISBN: 978-3-031-57718-5. DOI: [10.1007/978-3-031-57718-5_6](https://doi.org/10.1007/978-3-031-57718-5_6)
- [SP21] M. Sedaghat and B. Preneel. “Cross-Domain Attribute-Based Access Control Encryption”. In: *Cryptology and Network Security (CANS)*. Ed. by M. Conti, M. Stevens, and S. Krenn. Springer International Publishing, 2021, pp. 3–23. DOI: [10.1007/978-3-030-92548-2_1](https://doi.org/10.1007/978-3-030-92548-2_1)

Journals

- [Agh+22] S. F. Aghili, M. Sedaghat, D. Singelée, and M. Gupta. “MLS-ABAC: Efficient Multi-Level Security Attribute-Based Access Control scheme”. In: *Future Generation Computer Systems* (2022). ISSN: 0167-739X. DOI: <https://doi.org/10.1016/j.future.2022.01.003>. URL: <https://www.sciencedirect.com/science/article/pii/S0167739X22000115>

References

- [Abd+19] B. Abdolmaleki, K. Baghery, H. Lipmaa, J. Siim, and M. Zajac. “UC-Secure CRS Generation for SNARKs”. In: *AFRICACRYPT 19*. Ed. by J. Buchmann, A. Nitaj, and T.-e. Rachidi. Vol. 11627. LNCS. Springer, Heidelberg, July 2019, pp. 99–117. DOI: [10.1007/978-3-030-23696-0_6](https://doi.org/10.1007/978-3-030-23696-0_6)
- [Abd+17] B. Abdolmaleki, K. Baghery, H. Lipmaa, and M. Zajac. “A Subversion-Resistant SNARK”. In: *ASIACRYPT 2017, Part III*. Ed. by T. Takagi and T. Peyrin. Vol. 10626. LNCS. Springer, Heidelberg, Dec. 2017, pp. 3–33. DOI: [10.1007/978-3-319-70700-6_1](https://doi.org/10.1007/978-3-319-70700-6_1)
- [Abd+23] B. Abdolmaleki, N. Glaeser, S. Ramacher, and D. Slamanig. “Circuit-Succinct Universally-Composable NIZKs with Updatable CRS”. In: (2023). <https://eprint.iacr.org/2023/097>. URL: <https://eprint.iacr.org/2023/097>
- [ARS20] B. Abdolmaleki, S. Ramacher, and D. Slamanig. “Lift-and-Shift: Obtaining Simulation Extractable Subversion and Updatable SNARKs Generically”. In: *ACM CCS 2020*. Ed. by J. Ligatti, X. Ou, J. Katz, and G. Vigna. ACM Press, Nov. 2020, pp. 1987–2005. DOI: [10.1145/3372297.3417228](https://doi.org/10.1145/3372297.3417228)
- [Abe+18a] M. Abe, M. Ambrona, M. Ohkubo, and M. Tibouchi. “Lower Bounds on Structure-Preserving Signatures for Bilateral Messages”. In: *SCN 18*. Ed. by D. Catalano and R. De Prisco. Vol. 11035. LNCS. Springer, Heidelberg, Sept. 2018, pp. 3–22. DOI: [10.1007/978-3-319-98113-0_1](https://doi.org/10.1007/978-3-319-98113-0_1)
- [Abe+12] M. Abe, M. Chase, B. David, M. Kohlweiss, R. Nishimaki, and M. Ohkubo. “Constant-Size Structure-Preserving Signatures: Generic Constructions and Simple Assumptions”. In: *ASIACRYPT 2012*. Ed. by X. Wang and K. Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 4–24. DOI: [10.1007/978-3-642-34961-4_3](https://doi.org/10.1007/978-3-642-34961-4_3)

- [Abe+11a] M. Abe, S. S. M. Chow, K. Haralambiev, and M. Ohkubo. “Double-Trapdoor Anonymous Tags for Traceable Signatures”. In: *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*. Ed. by J. López and G. Tsudik. Vol. 6715. Lecture Notes in Computer Science. 2011, pp. 183–200. DOI: [10.1007/978-3-642-21554-4_11](https://doi.org/10.1007/978-3-642-21554-4_11). URL: https://doi.org/10.1007/978-3-642-21554-4_11
- [Abe+10] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. “Structure-Preserving Signatures and Commitments to Group Elements”. In: *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*. Ed. by T. Rabin. Vol. 6223. Lecture Notes in Computer Science. Springer, 2010, pp. 209–236. DOI: [10.1007/978-3-642-14623-7_12](https://doi.org/10.1007/978-3-642-14623-7_12). URL: https://doi.org/10.1007/978-3-642-14623-7_12
- [Abe+11b] M. Abe, J. Groth, K. Haralambiev, and M. Ohkubo. “Optimal Structure-Preserving Signatures in Asymmetric Bilinear Groups”. In: *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*. Ed. by P. Rogaway. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 649–666. DOI: [10.1007/978-3-642-22792-9_37](https://doi.org/10.1007/978-3-642-22792-9_37). URL: https://doi.org/10.1007/978-3-642-22792-9_37
- [AGO11] M. Abe, J. Groth, and M. Ohkubo. “Separating Short Structure-Preserving Signatures from Non-interactive Assumptions”. In: *ASIACRYPT 2011*. Ed. by D. H. Lee and X. Wang. Vol. 7073. LNCS. Springer, Heidelberg, Dec. 2011, pp. 628–646. DOI: [10.1007/978-3-642-25385-0_34](https://doi.org/10.1007/978-3-642-25385-0_34)
- [Abe+14] M. Abe, J. Groth, M. Ohkubo, and M. Tibouchi. “Unified, Minimal and Selectively Randomizable Structure-Preserving Signatures”. In: *TCC 2014*. Ed. by Y. Lindell. Vol. 8349. LNCS. Springer, Heidelberg, Feb. 2014, pp. 688–712. DOI: [10.1007/978-3-642-54242-8_29](https://doi.org/10.1007/978-3-642-54242-8_29)
- [Abe+17] M. Abe, D. Hofheinz, R. Nishimaki, M. Ohkubo, and J. Pan. “Compact Structure-Preserving Signatures with Almost Tight Security”. In: *CRYPTO 2017, Part II*. Ed. by J. Katz and H. Shacham. Vol. 10402. LNCS. Springer, Heidelberg, Aug. 2017, pp. 548–580. DOI: [10.1007/978-3-319-63715-0_19](https://doi.org/10.1007/978-3-319-63715-0_19)
- [Abe+19] M. Abe, C. S. Jutla, M. Ohkubo, J. Pan, A. Roy, and Y. Wang. “Shorter QA-NIZK and SPS with Tighter Security”. In: *ASIACRYPT 2019, Part III*. Ed. by S. D. Galbraith and S. Moriai.

- Vol. 11923. LNCS. Springer, Heidelberg, Dec. 2019, pp. 669–699. DOI: [10.1007/978-3-030-34618-8_23](https://doi.org/10.1007/978-3-030-34618-8_23)
- [Abe+18b] M. Abe, C. S. Jutla, M. Ohkubo, and A. Roy. “Improved (Almost) Tightly-Secure Simulation-Sound QA-NIZK with Applications”. In: *ASIACRYPT 2018, Part I*. Ed. by T. Peyrin and S. Galbraith. Vol. 11272. LNCS. Springer, Heidelberg, Dec. 2018, pp. 627–656. DOI: [10.1007/978-3-030-03326-2_21](https://doi.org/10.1007/978-3-030-03326-2_21)
- [Abr+22] D. Abram, A. Nof, C. Orlandi, P. Scholl, and O. Shlomovits. “Low-Bandwidth Threshold ECDSA via Pseudorandom Correlation Generators”. In: *2022 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2022, pp. 2554–2572. DOI: [10.1109/SP46214.2022.9833559](https://doi.org/10.1109/SP46214.2022.9833559)
- [Aki+13] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin. “Charm: a framework for rapidly prototyping cryptosystems”. In: *Journal of Cryptographic Engineering* 3.2 (2013), pp. 111–128
- [ADN06] J. F. Almansa, I. Damgard, and J. B. Nielsen. “Simplified Threshold RSA with Adaptive and Proactive Security”. In: *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*. Ed. by S. Vaudenay. Vol. 4004. Lecture Notes in Computer Science. Springer, 2006, pp. 593–611
- [AHI11] B. Applebaum, D. Harnik, and Y. Ishai. “Semantic Security under Related-Key Attacks and Applications”. In: *ICS 2011*. Ed. by B. Chazelle. Tsinghua University Press, Jan. 2011, pp. 45–60
- [AF24] Architecture and R. Framework. “European Digital Identity Wallet Architecture and Reference Framework”. In: 2024. URL: <https://eu-digital-identity-wallet.github.io/eudi-doc-architecture-and-reference-framework/1.4.0/>
- [AB19] S. Atapoor and K. Baghery. “Simulation Extractability in Groth’s zk-SNARK”. In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26-27, 2019, Proceedings*. Ed. by C. Perez-Sola, G. Navarro-Arribas, A. Biryukov, and J. Garcia-Alfaro. Vol. 11737. Lecture Notes in Computer Science. Springer, 2019, pp. 336–354. DOI: [10.1007/978-3-030-31500-9_22](https://doi.org/10.1007/978-3-030-31500-9_22). URL: https://doi.org/10.1007/978-3-030-31500-9_22

- [ALP12] N. Attrapadung, B. Libert, and T. Peters. “Computing on Authenticated Data: New Privacy Definitions and Constructions”. In: *ASIACRYPT 2012*. Ed. by X. Wang and K. Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 367–385. DOI: [10.1007/978-3-642-34961-4_23](https://doi.org/10.1007/978-3-642-34961-4_23)
- [ABK18] B. Auerbach, M. Bellare, and E. Kiltz. “Public-Key Encryption Resistant to Parameter Subversion and Its Realization from Efficiently-Embeddable Groups”. In: *PKC 2018, Part I*. Ed. by M. Abdalla and R. Dahab. Vol. 10769. LNCS. Springer, Heidelberg, Mar. 2018, pp. 348–377. DOI: [10.1007/978-3-319-76578-5_12](https://doi.org/10.1007/978-3-319-76578-5_12)
- [BL22] R. Bacho and J. Loss. “On the Adaptive Security of the Threshold BLS Signature Scheme”. In: *ACM CCS 2022*. Ed. by H. Yin, A. Stavrou, C. Cremers, and E. Shi. ACM Press, Nov. 2022, pp. 193–207. DOI: [10.1145/3548606.3560656](https://doi.org/10.1145/3548606.3560656)
- [Bac+24] R. Bacho, J. Loss, S. Tessaro, B. Wagner, and C. Zhu. “Twinkle: Threshold signatures from DDH with full adaptive security”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2024, pp. 429–459
- [BMW21] C. Badertscher, C. Matt, and H. Waldner. “Policy-Compliant Signatures”. In: *TCC 2021, Part III*. Ed. by K. Nissim and B. Waters. Vol. 13044. LNCS. Springer, Heidelberg, Nov. 2021, pp. 350–381. DOI: [10.1007/978-3-030-90456-2_12](https://doi.org/10.1007/978-3-030-90456-2_12)
- [Bag19a] K. Baghery. “On the Efficiency of Privacy-Preserving Smart Contract Systems”. In: *AFRICACRYPT 19*. Ed. by J. Buchmann, A. Nitaj, and T.-e. Rachidi. Vol. 11627. LNCS. Springer, Heidelberg, July 2019, pp. 118–136. DOI: [10.1007/978-3-030-23696-0_7](https://doi.org/10.1007/978-3-030-23696-0_7)
- [Bag19b] K. Baghery. “Subversion-Resistant Simulation (Knowledge) Sound NIZKs”. In: *17th IMA International Conference on Cryptography and Coding*. Ed. by M. Albrecht. Vol. 11929. LNCS. Springer, Heidelberg, Dec. 2019, pp. 42–63. DOI: [10.1007/978-3-030-35199-1_3](https://doi.org/10.1007/978-3-030-35199-1_3)
- [Bag20] K. Baghery. “Subversion-Resistant Commitment Schemes: Definitions and Constructions”. In: *Security and Trust Management - 16th International Workshop, STM 2020, Guildford, UK, September 17-18, 2020, Proceedings*. Ed. by K. Markantonakis and M. Petrocchi. Vol. 12386. Lecture Notes in Computer Science. Springer, 2020, pp. 106–122. DOI: [10.1007/978-3-030-59817-4_7](https://doi.org/10.1007/978-3-030-59817-4_7). URL: https://doi.org/10.1007/978-3-030-59817-4_7

- [BB22] K. Baghery and N. G. Bardeh. “Updatable NIZKs from Non-Interactive Zaps”. In: *CANS 22*. Ed. by A. R. Beresford, A. Patra, and E. Bellini. Vol. 13641. LNCS. Springer, Heidelberg, Nov. 2022, pp. 23–43. DOI: [10.1007/978-3-031-20974-1_2](https://doi.org/10.1007/978-3-031-20974-1_2)
- [Bag+21] K. Baghery, M. Kohlweiss, J. Siim, and M. Volkhov. “Another Look at Extraction and Randomization of Groth’s zk-SNARK”. In: LNCS. Springer, Heidelberg, 2021, pp. 457–475. DOI: [10.1007/978-3-662-64322-8_22](https://doi.org/10.1007/978-3-662-64322-8_22)
- [BPR20] K. Baghery, Z. Pindado, and C. Ràfols. “Simulation Extractable Versions of Groth’s zk-SNARK Revisited”. In: *CANS 20*. Ed. by S. Krenn, H. Shulman, and S. Vaudenay. Vol. 12579. LNCS. Springer, Heidelberg, Dec. 2020, pp. 453–461. DOI: [10.1007/978-3-030-65411-5_22](https://doi.org/10.1007/978-3-030-65411-5_22)
- [BS20] K. Baghery and M. Sedaghat. “Tiramisu: Black-Box Simulation Extractable NIZKs in the Updatable CRS Model.” In: *Cryptology ePrint Archive, Report 2020/474* (2020). <https://eprint.iacr.org/2020/474>
- [BLS03] P. S. L. M. Barreto, B. Lynn, and M. Scott. “Constructing Elliptic Curves with Prescribed Embedding Degrees”. In: *SCN 02*. Ed. by S. Cimato, C. Galdi, and G. Persiano. Vol. 2576. LNCS. Springer, Heidelberg, Sept. 2003, pp. 257–267. DOI: [10.1007/3-540-36413-7_19](https://doi.org/10.1007/3-540-36413-7_19)
- [BN06] P. S. L. M. Barreto and M. Naehrig. “Pairing-Friendly Elliptic Curves of Prime Order”. In: *SAC 2005*. Ed. by B. Preneel and S. Tavares. Vol. 3897. LNCS. Springer, Heidelberg, Aug. 2006, pp. 319–331. DOI: [10.1007/11693383_22](https://doi.org/10.1007/11693383_22)
- [BN05] P. S. Barreto and M. Naehrig. “Pairing-friendly elliptic curves of prime order”. In: *International Workshop on Selected Areas in Cryptography*. Springer. 2005, pp. 319–331
- [Bar+15] G. Barthe, E. Fagerholm, D. Fiore, A. Scedrov, B. Schmidt, and M. Tibouchi. “Strongly-Optimal Structure Preserving Signatures from Type II Pairings: Synthesis and Lower Bounds”. In: *PKC 2015*. Ed. by J. Katz. Vol. 9020. LNCS. Springer, Heidelberg, Mar. 2015, pp. 355–376. DOI: [10.1007/978-3-662-46447-2_16](https://doi.org/10.1007/978-3-662-46447-2_16)
- [BFL20] B. Bauer, G. Fuchsbauer, and J. Loss. “A Classification of Computational Assumptions in the Algebraic Group Model”. In: *CRYPTO 2020, Part II*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12171. LNCS. Springer, Heidelberg, Aug. 2020, pp. 121–151. DOI: [10.1007/978-3-030-56880-1_5](https://doi.org/10.1007/978-3-030-56880-1_5)

- [Bel+09] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. “Randomizable Proofs and Delegatable Anonymous Credentials”. In: *CRYPTO 2009*. Ed. by S. Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 108–125. DOI: [10.1007/978-3-642-03356-8_7](https://doi.org/10.1007/978-3-642-03356-8_7)
- [Bel+22] M. Bellare, E. C. Crites, C. Komlo, M. Maller, S. Tessaro, and C. Zhu. “Better than Advertised Security for Non-interactive Threshold Signatures”. In: *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*. Ed. by Y. Dodis and T. Shrimpton. Vol. 13510. Lecture Notes in Computer Science. Springer, 2022, pp. 517–550. DOI: [10.1007/978-3-031-15985-5_18](https://doi.org/10.1007/978-3-031-15985-5_18). URL: https://doi.org/10.1007/978-3-031-15985-5_18
- [BFS16] M. Bellare, G. Fuchsbauer, and A. Scafuro. “NIZKs with an Untrusted CRS: Security in the Face of Parameter Subversion”. In: *ASIACRYPT 2016, Part II*. Ed. by J. H. Cheon and T. Takagi. Vol. 10032. LNCS. Springer, Heidelberg, Dec. 2016, pp. 777–804. DOI: [10.1007/978-3-662-53890-6_26](https://doi.org/10.1007/978-3-662-53890-6_26)
- [BGR98] M. Bellare, J. A. Garay, and T. Rabin. “Batch Verification with Applications to Cryptography and Checking”. In: *LATIN 1998*. Ed. by C. L. Lucchesi and A. V. Moura. Vol. 1380. LNCS. Springer, Heidelberg, Apr. 1998, pp. 170–191
- [BG90] M. Bellare and S. Goldwasser. “New Paradigms for Digital Signatures and Message Authentication Based on Non-Interactive Zero Knowledge Proofs”. In: *CRYPTO’89*. Ed. by G. Brassard. Vol. 435. LNCS. Springer, Heidelberg, Aug. 1990, pp. 194–211. DOI: [10.1007/0-387-34805-0_19](https://doi.org/10.1007/0-387-34805-0_19)
- [Bel+03] M. Bellare, C. Namprempe, D. Pointcheval, and M. Semanko. “The One-More-RSA-Inversion Problems and the Security of Chaum’s Blind Signature Scheme”. In: *Journal of Cryptology* 16.3 (June 2003), pp. 185–215. DOI: [10.1007/s00145-002-0120-1](https://doi.org/10.1007/s00145-002-0120-1)
- [BR93] M. Bellare and P. Rogaway. “Random Oracles are Practical: A Paradigm for Designing Efficient Protocols”. In: *ACM CCS 93*. Ed. by D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby. ACM Press, Nov. 1993, pp. 62–73. DOI: [10.1145/168588.168596](https://doi.org/10.1145/168588.168596)
- [Ben+14] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. “Zerocash: Decentralized Anonymous Payments from Bitcoin”. In: *2014 IEEE Symposium on Security*

- and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, 2014, pp. 459–474. DOI: [10.1109/SP.2014.36](https://doi.org/10.1109/SP.2014.36). URL: <https://doi.org/10.1109/SP.2014.36>
- [Ben+15] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza. “Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs”. In: *2015 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2015, pp. 287–304. DOI: [10.1109/SP.2015.25](https://doi.org/10.1109/SP.2015.25)
- [Ben+13] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. “Succinct Non-Interactive Arguments for a von Neumann Architecture”. In: (2013). <https://eprint.iacr.org/2013/879>
- [Ben+22] Benarroch, Brandão, Maller, and Tromer. *ZKProof. ZKProof Community Reference, Version 0.3*. Tech. rep. available at <https://docs.zkproof.org/reference>. Pub. by zkproof.org. July 2022., 2022
- [Bla79] G. R. Blakley. “Safeguarding Cryptographic Keys”. In: *Proceedings of AFIPS 1979 National Computer Conference* 48 (1979), pp. 313–317
- [Bla+11] O. Blazy, S. Canard, G. Fuchsbauer, A. Gouget, H. Sibert, and J. Traoré. “Achieving Optimal Anonymity in Transferable E-Cash with a Judge”. In: *AFRICACRYPT 11*. Ed. by A. Nitaj and D. Pointcheval. Vol. 6737. LNCS. Springer, Heidelberg, July 2011, pp. 206–223
- [BFM88] M. Blum, P. Feldman, and S. Micali. “Non-interactive zero-knowledge and its applications”. In: *Proceedings of the twentieth annual ACM symposium on Theory of computing*. ACM. 1988, pp. 103–112
- [Bol03] A. Boldyreva. “Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme”. In: *PKC 2003*. Ed. by Y. Desmedt. Vol. 2567. LNCS. Springer, Heidelberg, Jan. 2003, pp. 31–46. DOI: [10.1007/3-540-36288-6_3](https://doi.org/10.1007/3-540-36288-6_3)
- [Bon98] D. Boneh. “The decision Diffie-Hellman problem”. In: *Third Algorithmic Number Theory Symposium (ANTS)*. Vol. 1423. LNCS. Invited paper. Springer, Heidelberg, 1998
- [BDN18] D. Boneh, M. Drijvers, and G. Neven. “Compact Multi-signatures for Smaller Blockchains”. In: *ASIACRYPT 2018, Part II*. Ed. by T. Peyrin and S. Galbraith. Vol. 11273. LNCS. Springer, Heidelberg, Dec. 2018, pp. 435–464. DOI: [10.1007/978-3-030-03329-3_15](https://doi.org/10.1007/978-3-030-03329-3_15)

- [BF01] D. Boneh and M. K. Franklin. “Identity-Based Encryption from the Weil Pairing”. In: *CRYPTO 2001*. Ed. by J. Kilian. Vol. 2139. LNCS. Springer, Heidelberg, Aug. 2001, pp. 213–229. DOI: [10.1007/3-540-44647-8_13](https://doi.org/10.1007/3-540-44647-8_13)
- [BK22] D. Boneh and C. Komlo. “Threshold Signatures with Private Accountability”. In: *CRYPTO 2022, Part IV*. Ed. by Y. Dodis and T. Shrimpton. Vol. 13510. LNCS. Springer, Heidelberg, Aug. 2022, pp. 551–581. DOI: [10.1007/978-3-031-15985-5_19](https://doi.org/10.1007/978-3-031-15985-5_19)
- [BLS01] D. Boneh, B. Lynn, and H. Shacham. “Short Signatures from the Weil Pairing”. In: *ASIACRYPT 2001*. Ed. by C. Boyd. Vol. 2248. LNCS. Springer, Heidelberg, Dec. 2001, pp. 514–532. DOI: [10.1007/3-540-45682-1_30](https://doi.org/10.1007/3-540-45682-1_30)
- [BLS04] D. Boneh, B. Lynn, and H. Shacham. “Short Signatures from the Weil Pairing”. In: *Journal of Cryptology* 17.4 (Sept. 2004), pp. 297–319. DOI: [10.1007/s00145-004-0314-9](https://doi.org/10.1007/s00145-004-0314-9)
- [BPR22] D. Boneh, A. Partap, and L. Rotem. “Accountable Threshold Signatures with Proactive Refresh”. In: (2022). <https://eprint.iacr.org/2022/1656>
- [BS23] A. Bouez and K. Singh. “One Round Threshold ECDSA Without Roll Call”. In: *Topics in Cryptology - CT-RSA 2023 - Cryptographers’ Track at the RSA Conference 2023, San Francisco, CA, USA, April 24-27, 2023, Proceedings*. Ed. by M. Rosulek. Vol. 13871. Lecture Notes in Computer Science. Springer, 2023, pp. 389–414. DOI: [10.1007/978-3-031-30872-7_15](https://doi.org/10.1007/978-3-031-30872-7_15). URL: https://doi.org/10.1007/978-3-031-30872-7_15
- [Bow+20] S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, and H. Wu. “ZEXE: Enabling Decentralized Private Computation”. In: *2020 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2020, pp. 947–964. DOI: [10.1109/SP40000.2020.00050](https://doi.org/10.1109/SP40000.2020.00050)
- [BG18] S. Bowe and A. Gabizon. “Making Groth’s zk-SNARK Simulation Extractable in the Random Oracle Model”. In: (2018). <https://eprint.iacr.org/2018/187>
- [BGG19] S. Bowe, A. Gabizon, and M. D. Green. “A Multi-party Protocol for Constructing the Public Parameters of the Pinocchio zk-SNARK”. In: *FC 2018 Workshops*. Ed. by A. Zohar, I. Eyal, V. Teague, J. Clark, A. Bracciali, F. Pintore, and M. Sala. Vol. 10958. LNCS. Springer, Heidelberg, Mar. 2019, pp. 64–77. DOI: [10.1007/978-3-662-58820-8_5](https://doi.org/10.1007/978-3-662-58820-8_5)

- [BGM17] S. Bowe, A. Gabizon, and I. Miers. “Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model”. In: (2017). <https://eprint.iacr.org/2017/1050>
- [BP23] L. Brandão and R. Peralta. “NIST First Call for Multi-Party Threshold Schemes”. In: (2023)
- [BDV+20] L. T. Brandão, M. Davidson, A. Vassilev, et al. “NIST Roadmap Toward Criteria for Threshold Schemes for Cryptographic Primitives”. In: *National Institute of Standards and Technology Internal or Interagency Report 8214A*. 2020. DOI: <https://doi.org/10.6028/NIST.IR.8214A>
- [Bün+20] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. “Zether: Towards Privacy in a Smart Contract World”. In: *FC 2020*. Ed. by J. Bonneau and N. Heninger. Vol. 12059. LNCS. Springer, Heidelberg, Feb. 2020, pp. 423–443. DOI: [10.1007/978-3-030-51280-4_23](https://doi.org/10.1007/978-3-030-51280-4_23)
- [Cam+20] J. Camenisch, M. Drijvers, A. Lehmann, G. Neven, and P. Towa. “Short Threshold Dynamic Group Signatures”. In: *SCN 20*. Ed. by C. Galdi and V. Kolesnikov. Vol. 12238. LNCS. Springer, Heidelberg, Sept. 2020, pp. 401–423. DOI: [10.1007/978-3-030-57990-6_20](https://doi.org/10.1007/978-3-030-57990-6_20)
- [CDH12] J. Camenisch, M. Dubovitskaya, and K. Haralambiev. “Efficient Structure-Preserving Signature Scheme from Standard Assumptions”. In: *SCN 12*. Ed. by I. Visconti and R. D. Prisco. Vol. 7485. LNCS. Springer, Heidelberg, Sept. 2012, pp. 76–94. DOI: [10.1007/978-3-642-32928-9_5](https://doi.org/10.1007/978-3-642-32928-9_5)
- [Cam+15] J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss. “Composable and Modular Anonymous Credentials: Definitions and Practical Constructions”. In: *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*. Ed. by T. Iwata and J. H. Cheon. Vol. 9453. Lecture Notes in Computer Science. Springer, 2015, pp. 262–288. DOI: [10.1007/978-3-662-48800-3_11](https://doi.org/10.1007/978-3-662-48800-3_11). URL: https://doi.org/10.1007/978-3-662-48800-3_11
- [CL03] J. Camenisch and A. Lysyanskaya. “A Signature Scheme with Efficient Protocols”. In: *SCN 02*. Ed. by S. Cimato, C. Galdi, and G. Persiano. Vol. 2576. LNCS. Springer, Heidelberg, Sept. 2003, pp. 268–289. DOI: [10.1007/3-540-36413-7_20](https://doi.org/10.1007/3-540-36413-7_20)

- [CL04] J. Camenisch and A. Lysyanskaya. “Signature Schemes and Anonymous Credentials from Bilinear Maps”. In: *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*. Ed. by M. K. Franklin. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 56–72. DOI: [10.1007/978-3-540-28628-8_4](https://doi.org/10.1007/978-3-540-28628-8_4). URL: https://doi.org/10.1007/978-3-540-28628-8_4
- [Cam+21] M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez. “Lunar: A Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions”. In: *ASIACRYPT 2021, Part III*. Ed. by M. Tibouchi and H. Wang. Vol. 13092. LNCS. Springer, Heidelberg, Dec. 2021, pp. 3–33. DOI: [10.1007/978-3-030-92078-4_1](https://doi.org/10.1007/978-3-030-92078-4_1)
- [CFQ19] M. Campanelli, D. Fiore, and A. Querol. “LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs”. In: *ACM CCS 2019*. Ed. by L. Cavallaro, J. Kinder, X. Wang, and J. Katz. ACM Press, Nov. 2019, pp. 2075–2092. DOI: [10.1145/3319535.3339820](https://doi.org/10.1145/3319535.3339820)
- [Cam+23] M. Campanelli, C. Ganesh, H. Khoshakhlagh, and J. Siim. “Impossibilities in Succinct Arguments: Black-Box Extraction and More”. In: *Progress in Cryptology - AFRICACRYPT 2023 - 14th International Conference on Cryptology in Africa, Sousse, Tunisia, July 19-21, 2023, Proceedings*. Ed. by N. E. Mrabet, L. D. Feo, and S. Duquesne. Vol. 14064. Lecture Notes in Computer Science. Springer, 2023, pp. 465–489. DOI: [10.1007/978-3-031-37679-5_20](https://doi.org/10.1007/978-3-031-37679-5_20). URL: https://doi.org/10.1007/978-3-031-37679-5_20
- [Can01] R. Canetti. “Universally Composable Security: A New Paradigm for Cryptographic Protocols”. In: *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. IEEE Computer Society, 2001, pp. 136–145. DOI: [10.1109/SFCS.2001.959888](https://doi.org/10.1109/SFCS.2001.959888). URL: <https://doi.org/10.1109/SFCS.2001.959888>
- [Can+96] R. Canetti, U. Feige, O. Goldreich, and M. Naor. “Adaptively Secure Multi-Party Computation”. In: *28th ACM STOC*. ACM Press, May 1996, pp. 639–648. DOI: [10.1145/237814.238015](https://doi.org/10.1145/237814.238015)
- [Can+20] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled. “UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts”. In: *ACM CCS 2020*. Ed. by J. Ligatti, X. Ou, J. Katz, and G. Vigna. ACM Press, Nov. 2020, pp. 1769–1787. DOI: [10.1145/3372297.3423367](https://doi.org/10.1145/3372297.3423367)

- [Can+99] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. “Adaptive Security for Threshold Cryptosystems”. In: *CRYPTO’99*. Ed. by M. J. Wiener. Vol. 1666. LNCS. Springer, Heidelberg, Aug. 1999, pp. 98–115. DOI: [10.1007/3-540-48405-1_7](https://doi.org/10.1007/3-540-48405-1_7)
- [CHK03] R. Canetti, S. Halevi, and J. Katz. “A Forward-Secure Public-Key Encryption Scheme”. In: *EUROCRYPT 2003*. Ed. by E. Biham. Vol. 2656. LNCS. Springer, Heidelberg, May 2003, pp. 255–271. DOI: [10.1007/3-540-39200-9_16](https://doi.org/10.1007/3-540-39200-9_16)
- [Cha+12] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. “Malleable Proof Systems and Applications”. In: *EUROCRYPT 2012*. Ed. by D. Pointcheval and T. Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 281–300. DOI: [10.1007/978-3-642-29011-4_18](https://doi.org/10.1007/978-3-642-29011-4_18)
- [CL06] M. Chase and A. Lysyanskaya. “On Signatures of Knowledge”. In: *CRYPTO 2006*. Ed. by C. Dwork. Vol. 4117. LNCS. Springer, Heidelberg, Aug. 2006, pp. 78–96. DOI: [10.1007/11818175_5](https://doi.org/10.1007/11818175_5)
- [Cha85] D. Chaum. “Security Without Identification: Transaction Systems to Make Big Brother Obsolete”. In: *Commun. ACM* 28.10 (1985), pp. 1030–1044. DOI: [10.1145/4372.4373](https://doi.org/10.1145/4372.4373). URL: <https://doi.org/10.1145/4372.4373>
- [CL24] Y.-H. Chen and Y. Lindell. “Optimizing and Implementing Fischlin’s Transform for UC-Secure Zero-Knowledge”. In: (2024). <https://eprint.iacr.org/2024/526>. URL: <https://eprint.iacr.org/2024/526>
- [CF24] A. Chiesa and G. Fenzi. “zkSNARKs in the ROM with Unconditional UC-Security”. In: (2024). <https://eprint.iacr.org/2024/724>. URL: <https://eprint.iacr.org/2024/724>
- [Chi+19] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward. “Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS”. In: (2019). <https://eprint.iacr.org/2019/1047>
- [CLP22] A. Connolly, P. Lafourcade, and O. Perez-Kempner. “Improved Constructions of Anonymous Credentials from Structure-Preserving Signatures on Equivalence Classes”. In: *PKC 2022, Part I*. Ed. by G. Hanaoka, J. Shikata, and Y. Watanabe. Vol. 13177. LNCS. Springer, Heidelberg, Mar. 2022, pp. 409–438. DOI: [10.1007/978-3-030-97121-2_15](https://doi.org/10.1007/978-3-030-97121-2_15)

- [CH20] G. Couteau and D. Hartmann. “Shorter Non-interactive Zero-Knowledge Arguments and ZAPs for Algebraic Languages”. In: *CRYPTO 2020, Part III*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12172. LNCS. Springer, Heidelberg, Aug. 2020, pp. 768–798. DOI: [10.1007/978-3-030-56877-1_27](https://doi.org/10.1007/978-3-030-56877-1_27)
- [CDN01] R. Cramer, I. Damgard, and J. B. Nielsen. “Multiparty Computation from Threshold Homomorphic Encryption”. In: *EUROCRYPT 2001*. Ed. by B. Pfitzmann. Vol. 2045. LNCS. Springer, Heidelberg, May 2001, pp. 280–299. DOI: [10.1007/3-540-44987-6_18](https://doi.org/10.1007/3-540-44987-6_18)
- [Cra+96] R. Cramer, M. K. Franklin, B. Schoenmakers, and M. Yung. “Multi-Authority Secret-Ballot Elections with Linear Work”. In: *EUROCRYPT’96*. Ed. by U. M. Maurer. Vol. 1070. LNCS. Springer, Heidelberg, May 1996, pp. 72–83. DOI: [10.1007/3-540-68339-9_7](https://doi.org/10.1007/3-540-68339-9_7)
- [CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers. “A Secure and Optimally Efficient Multi-Authority Election Scheme”. In: *EUROCRYPT’97*. Ed. by W. Fumy. Vol. 1233. LNCS. Springer, Heidelberg, May 1997, pp. 103–118. DOI: [10.1007/3-540-69053-0_9](https://doi.org/10.1007/3-540-69053-0_9)
- [Cri+22] E. Crites, M. Kohlweiss, B. Preneel, M. Sedaghat, and D. Slamanig. “Threshold Structure-Preserving Signatures”. In: (2022). <https://eprint.iacr.org/2022/839>. URL: <https://eprint.iacr.org/2022/839>
- [CKM23] E. C. Crites, C. Komlo, and M. Maller. “Fully Adaptive Schnorr Threshold Signatures”. In: *CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*. Ed. by H. Handschuh and A. Lysyanskaya. Vol. 14081. LNCS. Springer, 2023, pp. 678–709. DOI: [10.1007/978-3-031-38557-5_22](https://doi.org/10.1007/978-3-031-38557-5_22)
- [CL19] E. C. Crites and A. Lysyanskaya. “Delegatable Anonymous Credentials from Mercurial Signatures”. In: *CT-RSA 2019*. Ed. by M. Matsui. Vol. 11405. LNCS. Springer, Heidelberg, Mar. 2019, pp. 535–555. DOI: [10.1007/978-3-030-12612-4_27](https://doi.org/10.1007/978-3-030-12612-4_27)
- [Dal+20] A. P. K. Dalskov, C. Orlandi, M. Keller, K. Shrishak, and H. Shulman. “Securing DNSSEC Keys via Threshold ECDSA from Generic MPC”. In: *ESORICS 2020, Part II*. Ed. by L. Chen, N. Li, K. Liang, and S. A. Schneider. Vol. 12309. LNCS. Springer, Heidelberg, Sept. 2020, pp. 654–673. DOI: [10.1007/978-3-030-59013-0_32](https://doi.org/10.1007/978-3-030-59013-0_32)

- [Dam92] I. Damgård. “Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks”. In: *CRYPTO’91*. Ed. by J. Feigenbaum. Vol. 576. LNCS. Springer, Heidelberg, Aug. 1992, pp. 445–456. DOI: [10.1007/3-540-46766-1_36](https://doi.org/10.1007/3-540-46766-1_36)
- [DHO16] I. Damgård, H. Haagh, and C. Orlandi. “Access Control Encryption: Enforcing Information Flow with Cryptography”. In: *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*. Ed. by M. Hirt and A. D. Smith. Vol. 9986. Lecture Notes in Computer Science. 2016, pp. 547–576. DOI: [10.1007/978-3-662-53644-5_21](https://doi.org/10.1007/978-3-662-53644-5_21). URL: https://doi.org/10.1007/978-3-662-53644-5_21
- [DK01] I. Damgård and M. Koprowski. “Practical Threshold RSA Signatures without a Trusted Dealer”. In: *EUROCRYPT 2001*. Ed. by B. Pfitzmann. Vol. 2045. LNCS. Springer, Heidelberg, May 2001, pp. 152–165. DOI: [10.1007/3-540-44987-6_10](https://doi.org/10.1007/3-540-44987-6_10)
- [DN03] I. Damgård and J. B. Nielsen. “Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption”. In: *CRYPTO 2003*. Ed. by D. Boneh. Vol. 2729. LNCS. Springer, Heidelberg, Aug. 2003, pp. 247–264. DOI: [10.1007/978-3-540-45146-4_15](https://doi.org/10.1007/978-3-540-45146-4_15)
- [Dam+12] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. “Multiparty Computation from Somewhat Homomorphic Encryption”. In: *CRYPTO 2012*. Ed. by R. Safavi-Naini and R. Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 643–662. DOI: [10.1007/978-3-642-32009-5_38](https://doi.org/10.1007/978-3-642-32009-5_38)
- [DR23] S. Das and L. Ren. “Adaptively Secure BLS Threshold Signatures from DDH and co-CDH”. In: 2023. URL: <https://eprint.iacr.org/2023/1553>
- [Daz+19] V. Daza, A. González, Z. Pindado, C. Ràfols, and J. Silva. “Shorter Quadratic QA-NIZK Proofs”. In: *PKC 2019, Part I*. Ed. by D. Lin and K. Sako. Vol. 11442. LNCS. Springer, Heidelberg, Apr. 2019, pp. 314–343. DOI: [10.1007/978-3-030-17253-4_11](https://doi.org/10.1007/978-3-030-17253-4_11)
- [DRZ20] V. Daza, C. Ràfols, and A. Zacharakis. “Updateable Inner Product Argument with Logarithmic Verifier and Applications”. In: *PKC 2020, Part I*. Ed. by A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas. Vol. 12110. LNCS. Springer, Heidelberg, May 2020, pp. 527–557. DOI: [10.1007/978-3-030-45374-9_18](https://doi.org/10.1007/978-3-030-45374-9_18)

- [De +94] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. “How to Share a Function Securely”. In: *26th ACM STOC*. ACM Press, May 1994, pp. 522–533. DOI: [10.1145/195058.195405](https://doi.org/10.1145/195058.195405)
- [De +01] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. “Robust Non-interactive Zero Knowledge”. In: *CRYPTO 2001*. Ed. by J. Kilian. Vol. 2139. LNCS. Springer, Heidelberg, Aug. 2001, pp. 566–598. DOI: [10.1007/3-540-44647-8_33](https://doi.org/10.1007/3-540-44647-8_33)
- [Den+21] Y. Deng, S. Ma, X. Zhang, H. Wang, X. Song, and X. Xie. “Promise Σ -Protocol: How to Construct Efficient Threshold ECDSA from Encryptions Based on Class Groups”. In: *ASIACRYPT 2021, Part IV*. Ed. by M. Tibouchi and H. Wang. Vol. 13093. LNCS. Springer, Heidelberg, Dec. 2021, pp. 557–586. DOI: [10.1007/978-3-030-92068-5_19](https://doi.org/10.1007/978-3-030-92068-5_19)
- [DS16] D. Derler and D. Slamanig. “Key-Homomorphic Signatures and Applications to Multiparty Signatures”. In: (2016). <https://eprint.iacr.org/2016/792>
- [Des90] Y. Desmedt. “Making Conditionally Secure Cryptosystems Unconditionally Abuse-Free in a General Context”. In: *CRYPTO’89*. Ed. by G. Brassard. Vol. 435. LNCS. Springer, Heidelberg, Aug. 1990, pp. 6–16. DOI: [10.1007/0-387-34805-0_2](https://doi.org/10.1007/0-387-34805-0_2)
- [DF90] Y. Desmedt and Y. Frankel. “Threshold Cryptosystems”. In: *CRYPTO’89*. Ed. by G. Brassard. Vol. 435. LNCS. Springer, Heidelberg, Aug. 1990, pp. 307–315. DOI: [10.1007/0-387-34805-0_28](https://doi.org/10.1007/0-387-34805-0_28)
- [Dix24] C. Dixon. *Read Write Own: Building the Next Era of the Internet*. Random House New York, 2024
- [Dob+21] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schl  ffer. “Ascon v1. 2: Lightweight authenticated encryption and hashing”. In: *Journal of Cryptology* 34 (2021), pp. 1–42
- [Doe+18] J. Doerner, Y. Kondi, E. Lee, and a. shelat. “Secure Two-party Threshold ECDSA from ECDSA Assumptions”. In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 980–997. DOI: [10.1109/SP.2018.00036](https://doi.org/10.1109/SP.2018.00036)
- [Doe+19] J. Doerner, Y. Kondi, E. Lee, and a. shelat. “Threshold ECDSA from ECDSA Assumptions: The Multiparty Case”. In: *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2019, pp. 1051–1066. DOI: [10.1109/SP.2019.00024](https://doi.org/10.1109/SP.2019.00024)

- [Dri+19] M. Drijvers, K. Edalatnejad, B. Ford, E. Kiltz, J. Loss, G. Neven, and I. Stepanovs. “On the Security of Two-Round Multi-Signatures”. In: *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2019, pp. 1084–1101. DOI: [10.1109/SP.2019.00050](https://doi.org/10.1109/SP.2019.00050)
- [EGK14] A. El Kaafarani, E. Ghadafi, and D. Khader. “Decentralized Traceable Attribute-Based Signatures”. In: *CT-RSA 2014*. Ed. by J. Benaloh. Vol. 8366. LNCS. Springer, Heidelberg, Feb. 2014, pp. 327–348. DOI: [10.1007/978-3-319-04852-9_17](https://doi.org/10.1007/978-3-319-04852-9_17)
- [ElG84] T. ElGamal. “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”. In: *CRYPTO’84*. Ed. by G. R. Blakley and D. Chaum. Vol. 196. LNCS. Springer, Heidelberg, Aug. 1984, pp. 10–18
- [Ern+23] J. Ernstberger, S. Chaliasos, G. Kadianakis, S. Steinhorst, P. Jovanovic, A. Gervais, B. Livshits, and M. Orrù. “zk-Bench: A Toolset for Comparative Evaluation and Performance Benchmarking of SNARKs”. In: *IACR Cryptol. ePrint Arch.* (2023), p. 1503. URL: <https://eprint.iacr.org/2023/1503>
- [Esc+13] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. “An Algebraic Framework for Diffie-Hellman Assumptions”. In: *CRYPTO 2013, Part II*. Ed. by R. Canetti and J. A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 129–147. DOI: [10.1007/978-3-642-40084-1_8](https://doi.org/10.1007/978-3-642-40084-1_8)
- [Esc+17] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. L. Villar. “An Algebraic Framework for Diffie-Hellman Assumptions”. In: *Journal of Cryptology* 30.1 (Jan. 2017), pp. 242–288. DOI: [10.1007/s00145-015-9220-6](https://doi.org/10.1007/s00145-015-9220-6)
- [EU24] EU. “Regulation (eu) 2024/1183 of the european parliament and of the council of 11 april 2024 amending regulation (eu) no 910/2014 as regards establishing the european digital identity framework”. In: 2024. URL: <https://eur-lex.europa.eu/eli/reg/2024/1183/oj>
- [Fau+19] P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi. “Quisquis: A New Design for Anonymous Cryptocurrencies”. In: *ASIACRYPT 2019, Part I*. Ed. by S. D. Galbraith and S. Moriai. Vol. 11921. LNCS. Springer, Heidelberg, Dec. 2019, pp. 649–678. DOI: [10.1007/978-3-030-34578-5_23](https://doi.org/10.1007/978-3-030-34578-5_23)

- [FS87] A. Fiat and A. Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *CRYPTO’86*. Ed. by A. M. Odlyzko. Vol. 263. LNCS. Springer, Heidelberg, Aug. 1987, pp. 186–194. DOI: [10.1007/3-540-47721-7_12](https://doi.org/10.1007/3-540-47721-7_12)
- [Fis06] M. Fischlin. “Round-Optimal Composable Blind Signatures in the Common Reference String Model”. In: *CRYPTO 2006*. Ed. by C. Dwork. Vol. 4117. LNCS. Springer, Heidelberg, Aug. 2006, pp. 60–77. DOI: [10.1007/11818175_4](https://doi.org/10.1007/11818175_4)
- [Fuc09] G. Fuchsbauer. “Automorphic Signatures in Bilinear Groups and an Application to Round-Optimal Blind Signatures”. In: (2009). <https://eprint.iacr.org/2009/320>
- [Fuc11] G. Fuchsbauer. “Commuting Signatures and Verifiable Encryption”. 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*. Ed. by K. G. Paterson. Vol. 6632. Lecture Notes in Computer Science. Springer, 2011, pp. 224–245. DOI: [10.1007/978-3-642-20465-4_14](https://doi.org/10.1007/978-3-642-20465-4_14). URL: https://doi.org/10.1007/978-3-642-20465-4_14
- [Fuc18] G. Fuchsbauer. “Subversion-Zero-Knowledge SNARKs”. In: *PKC 2018, Part I*. Ed. by M. Abdalla and R. Dahab. Vol. 10769. LNCS. Springer, Heidelberg, Mar. 2018, pp. 315–347. DOI: [10.1007/978-3-319-76578-5_11](https://doi.org/10.1007/978-3-319-76578-5_11)
- [Fuc+16] G. Fuchsbauer, C. Hanser, C. Kamath, and D. Slamanig. “Practical Round-Optimal Blind Signatures in the Standard Model from Weaker Assumptions”. In: *Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings*. Ed. by V. Zikas and R. D. Prisco. Vol. 9841. Lecture Notes in Computer Science. Springer, 2016, pp. 391–408. DOI: [10.1007/978-3-319-44618-9_21](https://doi.org/10.1007/978-3-319-44618-9_21). URL: https://doi.org/10.1007/978-3-319-44618-9_21
- [FHS15] G. Fuchsbauer, C. Hanser, and D. Slamanig. “Practical Round-Optimal Blind Signatures in the Standard Model”. In: *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*. Ed. by R. Gennaro and M. Robshaw. Vol. 9216. Lecture Notes in Computer Science. Springer, 2015, pp. 233–253. DOI: [10.1007/978-3-662-48000-7_12](https://doi.org/10.1007/978-3-662-48000-7_12). URL: https://doi.org/10.1007/978-3-662-48000-7_12

- [FHS19] G. Fuchsbauer, C. Hanser, and D. Slamanig. “Structure-Preserving Signatures on Equivalence Classes and Constant-Size Anonymous Credentials”. In: *Journal of Cryptology* 32.2 (Apr. 2019), pp. 498–546. DOI: [10.1007/s00145-018-9281-4](https://doi.org/10.1007/s00145-018-9281-4)
- [FKL18] G. Fuchsbauer, E. Kiltz, and J. Loss. “The Algebraic Group Model and its Applications”. In: *CRYPTO 2018, Part II*. Ed. by H. Shacham and A. Boldyreva. Vol. 10992. LNCS. Springer, Heidelberg, Aug. 2018, pp. 33–62. DOI: [10.1007/978-3-319-96881-0_2](https://doi.org/10.1007/978-3-319-96881-0_2)
- [Gab19] A. Gabizon. “AuroraLight: Improved prover efficiency and SRS size in a Sonic-like system”. In: (2019). <https://eprint.iacr.org/2019/601>
- [GWC19] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. “PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge”. In: (2019). <https://eprint.iacr.org/2019/953>
- [GPS08] S. D. Galbraith, K. G. Paterson, and N. P. Smart. “Pairings for cryptographers”. In: *Discrete Applied Mathematics* 156.16 (2008), pp. 3113–3121. ISSN: 0166-218X. DOI: <https://doi.org/10.1016/j.dam.2007.12.010>
- [Gan+22] C. Ganesh, H. Khoshakhlagh, M. Kohlweiss, A. Nitulescu, and M. Zajakac. “What Makes Fiat–Shamir zkSNARKs (Updatable SRS) Simulation Extractable?” In: *Security and Cryptography for Networks*. Ed. by C. Galdi and S. Jarecki. Cham: Springer International Publishing, 2022, pp. 735–760. ISBN: 978-3-031-14791-3
- [Gar+18] S. Garg, M. Mahmoody, D. Masny, and I. Meckler. “On the Round Complexity of OT Extension”. In: *CRYPTO 2018, Part III*. Ed. by H. Shacham and A. Boldyreva. Vol. 10993. LNCS. Springer, Heidelberg, Aug. 2018, pp. 545–574. DOI: [10.1007/978-3-319-96878-0_19](https://doi.org/10.1007/978-3-319-96878-0_19)
- [Gay+18] R. Gay, D. Hofheinz, L. Kohl, and J. Pan. “More Efficient (Almost) Tightly Secure Structure-Preserving Signatures”. In: *EUROCRYPT 2018, Part II*. Ed. by J. B. Nielsen and V. Rijmen. Vol. 10821. LNCS. Springer, Heidelberg, Apr. 2018, pp. 230–258. DOI: [10.1007/978-3-319-78375-8_8](https://doi.org/10.1007/978-3-319-78375-8_8)
- [GG18] R. Gennaro and S. Goldfeder. “Fast Multiparty Threshold ECDSA with Fast Trustless Setup”. In: *ACM CCS 2018*. Ed. by D. Lie, M. Mannan, M. Backes, and X. Wang. ACM Press, Oct. 2018, pp. 1179–1194. DOI: [10.1145/3243734.3243859](https://doi.org/10.1145/3243734.3243859)

- [GGN16] R. Gennaro, S. Goldfeder, and A. Narayanan. “Threshold-Optimal DSA/ECDSA Signatures and an Application to Bitcoin Wallet Security”. In: *ACNS 16*. Ed. by M. Manulis, A.-R. Sadeghi, and S. Schneider. Vol. 9696. LNCS. Springer, Heidelberg, June 2016, pp. 156–174. DOI: [10.1007/978-3-319-39555-5_9](https://doi.org/10.1007/978-3-319-39555-5_9)
- [Gen+01] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. “Robust Threshold DSS Signatures”. In: *Inf. Comput.* 164.1 (2001), pp. 54–84. DOI: [10.1006/INCO.2000.2881](https://doi.org/10.1006/inco.2000.2881). URL: <https://doi.org/10.1006/inco.2000.2881>
- [GW11] C. Gentry and D. Wichs. “Separating succinct non-interactive arguments from all falsifiable assumptions”. In: *43rd ACM STOC*. Ed. by L. Fortnow and S. P. Vadhan. ACM Press, June 2011, pp. 99–108. DOI: [10.1145/1993636.1993651](https://doi.org/10.1145/1993636.1993651)
- [Gha16] E. Ghadafi. “Short Structure-Preserving Signatures”. In: *CT-RSA 2016*. Ed. by K. Sako. Vol. 9610. LNCS. Springer, Heidelberg, Feb. 2016, pp. 305–321. DOI: [10.1007/978-3-319-29485-8_18](https://doi.org/10.1007/978-3-319-29485-8_18)
- [Gha17a] E. Ghadafi. “How Low Can You Go? Short Structure-Preserving Signatures for Diffie-Hellman Vectors”. In: *16th IMA International Conference on Cryptography and Coding*. Ed. by M. O’Neill. Vol. 10655. LNCS. Springer, Heidelberg, Dec. 2017, pp. 185–204
- [Gha17b] E. Ghadafi. “More Efficient Structure-Preserving Signatures - Or: Bypassing the Type-III Lower Bounds”. In: *ESORICS 2017, Part II*. Ed. by S. N. Foley, D. Gollmann, and E. Sneekenes. Vol. 10493. LNCS. Springer, Heidelberg, Sept. 2017, pp. 43–61. DOI: [10.1007/978-3-319-66399-9_3](https://doi.org/10.1007/978-3-319-66399-9_3)
- [Gha19] E. Ghadafi. “Further Lower Bounds for Structure-Preserving Signatures in Asymmetric Bilinear Groups”. In: *AFRICACRYPT 19*. Ed. by J. Buchmann, A. Nitaj, and T.-e. Rachidi. Vol. 11627. LNCS. Springer, Heidelberg, July 2019, pp. 409–428. DOI: [10.1007/978-3-030-23696-0_21](https://doi.org/10.1007/978-3-030-23696-0_21)
- [GGM84] O. Goldreich, S. Goldwasser, and S. Micali. “How to Construct Random Functions (Extended Abstract)”. In: *25th FOCS*. IEEE Computer Society Press, Oct. 1984, pp. 464–479. DOI: [10.1109/SFCS.1984.715949](https://doi.org/10.1109/SFCS.1984.715949)
- [GMR85] S. Goldwasser, S. Micali, and C. Rackoff. “The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)”. In: *17th ACM STOC*. ACM Press, May 1985, pp. 291–304. DOI: [10.1145/22145.22178](https://doi.org/10.1145/22145.22178)

- [GMR89] S. Goldwasser, S. Micali, and C. Rackoff. “The knowledge complexity of interactive proof systems”. In: *SIAM Journal on computing* 18.1 (1989), pp. 186–208
- [GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. “A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks”. In: *SIAM Journal on Computing* 17.2 (Apr. 1988), pp. 281–308
- [Gro06] J. Groth. “Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures”. In: *ASIACRYPT 2006*. Ed. by X. Lai and K. Chen. Vol. 4284. LNCS. Springer, Heidelberg, Dec. 2006, pp. 444–459. DOI: [10.1007/11935230_29](https://doi.org/10.1007/11935230_29)
- [Gro10] J. Groth. “Short Pairing-Based Non-interactive Zero-Knowledge Arguments”. In: *ASIACRYPT 2010*. Ed. by M. Abe. Vol. 6477. LNCS. Springer, Heidelberg, Dec. 2010, pp. 321–340. DOI: [10.1007/978-3-642-17373-8_19](https://doi.org/10.1007/978-3-642-17373-8_19)
- [Gro15] J. Groth. “Efficient Fully Structure-Preserving Signatures for Large Messages”. In: *ASIACRYPT 2015, Part I*. Ed. by T. Iwata and J. H. Cheon. Vol. 9452. LNCS. Springer, Heidelberg, Nov. 2015, pp. 239–259. DOI: [10.1007/978-3-662-48797-6_11](https://doi.org/10.1007/978-3-662-48797-6_11)
- [Gro16] J. Groth. “On the size of pairing-based non-interactive arguments”. In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2016, pp. 305–326
- [Gro+18] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. “Updatable and Universal Common Reference Strings with Applications to zk-SNARKs”. In: *CRYPTO 2018, Part III*. Ed. by H. Shacham and A. Boldyreva. Vol. 10993. LNCS. Springer, Heidelberg, Aug. 2018, pp. 698–728. DOI: [10.1007/978-3-319-96878-0_24](https://doi.org/10.1007/978-3-319-96878-0_24)
- [GM17] J. Groth and M. Maller. “Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs”. In: *CRYPTO 2017, Part II*. Ed. by J. Katz and H. Shacham. Vol. 10402. LNCS. Springer, Heidelberg, Aug. 2017, pp. 581–612. DOI: [10.1007/978-3-319-63715-0_20](https://doi.org/10.1007/978-3-319-63715-0_20)
- [GOS06] J. Groth, R. Ostrovsky, and A. Sahai. “Perfect Non-interactive Zero Knowledge for NP”. In: *EUROCRYPT 2006*. Ed. by S. Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, May 2006, pp. 339–358. DOI: [10.1007/11761679_21](https://doi.org/10.1007/11761679_21)

- [GS08] J. Groth and A. Sahai. “Efficient Non-interactive Proof Systems for Bilinear Groups”. In: *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*. Ed. by N. P. Smart. Vol. 4965. Lecture Notes in Computer Science. Springer, 2008, pp. 415–432. DOI: [10.1007/978-3-540-78967-3_24](https://doi.org/10.1007/978-3-540-78967-3_24). URL: https://doi.org/10.1007/978-3-540-78967-3_24
- [Gur+21] K. Gurkan, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomescu. “Aggregatable Distributed Key Generation”. In: *EUROCRYPT 2021, Part I*. Ed. by A. Canteaut and F.-X. Standaert. Vol. 12696. LNCS. Springer, Heidelberg, Oct. 2021, pp. 147–176. DOI: [10.1007/978-3-030-77870-5_6](https://doi.org/10.1007/978-3-030-77870-5_6)
- [HS14] C. Hanser and D. Slamanig. “Structure-Preserving Signatures on Equivalence Classes and Their Application to Anonymous Credentials”. In: *ASIACRYPT 2014, Part I*. Ed. by P. Sarkar and T. Iwata. Vol. 8873. LNCS. Springer, Heidelberg, Dec. 2014, pp. 491–511. DOI: [10.1007/978-3-662-45611-8_26](https://doi.org/10.1007/978-3-662-45611-8_26)
- [HS21] L. Hanzlik and D. Slamanig. “With a Little Help from My Friends: Constructing Practical Anonymous Credentials”. In: *ACM CCS 2021*. Ed. by G. Vigna and E. Shi. ACM Press, Nov. 2021, pp. 2004–2023. DOI: [10.1145/3460120.3484582](https://doi.org/10.1145/3460120.3484582)
- [Her+14] G. Herold, J. Hesse, D. Hofheinz, C. Ràfols, and A. Rupp. “Polynomial Spaces: A New Framework for Composite-to-Prime-Order Transformations”. In: *CRYPTO 2014, Part I*. Ed. by J. A. Garay and R. Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 261–279. DOI: [10.1007/978-3-662-44371-2_15](https://doi.org/10.1007/978-3-662-44371-2_15)
- [HJ12] D. Hofheinz and T. Jager. “Tightly Secure Signatures and Public-Key Encryption”. In: *CRYPTO 2012*. Ed. by R. Safavi-Naini and R. Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 590–607. DOI: [10.1007/978-3-642-32009-5_35](https://doi.org/10.1007/978-3-642-32009-5_35)
- [Ica09] T. Icart. “How to Hash into Elliptic Curves”. In: *CRYPTO 2009*. Ed. by S. Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 303–316. DOI: [10.1007/978-3-642-03356-8_18](https://doi.org/10.1007/978-3-642-03356-8_18)
- [JL00] S. Jarecki and A. Lysyanskaya. “Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures”. In: *EUROCRYPT 2000*. Ed. by B. Preneel. Vol. 1807. LNCS. Springer, Heidelberg, May 2000, pp. 221–242. DOI: [10.1007/3-540-45539-6_16](https://doi.org/10.1007/3-540-45539-6_16)

- [Jou00] A. Joux. “A One Round Protocol for Tripartite Diffie-Hellman”. In: *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings*. Ed. by W. Bosma. Vol. 1838. Lecture Notes in Computer Science. Springer, 2000, pp. 385–394. URL: https://doi.org/10.1007/10722028_23
- [JKS16] A. Juels, A. E. Kosba, and E. Shi. “The Ring of Gyges: Investigating the Future of Criminal Smart Contracts”. In: *ACM CCS 2016*. Ed. by E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi. ACM Press, Oct. 2016, pp. 283–295. DOI: [10.1145/2976749.2978362](https://doi.org/10.1145/2976749.2978362)
- [JOR18] C. S. Jutla, M. Ohkubo, and A. Roy. “Improved (Almost) Tightly-Secure Structure-Preserving Signatures”. In: *PKC 2018, Part II*. Ed. by M. Abdalla and R. Dahab. Vol. 10770. LNCS. Springer, Heidelberg, Mar. 2018, pp. 123–152. DOI: [10.1007/978-3-319-76581-5_5](https://doi.org/10.1007/978-3-319-76581-5_5)
- [JR17] C. S. Jutla and A. Roy. “Improved Structure Preserving Signatures Under Standard Bilinear Assumptions”. In: *PKC 2017, Part II*. Ed. by S. Fehr. Vol. 10175. LNCS. Springer, Heidelberg, Mar. 2017, pp. 183–209. DOI: [10.1007/978-3-662-54388-7_7](https://doi.org/10.1007/978-3-662-54388-7_7)
- [KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. “Constant-Size Commitments to Polynomials and Their Applications”. In: *ASIACRYPT 2010*. Ed. by M. Abe. Vol. 6477. LNCS. Springer, Heidelberg, Dec. 2010, pp. 177–194. DOI: [10.1007/978-3-642-17373-8_11](https://doi.org/10.1007/978-3-642-17373-8_11)
- [KKK21] T. Kerber, A. Kiayias, and M. Kohlweiss. “KACHINA - Foundations of Private Smart Contracts”. In: *CSF 2021 Computer Security Foundations Symposium*. Ed. by R. Küsters and D. Naumann. IEEE Computer Society Press, 2021, pp. 1–16. DOI: [10.1109/CSF51468.2021.00002](https://doi.org/10.1109/CSF51468.2021.00002)
- [Ker+19] T. Kerber, A. Kiayias, M. Kohlweiss, and V. Zikas. “Ouroboros Cryptosynous: Privacy-Preserving Proof-of-Stake”. In: *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2019, pp. 157–174. DOI: [10.1109/SP.2019.00063](https://doi.org/10.1109/SP.2019.00063)
- [KKS22] A. Kiayias, M. Kohlweiss, and A. Sarencheh. “PEReDi: Privacy-Enhanced, Regulated and Distributed Central Bank Digital Currencies”. In: *ACM CCS 2022*. Ed. by H. Yin, A. Stavrou, C. Cremers, and E. Shi. ACM Press, Nov. 2022, pp. 1739–1752. DOI: [10.1145/3548606.3560707](https://doi.org/10.1145/3548606.3560707)

- [KPW15] E. Kiltz, J. Pan, and H. Wee. “Structure-Preserving Signatures from Standard Assumptions, Revisited”. In: *CRYPTO 2015, Part II*. Ed. by R. Gennaro and M. J. B. Robshaw. Vol. 9216. LNCS. Springer, Heidelberg, Aug. 2015, pp. 275–295. DOI: [10.1007/978-3-662-48000-7_14](https://doi.org/10.1007/978-3-662-48000-7_14)
- [KW15] E. Kiltz and H. Wee. “Quasi-Adaptive NIZK for Linear Subspaces Revisited”. In: *EUROCRYPT 2015, Part II*. Ed. by E. Oswald and M. Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 101–128. DOI: [10.1007/978-3-662-46803-6_4](https://doi.org/10.1007/978-3-662-46803-6_4)
- [Kim+20] H. Kim, Y. Lee, M. Abdalla, and J. H. Park. “Practical Dynamic Group Signature with Efficient Concurrent Joins and Batch Verifications”. In: (2020). <https://eprint.iacr.org/2020/921>
- [Kim+22] H. Kim, O. Sanders, M. Abdalla, and J. H. Park. “Practical Dynamic Group Signatures Without Knowledge Extractors”. In: *Designs, Codes and Cryptography* (Oct. 2022). ISSN: 1573-7586. DOI: <https://doi.org/10.1007/s10623-022-01129-w>
- [Koh+21] M. Kohlweiss, M. Maller, J. Siim, and M. Volkhov. “Snarky Ceremonies”. In: *ASIACRYPT 2021, Part III*. Ed. by M. Tibouchi and H. Wang. Vol. 13092. LNCS. Springer, Heidelberg, Dec. 2021, pp. 98–127. DOI: [10.1007/978-3-030-92078-4_4](https://doi.org/10.1007/978-3-030-92078-4_4)
- [KG20] C. Komlo and I. Goldberg. “FROST: Flexible Round-Optimized Schnorr Threshold Signatures”. In: *SAC 2020*. Ed. by O. Dunkelman, M. J. J. Jr., and C. O’Flynn. Vol. 12804. LNCS. Springer, Heidelberg, Oct. 2020, pp. 34–65. DOI: [10.1007/978-3-030-81652-0_2](https://doi.org/10.1007/978-3-030-81652-0_2)
- [Kon+21] Y. Kondi, B. Magri, C. Orlandi, and O. Shlomovits. “Refresh When You Wake Up: Proactive Threshold Wallets with Offline Devices”. In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 608–625. DOI: [10.1109/SP40001.2021.00067](https://doi.org/10.1109/SP40001.2021.00067)
- [Kos+15] A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. Papamanthou, R. Pass, abhi shelat, and E. Shi. “C0C0: A Framework for Building Composable Zero-Knowledge Proofs”. In: (2015). <https://eprint.iacr.org/2015/1093>. URL: <https://eprint.iacr.org/2015/1093>
- [Kos+16] A. E. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. “Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts”. In: *2016 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2016, pp. 839–858. DOI: [10.1109/SP.2016.55](https://doi.org/10.1109/SP.2016.55)

- [Lib+16] B. Libert, F. Mouhartem, T. Peters, and M. Yung. “Practical “Signatures with Efficient Protocols” from Simple Assumptions”. In: *ASIACCS 16*. Ed. by X. Chen, X. Wang, and X. Huang. ACM Press, May 2016, pp. 511–522
- [Lib+13] B. Libert, T. Peters, M. Joye, and M. Yung. “Linearly Homomorphic Structure-Preserving Signatures and Their Applications”. In: *CRYPTO 2013, Part II*. Ed. by R. Canetti and J. A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 289–307. DOI: [10.1007/978-3-642-40084-1_17](https://doi.org/10.1007/978-3-642-40084-1_17)
- [LJY16] B. Libert, M. Joye, and M. Yung. “Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares”. In: *Theor. Comput. Sci.* 645 (2016), pp. 1–24. DOI: [10.1016/J.TCS.2016.02.031](https://doi.org/10.1016/j.tcs.2016.02.031). URL: <https://doi.org/10.1016/j.tcs.2016.02.031>
- [LPY15] B. Libert, T. Peters, and M. Yung. “Short Group Signatures via Structure-Preserving Signatures: Standard Model Security from Simple Assumptions”. In: *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*. Ed. by R. Gennaro and M. Robshaw. Vol. 9216. Lecture Notes in Computer Science. Springer, 2015, pp. 296–316. DOI: [10.1007/978-3-662-48000-7_15](https://doi.org/10.1007/978-3-662-48000-7_15). URL: https://doi.org/10.1007/978-3-662-48000-7_15
- [Lin17] Y. Lindell. “Fast Secure Two-Party ECDSA Signing”. In: *CRYPTO 2017, Part II*. Ed. by J. Katz and H. Shacham. Vol. 10402. LNCS. Springer, Heidelberg, Aug. 2017, pp. 613–644. DOI: [10.1007/978-3-319-63715-0_21](https://doi.org/10.1007/978-3-319-63715-0_21)
- [Lip12] H. Lipmaa. “Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments”. In: *TCC 2012*. Ed. by R. Cramer. Vol. 7194. LNCS. Springer, Heidelberg, Mar. 2012, pp. 169–189. DOI: [10.1007/978-3-642-28914-9_10](https://doi.org/10.1007/978-3-642-28914-9_10)
- [Lip19] H. Lipmaa. “Simulation-Extractable SNARKs Revisited”. In: (2019)
- [Lip22] H. Lipmaa. “A Unified Framework for Non-universal SNARKs”. In: *PKC 2022, Part I*. Ed. by G. Hanaoka, J. Shikata, and Y. Watanabe. Vol. 13177. LNCS. Springer, Heidelberg, Mar. 2022, pp. 553–583. DOI: [10.1007/978-3-030-97121-2_20](https://doi.org/10.1007/978-3-030-97121-2_20)
- [LPS23a] H. Lipmaa, R. Parisella, and J. Siim. “Algebraic Group Model with Oblivious Sampling”. In: *Theory of Cryptography*. Ed. by G. Rothblum and H. Wee. Cham: Springer Nature Switzerland, 2023, pp. 363–392. ISBN: 978-3-031-48624-1

- [LPS23b] H. Lipmaa, R. Parisella, and J. Siim. “Algebraic Group Model with Oblivious Sampling”. In: *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part IV*. Ed. by G. N. Rothblum and H. Wee. Vol. 14372. Lecture Notes in Computer Science. Springer, 2023, pp. 363–392. DOI: [10.1007/978-3-031-48624-1_14](https://doi.org/10.1007/978-3-031-48624-1_14). URL: https://doi.org/10.1007/978-3-031-48624-1_14
- [LSZ22] H. Lipmaa, J. Siim, and M. Zajac. “Counting Vampires: From Univariate Sumcheck to Updatable ZK-SNARK”. In: *ASIACRYPT 2022, Part II*. Ed. by S. Agrawal and D. Lin. Vol. 13792. LNCS. Springer, Heidelberg, Dec. 2022, pp. 249–278. DOI: [10.1007/978-3-031-22966-4_9](https://doi.org/10.1007/978-3-031-22966-4_9)
- [LP01] A. Lysyanskaya and C. Peikert. “Adaptive Security in the Threshold Setting: From Cryptosystems to Signature Schemes”. In: *ASIACRYPT 2001*. Ed. by C. Boyd. Vol. 2248. LNCS. Springer, Heidelberg, Dec. 2001, pp. 331–350. DOI: [10.1007/3-540-45682-1_20](https://doi.org/10.1007/3-540-45682-1_20)
- [MR01] P. D. MacKenzie and M. K. Reiter. “Two-Party Generation of DSA Signatures”. In: *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*. Ed. by J. Kilian. Vol. 2139. Lecture Notes in Computer Science. Springer, 2001, pp. 137–154. DOI: [10.1007/3-540-44647-8_8](https://doi.org/10.1007/3-540-44647-8_8). URL: https://doi.org/10.1007/3-540-44647-8_8
- [Mal+19] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. “Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings”. In: *ACM CCS 2019*. Ed. by L. Cavallaro, J. Kinder, X. Wang, and J. Katz. ACM Press, Nov. 2019, pp. 2111–2128. DOI: [10.1145/3319535.3339817](https://doi.org/10.1145/3319535.3339817)
- [MKW22] R. McKean, E. Kurowska-Tober, and H. Waem. “DLA Piper GDPR fines and data breach survey: January 2022”. In: 2022. URL: <https://www.dlapiper.com/en-gb/insights/publications/2022/1/dla-piper-gdpr-fines-and-data-breach-survey-2022>
- [MRV99] S. Micali, M. O. Rabin, and S. P. Vadhan. “Verifiable Random Functions”. In: *40th FOCS*. IEEE Computer Society Press, Oct. 1999, pp. 120–130. DOI: [10.1109/SFFCS.1999.814584](https://doi.org/10.1109/SFFCS.1999.814584)
- [MTT19] T. Mizuide, A. Takayasu, and T. Takagi. “Tight Reductions for Diffie-Hellman Variants in the Algebraic Group Model”. In: *CT-RSA 2019*. Ed. by M. Matsui. Vol. 11405. LNCS. Springer, Heidelberg, Mar. 2019, pp. 169–188. DOI: [10.1007/978-3-030-12612-4_9](https://doi.org/10.1007/978-3-030-12612-4_9)

- [MRV16] P. Morillo, C. Ràfols, and J. L. Villar. “The Kernel Matrix Diffie-Hellman Assumption”. In: *ASIACRYPT 2016, Part I*. Ed. by J. H. Cheon and T. Takagi. Vol. 10031. LNCS. Springer, Heidelberg, Dec. 2016, pp. 729–758. DOI: [10.1007/978-3-662-53887-6_27](https://doi.org/10.1007/978-3-662-53887-6_27)
- [Nan+24] M. Nanri, O. P. Kempner, M. Tibouchi, and M. Abe. “Interactive Threshold Mercurial Signatures and Applications”. In: (2024). <https://eprint.iacr.org/2024/625>. URL: <https://eprint.iacr.org/2024/625>
- [Nao03] M. Naor. “On Cryptographic Assumptions and Challenges (Invited Talk)”. In: *CRYPTO 2003*. Ed. by D. Boneh. Vol. 2729. LNCS. Springer, Heidelberg, Aug. 2003, pp. 96–109. DOI: [10.1007/978-3-540-45146-4_6](https://doi.org/10.1007/978-3-540-45146-4_6)
- [NCC19] NCC. “Zcash Overwinter Consensus and Sapling Cryptography Review”. In: (2019)
- [Par+13] B. Parno, J. Howell, C. Gentry, and M. Raykova. “Pinocchio: Nearly Practical Verifiable Computation”. In: *2013 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2013, pp. 238–252. DOI: [10.1109/SP.2013.47](https://doi.org/10.1109/SP.2013.47)
- [Ped92] T. P. Pedersen. “Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing”. In: *CRYPTO’91*. Ed. by J. Feigenbaum. Vol. 576. LNCS. Springer, Heidelberg, Aug. 1992, pp. 129–140. DOI: [10.1007/3-540-46766-1_9](https://doi.org/10.1007/3-540-46766-1_9)
- [PS16] D. Pointcheval and O. Sanders. “Short Randomizable Signatures”. In: *CT-RSA 2016*. Ed. by K. Sako. Vol. 9610. LNCS. Springer, Heidelberg, Feb. 2016, pp. 111–126. DOI: [10.1007/978-3-319-29485-8_7](https://doi.org/10.1007/978-3-319-29485-8_7)
- [PS00] D. Pointcheval and J. Stern. “Security Arguments for Digital Signatures and Blind Signatures”. In: *Journal of Cryptology* 13.3 (June 2000), pp. 361–396. DOI: [10.1007/s001450010003](https://doi.org/10.1007/s001450010003)
- [RZ21] C. Ràfols and A. Zapico. “An Algebraic Framework for Universal and Updatable SNARKs”. In: *CRYPTO 2021, Part I*. Ed. by T. Malkin and C. Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 774–804. DOI: [10.1007/978-3-030-84242-0_27](https://doi.org/10.1007/978-3-030-84242-0_27)
- [RP23] A. Rial and A. M. Piotrowska. “Compact and Divisible E-Cash with Threshold Issuance”. In: *Proceedings on Privacy Enhancing Technologies* (2023)
- [RP22] A. Rial and A. M. Piotrowska. “Security Analysis of Coconut, an Attribute-Based Credential Scheme with Threshold Issuance”. In: (2022). <https://eprint.iacr.org/2022/011>

- [San20] O. Sanders. “Efficient Redactable Signature and Application to Anonymous Credentials”. In: *PKC 2020, Part II*. Ed. by A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas. Vol. 12111. LNCS. Springer, Heidelberg, May 2020, pp. 628–656. DOI: [10.1007/978-3-030-45388-6_22](https://doi.org/10.1007/978-3-030-45388-6_22)
- [SKK23] A. Sarencheh, A. Kiayias, and M. Kohlweiss. “PARScoin: A Privacy-preserving, Auditable, and Regulation-friendly Stablecoin”. In: (2023). <https://eprint.iacr.org/2023/1908>. URL: <https://eprint.iacr.org/2023/1908>
- [Sch90] C.-P. Schnorr. “Efficient Identification and Signatures for Smart Cards”. In: *CRYPTO’89*. Ed. by G. Brassard. Vol. 435. LNCS. Springer, Heidelberg, Aug. 1990, pp. 239–252. DOI: [10.1007/0-387-34805-0_22](https://doi.org/10.1007/0-387-34805-0_22)
- [Sch91] C.-P. Schnorr. “Efficient Signature Generation by Smart Cards”. In: *Journal of Cryptology* 4.3 (Jan. 1991), pp. 161–174. DOI: [10.1007/BF00196725](https://doi.org/10.1007/BF00196725)
- [Sha79] A. Shamir. “How to Share a Secret”. In: *Communications of the Association for Computing Machinery* 22.11 (Nov. 1979), pp. 612–613
- [Sho97] V. Shoup. “Lower Bounds for Discrete Logarithms and Related Problems”. In: *EUROCRYPT’97*. Ed. by W. Fumy. Vol. 1233. LNCS. Springer, Heidelberg, May 1997, pp. 256–266. DOI: [10.1007/3-540-69053-0_18](https://doi.org/10.1007/3-540-69053-0_18)
- [Sho00] V. Shoup. “Practical Threshold Signatures”. In: *EUROCRYPT 2000*. Ed. by B. Preneel. Vol. 1807. LNCS. Springer, Heidelberg, May 2000, pp. 207–220. DOI: [10.1007/3-540-45539-6_15](https://doi.org/10.1007/3-540-45539-6_15)
- [SG98] V. Shoup and R. Gennaro. “Securing Threshold Cryptosystems against Chosen Ciphertext Attack”. In: *EUROCRYPT’98*. Ed. by K. Nyberg. Vol. 1403. LNCS. Springer, Heidelberg, May 1998, pp. 1–16. DOI: [10.1007/BFb0054113](https://doi.org/10.1007/BFb0054113)
- [Son+19] A. Sonnino, M. Al-Bassam, S. Bano, S. Meiklejohn, and G. Danezis. “Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers”. In: *NDSS 2019*. The Internet Society, Feb. 2019
- [Swi97] P. Swire. “Markets, Self-Regulation, and Government Enforcement in the Protection of Personal Information, in Privacy and Self-Regulation in the Information Age by the US Department of Commerce.” In: *Privacy and Self-Regulation in the Information Age by the US Department of Commerce* (1997)

- [Tom+22] A. Tomescu, A. Bhat, B. Applebaum, I. Abraham, G. Gueta, B. Pinkas, and A. Yanai. “UTT: Decentralized Ecash with Accountable Privacy”. In: (2022). <https://eprint.iacr.org/2022/452>. URL: <https://eprint.iacr.org/2022/452>
- [Tro+17] C. Troncoso, M. Isaakidis, G. Danezis, and H. Halpin. “Systematizing Decentralization and Privacy: Lessons from 15 Years of Research and Deployments”. In: *PoPETs 2017.4* (Oct. 2017), pp. 404–426. DOI: [10.1515/popets-2017-0056](https://doi.org/10.1515/popets-2017-0056)
- [TS21] D. Tymokhanov and O. Shlomovits. “Alpha-Rays: Key Extraction Attacks on Threshold ECDSA Implementations”. In: (2021). <https://ia.cr/2021/1621>
- [Uni16] E. Union. “General Data Protection Regulation”. In: *Coping with complexity: model reduction and data analysis*. Official Journal of the European Union, 2016
- [Won+23] H. W. H. Wong, J. P. K. Ma, H. H. F. Yin, and S. S. M. Chow. “Real Threshold ECDSA”. In: *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. URL: <https://www.ndss-symposium.org/ndss-paper/real-threshold-ecdsa/>
- [Xio+23] A. L. Xiong, B. Chen, Z. Zhang, B. Bünz, B. Fisch, F. Krell, and P. Camacho. “VeriZexe: Decentralized Private Computation with Universal Setup”. In: *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*. Ed. by J. A. Calandrino and C. Troncoso. USENIX Association, 2023, pp. 4445–4462. URL: <https://www.usenix.org/conference/usenixsecurity23/presentation/xiong>

Curriculum

Mahdi Sedaghat earned his Bachelor's degree in Electrical Engineering, majoring in Telecommunications, from Birjand University, Iran, in 2014. He then pursued a Master's degree in Secure Telecommunications and Cryptology at Sharif University of Technology in Tehran, completing his thesis on the topic of outsourceable Attribute-Based encryption schemes in 2017. In 2019, he visited Charles University in Prague working on transparent Zero-Knowledge proofs under the supervision of prof. Pavel Hubáček.

He started his PhD at COSIC, KU Leuven, on January 2020 under the supervision of prof. Bart Preneel focusing on privacy-preserving distributed systems. In February and March 2023, he visited the ZK-lab at the University of Edinburgh, hosted by prof. Markulf Kohlweiss. From April to August 2023, he was an intern at Mysten Labs, cryptography team working on zero-knowledge proofs and efficient multi-signatures. In December 2023, he had a short visit from Chinese University of Hong Kong (CUHK) hosted by prof. Sherman S. M. Chow. In June 2024, he visited Foundations of Cryptography group at ETH Zürich, hosted by Prof. Dennis Hofheinz.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING

COSIC

Kasteelpark Arenberg 10, bus 2452
B-3001 Leuven

<https://www.esat.kuleuven.be/cosic/>

