DISSERTATIONES
INFORMATICAE
UNIVERSITATIS
TARTUENSIS
16

KARIM BAGHERY

Reducing Trust and Improving Security in zk-SNARKs and Commitments

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

16

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS
16

KARIM BAGHERY

Reducing Trust and Improving Security in zk-SNARKs and Commitments

UNIVERSITY OF TARTU
Press

Institute of Computer Science, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in Computer Science on 16th of June, 2020 by the Council of the Institute of Computer Science, University of Tartu.

Supervisor

Prof.	Helger Lipmaa Simula UiB, Bergen Norway

Opponents

Assoc. Prof.	Markulf Kohlweiss School of Informatics University of Edinburgh, Edinburgh United Kingdom
Assoc. Prof.	Georg Fuchsbauer Security & Privacy Group, TU Wien Favoritenstr. 9, 1040 Wien Austria

The public online defense will take place on August 13, 2020 at 14.15 on Zoom.

The publication of this dissertation was financed by the Institute of Computer Science, University of Tartu.

ISSN 2613-5906
ISBN 978-9949-03-373-7 (print)
ISBN 978-9949-03-374-4 (PDF)

University of Tartu Press
http://www.tyk.ee/

Sevgili anama və atama.

To my lovely mother and father.

Minu kallile emale ja isale.

تقدیم به مادر و پدر دوست داشتنی‌ام.

ABSTRACT

Zero-knowledge proofs are one of the essential tools in modern cryptography that allow a party to prove the validity of a statement without revealing secret information related to himself/herself. Due to their impressive advantages, Non-Interactive Zero-Knowledge (NIZK) proof systems ubiquitously have appeared in various novel applications. Verifiable computation systems like Pinocchio, the privacy-preserving coin Zcash, smart contract systems Hawk and Gyges, and the private proof-of-stake system Ouroboros Crypsinous are some of practical applications that use an efficient family of zero-knowledge proofs, called zk-SNARKs, to prove various statements in NP-complete languages. Zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs) are an efficient family of NIZK proof systems that are constructed in the Common Reference String (CRS) model and due to their *succinct* proofs and (consequently) very efficient verification, they are widely adopted in large-scale practical applications.

In the CRS model, the construction of zk-SNARKs relies on a setup phase and the users should trust the output of the setup phase that is supposed to be done by a trusted third party or distributed authority. Additionally, in different applications, it is shown that the default security of zk-SNARKs is not sufficient to deploy them directly in larger protocols that aim to achieve a stronger notion of security, say non-malleability of proofs, or even stronger Universal Composability (UC). UC-secure protocols guarantee a very strong security property, as they remain secure even if arbitrarily composed with other protocols.

In this thesis, we study zk-SNARKs from two perspectives, namely *reducing trust* and *improving security* in them. In the first direction, we investigate how much one can mitigate trust in pairing-based zk-SNARKs without sacrificing their efficiency. In such constructions, the parties of the protocol will obtain a certain level of security even if the setup phase was done maliciously or the secret information of the setup phase was revealed. As a result of this direction, we present some efficient constructions that can resist against subverting of the setup phase of zk-SNARKs and achieve a certain level of security which is stronger than before. We also show that similar techniques will allow us to mitigate the trust in trapdoor commitment schemes that are another prominent family of cryptographic primitives that require a trusted setup phase. In the second direction, we present some efficient constructions that achieve more security with minimal overhead. Some of the presented constructions allow to simplify the construction of current UC-secure protocols and improve their efficiency. New constructions can be directly deployed in any novel protocols that aim to use zk-SNARKs.

Some of the proposed zk-SNARKs are implemented in Libsnark, the state-of-the-art library for zk-SNARKs, and empirical experiences confirm that the computational cost to mitigate the trust or to achieve more security is practical.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS & ALGORITHMS

Abbreviation	Description	Section
RO	**R**andom **O**racle	2.2
CRS	**C**ommon **R**eference **S**tring	2.2
NIZK	**N**on-**I**nteractive **Z**ero-**K**nowledge	2.4.1
SNARK	**S**uccinct **N**on-interactive **AR**gument of **K**nowledge	2.4.2
zk-SNARK	**Z**ero-**K**nowledge **SNARK**	2.4.2
SE zk-SNARK	**S**imulation **E**xtractable **zk-SNARK**	2.4.2
QAP/SAP	**Q**uadratic/**S**quare **A**rithmetic **P**rogram	2.4.3
QSP/SSP	**Q**uadratic/**S**quare **S**pan **P**rogram	2.4.3
ZK	**Z**ero-**K**nowledge	2.4.1
Sub-$\{\cdots\}$	**Sub**version-$\{\cdots\}$, e.g., **Sub-ZK** for **Sub**version **ZK**	3.3
IND	**Ind**istinguishability	2.3.2
IND-CPA	**Ind**istinguishability under **C**hosen **P**laintext **A**ttack	2.3.2
NP	**N**ondeterministic **P**olynomial time	2.4
PPT	**P**robabilistic **P**olynomial **T**ime	2.1
NUPPT	**N**on-**U**niform **P**robabilistic **P**olynomial **T**ime	2.1
UC	**U**niversal **C**omposability	2.4.5
MPC	**M**ulti-**P**arty **C**omputation	3
BB	**B**lack-**B**ox	2.4.2
nBB	**n**on-**B**lack-**B**ox	2.4.2

Algorithms	Description	Section
$\mathscr{A}$	Adversary	all
KGen_{Π}	Key generation for primitive Π	all
CV	CRS verifier (CRS verification algorithm)	3, 4
P	Prover (proof generation algorithm)	3,4, 6
V	Verifier (proof verification algorithm)	3,4, 6
Sim	Simulator (proof simulation algorithm)	3,4, 6
$\mathsf{Ext}_{\mathscr{A}}$	Non-black-box Extractor dependent on $\mathscr{A}$	3, 4
Ext	Black-box Extractor	6
Sig	Signing algorithm	4
Vf	Signature verification algorithm	4
CKVer	Commitment key verifier	5
Com	Committing algorithm	5
Ver	Opening verification algorithm	5
Com^*	Trapdoor committing algorithm	5
Equiv	Trapdoor opening algorithm	5
Enc	Encryption algorithm	6
Dec	Decryption algorithm	6

LIST OF ORIGINAL PUBLICATIONS

Publications included in the thesis

1. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa and Michal Zajac. A Subversion-Resistant SNARK. In Thomas Peyrin and Tsuyoshi Takagi, editors, ASIACRYPT 2017 , volume 10626 of Lecture Notes in Computer Science, pages 3–33, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg. The full version is available on `https://eprint.iacr.org/2017/599`.
2. Shahla Atapoor and Karim Baghery. Simulation Extractability in Groth's zk-SNARK. In Cristina Perez-Sola, Guillermo Navarro-Arribas, Alex Biryukov, and Joaquin Garcia-Alfaro, editors, Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26-27, 2019, Proceedings, volume 11737 of Lecture Notes in Computer Science, pages 336–354. Springer, 2019. The latest version is available on `https://eprint.iacr.org/2019/641`.
3. Karim Baghery. Subversion-Resistant Simulation (Knowledge) Sound NI ZKs. In Martin Albrecht, editors, 17th IMA Conference on Cryptography and Coding Theory - IMACC 2019, volume 11929 of Lecture Notes in Computer Science, pages 42–63, Oxford, December 16–18, 2019; Springer, Heidelberg. The latest version is available on `https://eprint.iacr.org/2019/1162`.
4. Karim Baghery. Subversion-Resistant Commitment Schemes: Definitions and Constructions. Cryptology ePrint Archive, Report 2019/1065, 2019. The latest version is available on `https://eprint.iacr.org/2019/1065`.
5. Karim Baghery. On the Efficiency of Privacy-Preserving Smart Contract Systems. In Johannes Buchmann, Abderrahmane Nitaj and Tajjeeddine Rachidi, editors, AFRICACRYPT 2019 , volume 11627 of Lecture Notes in Computer Science, pages 118–136, Rabat, Morocco, July 9–11, 2019. Springer, Heidelberg. The latest version is available on `https://eprint.iacr.org/2019/480`.

Publications not included in the thesis

6. Karim Baghery, Behzad Abdolmaleki. An AKARI-based Secure Communication Scheme for EPC Tags. Advances in Wireless and Optical Communications, RTUWO 2017 , pages 208–213, Riga, Latvia, November 2–3, 2017. IEEE.
7. Karim Baghery, Behzad Abdolmaleki, Shahram Khazaei, Mohammad Reza Aref; Breaking Anonymity of Some Recent Lightweight RFID Authenti-

cation Protocols. Journal of Wireless Networks , Vol. 25, No. 3, pp. 1235–52, 2019. Springer US. The latest version is available on https://eprint.iacr.org/2019/1125.

8. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim and Michal Zajac. DL-Extractable UC-Commitment Schemes. In Robert Deng and Moti Yung, editors, ACNS 2019, volume 11464 of Lecture Notes in Computer Science, pages 385–405, Bogotá , Colombia, June 5–7, 2019. Springer, Heidelberg. The latest version is available on https://eprint.iacr.org/2019/201.
9. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim and Michal Zajac. UC-Secure CRS Generation for SNARKs. In Johannes Buchmann, Abderrahmane Nitaj and Tajjeeddine Rachidi, editors, AFRICACRYPT 2019, volume 11627 of Lecture Notes in Computer Science, pages 99–117, Rabat, Morocco, July 9–11, 2019. Springer, Heidelberg. The latest version is available on https://eprint.iacr.org/2019/471.
10. Karim Baghery, Alonso González, Zaira Pindado and Carla Ràfols. Signatures of Knowledge for Boolean Circuits under Standard Assumptions. In Abderrahmane Nitaj and Amr Youssef, editors, AFRICACRYPT 2020, Lecture Notes in Computer Science, Cairo, Egypt, July 20–22, 2020. Springer, Heidelberg.
11. Karim Baghery, and Mahdi Sedaghat. TIRAMISU: Black-Box Simulation Extractable NIZKs in the Updatable CRS Model. Cryptology ePrint Archive, Report 2020/474, 2020. The latest version is available on https://eprint.iacr.org/2020/474.

1. INTRODUCTION

It is undeniable that these days technological advances are changing our lifestyles quickly. Interactions and communications over the internet are part of our daily habits. Either interacting with a computer, e.g., when doing online shopping or communicating with a person or a group of people, e.g., in private meetings. Such kind of remote activities require cryptographic protocols to protect the exchanged messages, to guarantee the integrity of delivered messages, and to authenticate the parties while ensuring their privacy. Cryptography deals with constructing such cryptographic protocols that allow us to transmit information to a computer or person while guaranteeing its originality and safeguarding it from intruders. For instance, encryption schemes are cryptographic protocols that are constructed to protect messages from unintended receivers, and digital signatures are cryptographic protocols that allow one to digitally sign a transmitted message and guarantee the originality of the delivered message.

1.1. Zero-knowledge Proofs and Verifiable Computations

In many cases, users are interested to only transfer a particular piece of information or knowledge to the receiver, without leaking extra information about themselves or other information that might be secret in that case. For example, in some businesses, companies are interested to enter a bid with a proposal, but it is necessary that the proposals are secret (at least till some particular time) and do not reveal extra information about the companies (e.g., the proposals do not reveal information about the account balance of the bidders) and that bid values are non-malleable. From the bid manager's view, it is required that a company without sufficient balance (smaller than a threshold value) should not be able to participate in the bidding. In another example, while using digital coins to pay for a service [27], payers are interested to guarantee their privacy but prove to the service providers that they have paid the service fee.

Zero-Knowledge (ZK) proofs [71, 72] are one of the elegant tools in cryptography that is proposed and constructed for such practical scenarios. They enable to prove the validity of a statement (e.g., $x \in \mathscr{L}$ where $\mathscr{L}$ is an NP language) such that the verifiers learn nothing more than the truth of the statement (e.g., $x \in \mathscr{L}$) from the proof. As a prominent instance, a ZK proof system allows proving that for a public value x, $\mathsf{x} = H(\mathsf{w})$ without leaking any information about the secret input w, where H can be a standard and secure *Hash* function (e.g., SHA256). In practice, ZK *proofs* achieve security against computationally unbounded provers, while ZK *arguments* guarantee security only against a bounded prover. In the previous example of secure bidding, using a ZK proof system, each bidder can hide their bid and prove that the bidded value is larger than a threshold value, and the bid manager will be convinced that all bidders have the threshold value needed to enter the bidding without learning extra information about the bidders. Similarly,

a ZK proof system allows a spender of the digital coin to spend his coin without revealing personal secret information. In practice, it is shown that generating a ZK proof for $\mathsf{x} = H(\mathsf{w})$ without leaking any information about w, but giving the possibility to verify the correctness of computations, has great potential for deployment in various practical applications [27, 90, 93]. This concept is known as *verifiable computation* and it is shown that a particular type of ZK proof systems allows generating very efficient proofs for proving the correctness of computations that can be encoded as an arithmetic or Boolean circuit.

1.2. Non-Interactive Proof Systems and zk-SNARKs

Zero-knowledge proof systems are constructed either in *interactive* or *non-interactive* manners. While in the interactive cases there are several message exchange rounds between the prover and a verifier, in the non-interactive cases the prover publishes a proof in one shot by sending it to the verifier or by posting it on some public bulletin boards. Due to the impressive advantages of NIZK proof systems, they ubiquitously have appeared in numerous practical applications. Verifiable computation systems like Pinocchio [105], privacy-preserving cryptocurrencies such as Zcash [27], privacy-preserving smart contract systems Hawk and Gyges [90, 93], private proof-of-stake system Ouroboros Crypsinous [92], and cyptocurrencies with succinct ledgers [102] are a few of known practical applications that use an efficient family of NIZK proof system, called zk-SNARK, to prove various statements for satisfiability of various circuits that encode NP languages.

Among various NIZK arguments, Zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs) [31, 52, 78, 79, 82, 96, 105] are the most efficient and widely used ones that are constructed either in the Common Reference String (CRS) model or in the Random Oracle (RO) model. In the RO model, it is assumed that all parties of the protocol have access to a *random oracle* that given an input returns a random output. While in the CRS model there exists a trusted setup phase that generates a string from a particular distribution and shares it among all parties of the protocol. All of our studied constructions are built in the CRS model.

A zk-SNARK is a three-party non-interactive protocol that besides the prover P and verifier V there additionally exists a trusted third party $\mathsf{KGen}_{\mathsf{NIZK}}$ that executes the setup phase of the protocol and generates some public parameters, called Common Reference String (CRS). After a trusted setup phase, the CRS is shared among possible provers and verifiers for proof generation and proof verification. Given the CRS, a statement and a witness, the prover P generates a non-interactive proof π and sends it to the verifier V. Then, given the CRS, the statement and the proof π, the verifier V returns either $\mathsf{Accepted}$ or $\mathsf{Rejected}$. To guarantee the end-users' security and privacy and have practical efficiency, the common requirements from a zk-SNARK are that they guarantee *completeness*, *knowledge-soundness*, *zero-knowledge*, and *succinctness* that informally can be expressed as

below (their formal definitions can be found in Section 2.4.2).

Completeness: This property ensures that if both prover P and verifier V behave as described in the protocol, the prover will generate a proof that the verifier will always accept.

Knowledge Soundness: This property guarantees that a malicious prover P' will not be able to generate an acceptable proof for the statement, unless he/she *knows* a witness for the statement.

Zero-knowledge: This property gives the guarantee to the prover that the verifier does not learn anything beyond the truth of statement from an honestly generated proof (based on the proof generation procedure described in the protocol).

Succinctness: This is an efficiency feature that guarantees the proof is short and its size is constant (e.g., less than 1KB for different sizes of statement and witness for 128-bit security).

In some applications, particularly verifiable computation systems like Pinocchio [105] the setup phase also can be done by the verifier. This is because it does not claim zero-knowledge. But in general, it is expected that the CRS elements are generated honestly by a trusted third party.

Besides the above notions for zk-SNARKs, recently there have been some constructions that can guarantee simulation knowledge-soundness (a.k.a. simulation extractability) that is a stronger version of knowledge-soundness and guarantees non-malleability of proofs. Informally, the notion can be defined as follows,

Simulation Knowledge Soundness (a.k.a. simulation extractability): This property guarantees that a malicious prover P' will not be able to generate an acceptable proof for the statement, even if he/she has already seen arbitrary number of simulated proofs, unless he/she *knows* a witness for the statement.

Similar to the definition of knowledge-soundness, the concept of *knowing* is formalized by showing that there exists an efficient extraction algorithm Ext (either non-black-box [82] or black-box [94]) that can extract the witness. In non-black-box extraction, the extractor needs to get access to the source code of the adversary, while in black-box extraction a universal extractor works for all adversaries. In practice, it is shown that to be able to deploy zk-SNARKs (more generally NIZKs) in the protocols that aim to achieve stronger security guarantees such as universal composability [44], the zk-SNARK should guarantee simulation knowledge-soundness (a.k.a. simulation extractability) and the constructed extraction algorithm Ext should work in a black-box manner [45, 76, 85].

1.3. Non-Interactive Equivocal Commitment Schemes

Similar to the zk-SNARKs in the CRS model, equivocal commitment schemes are fundamental and widely used primitives in cryptography that require a trusted

setup phase [47]. Equivocal commitment schemes (a.k.a. trapdoor commitments) are deployed in a wide range of cryptographic protocols [27, 48, 53, 56–58, 68, 69, 81, 86, 93, 98, 110].

Similar to zk-SNARKs, in the CRS model, an equivocal commitment scheme is a three-party non-interactive protocol where besides committer Com and verifier Ver there exists a trusted third party $\mathsf{KGen}_{\mathsf{Com}}$ that executes the setup phase of the commitment scheme and generates some public commitment key ck. Given the commitment key ck and a message m, the commuting algorithm Com returns a commitment c and an opening op (a.k.a. decommitment string). Later, given the commitment key ck, the message m and the opening op the verification algorithm Ver verifies the validity of commitment c with the corresponding opening and returns $\mathsf{Accepted}$ or $\mathsf{Rejected}$.

Similar to other cryptographic protocols, under a trusted setup phase, there are some security requirements that each secure equivocal commitment scheme should guarantee. For common equivocal commitment schemes, the expected security requirements are *hiding*, *binding* and *equivocality* that informally can be defined as follows (their formal definitions can be found in Section 2.3.4).

Hiding: This property ensures that a malicious verifier cannot distinguish between two commitments that commit to m_0 and m_1, even if the verifier picks the messages.

Equivocality: This property guarantees that a verifier cannot learn anything from the commitment c about the message m and its opening op. Technically speaking, it says that there is an alternative way to open commitment and it is indistinguishable form the original opening. One may notice that hiding implied by equivocality.

Binding: This property guarantees that given an honestly generated commitment key ck, a malicious committer cannot open a commitment c to two different messages $m_0 \neq m_1$ (double opening).

1.4. Thesis Scope and Contributions

As we discussed above, in the CRS model, the construction of zk-SNARKs and equivocal commitment schemes rely on a trusted setup phase. In various cases, it is shown that along with developing those cryptographic primitives in some sensitive applications, there have been various attacks or flaw reports on the setup phase of cryptographic systems that were supposed to be done honestly. Indeed, finding a universally trusted party and minimizing the trust is one of the challenges that one needs to deal with when using zk-SNARKs in practice.

Another key challenge about using zk-SNARKs in practice is that in many cases their default security is not sufficient to directly deploy in larger cryptographic protocols. In some cases, simply because their proofs are malleable [27], while in some other cases [90, 92, 93], even with non-malleable proofs, their secu-

rity is weak to use in the protocols that aim to guarantee Universal Composability (UC) [44], mainly because they do not achieve black-box extraction.

The main result of this thesis is constructing zk-SNARKs and commitment schemes in the CRS model that require less trust and also achieve stronger security notions. Indeed, the above two challenges are the main scopes of this thesis. Particularly, in one scope we consider how much one can mitigate the trust in the setup phase of zk-SNARKs and equivocal commitment schemes that are constructed in the CRS model. While, in another scope, we consider how we can efficiently improve the security of zk-SNARKs such that they can be used in either non-UC-secure or UC-secure protocols. In both directions, we propose some efficient constructions by defining or revisiting the security notions of existing schemes. Some of the proposed constructions require less trust and some others achieve a stronger security notion that is sufficient to preserve universal composability.

In the rest, we describe the contribution of the thesis along with the author's main contribution towards the co-authored papers.

1.4.1. Subversion-Resistant Knowledge Sound SNARKs

Until 2016, all known zk-SNARKs were constructed such that under the assumption that both prover and verifier trust the setup phase, the constructions can achieve ZK and knowledge-soundness [31, 52, 78, 79, 96, 105]. For instance, to achieve ZK and guarantee the prover's privacy, the prover needed to trust the CRS generators.

In Chapter 3, we first present a necessary definitions for subversion-resistant zk-SNARKs and then present an efficient construction to can achieve our defined definitions. Roughly speaking, we show that one can construct a zk-SNARK that its prover does not need to trust the CRS generators to achieve ZK, while the verifier can achieve knowledge-soundness as before. The chapter refers to the following paper included in the thesis [3],

- Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa and Michal Zajac. A Subversion-Resistant SNARK. In Thomas Peyrin and Tsuyoshi Takagi, editors, ASIACRYPT 2017 , volume 10626 of Lecture Notes in Computer Science, pages 3–33, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg.

The paper first presents a variation of previous definitions for subversion-resistant zk-SNARKs and then propose a modified version of the state-of-the-art zk-SNARK proposed by Groth [79] and show that new scheme can guarantee ZK without trusting to the third party. The main change in our definition is that it has an extra algorithm called CRS Verification (CV) that a prover needs to execute instead of trusting the CRS generators. We also evaluate the construction with a sample implementation and show that in practice the computational cost that a prover needs to pay is comparable with the cost of proof generation, and in many cases

even less. The author's contribution is in constructing a new proposed CV algorithm, implementing the proposed construction along with some comparison with another different approach that chases the same goal.

1.4.2. Subversion-Resistant Simulation-Extractable SNARKs

The first contribution led into subversion-resistant zk-SNARKs that can guarantee ZK without trusting the CRS generators, and knowledge-soundness by trusting the setup phase. But, in practice, it is necessary to guarantee the non-malleability of proofs generated by a zk-SNARK, which is not guaranteed with knowledge-soundness. In the first part of Chapter 4, we present a variation of Groth's [79] zk-SNARK that can achieve ZK along with *simulation* knowledge-soundness, that can guarantee non-malleability of the proofs. The first part of the chapter refers to the following paper included in the thesis [6],

- Shahla Atapoor and Karim Baghery. Simulation Extractability in Groth's zk-SNARK. In Cristina Perez-Sola, Guillermo Navarro-Arribas, Alex Biryu kov, and Joaquin Garcia-Alfaro, editors, Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26-27, 2019, Proceedings, volume 11737 of Lecture Notes in Computer Science, pages 336–354. Springer, 2019.

The paper also contains an implementation of a the proposed construction which shows that in practical scenarios the proposed variation has close efficiency to the original one. As the main author of the paper, the author's contribution is giving the main idea, constructing the proposed protocol, finalizing the security proofs and doing proof-of-concept implementation.

In the second part of Chapter 4, we show that with a similar construction used in the first part but with different technique, one can construct subversion-resistant NIZK arguments that can guarantee ZK without trusting the third party along with *simulation* knowledge-soundness. The second part of the chapter refers to the following paper included in the thesis [12],

- Karim Baghery. Subversion-Resistant Simulation (Knowledge) Sound NIZKs. In Martin Albrecht, editors, 17th IMA Conference on Cryptography and Coding Theory - IMACC 2019, volume 11929 of Lecture Notes in Computer Science, pages 42–63, Oxford, December 16–18, 2019; Springer, Heidelberg.

The presented construction shows that we can lift knowledge-soundness of the presented construction in Section 3 to *simulation* knowledge-soundness while keeping it subversion ZK (ZK without trusting the third party). From a different perspective, the paper shows that we can amplify the best positive result about constructing subversion-resistant NIZK arguments, studied by Bellare, Fuchsbauer, and Scafuro [20]. The author is the only author of this paper and as such, both the constructions and security proofs are completed by the author himself.

1.4.3. Subversion-Resistant Commitment Schemes

In Chapters 3 and 4, we study constructing subversion-resistant zk-SNARKs with less trust on a third party and more stronger security properties. As mentioned before, equivocal commitment schemes in the CRS model are another prominent primitive in cryptography that similar to NIZK arguments need a trusted setup phase. As the next contribution of the thesis, in Chapter 5, we study the security of equivocal commitment schemes in the face of subverted commitment keys. Similar to the results for NIZK arguments [20], we present some negative and positive results along with some subversion-resistant equivocal commitment schemes. The chapter refers to the following paper included in the thesis [11],

- Karim Baghery. Subversion-Resistant Commitment Schemes: Definitions and Constructions. Cryptology ePrint Archive, Report 2019/1065, 2019. Available on `https://eprint.iacr.org/2019/1065`.

The paper first presents a new variation of notions i.e. binding, hiding, and equivocality for subversion-resistant equivocal commitment schemes. Then, similar to the case in NIZKs [20], it shows that some definitions are not compatible (a negative result) while presenting subversion-resistant commitment schemes that in the best case can achieve equivocality without trusting the key generators, while achieving (standard) binding. This is a single-authored paper and as such, all security proofs for negative or positive results along with constructions are completed by the author himself.

1.4.4. Efficient zk-SNARKs for UC-secure Protocols

As discussed before, a key challenge about using zk-SNARK in practice is that they are not secure enough to be deployed in cryptographic protocols that aim to achieve UC-security [44]. In 2015, Kosba et al. [94] proposed a framework called C∅C∅, which allows lifting a sound NIZK argument to a new construction that would guarantee simulation knowledge-soundness with universal extraction and can be deployed in UC-secure protocols directly. By considering recent progress in zk-SNARKs, in Chapter 6, we propose a construction along with two efficient instantiations for zk-SNARKs that has simpler constructions and can be directly used in applications that aim to guarantee UC-security. The chapter refers to the following paper included in the thesis [9],

- Karim Baghery. On the Efficiency of Privacy-Preserving Smart Contract Systems. In Johannes Buchmann, Abderrahmane Nitaj and Tajjeeddine Rachidi, editors, AFRICACRYPT 2019 , volume 11627 of Lecture Notes in Computer Science, pages 118–136, Rabat, Morocco, July 9–11, 2019. Springer, Heidelberg.

The paper first presents a new construction for building NIZK arguments that will guarantee simulation knowledge-soundness with black-box extraction, which is shown to be sufficient for a NIZK to be deployed in UC-secure protocols.

Then, the proposed construction is instantiated with two efficient zk-SNARKs which resulted in two efficient zk-SNARKs that can be used for any UC-secure application. Then, it is shown that the proposed construction allows one to simplify the construction of privacy-preserving smart contract systems like Hawk and Gyges [90,93] and also improve their efficiency in some cases. New constructions can be of independent interest and can be used in different UC-secure protocols; e.g., private proof-of-stake systems [92]. This is a single-authored paper and both the constructions and security proofs are completed by the author himself.

1.4.5. Other Contributions

In addition to the contributions outlined above and developed in the thesis, the author has worked during the thesis on other problems related to zk-SNARKs and NIZKs [1,2,15,16], and Radio Frequency IDentification (RFID) systems [13,14].

In Chapter 6, we constructed zk-SNARKs that can be deployed in UC-secure protocols. In those constructions, both prover and verifier need to trust the CRS generators. To generate the public parameters of such constructions with minimal trust, in [2] we construct a UC-secure MPC protocol. The proposed protocol allows us to safely compose the CRS-generation protocol with the main (UC-secure) zk-SNARK in a black-box manner.

In order to construct such a UC-secure CRS generation protocol, we needed a UC-secure commitment scheme with particular functionalities. In [1], we construct a new type of UC-secure commitment scheme that fulfills our requirements in the proposed UC-secure CRS generation protocol [2]. Briefly speaking, the proposed UC-secure commitment scheme allows a committer to commit a message m and open the commitment to a group element g^m; however, the simulator can extract its discrete logarithm m. The new commitment scheme can be used in situations where the secrecy of the committed message m is important, as the knowledge of m enables to break privacy while the simulator needs to know m to simulate the corrupted committer.

The constructions that achieve non-black-box (nBB) simulation extractability, including the ones that we proposed in [6, 12], allows one to build *succinct* (subversion-resistant) Signature-of-Knowledge (SoK) schemes [33, 46, 82]. But, the *succinct* SoK schemes are constructed under non-falsifiable assumptions and cannot be used directly in UC-secure protocols. In [15], we constructed an unbounded simulation sound quasi adaptive NIZK arguments for Boolean circuit satisfiability under standard assumptions with proof size $O(n+d)$ bilinear group elements, where d is the depth and n is the input size of the circuit. Our proposed scheme allows building the most efficient Signature-of-Knowledge based on standard assumptions that can also achieve UC-security.

In Chapter 6, we construct zk-SNARKs that can achieve black-box simulation extractability which is shown to be sufficient to deploy a NIZK argument in UC-secure protocols [76]. In those constructions, both prover and verifier need to

trust the CRS generators. In [2], we construct a UC-secure MPC protocol that can be used to sample the public parameters of such constructions. However, in such MPC protocol still, both prover and verifier need to trust at least 1 out-of *n* parties that are participated in the MPC protocol. In CRYPTO'18, Groth et al. [80] introduced the *updatable* CRS model that allows to circumvent the trust on setup phase of NIZK arguments. In [16], we present TIRAMISU [1] as a construction to build NIZK arguments (zk-SNARKs) that can achieve black-box simulation extractability but in the *updatable* CRS model. We show that TIRAMISU is suitable for modular use in larger cryptographic systems and allows to build NIZK arguments for UC-protocols, but with *updatable* parameters. From a different point of view, constructing TIRAMISU shows that one can bypass the impossibility of achieving *subversion* ZK and BB extractability, discussed in [20], in the *updatable* CRS model. In new constructions, in the cost of updating, all parties can eliminate the trust on a third-party and the protocol satisfies ZK and black-box simulation extractability. Using TIRAMISU, we present efficient black-box simulation extractable zk-SNARKs with updatable parameters that can be used in protocols like Hawk [93], Gyges [90], and Ouroboros Crypsinous [92], while allowing the users to update the parameters and eliminate the needed trust.

1.5. Organization of the Thesis

The rest of the thesis is organized as follows.

Chapter 2 presents some basic preliminaries related to the studied topics. Chapter 3 presents definitions for subversion-resistant zk-SNARKs along with an efficient construction that guarantees Sub-ZK and knowledge-soundness. In Chapter 4, we first discuss a general way to achieve simulation knowledge-soundness in pairing-based zk-SNARKs, and then prove that the same technique can be used to construct subversion-resistant zk-SNARKs (more generally NIZKs) which will satisfy Sub-ZK (ZK without trusting to the CRS generators) and *simulation* knowledge-soundness at the same time. Next, in Chapter 5, we study achievable security in (equivocal) commitment schemes in the face of parameter subversion and present both negative and positive results on constructing subversion-resistant commitment schemes, by showing that some of the definitions are not compatible while presenting some constructions that require less trust in comparison with current ones. Chapter 6 is more about improving the security of zk-SNARKs such that they can be used in applications that aim to achieve UC-security. Indeed, the chapter presents a simpler approach (in comparison with C∅C∅ framework [94]) to construct zk-SNARKs that can guarantee black-box simulation knowledge-soundness which is a necessary requirement to adopt zk-SNARKs in applications that aim to achieve UC-security. Finally, Chapter 7 concludes the thesis and outlines several open questions that can extend the studied topics.

[1] In Italian, TIRAMISU literally means "pull me up, lift me up".

2. PRELIMINARIES

In this chapter, we describe notations and some cryptographic concepts and definitions that are relevant to the rest of the thesis. We start by describing the notations and bilinear groups that frequently are used throughout the thesis. Next, we discuss assumptions behind the proposed zk-SNARKs and commitment schemes. Then we summarize the definitions of various cryptographic primitives including one-way functions, encryption schemes, digital signatures, commitment schemes that are used in Chapters 4, 5 and 6. Finally, we review the standard definitions and known security properties of zk-SNARKs as the main background for the thesis and more importantly as the starting point for the definition of subversion-resistant zk-SNARKs discussed in Chapters 3 and 4.

2.1. Notations, Bilinear Groups, Interpolation

Basics Notations. We denote the set of integers with $\mathbb{Z}$, the set of real numbers with $\mathbb{R}$, and the set of non-negative integers with $\mathbb{N}$. $|S|$ denotes the size of a set S. $\{0,1\}^*$ denotes all binary strings, while $\{0,1\}^n$ denotes the bit strings with length $n \in \mathbb{N}$. For a tuple of integers $\Gamma = (\gamma_1, \ldots, \gamma_n)$ with $\gamma_i \leq \gamma_{i+1}$, let $(a_i)_{i\in\Gamma} = (a_{\gamma_1}, \ldots, a_{\gamma_n})$. We sometimes denote $(a_i)_{i\in[n]}$ as $\vec{a}$. We say that $\Gamma = (\gamma_1, \ldots, \gamma_n) \in \mathbb{Z}$ is an (n,λ)-*nice* tuple, if $0 \leq \gamma_1 \leq \cdots \leq \gamma_i \leq \gamma_n = \mathsf{poly}(\lambda)$. For distributions A and B, $A \approx_c B$ means that they are computationally indistinguishable. In formal definitions, $\Pr[\mathsf{Exp} : y]$ shows the probability that y happens for experiment Exp. Let $\lambda \in \mathbb{N}$ be the information-theoretic security parameter, say $\lambda = 128$.

Algorithms. A probabilistic polynomial time (PPT) algorithm is an efficient probabilistic algorithm that runs in polynomial time. Consequently, NUPPT denotes non-uniform PPT. For an algorithm $\mathscr{A}$, let $\mathrm{im}(\mathscr{A})$ be the image of $\mathscr{A}$, i.e. the set of valid outputs of $\mathscr{A}$, let $\mathsf{RND}(\mathscr{A})$ denote the random tape of $\mathscr{A}$, and let $r \leftarrow_r \mathsf{RND}(\mathscr{A})$ denote sampling of a randomness r of sufficient length for $\mathscr{A}$'s needs. By $y \leftarrow \mathscr{A}(x;r)$ we denote the fact that $\mathscr{A}$, given an input x and a randomness r, outputs y. In case $\mathscr{A}$ is a deterministic algorithm we write $y = \mathscr{A}(x)$. For algorithms $\mathscr{A}$ and $\mathsf{Ext}_{\mathscr{A}}$, we write $(y \parallel y') \leftarrow (\mathscr{A} \parallel \mathsf{Ext}_{\mathscr{A}})(x;r)$ as a shorthand for "$y \leftarrow \mathscr{A}(x;r)$, $y' \leftarrow \mathsf{Ext}_{\mathscr{A}}(x;r)$". A function $f : \mathbb{N} \to \mathbb{Z}$ is *negligible*, denoted by $\mathsf{negl}(\lambda)$, if it grows slower than $1/P(\lambda)$ for any polynomial P; formally, f is $\mathsf{negl}(\lambda)$ if for any P there exists a constant $n_0 \in \mathbb{N}$ s.t. for all $\lambda > n_0$, $|f(\lambda)| < 1/P(\lambda)$. A function $f : \mathbb{N} \to \mathbb{Z}$ is *overwhelming*, if $1 - f$ is $\mathsf{negl}(\lambda)$.

Bilinear Groups: Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three finite Abelian groups, wherein all of them the discrete logarithm is conjectured to be hard. In a single group, Ggen denotes the generator of a group that given security parameter λ returns $(p, \mathbb{G})$, where $\mathbb{G}$ is a cyclic group of order p. In bilinear groups, we use additive notation together with the bracket notation, i.e., in group $\mathbb{G}_\mu$, $[a]_\mu = a[1]_\mu$, where $[1]_\mu$ is a fixed generator of $\mathbb{G}_\mu$, for $\mu \in \{1,2,T\}$, e.g. $a[1]_T = [a]_T$. A *bilinear group*

Algorithm 1: Computing $(\ell_i(\chi))_{i=1}^n$

$\zeta \leftarrow (\chi^n - 1)/n$; $\omega' \leftarrow 1$;
if $\chi = \omega'$ **then** $\ell_1(\chi) \leftarrow 1$; **else** $\ell_1(\chi) \leftarrow \zeta/(\chi - \omega')$;
for $i = 2$ **to** n **do**
  $\zeta \leftarrow \omega\zeta$; $\omega' \leftarrow \omega\omega'$;
  if $\chi = \omega'$ **then** $\ell_i(\chi) \leftarrow 1$; **else** $\ell_i(\chi) \leftarrow \zeta/(\chi - \omega')$;

generator $\mathsf{BGgen}(1^\lambda)$ returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where p (a large prime) is the order of three finite Abelian groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and $[1]_1, [1]_2$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. Finally, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient non-degenerate bilinear pairing, s.t. $\hat{e}([a]_1, [b]_2) = [ab]_T$, which satisfies the following properties:

(i) For all $[1]_1 \in \mathbb{G}_1$, $[1]_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$: $\hat{e}([a]_1, [b]_2) = [ab]_T$,

(ii) The *paring* $\hat{e}$ is efficiently computable,

(iii) The *paring* $\hat{e}$ is non-degenerative; $\hat{e}([a]_1, [b]_2) \neq [0]_T$ if $a, b \neq 0$.

We sometimes denote parings as $\hat{e}([a]_1, [b]_2) = [a]_1 \bullet [b]_2$.

Lagrange Interpolation: Assume n is a power of two, and let ω be the n-th primitive root of unity modulo p. Such ω exists, given that $n \mid (p-1)$. Then,

- $\ell(X) := \prod_{i=1}^{n}(X - \omega^{i-1}) = X^n - 1$ is the unique degree n monic polynomial such that $\ell(\omega^{i-1}) = 0$ for all $i \in [1..n]$.
- For $i \in [1..n]$, let $\ell_i(X)$ be the *i-th Lagrange basis polynomial*, i.e., the unique degree $n-1$ polynomial s.t. $\ell_i(\omega^{i-1}) = 1$ and $\ell_i(\omega^{j-1}) = 0$ for $i \neq j$. Clearly,
$$\ell_i(X) := \frac{\ell(X)}{\ell'(\omega^{i-1})(X - \omega^{i-1})} = \frac{(X^n - 1)\omega^{i-1}}{n(X - \omega^{i-1})} \quad . \tag{2.1}$$
where $\ell'(x_j) = \dfrac{d\ell(x)}{dx}$ at point $x = x_j$. Thus, $\ell_i(\omega^{i-1}) = 1$ while $\ell_i(\chi) = (\chi^n - 1)\omega^{i-1}/(n(\chi - \omega^{i-1}))$ for $\chi \neq \omega^{i-1}$,

Given any $\chi \in \mathbb{Z}_p$, Alg. 1 (see [28] for more details) computes $\ell_i(\chi)$ for $i \in [1..n]$. It can be implemented by using $4n-2$ multiplications and divisions in $\mathbb{Z}_p$.

Clearly, $L_{\vec{a}}(X) := \sum_{i=1}^{n} a_i \ell_i(X)$ is the interpolating polynomial of $\vec{a}$ at points ω^{i-1}, with $L_{\vec{a}}(\omega^{i-1}) = a_i$, and its coefficients can thus be computed by executing an inverse Fast Fourier Transform in time $\Theta(n \log n)$. Moreover, $(\ell_j(\omega^{i-1}))_{i=1}^n = \vec{e}_j$ (the j-th unit vector) and $(\ell(\omega^{i-1}))_{i=1}^n = \vec{0}_n$.

2.2. Provable Security and Hardness Assumptions

In modern cryptography, usually after constructing a cryptographic protocol (or primitive), the security of the protocol is defined and proven based on some inter-

actions between an adversary and an honest party that is called a *security game* (a.k.a. *experiment*). In some security games, while an adversary interacts with the honest party, it might have black-box access, a.k.a. *oracle* access, to some additional functions that return output for a particular input chosen by the adversary. Some of the known security games can be found in defining the security of cryptographic primitives such as public-key cryptosystems, digital signatures, and pseudo-random number generators [37, 70, 73, 74, 107].

While proving the security of a cryptographic protocol, using consecutive security games or experiments, more often the security of the target protocol is reduced to some known long-standing mathematical problems that are conjectured to be *computationally hard* to solve. By being *computationally hard*, we mean the problem cannot be solved efficiently, where efficiently usually means in polynomial time. Unlike in complexity theory case where the hardness of problems usually are measured in *worst case*, in cryptography the problems are supposed to be hard in the *average case* [1].

Reduction in Security Proof of Cryptographic Protocols: The procedure of reducing the security of a cryptographic protocol to a computationally hard problem is called *reduction* which is the main topic of research in the subfield *provable security*. Essentially, in a reduction one shows that if someone with either bounded/unbounded computational power, e.g., a polynomial-time adversary $\mathscr{A}$, can *break* security of the target cryptographic protocol with non-negligible probability, then the same adversary can be used to break a particular known problem that was supposed to be computationally hard. If a reduction exists for a cryptographic protocol we say that the cryptographic protocol is secure as long as the underlying computational problem is hard. In cryptography, computational hard problems are also knows as *computational assumptions*.

Security Parameter in Cryptographic Protocols: As mentioned before, the majority of cryptographic protocols are proven to be secure against a computationally bounded adversary, e.g., a polynomial-time adversary. In computationally secure constructions, the number of computations that a computationally bounded adversary needs to perform to break a particular protocol is expressed by a system parameter, called *security parameter*, denoted with λ. Roughly speaking, we say a cryptographic protocol provides λ bits of security if breaking its security (equivalently, breaking the hard problem behind the protocol) enforces a computationally bounded adversary to perform 2^{λ} mathematical operations. While designing a cryptographic protocol that needs to be computationally secure, the system parameters (e.g., length of keys or randomnesses) are chosen in a way that the protocol is estimated to guarantee λ bits of security. These days $\lambda = 128$ is a widely accepted security parameter in different cryptographic protocols that

[1] If the hardness of a problem is measured in average-case, it means the problem is hard on most instances from some explicit distribution. While when the hardness of a problem is measured in the worst case, it states that the problem is hard on some instances.

achieve computational security, which means by current computers performing 2^{128} steps, it will require considerable resources which are infeasible at the moment.

Computational Models Behind Security Proofs: While constructing a cryptographic protocol based on a computationally hard problem, namely a computational assumption, the security of the protocol can be proven under some restricted computational models [101]. The standard model, Random Oracle (RO) model [26], CRS model [50, 59], Generic Group Model (GGM) [109], and Algebric Group Model (AGM) [62] are some of the known computational models considered in proving security of various cryptographic protocols.

In the standard model, the security of cryptographic schemes is proven under some computational assumptions and only computational power and running time of adversary are restricted. Unfortunately not so many cryptographic protocols are constructed that their security proofs are done in the standard model, and due to this fact, in many cases, the protocols are constructed under other computational models that are more restricted models in comparison with the standard model.

In the RO model, it is assumed that all parties of the target cryptographic protocol have access to a *random oracle* that given an input returns a unique truly random output. In real life, random oracles do not exist, but in practice they are instantiated with some secure hash functions under some heuristic assumptions.

The GGM [101] is another known computational model where the adversary is restricted to have access to a randomly chosen encoding of a group. Basically it only allows the adversary to execute the group operations. In pairing-based groups, a paring operation is defined as an extra oracle which can be accessed by the adversary, and usually, this model is called the Generic Bilinear Group Model (GBGM) [39, 109]. The generic group model also is used in analyzing computational assumptions. Roughly speaking, in this case, one analyzes the fastest generic adversary [2], against a particular computational assumption.

Another known computational model is called the CRS model, where almost all of our studied and constructed constructions are in this model. In the CRS model, there exists a trusted setup that generates a string, known as crs, from a particular distribution and then shares it among all parties of the protocol. Later, parties involved in the protocol use the common reference string crs to run their target algorithms, e.g., proof generation and proof verification in NIZKs by a prover and verifier. Security proofs of cryptographic protocols done in the CRS model rely on the fact that the string crs is generated correctly by a trusted party or a distributed authority. If the crs is sampled from a uniformly random distribution, the model is called the common *random* string model.

Next, we recall the definitions of some cryptographic assumptions that are used in the protocols that we present in this thesis.

[2] An adversary who only does generic operations.

2.2.1. Computational Assumptions

One of the most known and primary computational assumptions in cryptography is called Discrete Logarithm (DL) which is defined as follows,

Assumption 1 (Discrete Logarithm (DL) Assumption)**.** *Let* Ggen *be a group generator which outputs a group description* $\mathsf{gk} := (g, \mathbb{G})$ *and* $\mathbb{G}$ *is a cyclic group of order* $p \in \mathbb{N}$*, with a generator* g*. For any NUPPT algorithm* $\mathscr{A}$*, the Discrete Logarithm Assumption holds relative to the group generator* Ggen *if:*

$$\mathsf{Adv}^{\mathrm{dl}}_{\mathscr{A}}(\lambda) = \Pr\left[\begin{matrix}\mathsf{gk} := (g,\mathbb{G}) \leftarrow \mathsf{Ggen}(1^\lambda), x \leftarrow_r \mathbb{Z}_p, \\ X = g^x,\ x' \leftarrow \mathscr{A}(\mathsf{gk}, X) : x' = x\end{matrix}\right] = \mathsf{negl}(\lambda) \quad .$$

Here we recall the setting where p is a large prime and $\mathbb{Z}_p$ is a field with p elements, and the multiplicative group $\mathbb{Z}_p^*$ is a cyclic group of order $p-1$ where the DL problem is assumed to be hard. We note that the DL assumption and all assumptions based on the DL assumption do not hold against a polynomial-time quantum adversary due to Shor's algorithm [108].

Assumption 2 (Computational Diffie-Hellman (CDH) Assumption [55])**.** *Let* Ggen *be a group generator which outputs a group description* $\mathsf{gk} := (g, \mathbb{G})$ *and* $\mathbb{G}$ *is a cyclic group of order* $p \in \mathbb{N}$*, with a generator* g*. For any NUPPT algorithm* $\mathscr{A}$*, the Computational Diffie-Hellman (CDH) Assumption holds relative to a group generator* Ggen *if:*

$$\mathsf{Adv}^{\mathrm{cdh}}_{\mathscr{A}}(\lambda) = \Pr\left[\begin{matrix}\mathsf{gk} := (g,\mathbb{G}) \leftarrow \mathsf{Ggen}(1^\lambda), (x,y) \leftarrow_r \mathbb{Z}_p^2, \\ (X,Y) = (g^x, g^y), Z \leftarrow \mathscr{A}(\mathsf{gk}, X, Y) : Z = g^{xy}\end{matrix}\right] = \mathsf{negl}(\lambda) \quad .$$

One may observe that the DL assumption implies the CDH assumption.

In the rest, we recall the definition of Power Symmetric Discrete Logarithm (PSDL) assumption which is based on the DL assumption and holds in the bilinear groups. The assumption is deployed in constructing some cryptographic protocols in the CRS model [96].

Assumption 3 (Γ-Power (Symmetric) Discrete Logarithm Assumption [96])**.** *Let* Γ *be an* (n,λ) *tuple for some* $n = \mathsf{poly}(\lambda)$*. We say a bilinear group generator* BGgen *is* (n,λ)*-PDL secure in group* $\mathbb{G}_t$ *for* $t \in \{1,2\}$*, if for any NUPPT* $\mathscr{A}$*,*

$$\Pr\left[\begin{matrix}\mathsf{gk} := (p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\hat{e},[1]_1,[1]_2) \leftarrow \mathsf{BGgen}(1^\lambda), \\ x \leftarrow \mathbb{Z}_p,\ x' \leftarrow \mathscr{A}(\mathsf{gk}; (\left[x^\ell\right]_t)_{\ell\in\Gamma}) : x' = x\end{matrix}\right] = \mathsf{negl}(\lambda) \quad .$$

Similarly, we say a bilinear group generator BGgen *is* Γ*-PSDL secure, if for any NUPPT adversary* $\mathscr{A}$*,*

$$\Pr\left[\begin{matrix}\mathsf{gk} := (p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\hat{e},[1]_1,[1]_2) \leftarrow \mathsf{BGgen}(1^\lambda), \\ x \leftarrow \mathbb{Z}_p,\ x' \leftarrow \mathscr{A}(\mathsf{gk}, (\left[x^\ell\right]_1, \left[x^\ell\right]_2)_{\ell\in\Gamma}) : x' = x\end{matrix}\right] = \mathsf{negl}(\lambda) \quad .$$

In [96], Lipmaa proved that the Γ-PSDL assumption holds in the generic group model for any (n,λ)-*nice* tuple Γ given $n = poly(\lambda)$.

2.2.2. Knowledge Assumptions

Cryptographic assumptions can be categorized into two classes called *falsifiable* and *non-falsifiable* assumptions. Roughly speaking, a computational assumption is falsifiable if it can be written as a security game between an adversary and a challenger, which at the end, the challenger can *efficiently* determine whether the adversary won the game [103]. Standard assumptions such as DL, CDH, etc. are falsifiable assumptions. The assumptions that cannot be formalized as a security game are called non-falsifiable assumptions. Knowledge assumptions (a.k.a. extractability assumptions) are the most known and well-established non-falsifiable assumptions that appear in various cryptographic protocols such as NIZKs, digital signatures, and commitments. In such assumptions, if a PPT adversary $\mathscr{A}$ managed to output some *well-formed* values, then one assumes that $\mathscr{A}$ *knows* some secret values to generate those values. The concept of *knowing* is formalized by assuming that there exists an efficient algorithm $\mathsf{Ext}_{\mathscr{A}}$ that given security parameter λ, public coins and random bits of $\mathscr{A}$ can extract the secret values.

In the rest, we recall definitions of some knowledge assumptions used in this thesis. The first assumption that we review is the Bilinear Diffie-Hellman Knowledge of Exponents (BDH-KE) Assumption (a.k.a. Bilinear Knowledge of Exponent Assumption (B-KEA)) which states that for an asymmetric bilinear group $(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\hat{e},[1]_1,[1]_2)$, given group elements $[1]_1$ and $[1]_2$, it is infeasible to generate $[a]_1$ and $[a]_2$ such that $[a]_1 \bullet [1]_2 = [1]_1 \bullet [a]_2$ without knowing a. The assumption is formalized bellow.

Let $\mathscr{R}$ be a relation generator, such that $\mathscr{R}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathbf{R} = \{(\mathsf{x},\mathsf{w})\}$ along with auxiliary information $\mathsf{aux}_{\mathbf{R}}$, where x is the statement and w is the corresponding witness. We assume one can deduce λ from the description of $\mathbf{R}$.

Assumption 4 (Bilinear Diffie-Hellman Knowledge of Exponents (BDH-KE) Assumption)**.** *A bilinear group generator* BGgen *is* BDH-KE secure *for the relation generator* $\mathscr{R}$ *if for any* λ, $(\mathbf{R},\mathsf{aux}_{\mathbf{R}}) \in \mathrm{im}(\mathscr{R}(1^\lambda))$*, and PPT adversary* $\mathscr{A}$ *there exists a PPT extractor* $\mathsf{Ext}_{\mathscr{A}}$*, such that*

$$\mathsf{Adv}^{\mathrm{bdh-ke}}_{\mathsf{BGgen},\mathscr{A},\mathsf{Ext}_{\mathscr{A}}}(\lambda) = \Pr\left[\begin{array}{l}\mathsf{gk} := (p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\hat{e},[1]_{1,2}) \leftarrow \mathsf{BGgen}(1^\lambda),\\ ([\alpha_1]_1,[\alpha_2]_2 \parallel a) \leftarrow (\mathscr{A} \parallel \mathsf{Ext}_{\mathscr{A}})(\mathbf{R},\mathsf{aux}_{\mathbf{R}},\mathsf{gk}) :\\ [\alpha_1]_1 \bullet [1]_2 = [1]_1 \bullet [\alpha_2]_2 \wedge a \neq \alpha_1\end{array}\right]$$

is $\mathsf{negl}(\lambda)$.

The BDH-KE assumption is an asymmetric-pairing version of the knowledge of exponent assumption (KEA) introduced by Damgard [49] and the special case of $q = 0$ of the q-Power Knowledge of Exponent (q-PKE) assumption in asymmetric bilinear groups introduced by Groth [78].

In [19], Bellare et al. presented a variation of KEA assumption which states that given $[1]_1, [a]_1, [s]_1, [as]_1$ it is infeasible to generate $[c_0 + c_1 s]_1, [c_0 a + c_1 as]_1$ without knowing c_0 and c_1. For a different purpose, recently Bellare, Fuchsbauer, and Scafuro [20] proposed a new variation of the KEA assumption called Diffie-Hellman Knowledge of Exponent Assumption (DH-KEA) which states that given $[1]_1$ if an adversary comes up with $[1]_1, [a]_1, [s]_1, [as]_1$, the adversary has to *either* know a or s. Later, in Chapter 3, we will discuss applications of the latter assumption in more detail.

2.3. Cryptographic Primitives

This section summarizes some necessary cryptographic primitives, namely one-way functions, pseudo-random functions, public-key encryption schemes, digital signatures and commitments schemes.

2.3.1. One-Way and Pseudo-Random Functions

In the rest, we define one-way and pseudo-random functions that are used in Chapter 6 while constructing zk-SNARKs that can be adopted in UC-protocols.

One-Way Functions. One-Way Functions (OWFs) are a fundamental family of functions in cryptography so that their computation is efficient but they cannot be inverted efficiently. By inverting, we mean finding any valid pre-image for a random image of a one-way function. More formally, for security parameter λ, a one-way function can be defined as below.

Definition 1 (One-way Function (OWF))**.** *A function $f : \{0,1\}^m \to \{0,1\}^n$ is OWF if the following conditions holds about it,*

- *The description of f is public and one does need secret trapdoors to execute it.*
- *Given the input x, one can efficiently compute $f(x)$.*
- *For all PPT adversaries $\mathscr{A}$,*

$$\Pr[x \leftarrow_r \{0,1\}^m, y = f(x), x' \leftarrow \mathscr{A}(f,y) : f(x') = y] = \mathsf{negl}(\lambda)$$

The existence of OWFs is an open problem, but this is a necessary assumption in proving the security of many cryptographic primitives.

Pseudo-Random Functions: A Pseudo-Random Function (PRF) family is a set of functions that can be computed efficiently and emulate a random oracle. In other words, no efficient algorithm can distinguish between the output of a function chosen randomly from the PRF family and an output of a truly random function. This is formally defined as follows.

Definition 2 (Pseudo-Random Functions (PRF))**.** *A function $f : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ is PRF if the following conditions holds about it,*

- *Given a key $K \in \{0,1\}^s$ and an input $X \in \{0,1\}^n$ there is an efficient algorithm to compute $f_K(X) = f(X,K)$.*
- *For any PPT oracle algorithm $\mathscr{A}$, we have*

$$|\Pr[K \leftarrow_r \{0,1\}^s, b' \leftarrow \mathscr{A}^{f_K} : b' = 1] - \Pr[f \leftarrow_\$ \mathscr{F}, b' \leftarrow \mathscr{A}^f : b' = 1]| = \mathsf{negl}(\lambda),$$

where $\mathscr{F} = \{f : \{0,1\}^n \to \{0,1\}^m\}$ and $\mathscr{A}$ makes at most q gueries to the oracle.

2.3.2. Public-Key Encryption Schemes

Encryption schemes (a.k.a. cryptosystems) are one of fundamental cryptographic primitives that allow one to encrypt a message (or data) using an encryption key such that only a desired receiver, who has access to the decryption key, can decrypt the encrypted message (a.k.a. ciphertext) and see the plain message. This allows two or several parties to communicate securely on a public channel once they have access to the encryption and decryption keys. Based on the type of encryption and decryption keys, there are two types of cryptosystems known as *symmetric* cryptosystems (a.k.a. secret key cryptosystems) or *public key* cryptosystems. In the former, both encryption and decryption keys are the same and it should securely be shared between only the encryptor and decryptor. But in the latter, the encryption key is public, called *public key*, but the decryption key, called *secret* key, is secret and only the intended recipient has access to the secret key.

In practice, it is shown that secret key cryptosystems are faster but public-key cryptosystems have easier key distribution. A public-key cryptosystem consists of a set of three algorithms $\Pi_{\mathsf{Enc}} = (\mathsf{KGen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ that are defined as follows,

- **Key Generation**, $(\mathsf{pk}_{\mathsf{Enc}}, \mathsf{sk}_{\mathsf{Enc}}) \leftarrow \mathsf{KGen}_{\mathsf{Enc}}(1^\lambda)$: Given the security parameter λ returns a public key $\mathsf{pk}_{\mathsf{Enc}}$, a secret key $\mathsf{sk}_{\mathsf{Enc}}$ associated with it. It also determines the message space $\mathscr{M}$, the ciphertext space $\mathscr{C}$ and the randomness space $\mathscr{R}$.
- **Encryption**, $c \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathsf{Enc}}, m)$: Given a public key $\mathsf{pk}_{\mathsf{Enc}}$ and a message $m \in \mathscr{M}$, algorithm Enc encrypts m and outputs a ciphertext $c \in \mathscr{C}$.
- **Decryption**, $m \leftarrow \mathsf{Dec}(\mathsf{sk}_{\mathsf{Enc}}, c)$: Given a secret key $\mathsf{sk}_{\mathsf{Enc}}$ and a ciphertext $c \in \mathscr{C}$, algorithm Dec decrypts ciphertext c an outputs the message m (or $\perp$ if decryption fails).

Various cryptosystems are constructed to guarantee different security notions but the primary security requirement common in all cryptosystems called indistinguishability under chosen-plaintext attacks (IND-CPA) which is defined as follows,

Definition 3 (IND-CPA)**.** *A public-key cryptosystem $\Pi_{\mathsf{Enc}} = (\mathsf{KGen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ guarantees indistinguishability under chosen-plaintext attacks (IND-CPA) if for*

any PPT adversary $\mathscr{A}$, it holds that

$$\mathsf{Adv}^{\mathrm{ind-cpa}}_{\mathscr{A}}(\lambda) := \left| \Pr \begin{bmatrix} (\mathsf{pk}_{\mathsf{Enc}}, \mathsf{sk}_{\mathsf{Enc}}, \mathsf{pp}_{\mathsf{Enc}}) \leftarrow \mathsf{KGen}_{\mathsf{Enc}}(1^{\lambda}), \\ (m_0, m_1) \leftarrow \mathscr{A}(\mathsf{pk}), b \leftarrow_r \{0,1\}, \\ b' \leftarrow \mathscr{A}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}_{\mathsf{Enc}}, m_b)) : b' = b \end{bmatrix} - \frac{1}{2} \right| = \mathsf{negl}(\lambda) \quad .$$

2.3.3. Digital Signatures

Digital signatures are one of the fundamental cryptographic primitives that are constructed based on public-key cryptography and have a key pair called *signing key* and *verification key*. A digital signature allows a signer to digitally sign a message using the signing key in a way that any person in possession of the verification key can check the validity of the signature. The *secret key* in public-key cryptography is the *signing key* and the *public key* is the *verification key* in digital signatures. In real life, digital signatures are used in different cryptographic protocols to authenticate the original sender of a message on a public channel. The most basic and critical security requirement for a digital signature is that the signature in unforgeable, such that only a valid signer (the one who knows the signing key) can generate a valid signature.

Technically speaking, a digital signature consists of a set of three algorithms $\Pi_{\mathsf{Sig}} = (\mathsf{KGen}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf})$ that are defined as follows,

- **Key Generation**, $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{vk}_{\mathsf{Sig}}) \leftarrow \mathsf{KGen}_{\mathsf{Sig}}(1^{\lambda})$: Given the security parameter λ returns a signing key $\mathsf{sk}_{\mathsf{Sig}}$ and a verification key $\mathsf{vk}_{\mathsf{Sig}}$.
- **Signing**, $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_{\mathsf{Sig}}, m)$: Given a signing key $\mathsf{sk}_{\mathsf{Sig}}$ and a message m, the algorithm Sig signs m and outputs a signature σ.
- **Verification**, $0/1 \leftarrow \mathsf{Vf}(\mathsf{vk}_{\mathsf{Sig}}, m, \sigma)$: Given a verification key $\mathsf{vk}_{\mathsf{Sig}}$, a message m, and a signature σ, algorithm Vf verifies if σ is a valid signature for message m and outputs 1 if so, and 0 otherwise.

The main security requirement for a digital signature is called *Unforgeability Against Chosen-Message Attacks* (UF-CMA) that is defined as follows.

Definition 4 (SUF-CMA)**.** *A digital signature scheme $\Pi_{\mathsf{Sig}} = (\mathsf{KGen}_{\mathsf{Sig}}, \mathsf{Enc}, \mathsf{Dec})$ guarantees strong unforgeability against chosen-message attacks (SUF-CMA) if for any PPT adversary $\mathscr{A}$ which makes $q = \mathsf{poly}(\lambda)$ signing queries, the following advantage is $\mathsf{negl}(\lambda)$:*

$$\mathsf{Adv}^{\mathrm{uf-cma}}_{\mathscr{A}}(\lambda) = \Pr \begin{bmatrix} (\mathsf{sk}_{\mathsf{Sig}}, \mathsf{vk}_{\mathsf{Sig}}) \leftarrow \mathsf{KGen}_{\mathsf{Sig}}(1^{\lambda}), \\ (m^*, \sigma^*) \leftarrow \mathscr{A}^{\mathsf{Sig}(\mathsf{sk}_{\mathsf{Sig}}, \cdot)}(\mathsf{vk}_{\mathsf{Sig}}) : \\ \mathsf{Vf}(\mathsf{vk}_{\mathsf{Sig}}, m^*, \sigma^*) = 1 \wedge ((m^*, \sigma^*) \notin \mathbb{M}) \end{bmatrix} = \mathsf{negl}(\lambda) \quad ,$$

where $\mathbb{M}$ shows the set of queried messages to the signing oracle $\mathsf{Sig}(\mathsf{sk}_{\mathsf{Sig}}, \cdot)$.

In constructions studied in Chapter 4, we use a *one-time* Strong Unforgeable Against Chosen-Message Attacks (SUF-1CMA) signature scheme to guarantee non-malleability of proofs. SUF-1CMA security is an especial case of SUF-CMA where an adversary can only make one query to the signing oracle $\mathsf{Sig}(\mathsf{sk}_{\mathsf{Sig}}, \cdot)$.

2.3.4. Equivocal Commitment Schemes

Commitment schemes are one of the fundamental and widely used concepts in cryptography [34]. A commitment scheme allows a committer to create a commitment to a secret value, and later may open and reveal the secret value in a verifiable manner [87, 106]. The procedure of generating the commitment is called the *committing* phase, and revealing a committed message and some secret information used in the committing phase (e.g., randomness) called the *opening* phase.

Let Setup be an algorithm that takes as input the security parameter λ and outputs some setup information $\mathsf{gk} \leftarrow \mathsf{Setup}(1^{\lambda})$. In the basic form, a commitment scheme consists of a tuple of polynomial-time algorithms $(\mathsf{KGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Ver})$.

- Algorithm $\mathsf{KGen}_{\mathsf{Com}}$ is a probabilistic algorithm that given the setup information gk generates a commitment key ck and a trapdoor tk and returns ck. The setup information can, for instance, describe a finite group over which we are working, or simply the security parameter written in unary representation [77]. We assume all parties have access to gk. The commitment key ck specifies a message space $\mathscr{M}$, a randomizer space $\mathscr{R}$ and a commitment space $\mathscr{C}$. It is usually assumed that it is easy to verify membership of the message space, randomizer space, and the commitment space and it is possible to sample randomizers uniformly at random from $\mathscr{R}$ [3].
- The algorithm Com takes as input the commitment key ck, a message m, a randomizer r and outputs a commitment c and an opening information op.
- Given ck, c, m and op, the algorithm Ver returns either 1 or 0.

In this thesis, we consider equivocal commitment schemes (a.k.a. trapdoor commitments) in the CRS model that additionally achieve equivocality. In such cases, a non-interactive commitment scheme Π_{Com} consists of a tuple of algorithms $(\mathsf{KGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Ver}, \mathsf{KGen}^{*}_{\mathsf{Com}}, \mathsf{Com}^{*}, \mathsf{Equiv})$. In an equivocal commitment scheme, given the trapdoor tk associated with key ck, a commitment can be opened to any message, with appropriate opening information. This property is considered by PPT algorithms Com^{*} and Equiv, where Com^{*} takes tk (generated by $\mathsf{KGen}^{*}_{\mathsf{Com}}$) as input and outputs an equivocal commitment c and an equivocation key ek. Then, Equiv on inputs ek, c and a message m creates an opening $\mathsf{op} := r$ of the commitment, so that $(c, \mathsf{op}) = \mathsf{Com}(\mathsf{ck}, m; r)$.

Technically speaking, an equivocal commitment scheme consists of a set of seven algorithms $\Pi_{\mathsf{Com}} = (\mathsf{KGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Ver}, \mathsf{KGen}^{*}_{\mathsf{Com}}, \mathsf{Com}^{*}, \mathsf{Equiv})$ that are

[3] Note that in some commitment schemes the description of key generation requires direct sampling of ck, but there is no guarantee that a malicious key generator can sample the ck such that he can keep a trapdoor tk for it.

defined as follows,

- **Key Generation**, $\mathsf{ck} \leftarrow \mathsf{KGen}_{\mathsf{Com}}(\mathsf{gk})$: Generates a commitment key ck and associated trapdoor tk. It returns ck and keeps secret or removes tk. It also specifies a message space $\mathscr{M}$, a randomness space $\mathscr{R}$, and a commitment space $\mathscr{C}$. The algorithm should be executed by a trusted authority.
- **Committing**, $(c, \mathsf{op}) \leftarrow \mathsf{Com}(\mathsf{ck}, m; r)$: Outputs a commitment c and opening information op. This algorithm specifies a function $\mathsf{Com} : \mathscr{M} \times \mathscr{R} \rightarrow \mathscr{C}$. Given a message $m \in \mathscr{M}$, the committer picks a randomness $r \in \mathscr{R}$ and computes the commitment $(c, \mathsf{op}) = \mathsf{Com}(\mathsf{ck}, m; r)$.
- **Opening Verification**, $0/1 \leftarrow \mathsf{Ver}(\mathsf{ck}, c, m, \mathsf{op})$: Outputs 1 if $m \in \mathscr{M}$ is the committed message in the commitment c with opening value op, and returns 0 if (c, m, op) does not correspond to a valid tuple of commitment, message and opening.
- **Simulation of Key Generation**, $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}^{*}_{\mathsf{Com}}(\mathsf{gk})$: Generates a commitment key ck and associated trapdoor tk. It also specifies a message space $\mathscr{M}$, a randomness space $\mathscr{R}$, and a commitment space $\mathscr{C}$.
- **Trapdoor Committing**, $(c, \mathsf{ek}) \leftarrow \mathsf{Com}^{*}(\mathsf{ck}, \mathsf{tk})$: Given commitment key ck and tk, outputs an equivocal commitment c and an equivocation key ek.
- **Trapdoor Opening**, $\mathsf{op} \leftarrow \mathsf{Equiv}(\mathsf{ek}, c, m)$: On inputs ek, c and a message m creates an opening $\mathsf{op} := r$ of the commitment, s.t. $(c, \mathsf{op}) = \mathsf{Com}(\mathsf{ck}, m; r)$ and returns op.

The basic requirement from a commitment scheme is *completeness* which implies for $\mathsf{ck} \leftarrow \mathsf{KGen}_{\mathsf{Com}}(\mathsf{gk})$ and any honestly generated commitment $c \in \mathscr{C}$ and opening $\mathsf{op}\mathscr{R}$ of $m \in \mathscr{M}$, they should successfully pass the verification by $\mathsf{Ver}(\mathsf{ck}, c, m, \mathsf{op})$. Additionally, an equivocal commitment scheme Π_{Com} is expected to satisfy security notions known as *binding*, *hiding*, and *equivocality* as follows.

Definition 5 (Binding)**.** *A commitment scheme* $\Pi_{\mathsf{Com}} = (\mathsf{KGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Ver})$ *is computationally binding if for any PPT adversary* $\mathscr{A}$*,*

$$\Pr\begin{bmatrix}\mathsf{gk} \leftarrow \mathsf{Setup}(1^{\lambda}), \mathsf{ck} \leftarrow \mathsf{KGen}_{\mathsf{Com}}(\mathsf{gk}), \\ (c, (m_0, \mathsf{op}_0), (m_1, \mathsf{op}_1)) \leftarrow \mathscr{A}(\mathsf{ck}) : (m_0 \neq m_1) \wedge \\ (\mathsf{Ver}(\mathsf{ck}, c, m_0, \mathsf{op}_0) = 1) \wedge (\mathsf{Ver}(\mathsf{ck}, c, m_1, \mathsf{op}_1) = 1)\end{bmatrix} = \mathsf{negl}(\lambda) \quad .$$

The commitment is *perfectly* binding if the above probability is equal to 0.

Definition 6 (Hiding)**.** *A commitment scheme* $\Pi_{\mathsf{Com}} = (\mathsf{KGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Ver})$ *is computationally hiding if for any PPT adversary* $\mathscr{A}$*,*

$$\left| 2\Pr\begin{bmatrix}\mathsf{gk} \leftarrow \mathsf{Setup}(1^{\lambda}), \mathsf{ck} \leftarrow \mathsf{KGen}_{\mathsf{Com}}(\mathsf{gk}), \\ (m_0, m_1) \leftarrow \mathscr{A}(\mathsf{ck}), b \leftarrow_{\$} \{0,1\}, r_b \leftarrow_{\$} \mathscr{R}, \\ (c_b, \mathsf{op}_b) \leftarrow \mathsf{Com}(\mathsf{ck}, m_b; r_b), b' \leftarrow \mathscr{A}(\mathsf{ck}, c_b) : b' = b\end{bmatrix} - 1 \right| = \mathsf{negl}(\lambda) \quad .$$

The commitment is *perfectly* hiding if the above probability is equal to 0.

Definition 7 (Equivocality)**.** *A commitment scheme* $\Pi_{\mathsf{Com}} = (\mathsf{KGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Ver}, \mathsf{KGen}^*_{\mathsf{Com}}, \mathsf{Com}^*, \mathsf{Equiv})$ *is equivocal if there exist PPT algorithms* Com^* *and* Equiv *that given the trapdoor of the commitment key, can come up with a fake commitment and a valid opening s.t. they would be indistinguishable from the real ones. More formally, for* $\mathsf{gk} \leftarrow \mathsf{Setup}(1^\lambda)$*, for any PPT adversary* $\mathscr{A}$*,*

$$\left| \Pr \left[\begin{array}{l} \mathsf{ck} \leftarrow \mathsf{KGen}_{\mathsf{Com}}(\mathsf{gk}), \\ m \leftarrow \mathscr{A}(\mathsf{ck}), r \leftarrow_{\$} \mathscr{R}, \\ (c, \mathsf{op}) \leftarrow \mathsf{Com}(\mathsf{ck}, m; r) : \\ \mathscr{A}(\mathsf{ck}, c, \mathsf{op}) = 1 \end{array} \right] - \Pr \left[\begin{array}{l} (\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}^*_{\mathsf{Com}}(\mathsf{gk}), \\ m \leftarrow \mathscr{A}(\mathsf{ck}), \\ (c, \mathsf{ek}) \leftarrow \mathsf{Com}^*(\mathsf{ck}, \mathsf{tk}), \\ \mathsf{op} \leftarrow \mathsf{Equiv}(\mathsf{ek}, c, m) : \\ \mathscr{A}(\mathsf{ck}, c, \mathsf{op}) = 1 \end{array} \right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathscr{A}$ *outputs* $m \in \mathscr{M}$*.*

One may notice that equivocality implies hiding, as a commitment is indistinguishable from an equivocal commitment that can be opened to any message [77].

Pedersen Commitments: Pedersen commitment scheme [106] is one of the most known schemes that satisfy hiding, equivocality, and binding under the discrete logarithm assumption (defined in Assumption 1). In Pedersen commitment scheme, the algorithms are defined as follows,

- **Key Generation**, $\mathsf{ck} \leftarrow \mathsf{KGen}_{\mathsf{Com}}(\mathsf{gk})$: Given the description of a group $\mathbb{G}$ of prime order p as $\mathsf{gk} := (\mathbb{G}, p)$, it samples two generators $(g, h) \leftarrow \mathbb{G}^*$. We let $\mathsf{ck} := (g, h)$;
- **Committing**, $(c, \mathsf{op}) \leftarrow \mathsf{Com}(\mathsf{ck}, m; r)$: Given (ck, m) where $m \in \mathbb{Z}_p$, it samples a randomness $r \leftarrow_{\$} \mathbb{Z}_p$ and computes the commitment $(c, \mathsf{op}) := (g^m h^r, r)$.
- **Opening Verification**, $0/1 \leftarrow \mathsf{Ver}(\mathsf{ck}, c, m, \mathsf{op})$: Outputs 1 if $g^m h^{\mathsf{op}} = c$, otherwise outputs 0;
- **Simulation of Key Generation**, $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}^*_{\mathsf{Com}}(\mathsf{gk})$: Generates two generators $g \leftarrow \mathbb{G}$ and $h = g^{\mathsf{sk}}$, where $\mathsf{sk} \leftarrow_{\$} \mathbb{Z}_p^*$ is the secret trapdoor of the commitment key ck. Then set $(\mathsf{ck}, \mathsf{tk}) := ((g, h), \mathsf{sk})$;
- **Trapdoor Committing**, $(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*(\mathsf{ck}, \mathsf{tk})$: Given commitment key ck and tk, outputs an equivocal commitment $c = g^s$ where $s \leftarrow_{\$} \mathbb{Z}_p$ and an equivocation key $\mathsf{ek} = s$.
- **Trapdoor Opening**, $\mathsf{op} \leftarrow \mathsf{Equiv}(\mathsf{ek}, c, m)$: On input equivocation key $\mathsf{ek} = s \in \mathbb{Z}_p$, $c \in \mathscr{C}$ and message m creates an opening $r \leftarrow (s - m) \cdot \mathsf{sk}^{-1}$, such that $(c, \mathsf{op}) = \mathsf{Com}(\mathsf{ck}, m; r)$, and returns $\mathsf{op} = r$.

2.4. NIZK Arguments and zk-SNARKs

Recall that in complexity theory, NP is the class of languages $\mathscr{L}$ that have a polynomial time decision algorithm $\mathsf{V}(\mathsf{x},\mathsf{w}) \in \{0,1\}$, and we say $\mathsf{x} \in \mathscr{L} \Leftrightarrow \exists \mathsf{w} : (\mathsf{x},\mathsf{w}) = 1$, where x is known as statement and w is called witness.

Zero-knowledge [71] proof systems are essential cryptographic tool that allows one (namely a *prover* P) to prove the validity of a statement $\mathsf{x} \in \mathscr{L}$ without leaking extra information. The term *zero-knowledge* refers to the fact that a verifier V does not learn anything from the proof more than the truth of the statement x. Non-Interactive Zero-Knowledge (NIZK) [36] is a popular type of zero-knowledge proofs that achieves the same goal but without the need for interaction between the prover P and verifier V. In the rest of this section, we summarize definitions of NIZKs and an efficient family of them, called zk-SNARKs, that are widely used in the rest of thesis.

2.4.1. NIZK Arguments in the CRS model

Non-interactive zero-knowledge arguments are usually constructed in either the Random Oracle (RO) [67] or Common Reference String (CRS) model [36]. In the RO model, it is assumed that both P and V have access to a random oracle (function) that returns a uniformly random value for each unique query. But in the CRS model, a NIZK argument requires a one-time setup phase which is supposed to be done by a trusted third party or distributed authority. A graphical representation of NIZKs in the CRS model is shown in Fig. 1

In the CRS model, once the setup phase was done by a trusted third party, the generated CRS crs is publicly shared between the prover P and verifier V which would allow them to generate and verify a proof without interaction with each other. Under a trusted setup phase, a standard NIZK argument is expected to guarantee three notions known as *completeness*, *zero-knowledge* and *soundness*. Completeness ensures an honest P will always convince an honest V. Soundness guarantees that a malicious P cannot convince an honest V except with negligible probability. As briefly mentioned before, zero-knowledge ensure that the honestly generated proof does leak extra information besides the validity of the statement.

Let $\mathscr{R}$ be a relation generator, such that $\mathscr{R}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathbf{R}_{\mathscr{L}}$. For an NP language $\mathscr{L} := \{\mathsf{x} \mid \exists \mathsf{w} : (\mathsf{x},\mathsf{w}) \in \mathbf{R}_{\mathscr{L}}\}$, a

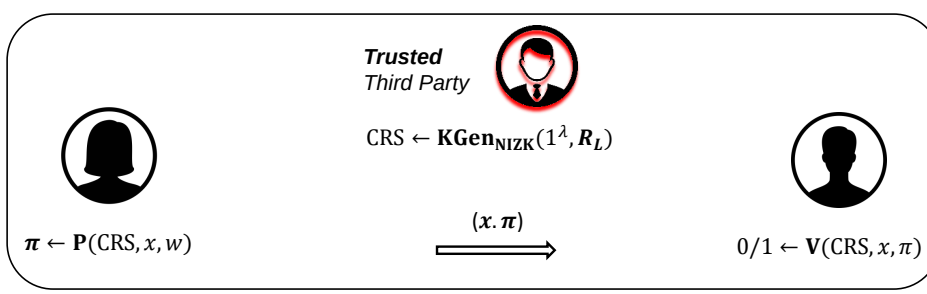

Figure 1. A NIZK argument in the CRS model.

NIZK argument consists of four algorithms $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ that are defined as follows,

- **CRS generator**, $\mathsf{crs} \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda, \mathbf{R}_{\mathscr{L}})$: Given the security parameter λ and a description of relation $\mathbf{R}_{\mathscr{L}}$ samples some secret trapdoors tc and use them to generate a common reference string crs; finally returns crs.
- **Prover**, $\pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{x}, \mathsf{w})$: Given the CRS crs, the statement x, and w if $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathscr{L}}$, it returns a proof π. Otherwise, it outputs $\perp$.
- **Verifier**, $0/1 \leftarrow \mathsf{V}(\mathsf{crs}, \mathsf{x}, \pi)$: Given the common reference string crs, the statement x, and a proof π returns either 0 (reject) or 1 (accept).
- **Simulator**, $\pi \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{x}, \mathsf{tc})$: Given the common reference string crs, the statement x, and the CRS trapdoor tc it returns a simulated proof π.

As mentioned above, the desired security requirement of a NIZK argument are completeness, soundness and zero-knowledge that formally are defined as below.

Definition 8 (Perfect Completeness)**.** *A non-interactive zero-knowledge argument* $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *is* perfectly complete for relation generator $\mathscr{R}$*, if for all* λ*, for all* $\mathbf{R}_{\mathscr{L}} \in \mathrm{im}(\mathscr{R}(1^\lambda))$*, and for all* $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathscr{L}}$*,*

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda, \mathbf{R}_{\mathscr{L}}), \pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{x}, \mathsf{w}) : \mathsf{V}(\mathsf{crs}, \mathsf{x}, \pi) = 1 \quad \right] = 1 \ .$$

Definition 9 (Computational Soundness)**.** *A non-interactive zero-knowledge argument* $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *is* (adaptively) computationally sound for relation generator $\mathscr{R}$*, if for all* λ*, for all* $\mathbf{R}_{\mathscr{L}} \in \mathrm{im}(\mathscr{R}(1^\lambda))$*, and for every PPT* $\mathscr{A}$*,*

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda, \mathbf{R}_{\mathscr{L}}), (\mathsf{x}, \pi) \leftarrow \mathscr{A}(\mathsf{crs}) : \\ \mathsf{V}(\mathsf{crs}, \mathsf{x}, \pi) = 1 \wedge (\mathsf{x} \notin \mathscr{L})\end{array}\right] = \mathsf{negl}(\lambda) \ .$$

Definition 10 (Perfect Zero-Knowledge)**.** *A non-interactive zero-knowledge argument* $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *guarantees* perfect zero-knowledge*, if for all* $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathscr{L}}$*, there exists a PPT algorithm* Sim*, such that for all non-uniform polynomial time adversary* $\mathscr{A}$*,*

$$\Pr\left[\begin{array}{l}(\mathsf{crs}, \mathsf{tc}) \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda, \mathbf{R}_{\mathscr{L}}) : \\ \mathscr{A}^{\mathsf{P}(\mathsf{crs}, \cdot, \cdot)}(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}) = 1\end{array}\right] = \Pr\left[\begin{array}{l}(\mathsf{crs}, \mathsf{tc}) \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda, \mathbf{R}_{\mathscr{L}}) : \\ \mathscr{A}^{\mathsf{Sim}(\mathsf{crs}, \mathsf{tc}, \cdot)}(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}) = 1\end{array}\right] \ ,$$

where both $\mathsf{P}(\mathsf{crs}, \cdot, \cdot)$ *and* $\mathsf{Sim}(\mathsf{crs}, \mathsf{tc}, \cdot)$ *returns* $\perp$ *if* $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathscr{L}}$*, and otherwise they return* $\mathsf{P}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \mathsf{w})$ *and* $\mathsf{Sim}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \mathsf{tc})$*, respectively.*

Intuitively, the existence of simulator Sim shows that a valid proof could have been generated by Sim that knows CRS trapdoors tc, but not the witness w. As a result, the proof does not reveal any information about the witness to a malicious verifier $\mathscr{A}$.

2.4.2. Zk-SNARKs

Among various type of NIZKs, zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs) [31,65,78,79,82,96,105] are the most practically interesting ones that guarantee *succinctness*, *completeness*, *zero-knowledge*, and *knowledge-soundness*. Succinctness implies very short proofs and consequently very fast verification; knowledge-soundness guarantees that a successful prover who manages to come up with an acceptable proof, must *know* the witness. The concept of *knowing* is formalized by constructing an extraction algorithm Ext that can extract the witness from a successful adversary. The formal definitions of completeness and zero-knowledge are presented in definitions 8 and 10, so in the rest, we summarize the definition of succinctness [82] and (non-black-box) knowledge-soundness [79] that is achieved in zk-SNARKs.

Definition 11 (Succinctness [82])**.** *A non-interactive zero-knowledge argument* $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *guarantees* succinctness *if the proof size is polynominal in* λ *and the verifier's computation time is polynominal in security parameter* λ *and size of the statement* x*.*

Definition 12 (Computational Non-black-box Knowledge Soundness [79])**.** *A non-interactive zero-knowledge argument of knowledge* $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *guarantees* computational non-black-box knowledge-soundness for the relation generator $\mathscr{R}$*, for all* $\mathbf{R}_{\mathscr{L}} \in \mathrm{im}(\mathscr{R}(1^\lambda))$*, if for every PPT* $\mathscr{A}$*, there exists a PPT extractor* $\mathsf{Ext}_{\mathscr{A}}$ *s.t. for all* λ*,*

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda, \mathbf{R}_{\mathscr{L}}), ((\mathsf{x}, \pi) \parallel \mathsf{w}) \leftarrow (\mathscr{A} \parallel \mathsf{Ext}_{\mathscr{A}})(\mathsf{crs}) : \\ \mathsf{V}(\mathsf{crs}, \mathsf{x}, \pi) = 1 \wedge ((\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathscr{L}})\end{array}\right] = \mathsf{negl}(\lambda) \quad .$$

Recently, it was shown that some new constructions of zk-SNARKs (e.g., [82]) can achieve *simulation* knowledge-soundness (a.k.a. simulation extractability) which is a stronger version of knowledge-soundness, as it also can guarantee non-malleability of proofs. In the rest, we recall the definition of nBB simulation knowledge-soundness (a.k.a. nBB simulation extractability) that we aim to achieve in new constructions in Chapters 4 and 6.

Definition 13 (Computational Non-black-box Simulation Knowledge Soundness)**.** *A non-interactive zero-knowledge argument* $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *guarantees* non-black-box simulation knowledge-soundness *for the relation generator* $\mathscr{R}$*, if for all* $\mathbf{R}_{\mathscr{L}} \in \mathrm{im}(\mathscr{R}(1^\lambda))$*, for any PPT* $\mathscr{A}$*, there exists a PPT extractor* $\mathsf{Ext}_{\mathscr{A}}$ *s.t. for all* λ*,*

$$\Pr\left[\begin{array}{l}(\mathsf{crs}, \mathsf{tc}) \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda, \mathbf{R}_{\mathscr{L}}), \\ ((\mathsf{x}, \pi), \mathsf{w}) \leftarrow (\mathscr{A}^{\mathsf{Sim}(\mathsf{crs}, \mathsf{tc}, \cdot)}, \mathsf{Ext}_{\mathscr{A}})(\mathsf{crs}) : \\ ((\mathsf{x}, \pi) \notin Q) \wedge ((\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathscr{L}}) \wedge \mathsf{V}(\mathsf{crs}, \mathsf{x}, \pi) = 1\end{array}\right] = \mathsf{negl}(\lambda) \quad ,$$

where Q *is the set of* (x, π)*-pairs generated by the adversary's queries to the simulation oracle* $\mathsf{Sim}(\mathsf{crs}, \mathsf{tc}, \cdot)$*.*

One may notice that nBB simulation knowledge-soundness implies nBB *knowledge-soundness* (given in Def. 9), as the former additionally allows the adversary to send a query to the proof simulator $\mathsf{Sim}(\mathsf{crs},\mathsf{tc},\cdot)$. It is worth to mention that in both definitions nBB knowledge-soundness (given in Def. 9) and nBB simulation knowledge-soundness (given in Def. 13) the extractor $\mathsf{Ext}_{\mathscr{A}}$ is non-black-box and to be able to extract the witness w it requires to have full access to the source code and random coins of adversary $\mathscr{A}$.

Definition 14 (Computational Black-Box Simulation Knowledge Soundness)**.** *A non-interactive zero-knowledge argument* $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim}, \mathsf{Ext})$ *guarantees* black-box simulation knowledge-soundness *for the relation generator* $\mathscr{R}$*, if for all* $\mathbf{R}_{\mathscr{L}} \in \mathrm{im}(\mathscr{R}(1^{\lambda}))$*, there exists a PPT extractor* Ext *s.t., for all PPT* $\mathscr{A}$*, and for all* λ*,*

$$\Pr\left[\begin{array}{l}(\mathsf{crs},\mathsf{ts},\mathsf{tx}) \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^{\lambda},\mathbf{R}_{\mathscr{L}}), (\mathsf{x},\pi) \leftarrow \mathscr{A}^{\mathsf{Sim}(\mathsf{crs},\mathsf{ts},\cdot)}(\mathsf{crs}),\\ \mathsf{w} \leftarrow \mathsf{Ext}(\mathsf{crs},\mathsf{x},\pi,\mathsf{tx}) : ((\mathsf{x},\pi) \notin Q) \wedge ((\mathsf{x},\mathsf{w}) \notin \mathbf{R}_{\mathscr{L}})\\ \wedge \mathsf{V}(\mathsf{crs},\mathsf{x},\pi) = 1\end{array}\right] = \mathsf{negl}(\lambda) \ ,$$

where Q *is the set of* (x,π)*-pairs generated by the adversary's queries to the simulator* $\mathsf{Sim}(\mathsf{crs},\mathsf{tc},\cdot)$*. Above* tx *and* ts *denote the extraction and simulation trapdoors, respectively.*

2.4.3. NP Characterizations QAPs and SAPs

In order to construct a zk-SNARK for an NP language $\mathscr{L}$, one first needs to encode the language $\mathscr{L}$ to one of the characterizations that have an efficient reduction from either arithmetic or Boolean `Circuit-SAT` [52, 65, 79, 82, 97]. Among various characterizations, encoding from arithmetic circuits is easier to QAP (Quadratic Arithmetic Programs) and SAP (Square Arithmetic Programs), but while working with Boolean circuits encoding to SSP (Square Span Programs) and QSP (Quadratic Span Programs) are more convenient.

As the CRS length and prover computational complexity of pairing-based zk-SNARKs severely depends on the number of multiplication/Boolean gates in the circuit that encodes the language, therefore QAP-based zk-SNARKs are more efficient and practical. This is raised from the fact that a particular computation described as an arithmetic circuit more often requires a smaller number of multiplication gates in comparison with the equivalent Boolean circuit. As an instance, currently verifying $y = \mathsf{SHA256}(x)$ requires an arithmetic circuit which has around 25.550 multiplication gates [4], while doing the same with Boolean circuits requires around 120.000 Boolean gates [5].

In the rest, we summarize the characterizations QAPs and SAPs that both define NP-complete languages specified by a quadratic equation over polynomials

[4] Available on: `https://github.com/akosba/xjsnark`

[5] Available on: `http://stevengoldfeder.com/projects/circuits/sha2circuit.html`

and used in zk-SNARKs that we have studied in this thesis.

Quadratic Arithmetic Programs. QAP [65] is a language where for an input x and witness w, validity of $(\mathsf{x},\mathsf{w}) \in \mathbf{R}_{\mathscr{Q}}$ can be checked via a parallel quadratic check. By considering the fact that there is an efficient reduction from CIRCUIT-SAT to QAPs, any QAP-based zk-SNARK is a zk-SNARK for CIRCUIT-SAT. The key idea behind such reduction is that one assigns variable to the input and output wires of each gate and then rewrite each gate as an equation. Then, instead of giving proof for the circuit, a prover gives a proof for a set of equations.

Technically speaking, a QAP instance $\mathscr{Q}_p$ is defined as $(\mathbb{Z}_p, m_0, \ell, \{u_j, v_j, w_j\}_{j=0}^m)$, where m_0 is the length of the statement (e.g., public inputs and outputs in an arithmetic circuit), ℓ is a target polynomial (specified via the number of constraints, namely the number of multiplication gates in an arithmetic circuit), and u_j, v_j, w_j are three sets of polynomials degree $n-1$ that encode the input and output wires in the target arithmetic circuit that has n multiplication gates [65]. For a QAP instance $\mathscr{Q}_p$ the following relation is defined, where we assume $A_0 = 1$:

$$\mathbf{R}_{\mathscr{Q}} = \left\{ \begin{array}{l} (\mathsf{x},\mathsf{w}) : \mathsf{x} = (A_1,\ldots,A_{m_0})^\top \wedge \mathsf{w} = (A_{m_0+1},\ldots,A_m)^\top \wedge \\ \left(\sum_{j=0}^m A_j u_j(X)\right)\left(\sum_{j=0}^m A_j v_j(X)\right) \equiv \sum_{j=0}^m A_j w_j(X) \pmod{\ell(X)} \end{array} \right\}.$$

Alternatively, $(\mathsf{x},\mathsf{w}) \in \mathbf{R}_{\mathscr{Q}}$ if there exists a (degree $\leq n-2$) polynomial $h(X)$, s.t.

$$\left(\sum_{j=0}^m A_j u_j(X)\right)\left(\sum_{j=0}^m A_j v_j(X)\right) - \sum_{j=0}^m A_j w_j(X) = h(X)\ell(X)\ ,$$

where $\ell(X) = \prod_{i=1}^n (X - \omega^{i-1})$ is a polynomial related to Lagrange interpolation, and in practice ω usually is set to be n-th primitive root of unity modulo p (by assuming that n is a power of two).

In a QAP-based NIZKs (or zk-SNARKs [65]), the goal of the prover is to prove that for public $(A_1,\ldots,A_{m_0})$ and $A_0 = 1$, she knows $(A_{m_0+1},\ldots,A_m)$ and a degree $\leq n-2$ polynomial $h(X)$, such that the above equation holds.

Square Arithmetic Programs. While encoding an arithmetic circuit to a QAP instance, the circuit should have only addition and multiplication gates. It is shown that any quadratic arithmetic circuit with fan-in 2 over a finite field $\mathbb{Z}_p$, can be converted to a SAP instance over the same finite field [82]. Specifically for multiplication gates one can use the fact that $ab = ((a+b)^2 - (a-b)^2)/4$. A SAP instance is defined as $\mathscr{S}_p = (\mathbb{Z}_p, m_0, \{u_j, w_j\}_{j=0}^m)$, which can be considered as a particular case of a QAP instance where $\{u_j\}_{j=0}^m = \{v_j\}_{j=0}^m$. A SAP defines the following relation:

$$\mathbf{R}_{\mathscr{S}} = \left\{ \begin{array}{l} (\mathsf{x},\mathsf{w}) : \mathsf{x} = (A_1,\ldots,A_{m_0})^\top \wedge \mathsf{w} = (A_{m_0+1},\ldots,A_m)^\top \wedge \\ \left(\sum_{j=0}^m A_j u_j(X)\right)^2 \equiv \sum_{j=0}^m A_j w_j(X) \pmod{\ell(X)} \end{array} \right\}$$

where $\ell(X) := \prod_{i=1}^{n}(X - \omega^{i-1})$ a degree n polynomial which in the case that n is set to be a power of two, and ω is the n-th root of unity modulo p, it equal to $\ell(X) := X^n - 1$, such that $\ell(\omega^{i-1}) = 0$ for all $i \in [1..n]$. Alternatively, $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathscr{S}}$ if there exists a (degree $\leq n-2$) polynomial $h(X)$, s.t.

$$\left(\sum_{j=0}^{m} A_j u_j(X)\right)^2 - \sum_{j=0}^{m} A_j w_j(X) = h(X)\ell(X) \ .$$

Similar to QAP-based NIZKs (or zk-SNARKs [65]), in a SAP-based NIZK (or zk-SNARK [82]) the goal of the prover is to prove that for public $(A_1, \ldots, A_{m_0})$ and $A_0 = 1$, she knows $(A_{m_0+1}, \ldots, A_m)$ and a degree $\leq n-2$ polynomial $h(X)$, such that above equation holds. As discussed by Groth [79], usually a SAP-based argument leads to a more succinct proof [79, 104], but it increases the prover's computational complexity, as each multiplication gate require two squaring gates.

2.4.4. Zk-SNARKs in Practical Applications

In recent years, due to the practical efficiency of zk-SNARKs in proving the correctness of any computational task without leaking any information about the secret inputs, they have appeared in various applications. Initiated by Pinocchio, a system for verifiable computations [105], they are deployed in various blockchain protocols including privacy-preserving cryptocurrencies [27], privacy-preserving smart contract systems [90, 93], private proof-of-stake protocols [92], ledger verification protocols [102] and so on.

Using zk-SNARKs in Practical Applications: In all of the mentioned applications, the zk-SNARK is used essentially to prove the correctness of a particular computation that is written as an arithmetic or Boolean circuit. Fig. 2 shows a graphical representation of the steps that one needs to take when using zk-SNARKs to prove the correctness of a particular computation. For instance, in the privacy-preserving cryptocurrency Zcash [27] the procedure of spending a coin is written as an arithmetic circuit, which is called the *Pour* circuit. Whenever a network node aims to spend a coin, they need to execute the Pour circuit with their secret inputs (related to the coin that will be spent) and output a new coin for the receiver, but attached with a zk-SNARK proof to prove the correctness of computations without leaking information about the spender's secret inputs. It is worth to mention that due to the knowledge-soundness of zk-SNARKs a malicious spender cannot spend a coin without knowing the secret information of the spent coin. Similarly, due to the zero-knowledge property of zk-SNARKs, a malicious verifier does not learn anything about secret information of the spender. The same procedure is repeated in all other (privacy-preserving) applications that use zk-SNARKs [90, 92, 93, 102], but each one with a different arithmetic circuit.

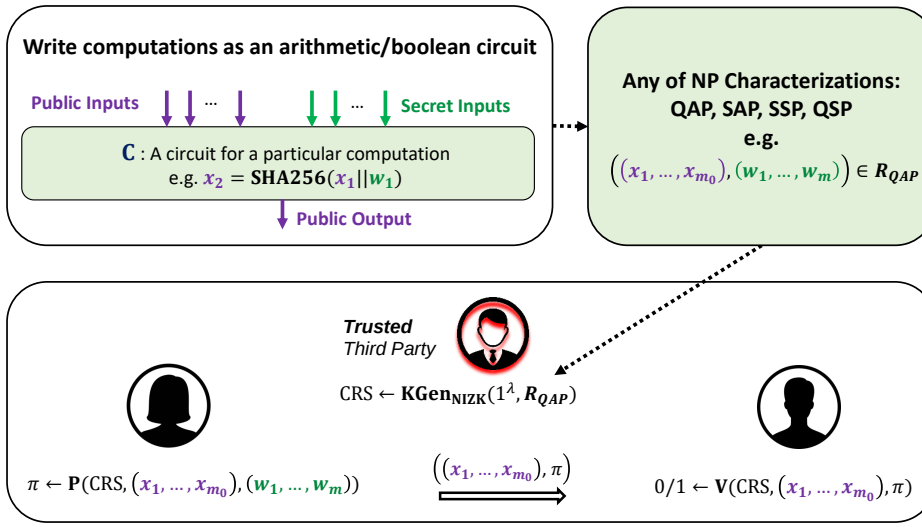

Figure 2. Using zk-SNARKs to prove the correctness of a particular computation that can be written as an arithmetic or Boolean circuit without leaking information about the secret inputs.

2.4.5. Zk-SNARKs in Universally Composable Protocols

Recall that Universal Composability (UC) [44] is a very strong security notion in cryptography which is an imperative and necessary requirement in constructing larger practical cryptographic systems. Briefly speaking, UC-security guarantees that a cryptographic primitive or protocol remains secure even if it is arbitrarily composed with other instances of the same or other primitives/protocols.

During the last few years, by developing zk-SNARKs, several practical cryptographic protocols are constructed that aimed to guarantee UC-security and deploy zk-SNARKs in their systems [90, 92, 93]. But as the default security of zk-SNARKs, namely non-black-box knowledge-soundness, was very weak to be directly used in UC-secure protocols, one needed to lift their security before using them in UC-secure protocols. A technical reason that zk-SNARKs cannot directly be used in UC-secure protocols is that the extraction algorithm constructed in security proofs of zk-SNARKs is non-black-box and depends on the source code of a particular adversary and it cannot work universally for all adversaries. But in the UC framework, the UC-simulator should be able to simulate all honest and corrupted parties, and to do so it should be able to extract witnesses from corrupted parties without requiring access to their source code. On the other hand, it is already observed and proven [45, 76, 85] that to be able to achieve UC-security in NIZKs, the NIZK argument should satisfy black-box simulation knowledge-soundness, defined in Def. 14. In other words, it is shown that black-box simulation knowledge-soundness is a sufficient requirement for a NIZK argument to realize the ideal functionality of NIZK arguments in the UC framework [76]. In such arguments, the extraction is done in a black-box manner (i.e. without a

knowledge assumption) and the proofs are non-malleable. By considering this result, in 2015, Kosba et al. [94] proposed a framework called C∅C∅, which consists of several constructions in which the strongest one gets a sound NIZK and lifts it to a NIZK that can achieve black-box simulation knowledge-soundness, sufficient to achieve UC-security.

In the rest, we review the most efficient and strong construction of the C∅C∅ framework used by various systems to construct black-box simulation knowledge-sound zk-SNARKs that are used in UC-secure protocols. Note that in the output constructions, the proof size is not *witness* succinct anymore.

The Strongest Construction in the C∅C∅ Framework. In summary, given a sound NIZK, the framework initially defines a new language $\mathscr{L}''$ based on the language $\mathscr{L}$ in the input NIZK, along with some cryptographic primitives. Let $\Pi_{\mathsf{Enc}} = (\mathsf{KGen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ be a set of algorithms for a semantically secure encryption scheme, $\Pi_{\mathsf{Sig}} = (\mathsf{KGen}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf})$ be a one-time signature scheme and $\Pi_{\mathsf{Com}} = (\mathsf{KGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Ver})$ be a perfectly binding commitment scheme. Given a language $\mathscr{L}$ with the corresponding NP relation $\mathbf{R}_{\mathscr{L}}$, define a new language $\mathscr{L}''$ such that $((\mathsf{x}, c, \mu, \mathsf{vk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Enc}}, \rho), (r, r_0, \mathsf{w}, s_0)) \in \mathbf{R}_{\mathscr{L}''}$ iff:

$$(c = \mathsf{Enc}(\mathsf{pk}_{\mathsf{Enc}}, \mathsf{w}; r)) \wedge ((\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathscr{L}} \vee (\mu = f_{s_0}(\mathsf{vk}_{\mathsf{Sig}}) \wedge \rho = \mathsf{Com}(s_0; r_0))),$$

where $\{f_s : \{0,1\}^* \to \{0,1\}^\lambda\}_{s \in \{0,1\}^\lambda}$ is a pseudo-random function family. The intuition behind the definition of language $\mathscr{L}''$ is that the prover has to send the encryption of witness with the public key given in the CRS along with a one-time secure signature of the proof. Attaching the encryption of witnesses allows the extractor to extract the witnesses from a valid proof with decryption, and the signature ensures the non-malleability of the proof. Now, a sound NIZK argument system Π_{NIZK} for $\mathscr{R}$ constructed from PPT algorithms $(\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ can be lifted to a BB simulation knowledge sound NIZK Π'_{NIZK} with PPT algorithms $(\mathsf{KGen}'_{\mathsf{NIZK}}, \mathsf{P}', \mathsf{V}', \mathsf{Sim}', \mathsf{Ext}')$ described in Fig. 3.

In Chapter 6, we compare our proposed constructions with the zk-SNARKs that are lifted with the C∅C∅ framework [90, 93].

- **CRS generator**, $\mathsf{crs}' \leftarrow \mathsf{KGen}'_{\mathsf{NIZK}}(1^\lambda, \mathbf{R}_{\mathscr{L}})$: Given the security parameter λ and description of relation $\mathbf{R}_{\mathscr{L}}$,
  - sample $(\mathsf{crs} \parallel \mathsf{ts}) \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda, \mathbf{R}_{\mathscr{L}''})$;
  - $(\mathsf{pk}_{\mathsf{Enc}}, \mathsf{sk}_{\mathsf{Enc}}) \leftarrow \mathsf{KGen}_{\mathsf{Enc}}(1^\lambda)$;
  - $s_0, r_0 \leftarrow_{\$} \{0,1\}^\lambda$; $\rho := \mathsf{Com}(s_0; r_0)$; and

  output $(\mathsf{crs}' \parallel \mathsf{ts}' \parallel \mathsf{tx}') := ((\mathsf{crs}, \mathsf{pk}_{\mathsf{Enc}}, \rho) \parallel (s_0, r_0) \parallel \mathsf{sk}_{\mathsf{Enc}})$. Note that here $\mathsf{tc}' := (\mathsf{ts}' \parallel \mathsf{tx}')$ can be considered as the CRS trapdoor.
- **Prover**, $\pi' \leftarrow \mathsf{P}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}, \mathsf{x}, \mathsf{w})$: Parse $\mathsf{crs}' := (\mathsf{crs}, \mathsf{pk}_{\mathsf{Enc}}, \rho)$; Abort if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathscr{L}}$;
  - $(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{KGen}_{\mathsf{Sig}}(1^\lambda)$;
  - sample $z_0, z_1, z_2, r_1 \leftarrow_{\$} \{0,1\}^\lambda$; compute $c = \mathsf{Enc}(\mathsf{pk}_{\mathsf{Enc}}, \mathsf{w}; r_1)$;
  - generate $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, c, z_0, \mathsf{vk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Enc}}, \rho), (r_1, z_1, w, z_2))$;
  - sign $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_{\mathsf{Sig}}, (\mathsf{x}, c, z_0, \pi))$;

  and output $\pi' := (c, z_0, \pi, \mathsf{vk}_{\mathsf{Sig}}, \sigma)$.
- **Verifier**, $0/1 \leftarrow \mathsf{V}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}', \mathsf{x}, \pi')$: Parse $\mathsf{crs}' := (\mathsf{crs}, \mathsf{pk}_{\mathsf{Enc}}, \rho)$ and $\pi' := (c, \mu, \pi, \mathsf{vk}_{\mathsf{Sig}}, \sigma)$;
  - Abort if $\mathsf{Vf}(\mathsf{vk}_{\mathsf{Sig}}, (\mathsf{x}, c, \mu, \pi), \sigma) = 0$;
  - call $\mathsf{V}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, c, \mu, \mathsf{vk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Enc}}, \rho), \pi)$ and abort if it outputs 0.
- **Simulator**, $\pi' \leftarrow \mathsf{Sim}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}', \mathsf{ts}', \mathsf{x})$: Parse $\mathsf{crs}' := (\mathsf{crs}, \mathsf{pk}_{\mathsf{Enc}}, \rho)$ and $\mathsf{ts}' := (s_0, r_0)$;
  - $(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{KGen}_{\mathsf{Sig}}(1^\lambda)$;
  - set $\mu = f_{s_0}(\mathsf{vk}_{\mathsf{Sig}})$;
  - sample $z_3, r_1 \leftarrow_{\$} \{0,1\}^\lambda$; compute $c = \mathsf{Enc}(\mathsf{pk}_{\mathsf{Enc}}, z_3; r_1)$;
  - generate $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, c, \mu, \mathsf{vk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Enc}}, \rho), (r_1, r_0, z_3, s_0))$;
  - sign $\sigma \leftarrow \mathsf{Sig}(\mathsf{vk}_{\mathsf{Sig}}, (\mathsf{x}, c, \mu, \pi))$;

  and output a simulated proof $\pi' := (c, \mu, \pi, \mathsf{vk}_{\mathsf{Sig}}, \sigma)$.
- **Extractor**, $\mathsf{w} \leftarrow \mathsf{Ext}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}', \mathsf{tx}', \mathsf{x}, \pi')$: Parse $\pi' := (c, \mu, \pi, \mathsf{vk}_{\mathsf{Sig}}, \sigma)$, $\mathsf{sk}_{\mathsf{Enc}} := \mathsf{tx}'$; decrypt $\mathsf{w} \leftarrow \mathsf{Dec}(\mathsf{sk}_{\mathsf{Enc}}, c)$; output w.

Figure 3. C∅C∅: a framework for constructing black-box simulation knowledge-sound (a.k.a. simulation extractable) NIZK arguments [94].

3. SUBVERSION-RESISTANT KNOWLEDGE SOUND SNARKS

3.1. Motivation

Due to very practical efficiency of zk-SNARKs, recently they are deployed in a large number of practical applications with different functionalities [27, 90, 92, 93, 102, 105]. Notably, Pinocchio [105] was the first nearly practical system for verifiable computations. Pinocchio has various compilers and encoders that allow a user to compile any computation (e.g., computations required in spending a digital coin, or computations inside a smart contract, or computations needed in verifying transactions on a ledger or encrypting a message with RSA cryptosystem) written in C/C++ language to an arithmetic/Boolean circuit and then encodes the circuit to one of the NP characterizations QAP/SAP/QSP/SSP and finally uses zk-SNARKs to prove the correctness of the computation. In a practical application that uses zk-SNARKs, beside succinct proofs (e.g., less than 288 bytes) and very efficient verification (e.g., in less than 10 milliseconds), *knowledge-soundness* (defined in Def. 12) and *zero-knowledge* (defined in Def.10) properties of the underlying zk-SNARK convinces a verifier that the prover has done the computations correctly and knows the values used in the computation while guaranteeing that a malicious verifier does not learn more than correctness of the computations from the proof.

As mentioned in Chapter 2, zk-SNARKs require a setup phase to generate a CRS crs, a.k.a. public parameters, that are used by prover and verifier for proof generation and proof verification [35]. This setup phase is supposed to be done by a trusted third party or by a distributed authority. Following this fact, any application and system that uses zk-SNARKs as a sub-protocol is required to have a (widely accepted) trusted setup phase which is a publicly trustable system. In other words, in any practical application where zk-SNARKs are deployed, for instance Zcash [27] or Hawk [93], all users of the application (e.g., nodes of Zcash network) either in the role of a prover or a verifier need to trust the party who runs the setup phase of zk-SNARKs.

Recently by increasing global surveillance disclosures, e.g., the Snowden revelations, users' sensitivity to cryptographic protocols and their public parameters has grown enormously. Following this fact, there have been various researches that reduce required trust in various cryptographic protocols or primitives and new variations that provide a level of security against active subversion [7, 8, 20, 25]. Roughly speaking, subversion mainly refers to manipulations on the public parameters or setup phase of the protocol.

From a different perspective, Multi-Party Computation (MPC) protocols are a standard and common approach widely used in various cryptographic systems to distribute and mitigate the level of needed trust. In the context of zk-SNARKs, in 2015, Ben Sasson et al. [29] proposed an efficient MPC protocol that can be used to sample the CRS for a majority of pairing-based zk-SNARKs. Later, applica-

tions such as the privacy-preserving cryptocurrency Zcash used Ben Sasson et al.'s MPC protocol to sample public parameters of their underlying zk-SNARK [42] which led to mitigating the required trust on setup phase for both prover and verifier from a *single party* to *1 out of n* parties, where n denotes the number of parties participating in the MPC protocol.

3.2. Problem Statement

To achieve zero-knowledge and knowledge-soundness in zk-SNARKs, by using Ben Sasson et al.'s [29] MPC protocol for CRS generation both prover and verifier need to trust at least 1 out of n players of the MPC protocol. Considering this required trust in the setup phase of zk-SNARKs by both prover and verifier, can we reduce the trust on the setup phase of zk-SNARK even more, so that users of zk-SNARKs will use them more confidentially in their practical applications? For instance, the nodes of the Zcash or the Hawk smart contract system will get the same security guarantees but with less trust in the generator of public parameters.

3.3. Previous Results

In 2016, Bellare, Fuchsbauer and Scafuro [20] studied achievable security properties in NIZK arguments in the case that the setup phase is subverted. They first defined three notions called subversion witness indistinguishability (Sub-WI), subversion zero-knowledge (Sub-ZK) and subversion soundness (Sub-SND) as a subversion-resistant variation of the standard notions witness indistinguishability (WI), zero-knowledge (ZK) and soundness in NIZK arguments. The key change in their new definitions is that unlike for the standard cases, in the new definitions it is assumed that the parameters are generated by an adversary. For instance, the notion Sub-ZK guarantees that even if an adversary generates the CRS elements, the NIZK argument still guarantees ZK.

Based on these new definitions, they proved a negative and several positive results about constructing subversion-resistant NIZKs. First, they showed that the definitions of Sub-SND and (even standard) ZK are not compatible and we cannot have a NIZK argument that will achieve Sub-SND and standard ZK at the same time. Intuitively, one may notice that achieving ZK in a NIZK argument confirms the existence of a simulator Sim that given trapdoors of the setup phase can generate a fake proof which will be indistinguishable from the real ones. On the other hand, Sub-SND requires that an adversary cannot forge the proof, even if he generates the CRS. But, by the definition of ZK one can notice that this is a contradiction: given the CRS trapdoors and the simulator Sim constructed in proving ZK, one can generate fake proofs indistinguishable from real ones without knowing the witness which will break Sub-SND.

On the positive side, in the best case, they showed that under an additional knowledge assumption on the setup phase, one can construct a NIZK argument

that can guarantee Sub-ZK and (knowledge) soundness. To achieve Sub-ZK, instead of giving the *simulation trapdoors* directly to the simulator Sim, their key idea was to use a knowledge assumption to extract the trapdoors from a maliciously generated CRS and then use them to simulate the argument. Constructing such NIZK arguments showed that we can completely eliminate the prover's required trust in the CRS generators. In other words, in a NIZK argument that guarantees Sub-ZK, a prover does not need to trust the CRS generators and it can achieve ZK without trusting anyone, but instead doing some initial checks. In another positive result, they showed that non-interactive arguments proposed by Groth, Ostrovsky, and Sahai [84] can achieve Sub-WI and Sub-SND, as they do not require a particular setup phase.

3.4. Subversion-Resistant SNARKs

By considering the discussed positive results about NIZKs with subverted parameters [20], we started to consider if we can construct subversion-resistant zk-SNARKs which achieve their best positive result, namely Sub-ZK along with soundness. Briefly speaking, we present a variation of the-state-of-the-art zk-SNARK, proposed by Groth [79], that can achieve Sub-ZK and knowledge-soundness.

A full detailed description of the result can be found in the paper [3] which is joint work with Abdolmaleki, Lipmaa, and Zając. In the rest of the section, we first summarize the definition of a subversion-resistant SNARK and then explain our proposed variation of Groth's zk-SNARK that can achieve Sub-ZK and knowledge-soundness. Finally, we evaluate the practical performance of the proposed subversion-resistant zk-SNARK.

3.4.1. Definitions

A standard definition of zk-SNARKs is given in Section 2.4. In the rest, we present our definitions for subversion-resistant zk-SNARKs and their security requirements. To achieve Sub-ZK, we augment a zk-SNARK by requiring the existence of an efficient CRS verification algorithm, called CV. Our definition of Sub-ZK for SNARKs is motivated by the definitions presented in [20, 76].

As before, let $\mathscr{R}$ be a relation generator, such that $\mathscr{R}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathbf{R}$, with auxiliary information $\mathsf{aux}_{\mathbf{R}}$. The security parameter λ can be determined from the description of $\mathbf{R}$. The relation generator also outputs auxiliary information $\mathsf{aux}_{\mathbf{R}}$ that will be given to all algorithms including the adversary. In the rest, for simplicity we ignore writing $\mathsf{aux}_{\mathbf{R}}$ in the inputs of algorithms. Let $\mathscr{L}_{\mathbf{R}} = \{\mathsf{x} : \exists \mathsf{w}, (\mathsf{x}, \mathsf{w}) \in \mathbf{R}\}$ be an NP-language. A subversion-resistant *NIZK argument system* Π_{NIZK} for $\mathscr{R}$ consists of a tuple of PPT algorithms $(\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{CV}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$, that are defined as below,

- **CRS generator**, $\mathsf{crs} := (\mathsf{crs}_{\mathsf{P}}, \mathsf{crs}_{\mathsf{V}}, \mathsf{crs}_{\mathsf{CV}}) \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(\mathbf{R})$: Given $\mathbf{R} \in \mathrm{im}(\mathscr{R}(1^\lambda))$, samples the CRS trapdoor tc, then computes and returns crs

and stores trapdoors of crs as tc. We distinguish crs_P (needed by the prover), crs_V (needed by the verifier) and crs_CV (necessary for CRS verifier).

- **CRS Verifier**, $\{0,1\} \leftarrow \mathsf{CV}(\mathbf{R}, \mathsf{crs})$: Given $\mathbf{R} \in \mathrm{im}(\mathscr{R}(1^\lambda))$ and crs it verifies weather crs is well-formed and returns either 0 (the CRS is incorrectly formed) or 1 (the CRS is correctly formed).
- **Prover**, $\pi \leftarrow \mathsf{P}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \mathsf{w})$: Given $(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \mathsf{w})$ for $\mathsf{CV}(\mathbf{R}, \mathsf{crs}) = 1$ and $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$, outputs an argument π. Otherwise, it outputs $\perp$.
- **Verifier**, $\{0,1\} \leftarrow \mathsf{V}(\mathbf{R}, \mathsf{crs}_\mathsf{V}, \mathsf{x}, \pi)$: Given $(\mathbf{R}, \mathsf{crs}_\mathsf{V}, \mathsf{x}, \pi)$, returns either 0 (reject proof) or 1 (accept proof).
- **Simulator**, $\pi \leftarrow \mathsf{Sim}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \mathsf{ts})$: Given $(\mathbf{R}, \mathsf{crs}, \mathsf{x})$, and simulation trapdoors ts the algorithm outputs a simulated argument π. Note that simulation trapdoors can be a subset or equal to CRS trapdoors tc.

As discussed in Section 2.4, a (non subversion-resistant) NIZK argument is defined as a tuple $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$. One may notice that in above algorithms there are some differences in comparison with the established syntax of NIZK arguments. An essential difference in the new syntax is the existence of an efficient CV algorithm which is crucial for achieving Sub-ZK. The checks inside the CV algorithm verify the consistency of the elements of crs along with the well-formedness of some particular elements. From a high-level view, executing CV is the cost that a prover needs to pay to achieve Sub-ZK instead of ZK. In applications that a prover (e.g., a node in Zcash network) frequently uses a fixed CRS, the user only needs to run the CV algorithm once. Another difference is the separation of crs into three parts $(\mathsf{crs}_\mathsf{P}, \mathsf{crs}_\mathsf{V}, \mathsf{crs}_\mathsf{CV})$, which is done to highlight that to be able to construct an efficient CV algorithm, one may need to add some new elements to the CRS of the original zk-SNARK. Essentially, a verifier only needs crs_V which in practice usually has a considerably smaller size than crs_P, and crs_CV contains elements that are required to verify the well-formedness of CRS.

The desired security requirement of a zk-SNARK is *completeness*, *knowledge-soundness* and *zero-knowledge* that were formally defined in Section 2.4.2. In the rest, we present the definition of *subversion completeness* (an honest P convinces an honest V, and an honestly generated CRS gets accepted by the CV algorithm) and *Sub-ZK* (a malicious verifier does not learn more than the truth of the statement, even if it generates the CRS) that are two essential notions in subversion-security which we aim to achieve in our proposed subversion-resistant zk-SNARKs.

Definition 15 (Perfect Subversion-Completeness)**.** *A subversion-resistant non-interactive zero-knowledge argument* $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{CV}, \mathsf{V}, \mathsf{Sim})$ *is* perfectly subversion complete for $\mathscr{R}$*, if for all* λ*, all* $\mathbf{R} \in \mathrm{im}(\mathscr{R}(1^\lambda))$*, and* $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$*,*

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(\mathbf{R}), \pi \leftarrow \mathsf{P}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \mathsf{w}) : \\ \mathsf{CV}(\mathbf{R}, \mathsf{crs}) = 1 \wedge \mathsf{V}(\mathbf{R}, \mathsf{crs}, \mathsf{x}, \pi) = 1\end{array}\right] = 1 \ .$$

Definition 16 (Statistically Sub-ZK)**.** *A subversion-resistant non-interactive zero-knowledge argument* $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{CV}, \mathsf{V}, \mathsf{Sim})$ *is* statistically Sub-ZK for $\mathscr{R}$*, if for any PPT subverter* Sub *there exists a PPT extractor* $\mathsf{Ext}_{\mathsf{Sub}}$*, such that for all* λ*, all* $\mathbf{R} \in \mathrm{im}(\mathscr{R}(1^\lambda))$*, and for all PPT* $\mathscr{A}$*,* $\varepsilon_0 \approx_\lambda \varepsilon_1$*, where*

$$\varepsilon_b = \Pr\left[\begin{array}{l} r \leftarrow_r \mathsf{RND}(\mathsf{Sub}), (\mathsf{crs}, \mathsf{aux}_{\mathsf{Sub}} \parallel \mathsf{ts}) \leftarrow (\mathsf{Sub} \parallel \mathsf{Ext}_{\mathsf{Sub}})(\mathbf{R}; r) : \\ \mathsf{CV}(\mathbf{R}, \mathsf{crs}) = 1 \wedge \mathscr{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathbf{R}, \mathsf{crs}, \mathsf{ts}, \mathsf{aux}_{\mathsf{Sub}}) = 1 \end{array}\right].$$

Here, $r \leftarrow_r \mathsf{RND}(\mathscr{A})$ *denote sampling of a randomizer* r *of sufficient length for* $\mathscr{A}$*'s needs,* $\mathsf{aux}_{\mathsf{Sub}}$ *is auxiliary information generated by the subverter* Sub*, and the oracle* $\mathsf{O}_0(\mathsf{x}, \mathsf{w})$ *returns* $\perp$ *(reject) if* $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}$*, and otherwise it returns* $\mathsf{P}(\mathbf{R}, \mathsf{crs}_{\mathsf{P}}, \mathsf{x}, \mathsf{w})$*. Similarly,* $\mathsf{O}_1(\mathsf{x}, \mathsf{w})$ *returns* $\perp$ *(reject) if* $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}$*, and otherwise it returns* $\mathsf{Sim}(\mathbf{R}, \mathsf{crs}, \mathsf{ts}, \mathsf{x})$*.* Π_{NIZK} *is* perfectly Sub-ZK for $\mathscr{R}$ *if one requires that* $\varepsilon_0 = \varepsilon_1$*.*

Note that in Def. 16 the subverter Sub and adversary $\mathscr{A}$ can be a single party, consequently tc are provided to the adversary $\mathscr{A}$.

3.4.2. An Efficient Construction

Next, we present a minimally modified version of the state-of-the-art QAP-based zk-SNARK proposed by Groth [79] so that it achieves Sub-ZK (defined in Def. 16) and subversion completeness (defined in Def. 15). Note that currently Groth's zk-SNARK is the most efficient, which was the main reason that we focus on it[1].

Main Strategy to Achieve Sub-ZK in SNARKs: Before going through the presented construction, it is worth to mention that in a NIZK argument achieving Sub-ZK is a bit more challenging than achieving standard ZK. The reason is that in the definition of standard ZK, the CRS generator is trusted and the CRS trapdoors tc (consequently simulation trapdoors ts) are honestly provided to the simulator Sim. But in order to achieve Sub-ZK, since the prover does not trust the CRS generator anymore, consequently the simulator Sim cannot trust the provided trapdoors. To address this issue, the proposed solution [20] is that the prover checks the well-formedness of CRS elements before using them, and in simulating arguments the simulator uses a non-black-box extraction algorithm $\mathsf{Ext}_{\mathsf{Sub}}$ to extract the simulation trapdoors directly from the (possibly malicious) CRS generator Sub and then uses them for the simulation [20]. A known approach for constructing non-black-box extraction algorithm $\mathsf{Ext}_{\mathsf{Sub}}$ is to use a knowledge assumption [20, 49]. In our construction, we use the Bilinear Diffie-Hellman Knowledge of Exponents (BDH-KE) assumption [49] (defined in Assumption 4), while one may use a different knowledge assumption, as was later observed in a concurrent work by

[1] While less efficient zk-SNARKs like Pinocchio [105] are used more widely, this situation might change and one should precisely analyze for its construction.

Fuchsbauer [60] [2]. But, intuitively in all cases, one relies on the fact that if a (malicious) CRS generator manages to come out with some *well-formed* CRS elements, they must know the secret information used in CRS generation. Namely, there exists a non-black-box extractor $\mathsf{Ext}_{\mathsf{Sub}}$ that given access to the source code and random coins of the (malicious) CRS generator Sub, can extract the simulation trapdoors ts. Once the simulation trapdoors are extracted, the simulator Sim can simulate the proofs. It is worth to mention that the well-formedness of CRS elements are checked by a public efficient CRS verification algorithm known as CV. While making the CRS of the underlying NIZK well-formed, one may need to modify CRS elements of the construction which will require redoing the soundness proof. Note that, using different knowledge assumptions to achieve Sub-ZK in a particular NIZK argument, will result in different CV algorithms.

Construction of New Sub-ZK Secure SNARK: Next, we describe the construction of a subversion-resistant SNARK for $\mathscr{R}$ that satisfies Sub-ZK and is closely based on Groth's construction [79]. Fig. 4 and Fig. 5 describe the algorithms of the new construction $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{CV}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$. Importantly, the prover and the verifier of the new zk-SNARK are unchanged compared to Groth's SNARK [79]. We assume implicitly that each algorithm checks that their inputs belong to correct groups and that $\mathbf{R} \in \mathrm{im}(\mathscr{R}(1^{\lambda}))$.

The proposed Sub-ZK SNARK is closely based on Groth's zk-SNARK. In fact, the differences between the construction of the two SNARKs can be summarized very briefly:

- We proposed an efficient CRS verification algorithm CV which is constructed based on the BDH-KE assumption (see Fig. 4).
- We extend the CRS of the original scheme with $2n+3$ new elements (see the variable $\mathsf{crs}_{\mathsf{CV}}$ in Fig. 4) that are needed to make the original CRS well-formedness verifiable.

[2] He uses a different knowledge assumption called Square Knowledge of Exponent (SKE) which states for an asymmetric bilinear group, given $[1]_1$, if an adversary manages to come out with $[a]_1$ and $\left[a^2\right]_1$, he must know a.

Algorithm 2: Checking that $\left[(\ell_j(\chi))_{j=1}^{n}\right]_1$ is correctly computed (based on equation 2.1)

Input: $(\left[\chi^{n-1}, (\ell_j(\chi))_{j=1}^{n}\right]_1, [1,\chi]_2, [1]_T)$

// $j = 1$

$[\zeta]_T \leftarrow (\left[\chi^{n-1}\right]_1 \bullet [\chi]_2 - [1]_T)/n$; $\left[\omega'\right]_2 \leftarrow [1]_2$;

Check that $[\ell_1(\chi)]_1 \bullet ([\chi]_2 - \left[\omega'\right]_2) = [\zeta]_T$;

for $j = 2$ **to** n **do**

  $[\zeta]_T \leftarrow \omega\,[\zeta]_T$; $\left[\omega'\right]_2 \leftarrow \omega\left[\omega'\right]_2$;

  Check that $[\ell_j(\chi)]_1 \bullet ([\chi]_2 - \left[\omega'\right]_2) = [\zeta]_T$;

CRS generator, $\mathsf{crs} := (\mathsf{crs_P}, \mathsf{crs_V}, \mathsf{crs_{CV}}) \leftarrow \mathsf{KGen_{NIZK}}(\mathbf{R})$**:** Given a QAP $\mathbf{R} \in \mathrm{im}(\mathscr{R}(1^\lambda))$, first sample CRS trapdoors $\mathsf{tc} = (\chi, \alpha, \beta, \gamma, \delta) \leftarrow_r \mathbb{Z}_p^3 \times (\mathbb{Z}_p^*)^2$ and then set simulation trapdoors $\mathsf{ts} = (\chi, \alpha, \beta, \delta)$. Next, compute Lagrange basis $(\ell_i(\chi))_{i=1}^n$ and set $u_j(\chi) \leftarrow \sum_{i=1}^n U_{ij}\ell_i(\chi)$, $v_j(\chi) \leftarrow \sum_{i=1}^n V_{ij}\ell_i(\chi)$, $w_j(\chi) \leftarrow \sum_{i=1}^n W_{ij}\ell_i(\chi)$ for all $j \in \{0,\ldots,m\}$. Note that U_{ij}, V_{ij} and W_{ij} are the entries of QAP matrices when the circuit is encoded as three sets of matrices U, V, W with dimensions $n \times m$. Finally, compute $\mathsf{crs} = (\mathsf{crs_P}, \mathsf{crs_V}, \mathsf{crs_{CV}})$, where

$$\mathsf{crs_P} \leftarrow \left(\begin{bmatrix} \alpha, \beta, \delta, \left(\frac{u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi)}{\delta}\right)_{j=m_0+1}^{m} \end{bmatrix}_1, \\ \left[(\chi^i \ell(\chi)/\delta)_{i=0}^{n-2}, (u_j(\chi), v_j(\chi))_{j=0}^{m} \right]_1, \left[\beta, \delta, (v_j(\chi))_{j=0}^{m} \right]_2 \end{pmatrix},$$

$$\mathsf{crs_V} \leftarrow \left(\left[\left(\frac{u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi)}{\gamma}\right)_{j=0}^{m_0} \right]_1, [\gamma, \delta]_2, [\alpha\beta]_T \right) ,$$

$$\mathsf{crs_{CV}} \leftarrow \left([\gamma, (\chi^i)_{i=1}^{n-1}, (\ell_i(\chi))_{i=1}^n]_1, [\alpha, \chi, \chi^{n-1}]_2 \right) .$$

Return $\mathsf{crs} \leftarrow (\mathsf{crs_{CV}}, \mathsf{crs_P}, \mathsf{crs_V})$.

CRS Verifier, $\{0,1\} \leftarrow \mathsf{CV}(\mathbf{R}, \mathsf{crs})$**:**

1. For $\iota \in \{\gamma, \delta\}$: check that $[\iota]_1 \neq [0]_1$
2. For $\iota \in \{\alpha, \beta, \gamma, \delta\}$: check that $[\iota]_1 \bullet [1]_2 = [1]_1 \bullet [\iota]_2$,
3. For $i = 1$ to $n-1$: check that $\left[\chi^i\right]_1 \bullet [1]_2 = \left[\chi^{i-1}\right]_1 \bullet [\chi]_2$,
4. Check that $([\ell_i(\chi)]_1)_{i=1}^n$ is correctly computed by using Alg. 2, that is designed based on equation 2.1.)
5. For $j = 0$ to m:
   (a) Check that $[u_j(\chi)]_1 = \sum_{i=1}^n U_{ij}[\ell_i(\chi)]_1$,
   (b) Check that $[v_j(\chi)]_1 = \sum_{i=1}^n V_{ij}[\ell_i(\chi)]_1$,
   (c) Set $[w_j(\chi)]_1 \leftarrow \sum_{i=1}^n W_{ij}[\ell_i(\chi)]_1$,
   (d) Check that $[v_j(\chi)]_1 \bullet [1]_2 = [1]_1 \bullet [v_j(\chi)]_2$,
6. For $j = m_0 + 1$ to m: check that $[(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta]_1 \bullet [\delta]_2 = [u_j(\chi)]_1 \bullet [\beta]_2 + [v_j(\chi)]_1 \bullet [\alpha]_2 + [w_j(\chi)]_1 \bullet [1]_2$,
7. Check that $\left[\chi^{n-1}\right]_1 \bullet [1]_2 = [1]_1 \bullet \left[\chi^{n-1}\right]_2$,
8. For $i = 0$ to $n-2$: check that $\left[\chi^i \ell(\chi)/\delta\right]_1 \bullet [\delta]_2 = \left[\chi^{i+1}\right]_1 \bullet \left[\chi^{n-1}\right]_2 - \left[\chi^i\right]_1 \bullet [1]_2$,
9. Check that $[\alpha]_1 \bullet [\beta]_2 = [\alpha\beta]_T$.

Figure 4. The CRS generation and verification of the Sub-ZK SNARK for $\mathscr{R}$

Choosing which elements to add to the CRS is not straightforward since the zk-SNARK must remain knowledge-sound even given enlarged CRS, on the other hand the CRS must become publicly verifiable. Adding too many or just *wrong*

Prover, $\pi \leftarrow \mathsf{P}(\mathbf{R}, \mathsf{crs}, \mathsf{x} = (A_1, \ldots, A_{m_0}), \mathsf{w} = (A_{m_0+1}, \ldots, A_m))$**:** After executing CV successfully & assuming $A_0 = 1$, the prover does:

1. Let $a^{\dagger}(X) \leftarrow \sum_{j=0}^{m} A_j u_j(X)$, $b^{\dagger}(X) \leftarrow \sum_{j=0}^{m} A_j v_j(X)$,
2. Let $c^{\dagger}(X) \leftarrow \sum_{j=0}^{m} A_j w_j(X)$,
3. Set $h(X) = \sum_{i=0}^{n-2} h_i X^i \leftarrow (a^{\dagger}(X) b^{\dagger}(X) - c^{\dagger}(X))/\ell(X)$,
4. Set $[h(\chi)\ell(\chi)/\delta]_1 \leftarrow \sum_{i=0}^{n-2} h_i \left[\chi^i \ell(\chi)/\delta\right]_1$,
5. Set $r_a \leftarrow_r \mathbb{Z}_p$; Set $a \leftarrow \sum_{j=0}^{m} A_j [u_j(\chi)]_1 + [\alpha]_1 + r_a [\delta]_1$,
6. Set $r_b \leftarrow_r \mathbb{Z}_p$; Set $b \leftarrow \sum_{j=0}^{m} A_j [v_j(\chi)]_2 + [\beta]_2 + r_b [\delta]_2$,
7. Set $c \leftarrow r_b a + r_a \left(\sum_{j=0}^{m} A_j [v_j(\chi)]_1 + [\beta]_1\right) + \sum_{j=m_0+1}^{m} A_j [(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta]_1 + [h(\chi)\ell(\chi)/\delta]_1$,
8. Return $\pi \leftarrow (a, b, c)$.

Verifier, $\{0,1\} \leftarrow \mathsf{V}(\mathbf{R}, \mathsf{crs}_{\mathsf{V}}, \mathsf{x} = (A_1, \ldots, A_{m_0}), \pi = (a, b, c))$**:** assuming $A_0 = 1$, check that

$$a \bullet b = c \bullet [\delta]_2 + \left(\sum_{j=0}^{m_0} A_j \left[\frac{u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi)}{\gamma}\right]_1\right) \bullet [\gamma]_2 + [\alpha\beta]_T \quad .$$

Simulator, $\pi \leftarrow \mathsf{Sim}(\mathbf{R}, \mathsf{crs}, \mathsf{x} = (A_1, \ldots, A_{m_0}), \mathsf{ts} = (\chi, \alpha, \beta, \delta))$**:** Call the extraction algorithm $\mathsf{Ext}_{\mathsf{Sub}}$ which is constructed based on BDH-KE assumption and extract simulation trapdoors $\mathsf{ts} = (\chi, \alpha, \beta, \delta)$ of Π_{NIZK}; Next, pick $a', b' \leftarrow \mathbb{Z}_p^*$, and compute and return $\pi = (a, b, c)$, such that

$$a \leftarrow \left[a'\right]_1, b \leftarrow \left[b'\right]_2, c \leftarrow \left[\frac{a'b' - \alpha\beta - \sum_{j=0}^{m_0} A_j (u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))}{\delta}\right]_1.$$

Figure 5. The prover, the verifier and the simulator of the Sub-ZK SNARK for $\mathscr{R}$

elements to the CRS can break the knowledge-soundness. In the paper [3], we proved the following theorem and show that the construction proposed in Figures 4 and 5 satisfies completeness, Sub-ZK and knowledge-soundness.

Theorem 1 (Completeness, Subversion ZK, Knowledge Soundness)**.** *The zk-SNARK constructed in Figures 4 and 5 satisfies completeness, subversion zero-knowledge and computational knowledge soundness.*

Proof. The proof is given in [3] that is included in the thesis. □

It is worth to highlight that to achieve Sub-ZK, we added some elements to the CRS of the original scheme that all are in the span of other elements of the CRS. Considering this fact, we showed that the knowledge-soundness of the new variation follows closely the proof of original construction. One could also use the fact that knowledge-soundness of original scheme holds even in symmetric groups and since the elements that we added to the CRS are elements that had been given in the other group in the original scheme, so the new variation is would satisfy

Table 1. The length of CRS in the proposed subversion-resistant zk-SNARK

CRS Elements in	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	Total
crs_P	$3m+n-m_0+4$	$m+3$	0	$4m+n-m_0+7$
crs_V	m_0+1	2	1	m_0+4
crs_CV	$2n$	3	0	$2n+3$
Total	$3m+3n+5$	$m+7$	1	$4m+3n+13$

knowledge-soundness. The proof of subversion completeness is quite straightforward. But to prove Sub-ZK, as briefly mentioned at the beginning of Section 3.4.2, we needed to make the well-formedness of the CRS of the original scheme verifiable, such that we can extract the simulation trapdoor required by the simulator Sim constructed in the proof of standard ZK. To do so, we show that after adding our proposed elements to the CRS of the original scheme, we can construct an efficient extractor under a knowledge assumption that can extract the simulation trapdoors required by Sim and allows us to prove the Sub-ZK. We analyze the efficiency of the proposed SNARK in the next subsection.

3.4.3. Efficiency Evaluation

This section evaluates the performance of the proposed subversion-resistant zk-SNARK from different perspectives. To this end, we summarize the efficiency of each algorithm.

CRS Size: Tab. 1 summarizes the number of CRS elements in the three different groups. One element (namely, $[\delta]_2$) belongs both to crs_P and crs_V and thus the numbers in the "total" row are not equal to the sum of the numbers in previous rows. In Groth's zk-SNARK [79] the CRS consists of $m+2n$ elements of $\mathbb{G}_1$ and n elements of $\mathbb{G}_2$. On top of it, we added $2n+3$ group elements (highlighted with gray background in Fig. 4) to make the CRS verification possible and also some elements to speed up the prover's computation and the verifier's computation; the latter elements can alternatively be computed from the rest of the CRS.

CRS Generation and Computational Complexity: Assume that (bilinear) groups have already been computed. One can compute crs by first computing all CRS elements within brackets, and then compute their bracketed versions. One can evaluate $u_j(\chi)$, $v_j(\chi)$, and $w_j(\chi)$ for each $j \in [0\,..\,m]$ in time $\Theta(n)$ by using precomputed values $\ell_i(\chi)$ for $i \in [1\,..\,n]$ and the fact that the matrices U,V,W contain $\Theta(n)$ non-zero elements. The rest of the CRS can be computed efficiently by using straightforward algorithms.

By using Alg. 1, the whole CRS generation algorithm is dominated by $3m+3n+5$ exponentiations in $\mathbb{G}_1$, $m+7$ exponentiations in $\mathbb{G}_2$, and 1 exponentiation in $\mathbb{G}_T$ (one per CRS element) and $\Theta(n)$ multiplications/divisions in $\mathbb{Z}_p$.

CRS Verification Computational Complexity: As in [20], we assume that it is difficult to subvert the bilinear group description; this makes sense assuming that

the SNARK uses well-known bilinear groups (say, the Barreto-Naehrig curves). Consider the CRS verification algorithm in Fig. 4. It is clear that all other steps but Step 4 are efficient (computable in $\Theta(n)$ cryptographic operations); this follows from the fact that U, V, and W are sparse. Computation in those steps is dominated by $6m+5n-4m_0+8$ pairings. On top of it, one has to execute $s(U)+s(V)+s(W)$ exponentiations in $\mathbb{G}_1$, where $s(M)$ is the number of "large" (i.e., large enough so that exponentiating with them is expensive) entries in the matrix M. Often, $s(M)$ is very small.

By using Alg. 2, one can check that $[\ell_i(\chi)]_1$ has been correctly computed for all $i \in [1..n]$ in $n+1$ pairings, $n-1$ exponentiations in $\mathbb{G}_2$ and n exponentiations in $\mathbb{G}_T$. Note that, similar to [28] and `libsnark` library [32], to have an efficient implementation via suitable FFTs, we require $p-1$ should be divisible by a "large enough" power of 2. So, one needs to use a pairing-friendly elliptic curve that addresses such requirement. More discussions about finding a suitable curve can be found in section 3 of [28]. To sum up, the whole CRS verification algorithm is dominated by $6m+6n-4m_0+9$ pairings, $s(U)+s(V)+s(W)$ exponentiations in $\mathbb{G}_1$, $n-1$ exponentiations in $\mathbb{G}_2$, and n exponentiations in $\mathbb{G}_T$.

In addition, one can speed up CV by using batching [22]. Namely, clearly if $\sum_{i=1}^{s} t_i([a_i]_1 \bullet [b_i]_2) = \sum_{i=1}^{s} t_i [c]_T$ for uniformly random t_i, then w.h.p., $[a_i]_1 \bullet [b_i]_2 = [c]_T$ for each individual $i \in [1..s]$. In practice exponentiation is faster than pairing which leads to speed up in running time of CV algorithm. Moreover, one can further slightly optimize this by assuming $t_s = 1$. Full batched version of CV is described in the rest of the section, while reporting empirical analysis. As we show there, a batched CV will be dominated by $5(m+n)-4m_0+s(U)+s(V)$ (mostly, short-exponent) exponentiations in $\mathbb{G}_1$ and $m+s(W)$ (mostly, short-exponent) exponentiations in $\mathbb{G}_2$. Since exponentiation with a short exponent is significantly less costly than a pairing, this will decrease the execution time of CV significantly. More details will be shown in the second part of the section.

We note that after taking batching into account, CV will become a probabilistic algorithm, and will accept incorrect CRSs with negligible probability.

Prover, Proof Size, and Verifier: As in Groth's zk-SNARK [79], the prover's computational complexity is dominated by the need to compute $h(X)$ (3 interpolations, 1 polynomial multiplication, and 1 polynomial division; in total $\Theta(n \log n)$ non-cryptographic operations in $\mathbb{Z}_p$), followed by $(n-1)+(s(A)+1)+1+(s(A)+1)+s(A_1,\ldots,A_{m_0}) \leq n+3s(A)+2$ exponentiations in $\mathbb{G}_1$ and $s(A)+1$ exponentiations in $\mathbb{G}_2$, where $s(A)$ is the number of large elements in A (i.e., large enough so that exponentiating with them would be expensive). This means that the prover's computation is dominated by $\Theta(n \log n)$ non-cryptographic operations and $\Theta(n)$ cryptographic operations.

The verifier executes a single pairing equation that is dominated by 3 pairings and m_0 exponentiations in $\mathbb{G}_1$. The exponentiations can be done offline since they do not depend on the argument π but only on the common input $(A_1,\ldots,A_{m_0})$.

Hence, the verifier's computation is dominated by $\Theta(m_0)$ cryptographic operations but her online computation is only dominated by 3 pairings.

As the original version, the argument (proof) consists of 2 elements from $\mathbb{G}_1$ and 1 element from $\mathbb{G}_2$.

Comparison with MPC CRS Generation and a Concurrent Work: As already mentioned in Section 3.1, Ben-Sasson *et al.* [30] proposed an efficient MPC approach to generate CRS of zk-SNARKs, where both prover and verifier need to trust at least one CRS creator. We emphasize that [21] and the current paper study the scenario where you can trust no one. In such a case, the approach of [30] still works but it is not efficient. For example, the computational cost of CV in the new SNARK is very small compared to the cost of the joint CRS creation and verification protocol in [30]. Nevertheless, while the starting point of their approach is different, it actually resulted in a somewhat similar solution. A longer comparison will be provided in the next section with some concrete numbers.

Independently, Fuchsbauer [60] also constructed subversion-resistant SNARKs. His general approach is similar, but he modifies subtly the simulator of Groth's SNARK so that it does not require the knowledge of α and β, and then shows that Groth's original SNARK (without adding an extra element to the CRS) is knowledge-sound and Sub ZK.

The importance of subversion-resistant SNARKs was also recognized in [41, 43] and [43] gave a construction which they claimed achieves Sub-ZK. However, in [61], Fuchsbauer showed that their approach was flawed.

3.5. Batching CV Algorithm and an Implementation

3.5.1. Batching

We use batching techniques from [22] to make the CV algorithm more efficient, see Fig. 6 for implementation details. Especially, 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 with coefficients $X_1, \ldots, X_s$, $t_i \leftarrow_r \{1, \ldots, 2^\kappa\}$ for $i < s$, then $X_i = 0$ for each i, with probability $1 - 1/2^\lambda$. We also employ the following lemma (proven in [57]) to show that randomness generated in batching can be used multiple times.

Lemma 1. *Assume $1 < t < q$. Assume $\vec{t}$ is a vector chosen uniformly random from $[1..t]^{k-1} \times \{1\}$, $\vec{\chi}$ is a vector of integers in $\mathbb{Z}_q$, and f_i are some polynomials of degree* $\mathsf{poly}(\kappa)$. *If $\sum_{i=1}^{k} f_i(\vec{\chi}) t_i \cdot ([1]_1 \bullet [1]_2) = [0]_T$, then with probability $\geq 1 - \frac{1}{t}$, $f_i(\vec{\chi})([1]_1 \bullet [1]_2) = [0]_T$ for each $i \in [1..k]$.*

In the batched CV algorithm we execute Alg. 3 instead of Alg. 2. This algorithm needs 3 pairings, $2n$ exponentiations in $\mathbb{G}_1$, and 1 exponentiation in $\mathbb{G}_T$, and on top of it, 16 pairings, $5m + 5n - 4m_0 + s(U) + s(V) - 12$ (mostly, short-exponent) exponentiations in $\mathbb{G}_1$, and $m + s(W) + 2$ (mostly, short-exponent) ex-

ponentiations in $\mathbb{G}_2$. This makes, in total, 19 pairings, $5m+7n-4m_0+s(U)+s(V)-12$ (mostly, short-exponent) exponentiations in $\mathbb{G}_1$, $m+s(W)+2$ (mostly, short-exponent) exponentiations in $\mathbb{G}_2$, and 1 exponentiation in $\mathbb{G}_T$.

3.5.2. Implementation Results

We compare the efficiency of the new subversion-resistant zk-SNARK with Groth's (non-subversion) zk-SNARK from both the theoretical (as reported in [79]) and the empirical (as implemented in the `libsnark` [32] library) point of view. Similarly to the pre-existing implementation of Groth's zk-SNARK in `libsnark`, we implemented the new SNARK in the C++ language by using low-level subroutines of `libsnark`. All the following results were measured on a 64-bit `Linux Ubuntu 16.10` virtual machine with 8 GB RAM and a single core. The virtual machine was installed on a standard laptop (HP EliteBook 840), with the Intel Core i5-5200u 2.2 GHz CPU and 16 GB RAM. We built the `libsnark` library by using the option `CURVE=BN128` that provides an instantiation based on a Barreto-Naehrig curve at 128 (or 100 bits according to [17]) bits of security.

Table 2 compares the implementation of Groth's zk-SNARK in `libsnark` with our implementation of the new subversion-resistant zk-SNARK for several choices of m_0 and n. We report several measures including the running time of $\mathsf{KGen}_{\mathsf{NIZK}}$, CV, P, and V. All times are expressed in seconds. Fig. 7 compares the running time of (batched) CV with the running time of the prover.

In the case of the new Sub-ZK SNARK, we evaluated the performance of the CV algorithm by using both the non-batched (Fig. 4) and batched (Fig. 6) versions. In the execution of the batched CV, we first sample a vector $\vec{t}$ of random numbers from $[1\,..\,2^{80}]$. This vector has length m, since no verification equation needs more than m random values. As stated in Section 3.5.1, we reuse randomness, i.e., in every verification equation we use random values from the same $\vec{t}$. We computed the running times as averages over 10 iterations. We emphasize that while the non-batched CV is very slow (this is why we are not giving its timings for $n > 30000$), the batched CV is faster than P.

As mentioned, the randomness vector $\vec{t}$ takes inputs from $[1\,..\,2^{80}]$ which assures that the batching causes a security gap not bigger than 2^{-80}. This is a con-

Algorithm 3: Checking that $\left[(\ell_j(\chi))_{j=1}^n\right]_1$ is correctly computed: batched version

Input: $(\left[\chi^{n-1},(\ell_j(\chi))_{j=1}^n\right]_1,[1,\chi]_2,[1]_T)$

$\omega_0 \leftarrow 1/\omega$;
$[a]_1 \leftarrow [0]_1$; $[b]_1 \leftarrow [0]_1$; $c \leftarrow 0$;
for $j=1$ **to** n **do**
  $t_j \leftarrow_r \{1,\ldots,2^\kappa\}$; $\omega_j \leftarrow \omega\omega_{j-1}$;
  $[a]_1 \leftarrow [a]_1 + t_j[\ell_j(\chi)]_1$; $[b]_1 \leftarrow [b]_1 + t_j\omega_j[\ell_j(\chi)]_1$; $c \leftarrow c + t_j\omega_j$;
Check that $[a]_1 \bullet [\chi]_2 - [b]_1 \bullet [1]_2 = c/n \cdot (\left[\chi^{n-1}\right]_1 \bullet [\chi]_2 - [1]_T)$;

$\mathsf{CV}(\mathbf{R}, \mathsf{crs})$: // batched CV

1. For $\iota \in \{\gamma, \delta\}$: check that $[\iota]_1 \neq [0]_1$
2. To check $\{\alpha, \beta, \gamma, \delta\}$:
   (a) Generate $t_i \leftarrow_r \{1, \ldots, 2^\kappa\}$ for $i = 1 \ldots 3$, then set $t_4 \leftarrow 1$.
   (b) Check that $[t_1\alpha + t_2\beta + t_3\gamma + \delta]_1 \bullet [1]_2 = [1]_1 \bullet [t_1\alpha + t_2\beta + t_3\gamma + \delta]_2$,
3. To check the terms $\left[\chi^i\right]_1$ for $i = 1$ to $n-1$:
   (a) Generate $t_i \leftarrow_r \{1, \ldots, 2^\kappa\}$ for $i = 1$ to $n-2$, then set $t_{n-1} \leftarrow 1$.
   (b) Check that $(\sum_{i=1}^{n-1} t_i \left[\chi^i\right]_1) \bullet [1]_2 = (\sum_{i=1}^{n-1} t_i \left[\chi^{i-1}\right]_1) \bullet [\chi]_2$
4. Check that $\left[(\ell_j(\chi))_{j=1}^n\right]_1$ is correctly computed by using Alg. 3,
5. For $i = 0$ to m:
   (a) Check that $[u_i(\chi)]_1 = \sum_{j=1}^n U_{ij} [\ell_j(\chi)]_1$,
   (b) Check that $[v_i(\chi)]_1 = \sum_{j=1}^n V_{ij} [\ell_j(\chi)]_1$,
   (c) Set $[w_i(\chi)]_1 \leftarrow \sum_{j=1}^n W_{ij} [\ell_j(\chi)]_1$,
6. To check $[v_i(\chi)]_1$, for $i = 0$ to m:
   (a) Generate $t_k \leftarrow_r \{1, \ldots, 2^\kappa\}$ for $k = 0$ to $m-1$, then set $t_m \leftarrow 1$.
   (b) Check that $(\sum_{k=0}^m t_k [v_k(\chi)]_1) \bullet [1]_2 = [1]_1 \bullet (\sum_{k=0}^m t_k [v_k(\chi)]_2)$,
7. To check $[(u_i(\chi)\beta + v_i(\chi)\alpha + w_i(\chi))/\delta]_1$, for $i = m_0 + 1$ to m:
   (a) Generate $t_i \leftarrow_r \{1, \ldots, 2^\kappa\}$ for $i = m_0 + 1$ to $m-1$, then set $t_m \leftarrow 1$.
   (b) Check that $(\sum_{i=m_0+1}^m t_i [(u_i(\chi)\beta + v_i(\chi)\alpha + w_i(\chi))/\delta]_1) \bullet [\delta]_2 = (\sum_{i=m_0+1}^m t_i [u_i(\chi)]_1) \bullet [\beta]_2 + (\sum_{i=m_0+1}^m t_i [v_i(\chi)]_1) \bullet [\alpha]_2 + (\sum_{i=m_0+1}^m t_i [w_i(\chi)]_1) \bullet [1]_2$.
8. Check that $\left[\chi^{n-1}\right]_1 \bullet [1]_2 = [1]_1 \bullet \left[\chi^{n-1}\right]_2$.
9. To check $\left[\chi^i \ell(\chi)/\delta\right]_1$, for $i = 0$ to $n-2$:
   (a) Generate $t_i \leftarrow_r \{1, \ldots, 2^\kappa\}$ for $i = 0$ to $n-3$, then set $t_{n-2} \leftarrow 1$.
   (b) Check that $(\sum_{i=0}^{n-2} t_i \left[\chi^i \ell(\chi)/\delta\right]_1) \bullet [\delta]_2 = (\sum_{i=0}^{n-2} t_i \left[\chi^{i+1}\right]_1) \bullet \left[\chi^{n-1}\right]_2 - (\sum_{i=0}^{n-2} t_i \left[\chi^i\right]_1) \bullet [1]_2$,
   (c) Check that $[\alpha]_1 \bullet [\beta]_2 = [\alpha\beta]_T$.

Figure 6. A batched version of CV algorithm

servative approach. If the coordinates of $\vec{t}$ were chosen from $[1 \,..\, 2^{40}]$ then the CV algorithm would take 0.70, 1.25 and 2.44 seconds for (n, m_0) equal to $(7500, 100)$, $(15000, 100)$, $(30000, 100)$, respectively. These times are about 40% shorter than in the case $\vec{t} \leftarrow_r [1 \,..\, 2^{80}]$.

Tab. 3 summarizes the number of CRS elements in Groth's zk-SNARK and our new subversion zk-SNARK based on values given in [79] and `libsnark` implementation. (Note the slight difference between CRS in the original Groth paper and the implemented version).

Table 2. Performance of the implementations of Groth's non-subversion zk-SNARK and the new Sub-ZK SNARK in `libsnark` for different values of n and m_0. All running times are reported in seconds. Similarly to the pre-existing implementation of Groth's zk-SNARK in `libsnark`, we implemented the new SNARK in the C++ language by using low-level subroutines of `libsnark`. All the results were measured on a 64-bit `Linux Ubuntu 16.10` virtual machine with 8 GB RAM and a single core. The virtual machine was installed on a standard laptop (HP EliteBook 840), with the Intel Core i5-5200u 2.2 GHz CPU and 16 GB RAM.

Protocol	n, m_0	$\mathsf{KGen}_{\mathsf{NIZK}}$	CV	P	V
Groth's zk-SNARK [79]	7500, 100	1.31	—	1.29	0.005
Sub-ZK SNARK	7500, 100	1.74	156.2	1.29	0.005
Sub-ZK SNARK batched	7500, 100	1.74	1.14	1.29	0.005
Groth's zk-SNARK [79]	15000, 100	2.42	—	2.42	0.005
Sub-ZK SNARK	15000, 100	3.06	310.2	2.42	0.005
Sub-ZK SNARK batched	15000, 100	3.06	2.18	2.42	0.005
Groth's zk-SNARK [79]	30000, 100	4.45	—	4.60	0.005
Sub-ZK SNARK	30000, 100	5.72	623.4	4.60	0.005
Sub-ZK SNARK batched	30000, 100	5.72	3.89	4.60	0.005
Groth's zk-SNARK [79]	30000, 1000	4.47	—	4.60	0.035
Sub-ZK SNARK	30000, 1000	5.82	619.8	4.60	0.035
Sub-ZK SNARK batched	30000, 1000	5.82	3.89	4.60	0.035
Groth's zk-SNARK [79]	60000, 1000	8.39	—	8.88	0.035
Sub-ZK SNARK batched	60000, 1000	11.0	7.36	8.88	0.035
Groth's zk-SNARK [79]	120000, 1000	14.8	—	17.07	0.035
Sub-ZK SNARK batched	120000, 1000	18.9	14.04	17.07	0.035
Groth's zk-SNARK [79]	250000, 1000	28.1	—	33.96	0.035
Sub-ZK SNARK batched	250000, 1000	36.1	27.31	33.96	0.035
Groth's zk-SNARK [79]	500000, 1000	53.0	—	66.21	0.035
Sub-ZK SNARK batched	500000, 1000	68.1	53.48	66.21	0.035

Table 3. A comparison of number of crs elements in Groth's non-subversion zk-SNARK and new subversion-resistant zk-SNARK

	$\mathsf{crs}_{\mathsf{P}}$		$\mathsf{crs}_{\mathsf{V}}$			$\mathsf{crs}_{\mathsf{CV}}$		
SNARKs	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	$\mathbb{G}_1$	$\mathbb{G}_2$	Σ
Groth'16	$m+2n-m_0+2$	$n+1$	m_0+1	2	1	0	0	$m+3n+6$
Σ	$m+3n-m_0+3$		m_0+3			0		
Groth'16 imp.	$3m+n-m_0+4$	$m+1$	m_0	2	1	0	0	$4m+n+8$
Σ	$4m+n-m_0+5$		m_0+3			0		
This work	$3m+n-m_0+4$	$m+3$	m_0+1	2	1	$2n$	3	$4m+3n+14$
Σ	$4m+n-m_0+7$		m_0+4			$2n+3$		
This work imp.	$3m+n-m_0+4$	$m+3$	m_0+1	2	1	$2n+2$	3	
Σ	$4m+n-m_0+7$		m_0+4			$2n+5$		$4m+3n+16$

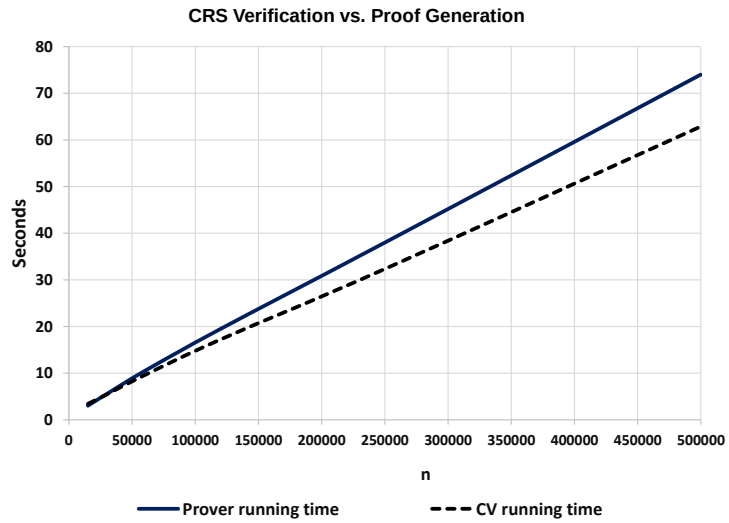

Figure 7. Efficiency of the *batched* CV algorithm in comparison with efficiency of the prover P.

The algorithm presented in Fig. 6 can be optimized even more. More precisely, the verification equations in Steps 2, 6, and 8 can be merged into a single equation. Such an operation adds 1 exponentiation in $\mathbb{G}_1$ and 1 exponentiation in $\mathbb{G}_2$ and needs 4 pairings less. Similarly, the verification equation $[\alpha]_1 \bullet [\beta]_2 = [\alpha\beta]_T$ in Step 9.c of the batched CV can be done inside Step 7.b of the algorithm, which will drop one more pairing. Thus, we need 5 pairings less and 1 extra exponentiation in $\mathbb{G}_1$ and 1 extra exponentiation in $\mathbb{G}_2$. The batched CV can be executed in total with 14 pairings, $5m+7n-4m_0+s(U)+s(V)-11$ (mostly, short-exponent) exponentiations in $\mathbb{G}_1$, $m+s(W)+3$ (mostly, short-exponent) exponentiations in $\mathbb{G}_2$, and 1 exponentiation in $\mathbb{G}_T$.

4. SUBVERSION-RESISTANT SIMULATION KNOWLEDGE SOUND SNARKS

4.1. Motivation

In Section 3, we defined subversion security in zk-SNARKs and presented a variation of the state-of-the-art zk-SNARK, proposed by Groth [79] which achieves Sub-ZK and knowledge-soundness. From the trust point of view, this means that the prover does not need to trust CRS generators to achieve zero-knowledge, but the verifier achieves knowledge-soundness assuming a trusted CRS generator. However, in practice a knowledge-sound proof is vulnerable to man-in-the-middle attacks [1]. Indeed, knowledge-soundness only guarantees that a successful prover knows the witness, and it does not guarantee that the proofs are non-malleable. Due to this fact, zk-SNARKs that only guarantee knowledge-soundness cannot be deployed in many practical applications straightforwardly [27, 90, 93]. Privacy-preserving cryptocurrencies such as Zerocash that use zk-SNARKs [31, 79] as a subroutine, takes extra steps to prevent malleability attacks in the SNARK proofs for *pour* transactions [27]. Similarly, smart contract systems like Hawk [90, 93] showed that knowledge-soundness of zk-SNARKs is not enough in their system.

Simulation knowledge-soundness (defined in Def. 13) which is also known as *simulation extractability*, is an amplified version of knowledge-soundness that also guarantees non-malleability of proofs. Technically speaking, a Simulation-Extractable (SE) zk-SNARK guarantees that the proof is succinct, zero-knowledge and *simulation-extractable*. Simulation extractability implies that an adversary cannot generate a new proof unless he knows a witness, even if he has seen an arbitrary number of simulated proofs.

So, by considering the importance of simulation-extractability and subversion-security in NIZKs, constructing a NIZK argument (particularly a zk-SNARK) that requires less trust but achieves stronger security notions is an interesting research question in cryptography. Constructing such zk-SNARKs that achieve Sub-ZK and simulation extractability can straightforwardly mitigate the trust and improve the security in practical applications that use zk-SNARKs as a building block in their systems. From a different perspective, as already summarized in Section 3.3, the best positive result of Bellare et al. [20] showed that in the face of subverted parameters one can construct NIZK arguments that can achieve Sub-ZK and (knowledge) soundness. So, an interesting research question can be if one can improve their best positive result.

[1] For instance, in the verification equations that include pairings such as $a \bullet b = c$, where a and b are proof elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ with prime orders, one can see that such a verification equation will also be satisfied for new proof elements such as $a' = a^r$ and $b' = b^{1/r}$, for arbitrary $r \leftarrow \mathbb{Z}_p$.

4.2. Problem Statement

By considering the above discussions, in this section, we chase one main research question which can also answer some other practically interesting research questions. Our main question is "Can one construct zk-SNARKs (or NIZKs) that can achieve Sub-ZK and *simulation* knowledge-soundness at the same time?" As the notion Sub-ZK implies standard ZK, so answering the above question can also answer the question "How can one efficiently achieve simulation-extractability in zk-SNARKs?".

Constructing such zk-SNARKs can be interesting from various perspectives. From one point of view, it will be an improved version of the construction presented in Section 3, as the prover again will achieve Sub-ZK, but the verifier will achieve simulation-extractability which is a stronger notion in comparison with knowledge-soundness that is achieved in the previous case. From a different perspective, this will amplify the best positive result about constructing subversion-resistant NIZKs, shown by Bellare, Fuchsbauer, and Scafuro [20].

4.3. Simulation-Extractable zk-SNARKs

In this subsection, we show how one can efficiently achieve simulation-extractability in zk-SNARKs that would guarantee the non-malleability of proofs. This would positively answer the second research question of this section.

A full detailed description of the result can be found in the paper [6] which is joint work with Atapoor.

4.3.1. Previous Results

In Crypto 2017, Groth and Maller [82] proposed the first zk-SNARK in the CRS model that can achieve nBB simulation-extractability (defined in Def. 13). They proved that a Simulation-Extractable (SE) zk-SNARK requires at least two verification equations. Their scheme is constructed in the bilinear groups for SAPs and achieves the lower bound in the number of verification equations. To verify a proof, V needs to check two equations that are dominated by 5 pairings [82]. To achieve simulation (knowledge) soundness, their scheme removes one of the bilinear group generators from the CRS, which might create some different challenges in some practical cases (e.g., in CRS generation by multi-party computation protocols [29], or in achieving subversion security as in Section 3. Above all, GM zk-SNARK is constructed for SAPs that require twice the number of multiplication (MUL) gates, as $ab = ((a+b)^2 - (a-b)^2)/4$ [79]. Implementations also prove that for a particular arithmetic circuit, QAP-based zk-SNARKs, namely Groth's zk-SNARK [79] has considerably better efficiency than Groth-Maller's zk-SNARK [82], but Groth's scheme does not achieve simulation extractability, which makes its proofs vulnerable to the malleability attacks. For a Rank-1 Con-

Table 4. Performance of zk-SNARKs proposed by Groth and Maller [82] and Groth [79] for arithmetic circuit satisfiability with an R1CS instance with 10^6 constraints and 10^6 variables, where 10 are input variables. SE: Simulation Extractable, KS: Knowledge Soundness, MB: Megabyte, s: seconds, ms: milliseconds, P: paring.

SNARK	CRS size, time	Proof , P's time	Verify, V's time	Notion
GM [82]	376 MB, 103 s	$2\mathbb{G}_1 + 1\mathbb{G}_2$, 120 s	5 P, 2.3 ms	SE
Groth [79]	196 MB, 75 s	$2\mathbb{G}_1 + 1\mathbb{G}_2$, 83 s	3 P, 1.4 ms	KS

straint System (R1CS) instance [2], efficiency metrics of both schemes are compared in Tab. 4.

4.3.2. Our Solution and Technique

Next, we present an efficient version of Groth's zk-SNARK that can achieve nBB simulation-extractability and outperforms Groth and Maller's zk-SNARK [82] considerably. To this end, we use folklore OR technique [24] with a particular instantiation from C∅C∅ framework [94] (reviewed in Section 2.4.5). Using the OR technique we define a new language $\mathscr{L}''$ based on the language $\mathscr{L}$ in Groth's zk-SNARK (more generally input NIZK) along with a PRF and perfectly binding commitment that commits to secret randomness as a key for the PRF [24, 54, 94].

Let $\Pi_{\mathsf{Sig}} = (\mathsf{KGen}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf})$ be a one-time signature scheme and $\Pi_{\mathsf{Com}} = (\mathsf{KGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Ver})$ be a perfectly binding commitment scheme.

Given the language $\mathscr{L}$ with the corresponding NP relation $\mathbf{R}_{\mathscr{L}}$, we define a new language $\mathscr{L}''$ such that $((\mathsf{x}, \mu, \mathsf{vk}_{\mathsf{Sig}}, \rho), (\mathsf{w}, s, r)) \in \mathbf{R}_{\mathscr{L}''}$ iff:

$$\big((\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathscr{L}} \vee (\mu = f_s(\mathsf{vk}_{\mathsf{Sig}}) \wedge \rho = \mathsf{Com}(s, r))\big),$$

where $\{f_s : \{0,1\}^* \to \{0,1\}^\lambda\}_{s \in \{0,1\}^\lambda}$ is a pseudo-random function family. The intuition for a pseudo-random function $f_s(\cdot)$ is that without the knowledge of the key s, $f_s(\cdot)$ behaves like a true random function. However, given s, one can compute $f_s(\cdot)$ easily. In the new language $\mathscr{L}''$, for a statement-witness pair to be valid, either a witness for $\mathbf{R}_{\mathscr{L}}$ is provided (by honest prover) or an opening to ρ together with the value of $\mu = f_s(\mathsf{vk}_{\mathsf{Sig}})$ is provided (by simulator), where s is the open value of ρ (in CRS). One may note that in order for a statement to pass the verification without a valid witness, the prover must generate $f_s(\mathsf{vk}_{\mathsf{Sig}})$ without the knowledge of s (thus breaking the pseudo-random function f_s). The $\mathsf{vk}_{\mathsf{Sig}}$ is a verification key of a one-time secure signature scheme that each time the prover samples and uses its associated signing key to sign the proof. By considering new language $\mathscr{L}''$, zk-SNARK of Groth for the relation $\mathbf{R}$ constructed from PPT algorithms $\Pi_{\mathsf{NIZK}} = (\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ can be lifted to a nBB simulation-extractable zk-SNARK Π'_{NIZK} with PPT algorithms $(\mathsf{KGen}'_{\mathsf{NIZK}}, \mathsf{P}', \mathsf{V}', \mathsf{Sim}')$ as

[2] A Rank One Constraint System (R1CS) is a way to express a computation that makes it amenable to zero knowledge proofs. An R1CS is a sequence of groups of three vectors $(\vec{a}, \vec{b}, \vec{c})$, and the solution to an R1CS is a vector $\vec{s}$, where $\vec{s}$ must satisfy the equation $\vec{s} \cdot \vec{a} * \vec{s} \cdot \vec{b} - \vec{s} \cdot \vec{c} = 0$, where $\cdot$ represents the dot product. More details on `https://github.com/scipr-lab/libsnark`.

CRS generator $\mathsf{KGen}'_{\mathsf{NIZK}}(\mathbf{R}_{\mathscr{L}})$**:**

- Sample $(\mathsf{crs} \parallel \mathsf{ts}) \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(\mathbf{R}_{\mathscr{L}''})$;
- $s, r \leftarrow \{0,1\}^{\lambda}$; $\rho := \mathsf{Com}(s, r)$;

and output $(\mathsf{crs}' \parallel \mathsf{tc}') := ((\mathsf{crs}, \rho) \parallel (\mathsf{ts}, (s, r)))$; where tc' is new CRS trapdoors.

Prover $\mathsf{P}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}', \mathsf{x}, \mathsf{w})$**:** Parse $\mathsf{crs}' := (\mathsf{crs}, \rho)$; abort if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathscr{L}}$;

- generate $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{vk}_{\mathsf{Sig}}) \leftarrow \mathsf{KGen}_{\mathsf{Sig}}(1^{\lambda})$;
- sample $z_0, z_1, z_2 \leftarrow \{0,1\}^{\lambda}$; generate $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, z_0, \mathsf{vk}_{\mathsf{Sig}}, \rho), (\mathsf{w}, z_1, z_2))$ using the prover of Groth's scheme;
- sign $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_{\mathsf{Sig}}, (\mathsf{x}, z_0, \pi))$;

and return $\pi' := (z_0, \pi, \mathsf{vk}_{\mathsf{Sig}}, \sigma)$.

Verifier $\mathsf{V}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}', \mathsf{x}, \pi')$**:** Parse $\mathsf{crs}' := (\mathsf{crs}, \rho)$ and $\pi' := (z_0, \pi, \mathsf{vk}_{\mathsf{Sig}}, \sigma)$;

- abort if $\mathsf{Vf}(\mathsf{vk}_{\mathsf{Sig}}, (\mathsf{x}, z_0, \pi), \sigma) = 0$;
- call the verifier of Groth's scheme $\mathsf{V}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, z_0, \mathsf{vk}_{\mathsf{Sig}}, \rho), \pi)$ and abort if it outputs 0.

Simulator $\mathsf{Sim}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}', \mathsf{tc}', \mathsf{x})$**:** Parse $\mathsf{crs}' := (\mathsf{crs}, \rho)$ and $\mathsf{tc}' := (\mathsf{ts}, (s, r))$ and using simulation trapdoor ts acts as follows:

- generate $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{vk}_{\mathsf{Sig}}) \leftarrow \mathsf{KGen}_{\mathsf{Sig}}(1^{\lambda})$;
- set $\mu = f_s(\mathsf{vk}_{\mathsf{Sig}})$;
- generate $\pi \leftarrow \mathsf{Sim}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, \mu, \mathsf{vk}_{\mathsf{Sig}}, \rho), \mathsf{ts})$;
- sign $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_{\mathsf{Sig}}, (\mathsf{x}, \mu, \pi))$;

and output $\pi' := (\mu, \pi, \mathsf{vk}_{\mathsf{Sig}}, \sigma)$.

Figure 8. The construction of lifted SNARK (more generally NIZK) that can achieve ZK and simulation-extractability.

described in Fig. 8. To simplify the description, we assume Com takes exactly λ random bits as randomness and that the witness for original language $\mathscr{L}$ is exactly λ bits; it is straight forward to adapt the proof when they are of different lengths [94]. Note that in the simulation-extractable zk-SNARK Π'_{NIZK}, the algorithms of the original scheme will be executed for a new arithmetic circuit which encodes new language $\mathscr{L}''$ and has slightly larger number of gates. Namely, the CRS generation algorithm of Groth's zk-SNARK will be executed with a new QAP instance that has larger parameters; $(\mathsf{crs} \parallel \mathsf{ts}) \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(\mathbf{R}_{\mathscr{L}''})$. Similarly the prover of the new variation will execute the prover of Groth's zk-SNARK with a new arithmetic circuit that has a larger number of gates; namely $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, z_0, \mathsf{vk}_{\mathsf{Sig}}, \rho), (\mathsf{w}, z_1, z_2))$, where z_1 and z_2 play the role of witnesses s and r for prover.

Note that, in the simulation of NIZK argument Π'_{NIZK}, one could also use the regular proof generation algorithm using the witness of the other OR branch,

as $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, \mu, \mathsf{vk}_{\mathsf{Sig}}, \rho), (s, r))$. However, here we use the simulation algorithm Sim of the input NIZK argument Π_{NIZK} along with the simulation trapdoor ts. In the second part of the section, we will use a similar simulation procedure along with a new technique to achieve Sub-ZK in the proposed construction.

In the paper [6], we prove the following theorem that results the completeness, ZK, and simulation-extractability of the presented construction.

Theorem 2 (Completness, ZK, nBB Simulation Extractability)**.** *If the NIZK argument Π_{NIZK} guarantees completeness, ZK, and (knowledge) soundness, the pseudo-random function family is secure, and the one-time signature scheme is secure, then the proposed construction in Fig. 8 achieves completeness, ZK, non-black-box simulation extractability.*

Proof. The proof is provided in [6] that is included in the thesis. □

4.3.3. An Efficient Instantiation

In the last subsection, we observed that the proposed construction (described in Fig. 8) uses a PRF, a commitment scheme, and a one-time secure signature scheme. In the rest, we discuss how the used primitives can be instantiated.

In similar practical cases, both pseudo-random function and commitment scheme are instantiated using an efficient SHA-256 circuit that has around $\approx 25 \times 10^3$ MUL gates for one block (512-bit input) [27, 93][3]. About the signature scheme, as Groth's zk-SNARK is paring-based and is constructed with bilinear groups, so we instantiate the signature scheme with Boneh and Boyen's signature [38] which works in bilinear groups and has very efficient verification; it requires only one pairing and one multi-exponentiation. Their signature scheme is proven to guarantee strong unforgeability under chosen message attack under the strong Diffie-Hellman assumption. Consequently, it satisfy unforgeability under strong *one-time* chosen message attack. The key generation, signing, and verification of Boneh and Boyen's signature scheme [38] is summarized below.

- **Key Generation**, $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{vk}_{\mathsf{Sig}}) \leftarrow \mathsf{KGen}_{\mathsf{Sig}}(1^{\lambda})$: Given system parameters for a prime-order bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, randomly selects $x, y \leftarrow_{\$} \mathbb{Z}_p^*$, and computes $u := x \cdot [1]_1$ and $v := y \cdot [1]_1$ and returns $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{vk}_{\mathsf{Sig}}) := ((x, y) \parallel (u, v, [1]_1, [1]_2, [1]_T))$.
- **Signing**, $([\sigma]_2, r) \leftarrow \mathsf{Sig}(\mathsf{sk}_{\mathsf{Sig}}, m)$: Given system parameters, a secret key $\mathsf{sk}_{\mathsf{Sig}} := (x, y)$, and a message $m \in \mathbb{Z}_p^*$, it first samples $r \leftarrow_{\$} \mathbb{Z}_p^*$ and then computes $[\sigma]_2 = [1/(x + m + yr)]_2$ and returns $([\sigma]_2, r)$ as the signature.
- **Verification**, $\{1, 0\} \leftarrow \mathsf{Vf}(\mathsf{vk}_{\mathsf{Sig}}, m, ([\sigma]_2, r))$: Given a verification key $\mathsf{vk}_{\mathsf{Sig}} := ([x]_1, [y]_1, [1]_1, [1]_2, [1]_T)$, a message m, and a signature $([\sigma]_2, r)$, verifies if $[x + m + yr]_1 \bullet [1/(x + m + yr)]_2 = [1]_T$; if so, it returns 1; otherwise it returns 0,

[3] It has 25.538 gates in the `xjsnark` library, `https://github.com/akosba/xjsnark`.

where • denotes the paring operation. In our case, we use the same bilinear group as in the original zk-SNARK and m would be the hash (e.g., with SHA224 or SHA256) of concatenations of the proof elements with the statement, i.e. $m := H(\mathsf{x}\|z_0\|\pi)$[4]. As can be seen, the scheme generates two elements signature from $\mathbb{G}_2$ and Z_p, its verification key consists of two elements from $\mathbb{G}_1$, and above all its verification only requires one paring; as $[1]_T$ can be preprocessed and shared in the CRS.

So by considering the above instantiation, in the new argument $\pi' = (\mu, \pi, \mathsf{vk}_{\mathsf{Sig}}, [\sigma]_1, r)$, where from the original scheme $\pi = ([a]_1, [b]_2, [c]_1)$, and μ is an output of the PRF $f_s(\cdot)$, which is instantiated with SHA-256 hash function [93]. As a result, the argument in new scheme will consists of 4 elements from $\mathbb{G}_1$, 2 elements from $\mathbb{G}_2$ and two 256-bit strings. Consequently, new changes add only one paring to the verification of the original scheme. To the best of our knowledge, this is the first simulation-extractable zk-SNARK whose verification is dominated with 4 pairings.

In an independent work, we show that the weak version of Boneh and Boyen's signature scheme [38] using a random oracle can satisfy unforgeability under strong *one-time* chosen message attack which allows to be used as an instantiation for our constructions. Namely, with instantiating the signature scheme with the weak version but using a RO that returns field elements, the proof will be 3 elements from $\mathbb{G}_1$, 2 elements from $\mathbb{G}_2$ and one 256-bit string.

4.3.4. Efficiency Evaluation and an Implementation

Next, we analyze the empirical performance of the proposed zk-SNARK from different perspectives. Tab. 5 summarizes asymptotic and empirical performance of the proposed scheme (in Section 4.3.2 and 4.3.3) and two zk-SNARKs proposed by Groth [79] and GM [82]. Implementations of Groth's [79] and GM [82] zk-SNARKs are available in `libsnark` library [31] [5], so implementation of the new scheme is done in the same library.

In Tab. 5, all empirical experiences are reported for a particular R1CS instance. In the rest of the analysis, we evaluate the empirical performance of the proposed scheme for different R1CS instances and compare with the *knowledge-sound* scheme of Groth [79] and simulation-extractable scheme of GM [82]. As the first efficiency metric, in Fig. 9 and Fig. 10, respectively, we plot CRS size and CRS generation time of the proposed scheme for different R1CS instances with 100 input variables and various number of constraints between 25×10^3 and 2×10^6 gates.

[4] As shown in [38], by taking hash of input message the signature scheme can be used to sign arbitrary messages in $\{0,1\}^*$.To do so, a collision-resistant hash function $H : \{0,1\}^* \to \{0,\ldots,2^b\}$ such that $2^b < p$ is sufficient [38]. By considering recent analysis on Barreto-Naehrig curves by Kim and Barbulescu [18], one can use different settings for various security levels which would need to use different hash functions for signing arbitrary messages in [38] signature scheme.

[5] Available on `https://github.com/scipr-lab/libsnark`

Table 5. An efficiency comparison of the proposed scheme (in Section 4.3.2 and 4.3.3) with Groth's [79] and GM [82] zk-SNARKs for arithmetic circuit satisfiability with m_0 elements instance, m wires, n MUL gates. In GM zk-SNARK [82], n MUL gates translate to $2n$ squaring gates. Implementations (Implem.) are done on a Laptop with 2.50 GHz Intel Core i5-7200U CPU, with 16GB RAM, in single-threaded mode, for an R1CS instance with $n = 10^6$ constraints and $m = 10^6$ variables, of which $m_0 = 10$ are input variables. $\mathbb{G}_1$ and $\mathbb{G}_2$: group elements, E: exponentiations, P: pairings, BS: λ-Bit String, B: Byte, MB: Megabyte, s: seconds, ms: milliseconds, SE: Simulation-Extractable, KS: Knowledge Soundness. In the new scheme, the statement contains $(\mathsf{x}, \mu, \mathsf{vk}_{\mathsf{Sig}}, \rho)$ which has 3 new elements $(\mu, \mathsf{vk}_{\mathsf{Sig}}, \rho)$, so $m'_0 = m_0 + 4$. All asymptotic analysis of new scheme are done based on our particular instantiation of the commitment scheme and PRF. So, as new changes add $\approx 50 \times 10^3$ MUL gates to n and m, so $n' = n + 50.000$ and $m' = m + 50.000$.

SNARK	CRS size & time	Proof	P & P$'s$ time	V	Sec.
GM [82] & Implem.	$m+4n$ $\mathbb{G}_1$ $2n$ $\mathbb{G}_2$	2 $\mathbb{G}_1$ 1 $\mathbb{G}_2$	$m+4n-m_0$ E_1 $2n$ E_2	m_0 E_1 5 P	SE
	376 MB, 103 s	127 B	120 s	2.3 ms	
Groth [79] & Implem.	$m+2n-m_0$ $\mathbb{G}_1$ n $\mathbb{G}_2$	2 $\mathbb{G}_1$ 1 $\mathbb{G}_2$	$m+3n-m_0$ E_1 n E_2	m_0 E_1 3 P	KS
	196 MB, 75 s	127 B	83 s	1.4 ms	
Sec.4.3.3 & Implem.	$m'+2n'-m'_0$ $\mathbb{G}_1$ n' $\mathbb{G}_2$	4 $\mathbb{G}_1$ 2 $\mathbb{G}_2$ 2 BS	$m'+3n'-m'_0$ E_1 n' E_2	m'_0 E_1 4 P	SE
	205 MB, 80.5 s	318 B	90.1 s	2.0 ms	

Similarly, in Fig. 11, we plot prover's running time in two mentioned zk-SNARKs and the proposed one for various R1CS instances with 100 input variables and the different number of constraints.

Finally, the plot in Fig. 12 compares the verification time of new SE zk-SNARK with two mentioned ones for various R1CS instances with 10^5 constraints and the various number of input variables.

To sum up, from the comparisons in Tab. 5 and empirical analysis in Fig. 9-12, one can observe that in order to give non-malleable proofs for an arithmetic circuit satisfiability in circuits with larger than 50×10^3 multiplication gates, the proposed SE zk-SNARK (described in Section 4.3.2 and 4.3.3) can outperform GM SE zk-SNARK considerably. It is worth to mention that, the new construction generates larger proofs in comparison with GM zk-SNARK, 318 bytes in comparison with 127 bytes, but as shown in Fig. 12, its verification still requires fewer pairings, and in the worst cases it is as efficient as verification of GM SE zk-SNARK [82].

4.4. Subversion-Resistant Simulation-Extractable SNARKs

In this rest of the section, we answer the main research question of the section (discussed in Section 4.2) by constructing zk-SNARKs that can achieve Sub-ZK

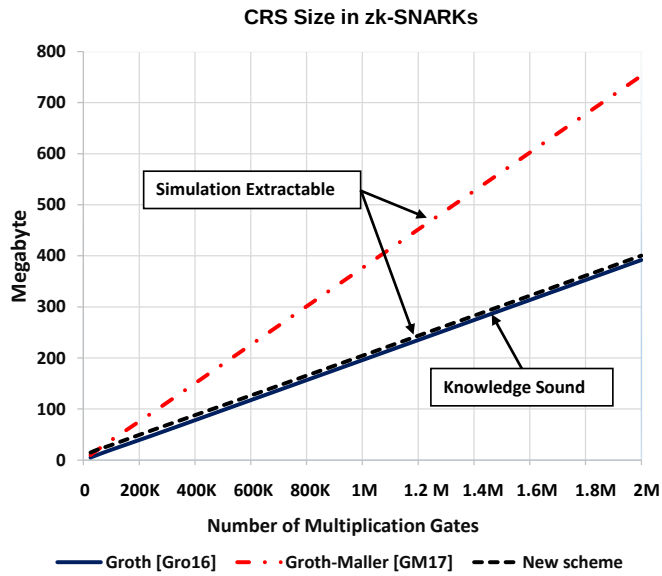

Figure 9. CRS size in zk-SNARKs of Groth [79], Groth-Maller [82] and the proposed SE zk-SNARK (in Section 4.3.3). The plot is drawn for R1CS instances with 10 input variables and various number of constraints.

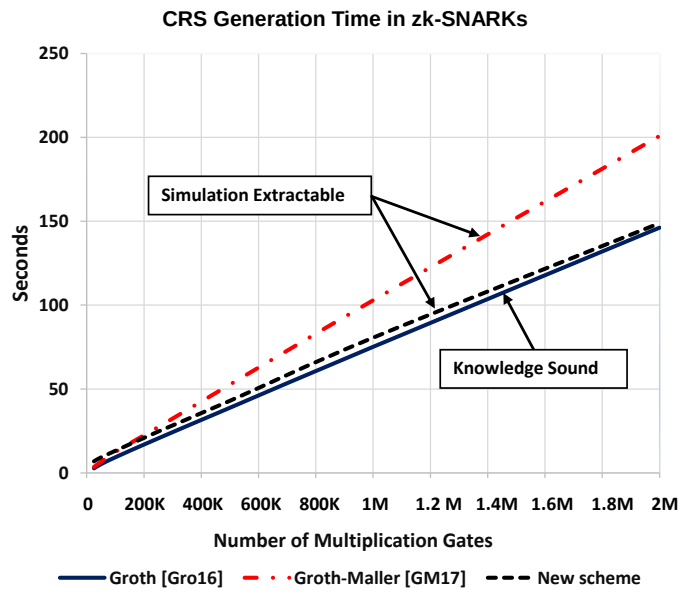

Figure 10. Average CRS generation time for 5 iterative in zk-SNARKs of Groth [79], Groth-Maller [82] and the proposed SE zk-SNARK (in Section 4.3.3). The plot is drawn for R1CS instances with 10 input variables and various number of constraints.

and nBB simulation-extractability at the same time.

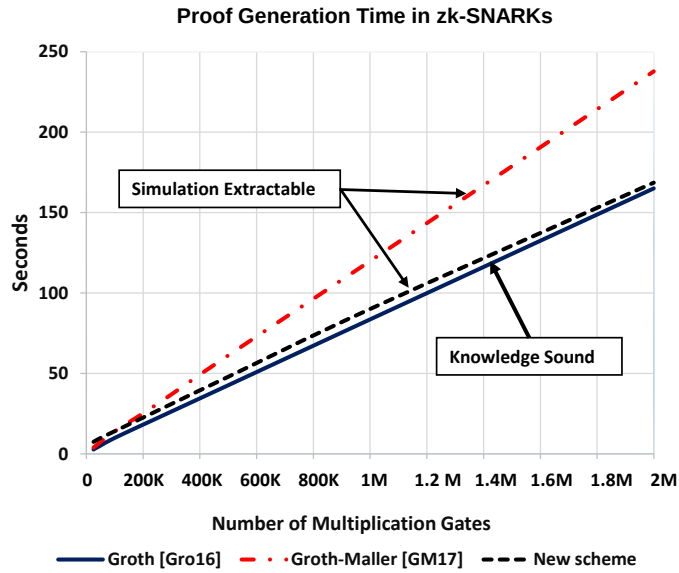

Figure 11. Average proof generation time for 5 iterative in zk-SNARKs of Groth [79], Groth-Maller [82] and the proposed SE zk-SNARK (in Section 4.3.3). The plot is drawn for R1CS instances with 10 input variables and various number of constraints.

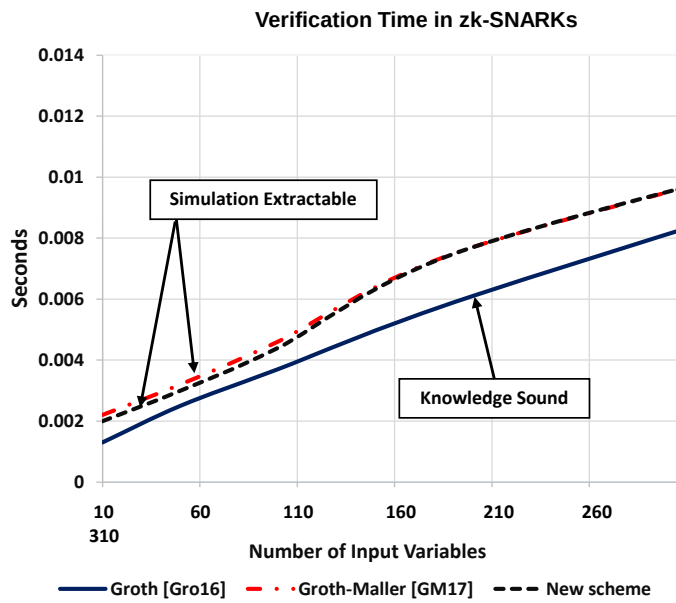

Figure 12. Average proof verification time for 5 iterative in zk-SNARKs of Groth [79], Groth-Maller [82] and the proposed SE zk-SNARK (in Section 4.3.3). The plot is drawn for various R1CS instances with 10^5 constraints and different number of inputs variables.

The new construction confirms that we can efficiently lift our presented subversion-

resistant zk-SNARK in Section 3, such that it would achieve nBB simulation-extractability and Sub-ZK. This also shows that one can amplify the best positive result of Bellare, Fuchsbauer, and Scafuro [20], with regards to constructing subversion-resistant NIZKs. A full detailed description of the result can be found in the paper [12].

4.4.1. Previous Results

As briefly discussed in Section 3.3, in ASIACRYPT 2016, Bellare, Fuchsbauer and Scafuro [20] studied achievable security in NIZKs in the face of subverted CRS. They defined the notions Sub-WI, Sub-ZK, and Sub-SND for subversion-resistant NIZKs. Then they showed that the definitions of Sub-SND and (standard) ZK are not compatible: as the former requires that a prover should not be able to generate a fake proof even if he generates the CRS, but the latter implies that there exists a simulation algorithm that given trapdoors of CRS can generate a (fake) simulated proof indistinguishable from the real ones. This resulted in a negative result that we cannot construct a NIZK argument satisfying ZK and Sub-SND simultaneously.

The above negative result opened two possible directions for positive results on subversion-resistant proof systems. One direction is achieving Sub-ZK and a version of soundness (i.e. either soundness, knowledge-soundness or simulation knowledge-soundness) and the second direction is achieving Sub-WI (the strongest notion weaker than ZK) and a notion of Sub-SND (either subversion soundness, subversion knowledge-soundness or subversion simulation knowledge-soundness).

Along the first direction, Bellare et al. showed that one can construct NIZK arguments that achieve Sub-ZK and (knowledge) SND at the same time [20]. After the mentioned positive result, Abdolmaleki et al. [3] showed that the state-of-the-art zk-SNARK [79] can achieve Sub-ZK and *knowledge* soundness with minimal changes in the CRS and executing an efficient public algorithm to check the well-formedness of CRS elements. In a concurrent work, Fuchsbauer [60] showed that most of the known pairing-based zk-SNARKs including Groth's scheme can achieve Sub-ZK and knowledge-soundness simultaneously. In the same direction, Abdolmaleki et al. [4] showed that one can achieve Sub-ZK and SND in Quasi-Adaptive NIZK (QA-NIZK) arguments.

In the second direction of possible positive results, Bellare et al. [20] showed that Zap schemes proposed by Groth, Ostrovsky, and Sahai [84] achieves Sub-WI and Sub-SND at the same time; as such proof systems do not require a CRS (consequently they do not require a trusted setup phase) but provides weaker security guarantees than ZK. Recently, Fuchsbauer and Orru [63] showed that one can achieve even more in this direction, by presenting a Sub-WI and (Sub-)*knowledge* sound Zap scheme.

4.4.2. Our Construction and Technique

In this section, we show that the language $\mathscr{L}''$ that we defined in Section 4.3.2 would also allow us to construct a subversion-resistant zk-SNARK (more generally, a NIZK argument) that will satisfy Sub-ZK and simulation-extractability (a.k.a. simulation knowledge-soundness). Indeed, we show that using such an OR-based construction, given a NIZK argument that guarantees Sub-ZK and (knowledge) soundness, we can construct a NIZK argument that will guarantee Sub-ZK and *simulation* (knowledge) soundness. Defining such OR-based language can be viewed as using the Bellare-Goldwasser paradigm [24], which was proposed to construct signatures from NIZK arguments, in a non-black-box way.

Roughly speaking, we use the same defined language $\mathscr{L}''$ which was constructed based on OR of the language $\mathscr{L}$ in the input non-interactive argument along with a PRF and a perfectly binding commitment scheme. But this time we assume that the input non-interactive argument satisfies Sub-ZK (instead of ZK) and (knowledge) soundness. Then we use the basic property of an OR construction, namely OR proofs can be simulated using the trapdoors of one branch, and show that if the input NIZK argument achieves Sub-ZK, then the lifted non-interactive argument also guarantees Sub-ZK. As in the notion of Sub-ZK, the prover does not trust the CRS generators and consequently, the simulation trapdoors are not trustable, so in the proof of Sub-ZK, different from the previous case (considered in Section 4.3.2), we use a technique from subversion-resistant schemes and simulate the protocol.

As an instantiation, we show that the variation of Groth's zk-SNARK that we presented in Section 4.3.3 [6] which could achieve ZK and simulation extractability can also achieve Sub-ZK with minimal extra computational cost. The cost is similar to NIZK arguments that achieve Sub-ZK and knowledge-soundness (e.g., the one proposed in Section 3.4.2 or any of the ones proposed in [60]), the prover only needs to execute an efficient CRS verification algorithm to check the well-formedness of CRS elements before using them.

Construction. Consider a subversion-resistant NIZK argument Π_{NIZK} for $\mathbf{R}_{\mathscr{L}}$ which consists of PPT algorithms $(\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{CV}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ and guarantees Sub-ZK and (knowledge) soundness (e.g., the one proposed in Section 3.4.2). Let $\Pi_{\mathsf{Sig}} = (\mathsf{KGen}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf})$ be a one-time signature scheme and $\Pi_{\mathsf{Com}} = (\mathsf{KGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Ver})$ be a perfectly binding commitment scheme.

Using a similar technique proposed by Bellare-Goldwasser [24] (also used in [94]), given the language $\mathscr{L}$ with the corresponding NP relation $\mathbf{R}_{\mathscr{L}}$, we define a new language $\mathscr{L}''$ such that $((\mathsf{x}, \mu, \mathsf{vk}_{\mathsf{Sig}}, \rho), (\mathsf{w}, s, r)) \in \mathbf{R}_{\mathscr{L}''}$ iff:

$$\big((\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathscr{L}} \vee (\mu = f_s(\mathsf{vk}_{\mathsf{Sig}}) \wedge \rho = \mathsf{Com}(s, r))\big),$$

where as mentioned before $\{f_s : \{0,1\}^* \to \{0,1\}^{\lambda}\}_{s \in \{0,1\}^{\lambda}}$ is a PRF family. Due to the OR-based construction of the new language $\mathscr{L}''$, in order to generate a valid proof, one would need either the witness w or the trapdoors of CRS, and

since it is assumed that the CRS trapdoors are kept secret, soundness is guaranteed as long as the CRS trapdoors are secret. We note that due to using the PRF f_s with a random secret trapdoor s, the output of f_s is indistinguishable from the outputs of a truly random function. By considering the new language, the subversion-resistant NIZK argument Π_{NIZK} for the relation $\mathbf{R}_{\mathscr{L}}$ with PPT algorithms $(\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{CV}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ that achieves Sub-ZK and (knowledge) soundness can be lifted to a subversion-resistant NIZK Π'_{NIZK} with PPT algorithms $(\mathsf{KGen}'_{\mathsf{NIZK}}, \mathsf{CV}', \mathsf{P}', \mathsf{V}', \mathsf{Sim}')$ that guarantees Sub-ZK and *simulation* (knowledge) soundness. Based on the language $\mathscr{L}''$, the construction of NIZK $\mathsf{KGen}'_{\mathsf{NIZK}}$ and the corresponding algorithms are described in Fig. 13.

Recall that one of two main differences between a Sub-ZK NIZK argument and a common NIZK argument is the existence of a public CRS verification algorithm CV in the former. Basically, given a CRS crs the algorithm CV verifies the well-formedness of its elements. Additionally, in the simulation of a Sub-ZK NIZK argument, there exists a (non-black-box) extractor $\mathsf{Ext}_{\mathsf{Sub}}$ (e.g., the one that we constructed in Section 3.4.2) that can extract the simulation trapdoors ts from a (possibly malicious) CRS generator Sub.

Similar to other subversion-resistant NIZK arguments [3, 20, 60], we aim to achieve Sub-ZK (not standard ZK) and simulation (knowledge) soundness in our constructions, so there are two key differences between new proposed constructions and the one presented in Section 4.3.2 (that are shown in highlighted form in Fig. 13). The first key difference is that we have an extra algorithm CV' which will be used by prover to check the well-formedness of CRS elements before using them. This is the cost that the prover needs to pay instead of trusting the CRS generators. The second key difference is that in new constructions, the simulator Sim' does not get simulation trapdoors directly, as the prover does not trust the CRS generators in this setting. Instead, the simulator Sim' calls the extraction algorithm $\mathsf{Ext}_{\mathsf{Sub}}$ constructed for the input NIZK argument Π_{NIZK} and extracts simulation trapdoors ts of it, and then uses them for the rest of simulation.

In the paper [12], we prove the following theorem and show that given a subversion-resistant NIZK argument that guarantees completeness, Sub-ZK, and (knowledge) soundness, the described construction in Fig. 13 results in a NIZK argument that achieves completeness, Sub-ZK and *simulation* (knowledge) soundness.

Theorem 3 (Completness, Sub-ZK, nBB Simulation Extractability)**.** *If the NIZK argument Π_{NIZK} guarantees completeness, Sub-ZK, and (knowledge) soundness, the pseudo-random function family is secure, and the one-time signature scheme is secure, then the proposed construction in Fig. 13 achieves completeness, Sub-ZK, non-black-box simulation extractability.*

Proof. The proof is provided in [12] that is included in the thesis. □

CRS Generator $\mathsf{KGen}'_{\mathsf{NIZK}}(\mathbf{R}_{\mathscr{L}})$**:**

- Call CRS generator of the subversion-resistant NIZK Π_{NIZK} and sample $(\mathsf{crs} \parallel \mathsf{tc}) \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(\mathbf{R}_{\mathscr{L}''})$; Set simulation trapdoors ts that are a subset of CRS trapdoors tc;
- $s, r \leftarrow_r \{0,1\}^{\lambda}$; $\rho := \mathsf{Com}(s, r)$;

and output $\mathsf{crs}' := (\mathsf{crs}, \rho)$.

CRS Verifier $\mathsf{CV}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}')$**:** Parse $\mathsf{crs}' := (\mathsf{crs}, \rho)$;

- abort if $\rho = 0$ or is not well-formed;
- call CV algorithm of the input subversion-resistant NIZK Π_{NIZK}

and return $b \leftarrow \mathsf{CV}(\mathbf{R}_{\mathscr{L}}, \mathsf{crs})$.

Prover $\mathsf{P}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}', \mathsf{x}, \mathsf{w})$**:** Parse $\mathsf{crs}' := (\mathsf{crs}, \rho)$;

- abort if $\mathsf{CV}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}') \neq 1$ or $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathscr{L}}$;
- generate $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{vk}_{\mathsf{Sig}}) \leftarrow \mathsf{KGen}_{\mathsf{Sig}}(1^{\lambda})$;
- sample $z_0, z_1, z_2 \leftarrow_r \{0,1\}^{\lambda}$; generate $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, z_0, \mathsf{vk}_{\mathsf{Sig}}, \rho), (\mathsf{w}, z_1, z_2))$ using the prover of subversion-resistant NIZK Π_{NIZK};
- sign $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_{\mathsf{Sig}}, (\mathsf{x}, z_0, \pi))$;

and return $\pi' := (z_0, \pi, \mathsf{vk}_{\mathsf{Sig}}, \sigma)$.

Verifier $\mathsf{V}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}', \mathsf{x}, \pi')$**:** Parse $\mathsf{crs}' := (\mathsf{crs}, \rho)$ and $\pi' := (z_0, \pi, pk_{\mathsf{Sig}}, \sigma)$;

- abort if $\mathsf{Vf}(\mathsf{vk}_{\mathsf{Sig}}, (\mathsf{x}, z_0, \pi), \sigma) = 0$;
- call the verifier of the input subversion-resistant NIZK argument $\mathsf{V}(\mathbf{R}_{\mathscr{L}''}, , \mathsf{crs}, (\mathsf{x}, z_0, \mathsf{vk}_{\mathsf{Sig}}, \rho), \pi)$ and abort if it outputs 0.

Simulator $\mathsf{Sim}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}', \mathsf{ts}, \mathsf{x})$**:** Parse $\mathsf{crs}' := (\mathsf{crs}, \rho)$;

- call the extraction algorithm $\mathsf{Ext}_{\mathsf{Sub}}$ constructed in the simulation of the subversion-resistant NIZK Π_{NIZK} and extract simulation trapdoors ts of Π_{NIZK} from the CRS generator Sub;
- generate $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{vk}_{\mathsf{Sig}}) \leftarrow \mathsf{KGen}_{\mathsf{Sig}}(1^{\lambda})$;
- sample $z_0 \leftarrow_r \{0,1\}^{\lambda}$; generate $\pi \leftarrow \mathsf{Sim}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, z_0, pk_{\mathsf{Sig}}, \rho), \mathsf{ts})$;
- sign $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}_{\mathsf{Sig}}, (\mathsf{x}, z_0, \pi))$;

and output $\pi' := (z_0, \pi, \mathsf{vk}_{\mathsf{Sig}}, \sigma)$.

Figure 13. An extension of the proposed construction in Section 4.3.2 that outputs a Sub-ZK and simulation (knowledge) sound NIZK argument Π'_{NIZK}. Note that in our construction, we assumed that the input NIZK Π_{NIZK} guarantees Sub-ZK and (knowledge) soundness. Due to this fact, we have a new algorithm CV' to verify the well-formedness of CRS elements, and a new simulation procedure by Sim' to simulate the proofs without trusting the CRS generators.

4.4.3. Two Efficient Instantiations

To construct a SNARK that would guarantee Sub-ZK and simulation knowledge-soundness, we instantiate the construction proposed in Section 4.4.2. As the definition of a new language $\mathscr{L}''$ is the same as the language defined in Section 4.3.2, so we use the same instantiation. The only difference is that in this case the input NIZK should be subversion-resistant and guarantee Sub-ZK. As we showed in Section 3, and Fuchsbauer showed in [60], Groth's zk-SNARK can achieve Sub-ZK (and knowledge-soundness), so one can use the variation of Groth's zk-SNARK proposed in either Section 3 or in [60] as an input subversion-resistant NIZK argument Π_{NIZK}. While proving Sub-ZK in Groth's zk-SNARK, in Section 3, we presented a CV algorithm (described in Fig. 4) which is constructed based on the BDH-KEA assumption (which was more convenient while using batching techniques) and needed to add around $2n$ elements to the CRS, where n is the number of multiplication gates in the arithmetic circuit that encodes the language. On the other hand, the CV algorithm that Fuchsbauer [60] proposed is constructed based on a different knowledge assumption[6] and does need to change the CRS elements.

By considering the above discussion, in order to lift Groth's zk-SNARK such that it would achieve Sub-ZK and simulation knowledge-soundness, one can use our construction described in Fig. 13 along with either of the following zk-SNARKs:

1. The variation we proposed in Section 3 along with our proposed CV algorithm expressed in Fig. 4 or its batched version in Fig. 6.
2. The variation proposed by Fuchsbauer [60] along with his proposed CV algorithm.

As in both above cases, the prover and the verifier (described in Fig. 5) are unchanged in comparison with the original protocol of Groth [79], so basically by picking each of them, the only difference would be in CRS elements and the CV' algorithm. In Figure 14, we present a CV' algorithms of which is constructed based on the CV algorithm that we presented in Fig. 4. Basically, in addition to CV algorithms, in CV' algorithms we check whether $\rho \neq 0$ and well-formed, and this is the only difference between CV' and CV algorithms, but CV' is executed for a larger CRS. In Section 3.5, we showed that using batching techniques a similar CV' algorithm can be executed very efficiently, especially faster than running time of prover.

To sum up, Tab. 6 summarizes current subversion-resistant constructions and compares them with an instantiation of our result in this section. The first row shows the negative result that achieving Sub-SND and ZK at the same time is impossible [20]. Next rows indicate the notions achieved in various non-interactive proof systems presented in Sec 3.4, Section 4.4, and [4, 20, 60, 63].

[6] Called Square Knowledge of Exponent (SKE) which states for an asymmetric bilinear group, given $[1]_1$, if an adversary manages to come out with $[a]_1$ and $\left[a^2\right]_1$, he must know a.

CRS of the variation of Groth's zk-SNARK presented in Section 3.4:

$$\mathsf{crs}_{\mathsf{P}} \leftarrow \left(\begin{bmatrix} \alpha, \beta, \delta, (\chi^i \ell(\chi)/\delta)_{i=0}^{n-2}, \left(\frac{u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi)}{\delta}\right)_{j=m_0+1}^{m} \end{bmatrix}_1, \\ \left[(u_j(\chi), v_j(\chi))_{j=0}^{m} \right]_1, \left[\beta, \delta, (v_j(\chi))_{j=0}^{m} \right]_2 \end{bmatrix} \right),$$

$$\mathsf{crs}_{\mathsf{V}} \leftarrow \left(\left[\left(\frac{u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi)}{\gamma}\right)_{j=0}^{m_0} \right]_1, [\gamma, \delta]_2, [\alpha\beta]_T \right) \ ,$$

$$\mathsf{crs}_{\mathsf{CV}} \leftarrow \left([\gamma, (\chi^i)_{i=1}^{n-1}, (\ell_i(\chi))_{i=1}^{n}]_1, [\alpha, \chi, \chi^{n-1}]_2 \right) \ .$$

Where $\mathsf{crs} = (\mathsf{crs}_{\mathsf{CV}}, \mathsf{crs}_{\mathsf{P}}, \mathsf{crs}_{\mathsf{V}})$.

CRS of the zk-SNARK constructed in Section 4.4: $\mathsf{crs}' = (\mathsf{crs}, \rho)$

CRS Verifier for the construction in Section 4.4, $\{0,1\} \leftarrow \mathsf{CV}'(\mathbf{R}, \mathsf{crs}')$**:**

Return 1 if all the following checks were successful; otherwise return 0.

1. For $\iota \in \{\gamma, \delta\}$: check that $[\iota]_1 \neq [0]_1$
2. For $\iota \in \{\alpha, \beta, \gamma, \delta\}$: check that $[\iota]_1 \bullet [1]_2 = [1]_1 \bullet [\iota]_2$,
3. For $i = 1$ to $n-1$: check that $\left[\chi^i\right]_1 \bullet [1]_2 = \left[\chi^{i-1}\right]_1 \bullet [\chi]_2$,
4. Check that $([\ell_i(\chi)]_1)_{i=1}^{n}$ is correctly computed by using Alg. 2,
5. For $j = 0$ to m:
   (a) Check that $[u_j(\chi)]_1 = \sum_{i=1}^{n} U_{ij} [\ell_i(\chi)]_1$,
   (b) Check that $[v_j(\chi)]_1 = \sum_{i=1}^{n} V_{ij} [\ell_i(\chi)]_1$,
   (c) Set $[w_j(\chi)]_1 \leftarrow \sum_{i=1}^{n} W_{ij} [\ell_i(\chi)]_1$,
   (d) Check that $[v_j(\chi)]_1 \bullet [1]_2 = [1]_1 \bullet [v_j(\chi)]_2$,
6. For $j = m_0 + 1$ to m: check that $[(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta]_1 \bullet [\delta]_2 = [u_j(\chi)]_1 \bullet [\beta]_2 + [v_j(\chi)]_1 \bullet [\alpha]_2 + [w_j(\chi)]_1 \bullet [1]_2$,
7. Check that $\left[\chi^{n-1}\right]_1 \bullet [1]_2 = [1]_1 \bullet \left[\chi^{n-1}\right]_2$,
8. For $i = 0$ to $n-2$: check that $\left[\chi^i \ell(\chi)/\delta\right]_1 \bullet [\delta]_2 = \left[\chi^{i+1}\right]_1 \bullet \left[\chi^{n-1}\right]_2 - \left[\chi^i\right]_1 \bullet [1]_2$,
9. Check that $[\alpha]_1 \bullet [\beta]_2 = [\alpha\beta]_T$.
10. Check that ρ is well-formed and $\rho \neq 0$.

Figure 14. A CRS verification algorithm for our construction presented in Section 4.4 when the input subversion-resistant NIZK is instantiated with our presented subversion-resistant zk-SNARK in Sec 3.4. Note that $\mathsf{crs}' := (\mathsf{crs}, \rho)$, where $\rho := \mathsf{Com}(s, r)$.

4.4.4. Efficiency Evaluation

Next, we analyze the efficiency of the instantiated construction from different perspectives.

Setup phase. Similar to the presented construction in Section 4.3, the setup phase needs to be done for a new arithmetic circuit that encodes the language $\mathscr{L}''$.

Table 6. A comparison of our results with current subversion-resistant non-interactive proof systems and their security guarantees. ZK: Zero-Knowledge, SND: Soundness, KS: Knowledge Soundness, SKS: Simulation Knowledge Soundness, (S)WI: (Subversion) Witness Indistinguishable, SZK: Subversion Zero-Knowledge, SSND: Sub-Soundness, SubKS: Subversion Knowledge Soundness.

	Prover's Security				Verifier's Security				
Result	WI	ZK	SWI	SZK	SND	KS	SKS	SSND	SubKS
– [20]		✓						✓	
+ [20]	✓		✓		✓			✓	
+ [63]	✓		✓		✓	✓		✓	✓
+ [20]	✓	✓	✓		✓				
+ [20]	✓	✓	✓	✓	✓				
+ [4]	✓	✓	✓	✓	✓				
+ Sec 3.4, [60]	✓	✓	✓	✓	✓	✓			
+ Sec 4.4	✓	✓	✓	✓	✓	✓	✓		

For the particular instantiation discussed in 4.3.3, the new changes will add around 52.000 multiplication gates to the QAP-based arithmetic circuits that encode $\mathscr{L}$. The number of gates is calculated from instantiating both the commitment scheme and the PRF with SHA512 hash function [93]. Implementations show that this will add a constant amount (e.g., 10 megabytes) to the size of the original CRS, that for arithmetic circuits with a large number of gates the overhead is negligible (see Fig. 9).

CRS Verification. To verify the well-formedness of CRS elements one needs to execute CV' algorithm which almost has the same efficiency as CV algorithm in original NIZK argument Π_{NIZK}. As we observed in Fig. 7, executing CV can be even more efficient than executing P.

Prover. The prover needs to generate a proof for a new arithmetic circuit that encodes new language $\mathscr{L}''$ and sign the proof with a one-time secure signature scheme. Our practical analysis in Fig. 11 shows that the overhead for a QAP-based zk-SNARK is very small in practical cases. For instance, for an R1CS instance with 10^6 constraints and 10 input variables, already proof generation needed 83 seconds, but in the new construction, this will take ≈ 90.1 seconds.

Proof size. By considering the considered instantiation, in the new construction the size of new proof $\pi' := (z_0, \pi, \mathsf{vk}_{\mathsf{Sig}}, [\sigma]_1, r)$ will be equal to the size of the original proof π plus two elements from $\mathbb{G}_1$, one element from $\mathbb{G}_2$ and two 256-bit bit-strings that for 128-bit security will have size less than 318 bytes. Using a RO and weak version of Boneh and Boyen's signature scheme [38], this proof size can be reduced to 256 bytes.

Verifier. In addition to the verification of input subversion-resistant NIZK argument Π_{NIZK}, the verifier of argument Π'_{NIZK} will verify the validity of a one-time secure signature which due to instantiating the signature scheme with Boneh

and Boyen's signature [38], verification of the signature scheme adds only 1 paring and 1 exponentiation to the original one.

5. COMMITMENT SCHEMES IN THE FACE OF PARAMETER SUBVERSION

5.1. Motivation

Our results in the last two sections showed that one can construct NIZK arguments that would guarantee Sub-ZK and simulation knowledge-soundness (a.k.a. simulation extractability). Roughly speaking, the result showed that a prover of a zk-SNARK in the CRS model can avoid trusting CRS generators by checking the CRS elements using an efficient CRS verification algorithm. On the other hand, the proofs will satisfy nBB simulation knowledge-soundness, defined in Def. 13.

Commitment schemes [34] are another fundamental and widely used primitives in cryptography that in the CRS model requires a *setup* phase which is supposed to be done by a trusted third party [47]. We reviewed definitions of commitment schemes in Section 2.3.4. During last few decades, we have seen various elegant non-interactive commitment schemes that are deployed as a sub-protocol in a wide range of cryptographic protocols and applications, contract signing [56], multi-party computation [68], zero-knowledge proofs [48, 69], commit-and-proof systems [53,86,98], e-voting [75,110], shuffle arguments [57,81,110], blockchains and their by-products (e.g., cryptocurrencies [27, 58] and smart contracts [93]), and many other sensitive practical applications.

Along with developing various cryptographic primitives in sensitive applications, recently there have been various attacks or flaw reports on the setup phase of cryptographic systems that rely on public parameters generated honestly. Particularly about commitment schemes, recently two results [88, 95] independently discovered that the implementation of shuffle arguments in the SwissPost-Scytl mix-net uses a trapdoor commitment scheme such that a malicious commitment key generator can store the trapdoor, which allows breaking security of the main system without being detected. Clearly speaking, in their case the used commitment scheme has a trapdoor that enables one to alter the votes but still can produce an acceptable shuffle proof. So, given such a trapdoor, a malicious party can do an undetectable vote manipulation by an authority who sets up the mixing network[1]. It is also possible to break voter's privacy by manipulating the commitment keys that will be used by the voters. Such flaw reports show the importance of commitment key generation in commitments.

5.2. Related Works

In order to mitigate the trust in public parameters generated by a third party, a known long-standing technique is distributed computations that are used in MPC

[1]More details in `https://people.eng.unimelb.edu.au/vjteague/SwissVote` and `https://e-voting.bfh.ch/publications/2019/`

protocols [64, 83, 91]. Another direction is subversion security that recently was studied for various cryptographic primitives. Initiated by Bellare et al. [25], recently this direction has received considerable attention with focus on different cryptographic primitives including symmetric encryption schemes [25], signature schemes [7], non-interactive zero-knowledge proofs [20], and public-key encryption schemes [8]. Each of the mentioned studies considers achievable security in a particular cryptographic primitive under subverted public parameters. As mentioned above, non-interactive (trapdoor) commitment schemes are another prominent family of cryptographic primitives whose public commitment keys are assumed to be generated honestly by a trusted third party [51, 77, 78, 84, 96, 106]. As such commitment schemes are deployed in various areas of cryptography, so their security is not only important on itself but also security of other practical systems relies on it (e.g., guaranteeing the security of shuffling in the mix-net of SwissPost-Scytl). Thus their security under subversion of public commitment key can have a crucial effect on the security of the complete system.

5.3. Problem Statement

Considering the fact that constructing equivocal commitments (a.k.a. trapdoor commitments) in the CRS model, requires a third party to generate the *public commitment key*, studying the security of them in the face of subverted parameters is an interesting research question from both theoretical and practical perspectives. As mentioned before, using MPC protocols [64, 83, 91] is an alternative approach to mitigate the trust on public parameters generated by a third party. But in general, the achievable security in the case that the public commitment key is generated maliciously was not considered clearly.

In this section we focus on the following research question. What security we can achieve in commitment schemes in the face of a maliciously chosen public commitment key?

5.4. Security of Commitments with an Untrusted Parameters

In the rest, we focus on the research question mentioned in the last subsection by studying achievable security in commitment schemes in the CRS model in the face of subverted commitment keys and presenting constructions that can resist against subverting public parameters.

To attain a clear understanding of achievable security, we first present a variety of current definitions called subversion hiding, subversion equivocality, and subversion binding. Then we provide both negative and positive results on constructing subversion-resistant commitment schemes, by showing that some combinations of notions are not compatible, while presenting subversion-resistant commitment schemes that can achieve other combinations.

A full detailed description of the results can be found in the paper [11].

5.4.1. Subversion-Resistant Notions: Sub-Hiding, Sub-Binding, Sub-Equivocality

Definitions for standard (equivocal) commitment schemes in the CRS model and their desired security guarantees were reviewed in Section 2.3.4. As discussed before in the equivocal constructions in the CRS model, a critical assumption is that the commitment key ck is honestly generated by a trusted third party. To consider achievable security in CRS-based commitment schemes with compromised setup phase, we first determine new goals and define subversion-resistance analogs sub-hiding, sub-equivocality, and sub-binding as a variation of the standard notions [77], reviewed in Section 2.3.4.

In the new notions, the key difference is that the setup is compromised and the key ck is selected by an adversary $\mathscr{A}$ (or a subverter) rather than via the honest key-generation algorithm $\mathsf{KGen}_{\mathsf{Com}}$ prescribed by Π_{Com}. We consider the *worst* case that the malicious key generator and the adversary are the same, but one can separate them and assume that the subverter provides (part of) his secret information to the adversary. Similar to the CRS verification algorithm in subversion-resistant NIZKs (studied in Section 3 and Section 4), there is a new algorithm CKVer which is used to verify the well-formedness of commitment key ck. In the rest, we formally define a subversion-resistant equivocal commitment scheme and the target goals. A *subversion-resistant* equivocal commitment scheme Π_{Com} consists of eight algorithms $\Pi_{\mathsf{Com}} = (\mathsf{KGen}_{\mathsf{Com}}, \mathsf{CKVer}, \mathsf{Com}, \mathsf{Ver}, \mathsf{KGen}^*_{\mathsf{Com}}, \mathsf{Com}^*, \mathsf{Equiv})$ that are defined as follows,

- **Key Generation**, $\mathsf{ck} \leftarrow \mathsf{KGen}_{\mathsf{Com}}(\mathsf{gk})$: Generates a commitment key ck and associated trapdoor tk. It returns ck and keeps secret or removes tk. It also specifies a message space $\mathscr{M}$, a randomness space $\mathscr{R}$, and a commitment space $\mathscr{C}$. The algorithm is supposed to be executed by a trusted authority.
- **Commitment Key Verification**, $0/1 \leftarrow \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck})$: CKVer is a deterministic polynomial-time algorithm that given setup information gk and the commitment key ck, returns either 0 (the ck is incorrectly formed) or 1 (the ck is correctly formed);
- **Committing**, $(c, \mathsf{op}) \leftarrow \mathsf{Com}(\mathsf{ck}, m; r)$: Outputs a commitment c and opening information op. This algorithm specifies a function $\mathsf{Com} : \mathscr{M} \times \mathscr{R} \rightarrow \mathscr{C}$. Given a message $m \in \mathscr{M}$, the committer picks a randomness $r \in \mathscr{R}$ and computes $(c, \mathsf{op}) \leftarrow \mathsf{Com}(\mathsf{ck}, m; r)$.
- **Opening Verification**, $0/1 \leftarrow \mathsf{Ver}(\mathsf{ck}, c, m, \mathsf{op})$: Outputs 1 if $m \in \mathscr{M}$ is the committed message in the commitment c with opening value op, and returns 0 if (c, m, op) does not correspond to a valid message, opening and commitment tuple.
- **Simulation of Key Generation**, $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}^*_{\mathsf{Com}}(\mathsf{gk})$: Generates and returns a commitment key ck and associated trapdoor tk. It also specifies a message space $\mathscr{M}$, a randomness space $\mathscr{R}$, and a commitment space $\mathscr{C}$.
- **Trapdoor Committing**, $(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*(\mathsf{ck}, \mathsf{tk})$: Given commitment key

ck and tk as input, outputs an equivocal commitment c and an equivocation key ek.

- **Trapdoor Opening**, $\mathsf{op} \leftarrow \mathsf{Equiv}(\mathsf{ek}, c, m)$: On inputs ek, c and a message m creates an opening $\mathsf{op} := r$ of the commitment, s.t. $(c, \mathsf{op}) = \mathsf{Com}(\mathsf{ck}, m; r)$ and returns op.

A (subversion-resistant equivocal) commitment scheme satisfies *completeness* if for $\mathsf{ck} \leftarrow \mathsf{KGen}_{\mathsf{Com}}(\mathsf{gk})$ and any honestly generated commitment of $m \in \mathscr{M}$, it successfully passes the verification, i.e., $\mathsf{Ver}(\mathsf{ck}, \mathsf{Com}(\mathsf{ck}, m; \mathsf{op}), m, \mathsf{op}) = 1$.

In the following definitions, let Setup be an algorithm that takes as input the security parameter λ and outputs some setup information $\mathsf{gk} \leftarrow \mathsf{Setup}(1^\lambda)$.

Definition 17 (Subversion Hiding (Sub-Hiding))**.** *A commitment scheme Π_{Com} guarantees computationally subversion hiding if for any PPT adversary $\mathscr{A}$,*

$$\left| 2 \Pr \left[\begin{array}{l} (\mathsf{ck}, (m_0, m_1)) \leftarrow \mathscr{A}(\mathsf{gk}), b \leftarrow_\$ \{0,1\}, \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1, \\ r_b \leftarrow_\$ \mathscr{R}, (c_b, \mathsf{op}_b) \leftarrow \mathsf{Com}(\mathsf{ck}, m_b; r_b), b' \leftarrow \mathscr{A}(c_b) : b' = b \end{array} \right] - 1 \right| = \mathsf{negl}(\lambda) \quad .$$

The scheme is perfectly subversion hiding if the above probability is equal to 0.

Intuitively, *subversion hiding* states that an adversary $\mathscr{A}$ will not be able to say which of two messages m_0 and m_1 is committed, even if it picks both messages itself and generates the (well-formed) commitment key ck. In new definitions, by *well-formedness* of ck we mean the CKVer will verify ck successfully.

Definition 18 (Subversion Binding (Sub-Binding))**.** *A commitment scheme Π_{Com} guarantees computationally subversion binding if for any PPT adversary $\mathscr{A}$,*

$$\Pr \left[\begin{array}{l} (\mathsf{ck}, c, (m_0, \mathsf{op}_0), (m_1, \mathsf{op}_1)) \leftarrow \mathscr{A}(\mathsf{gk}) : \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1 \wedge \\ (m_0 \neq m_1) \wedge (\mathsf{Ver}(\mathsf{ck}, c, m_0, \mathsf{op}_0) = 1) \wedge (\mathsf{Ver}(\mathsf{ck}, c, m_1, \mathsf{op}_1) = 1) \end{array} \right] = \mathsf{negl}(\lambda) \quad .$$

The commitment is perfectly subversion binding if the probability is equal to 0.

Intuitively, *subversion binding* states that an adversary $\mathscr{A}$ will not be able to do double open a commitment c, even if it generates the (well-formed) key ck.

Definition 19 (Subversion Equivocality (Sub-Equivocality))**.** *A commitment scheme Π_{Com} guarantees subversion equivocalability if for any PPT adversary $\mathscr{A}$,*

$$\left| \Pr \left[\begin{array}{l} (\mathsf{ck}, m) \leftarrow \mathscr{A}(\mathsf{gk}), r \leftarrow_\$ \mathscr{R}, \\ (c, \mathsf{op}) \leftarrow \mathsf{Com}(\mathsf{ck}, m; r) : \\ \mathscr{A}(\mathsf{ck}, c, \mathsf{op}) = 1 \wedge \\ \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1 \end{array} \right] - \Pr \left[\begin{array}{l} (\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}^*(\mathsf{gk}), m \leftarrow \mathscr{M}_{\mathsf{ck}} \\ (c, \mathsf{ek}) \leftarrow \mathsf{Com}^*(\mathsf{ck}, \mathsf{tk}), \\ \mathsf{op} \leftarrow \mathsf{Equiv}(\mathsf{ek}, c, m) : \\ \mathscr{A}(\mathsf{ck}, c, \mathsf{op}) = 1 \wedge \\ \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1, \end{array} \right] \right| \leq \mathsf{negl}(\lambda),$$

where $\mathscr{A}$ outputs $m \in \mathscr{M}$ and KGen^* is the key generator which also returns trapdoor tk.

Intuitively, subversion equivocality states that even if an adversary $\mathscr{A}$ (a malicious key generator) generates a (well-formed) commitment key ck, there still exists an efficient algorithm KGen^* which is able to produce the full view of the key generation phase. There also exist PPT algorithms Com^* and Equiv that given the trapdoor tk of commitment key ck, can come up with a fake commitment and a valid opening s.t. they would be indistinguishable from the real ones.

Lemma 2. *A commitment scheme that satisfies a security notion with subvertible setup also satisfies the security notion with honest setup.*

Proof. To prove the lemma, we must show that an adversary $\mathscr{A}$ against an honest setup can be used to construct an adversary $\mathscr{B}$ against a subvertible setup.

Adversary $\mathscr{B}$ first samples a commitment key ck honestly, i.e., $\mathsf{ck} \leftarrow \mathsf{KGen}_{\mathsf{Com}}(\mathsf{gk})$ and checks that $\mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1$. Next, sends ck to $\mathscr{A}$ and gets the answer and sends it to the challenger. Similarly, follows the rest of experiment and wins the game of (subversion security) with the same probability as the adversary $\mathscr{A}$ wins the standard game. □

Fig. 15 shows the relation between current and subversion-resistant notions in commitment schemes.

sub-equivocality	$\longrightarrow$	sub-hiding	sub-binding
$\downarrow$		$\downarrow$	$\downarrow$
equivocality	$\longrightarrow$	hiding	binding

Figure 15. Relation between current and new-defined subversion-resistant notions in commitment schemes.

5.5. Sub-binding with Equivocality are not Compatible

Next, we consider if we can construct commitment schemes that can guarantee sub-binding without degrading hiding, binding, and equivocality. Unfortunately, we show that the definitions of equivocality and sub-binding are not compatible and constructing a commitment scheme which achieves both at the same time is impossible.

The following theorem establishes the negative result, and its complete proof is presented in the manuscript [11].

Theorem 4 (Impossibility of Sub-binding along with Equivocality)**.** *There cannot exists a CRS-based commitment scheme* $\Pi_{\mathsf{Com}} = (\mathsf{KGen}_{\mathsf{Com}}, \mathsf{CKVer}, \mathsf{Com}, \mathsf{Ver}, \mathsf{KGen}^*_{\mathsf{Com}}, \mathsf{Com}^*, \mathsf{Equiv})$ *which can satisfy equivocality and sub-binding at the same time.*

Proof. (*Sketch.*) The definition of equivocality states that there exists KGen^* that given gk generates $(\mathsf{ck}, \mathsf{tk})$, and given trapdoor tk there exist two algorithms Com^* and Equiv that allow one to create a fake commitment and a valid opening which

are indistinguishable from an honestly generated commitment and opening. So, given those algorithms, an adversary of sub-binding can first generate ck and tk honestly. Then, it gives ck and tk as input to Com^* and calculates $(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*(ck, \mathsf{tk})$. After that, it samples $(m_0, m_1) \in \mathscr{M}_{\mathsf{ck}}$, where $m_0 \neq m_1$ and invokes the algorithm Equiv twice for two different messages, and generates $\mathsf{op}_0 \leftarrow \mathsf{Equiv}(\mathsf{ek}, c, m_0)$ and $\mathsf{op}_1 \leftarrow \mathsf{Equiv}(\mathsf{ek}, c, m_1)$ and sends $(c, (m_0, \mathsf{op}_0), (m_1, \mathsf{op}_1))$ to the challenger of sub-binding game and wins with probability 1, as each of the tuples (m_0, op_0) and (m_1, op_1) are a (distinct) valid opening for c.

On the other hand, sub-binding requires that an adversary should not be able to double open even if he generates the ck. But, one can observe that achieving equivocality implies that given tk one can use Com^* and Equiv and generate two valid opening with different messages which will bread sub-binding. □

5.6. Commitment Schemes with Sub-equivocality and Binding

By considering the mentioned incompatibility and the negative result in Section 5.5, we consider if we can construct subversion-resistant commitment schemes in the CRS model that can guarantee sub-equivocality and binding at the same time. Among the defined notions, this is the best one can achieve if they want to retain equivocality when the commitment key is subverted.

As the first positive result, we show that one can construct a sub-equivocal and binding commitment scheme in the CRS model. By considering the definition of sub-equivocality (given in Def. 19), to achieve sub-equivocality in a commitment scheme, there must be algorithms KGen^*, Com^* and Equiv, where KGen^* simulates a *malicious* setup phase, and Com^* and Equiv output a fake commitment and the corresponding valid opening, consequently. With a compromised setup phase, the algorithms Com^* and Equiv cannot get honestly generated trapdoors of ck, and they also cannot extract the trapdoors from the malicious key generator $\mathscr{A}$ by rewinding, as they do not have any interaction with $\mathscr{A}$. So instead, similar to the case in NIZK arguments [3, 12, 20, 60], we will rely on a knowledge assumption which allows extracting trapdoors of ck from a malicious key generator in a non-black-box manner. Once we extracted the tk, we can provide the extracted trapdoors to algorithms Com^* and Equiv to generate a pair of fake but acceptable commitment and opening. To guarantee binding, a minimal requirement is that an adversary cannot obtain the tk of ck from an honestly generated ck and this is the reason that still the versifier requires to trust the CRS generator yet.

Another key point is that in the case of compromised setup phase, similar to [20], we assume that the setup information gk (e.g., group descriptions) are generated in a deterministic way by each party of the protocol. In fact, the gk is considered as a part of the scheme specification.

Construction: In 2012, Lipmaa [96] presented a variation of knowledge commitment scheme proposed by Groth [78] (described in Fig. 16). In the rest, we show

Setup, $\mathsf{gk} \leftarrow \mathsf{BGgen}(1^\lambda)$**:** Given 1^λ, return $\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where p (a large prime) is the order of cyclic Abelian groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$; $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient non-degenerate bilinear pairing.

Key Generation, $\mathsf{ck} \leftarrow \mathsf{KGen}_{\mathsf{Com}}(\mathsf{gk})$**:** Let Γ be an (n, λ)-nice tuple for some $n = \mathsf{poly}(\lambda)$ with $\eta_i = i$ in the original version, for $i \in [0..n]$. Sample $\hat{a}, x \leftarrow \mathbb{Z}_p$. Let $t \in \{1,2\}$. Return the key $\mathsf{ck} = (\mathsf{ck}_1, \mathsf{ck}_2)$ where $\mathsf{ck}_t \leftarrow \{[x^i]_t, [\hat{a}x^i]_t\}$ for $i \in [0..n]$ and the corresponding trapdoor tk as $\mathsf{tk} = x$.

Commitment Key Verification, $0/1 \leftarrow \mathsf{CKVer}(\mathsf{gk}, \mathsf{ck})$**:** Given gk and the commitment key ck, first parse $ck := (\{[x^i]_1, [\hat{a}x^i]_1\}, \{[x^i]_2, [\hat{a}x^i]_2\})$ for $i \in [0..n]$ and then do the following verification on elements of the ck,

- Check whether $[\hat{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}]_2$
- For $i \in [1..n]$ check:
  1. $[x^i]_1 \bullet [1]_2 = [1]_1 \bullet [x^i]_2$
  2. $[\hat{a}x^i]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}x^i]_2$
  3. $[\hat{a}]_1 \bullet [x^i]_2 = [1]_1 \bullet [\hat{a}x^i]_2$
  4. $[\hat{a}x]_1 \bullet [x^{i-1}]_2 = [1]_1 \bullet [\hat{a}x^i]_2$

and return 1 if all checks passed successfully; otherwise return 0.

Committing, $(c, \mathsf{op}) \leftarrow \mathsf{Com}(\mathsf{ck}, \vec{m}; r)$**:** Given $(\mathsf{ck}, \vec{m})$ for $\mathsf{CKVer}(\mathsf{gk}, \mathsf{ck}) = 1$, to commit to $\vec{m} = (m_1, m_2, \ldots, m_n) \in \mathbb{Z}_p^n$ sample a random $r \leftarrow_\$ \mathbb{Z}_p$, and return $(c, \mathsf{op} := r)$ that are defined as follows,

$$c := (c_t^1, c_t^2) = (r[1]_t + \textstyle\sum_{i=1}^n m_i [x^i]_t, r[\hat{a}]_t + \sum_{i=1}^n m_i [\hat{a}x^i]_t)$$

Opening Verification, $0/1 \leftarrow \mathsf{Ver}(\mathsf{ck}, c, \vec{m}, \mathsf{op})$**:** Given c, $\vec{m}$ and $\mathsf{op} = r$, recompute c as original one and check if it is equal to given c and return $0/1$.

Simulation of Key Generation, $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}^*_{\mathsf{Com}}(\mathsf{gk})$**:** Use the simulation algorithm $\mathsf{Sim}_{\mathscr{A}}$ in Fig. 18 and generates a key pair $(\mathsf{ck}, \mathsf{tk} := (x, \hat{a}))$.

Trapdoor Committing, $(c, \mathsf{ek}) \leftarrow \mathsf{Com}^*(\mathsf{ck}, \mathsf{tk})$**:** Given the a key pair $(\mathsf{ck}, \mathsf{tk})$, output an equivocal commitment $c = [r]_t$ where $r \leftarrow \mathbb{Z}_p^2$ and an equivocation key $\mathsf{ek} = (\mathsf{tk}, r)$.

Trapdoor Opening, $\mathsf{op} \leftarrow \mathsf{Equiv}(\mathsf{ek}, c, \vec{m})$**:** On input equivocation key $\mathsf{ek} = (\mathsf{tk} := (x, \hat{a}), r \in \mathbb{Z}_p^2)$, $c \in \mathscr{C}^2$ and messages $\vec{m}$ create an opening $r' = r - \sum_{i=1}^n m_i x^i)$ for any $\vec{m}$, so that $(c, \mathsf{op}) = \mathsf{Com}(\mathsf{ck}, \vec{a}; r')$ and return $\mathsf{op} = r'$.

Figure 16. A variation of the commitment scheme of Groth [78] defined by Lipmaa [96] that achieves sub-equivocality and binding. We note that in this setting, $\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$ is part of the scheme specification, and in practice each party can run the deterministic algorithm BGgen and re-obtain gk.

Extraction algorithm, $\mathsf{tk} \leftarrow \mathsf{Ext}_{\mathscr{A}}(\mathsf{gk}, \mathsf{ck}, \mathsf{aux}_{\mathbf{R}})$:

Given source code and random coins of the malicious key generator $\mathscr{A}$, and some auxiliary information $\mathsf{aux}_{\mathbf{R}}$ it extracts $(x, \hat{a}) \leftarrow \mathsf{Ext}_{\mathscr{A}}(\mathsf{gk}, \mathsf{ck}, \mathsf{aux}_{\mathbf{R}})$ and set $\mathsf{tk} := (x, \hat{a})$; Finally, **Return** tk.

Figure 17. A BDH-KE assumption based extraction algorithm $\mathsf{Ext}_{\mathscr{A}}$ for the sub-equivocal commitment scheme described in Fig. 16

Simulator $\mathsf{Sim}_{\mathscr{A}}(\mathsf{gk})$:

$\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2, \mathsf{aux}_{\mathbf{R}}) \leftarrow \mathsf{BGgen}(1^{\lambda})$; $\mathsf{ck} \leftarrow \mathscr{A}(\mathsf{gk})$;
`# as in Fig. 16`

By executing $\mathsf{CKVer}(\mathsf{gk}, \mathsf{ck})$,

Check whether $[\hat{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}]_2$

For $i \in [1..n]$ check:

1. $[x^i]_1 \bullet [1]_2 = [1]_1 \bullet [x^i]_2$
2. $[\hat{a}x^i]_1 \bullet [1]_2 = [1]_1 \bullet [\hat{a}x^i]_2$
3. $[\hat{a}]_1 \bullet [x^i]_2 = [1]_1 \bullet [\hat{a}x^i]_2$
4. $[\hat{a}x]_1 \bullet [x^{i-1}]_2 = [1]_1 \bullet [\hat{a}x^i]_2$

if the checks pass, $\mathsf{tk} := (x, \hat{a}) \leftarrow \mathsf{Ext}_{\mathscr{A}}(\mathsf{gk}, \mathsf{ck}, \mathsf{aux}_{\mathbf{R}})$ `# as in Fig. 17`

Otherwise $\mathsf{tk} \leftarrow \perp$

Return $(\mathsf{ck}, \mathsf{tk})$

Figure 18. Sim of setup phase in the commitment scheme described in Fig. 16.

that his proposed variant for $\eta_i = i$ can achieve sub-equivocality and binding under the BDH-KE (defined in Assumption 4) and Γ-PDL (defined in Assumption 3) assumptions along with some well-formedness checking on ck.

The following theorem establishes the first positive result, and its complete proof is presented in the manuscript [11].

Theorem 5 (Sub-equivocal and Binding Commitment Scheme)**.** *Let* BGgen *be a bilinear group generator. Then the commitment scheme* Π_{Com} *described in Fig. 16 is binding in* $\mathbb{G}_t$ *for* $t \in \{1,2\}$*, under the* Γ*-PDL assumption and also satisfies sub-equivocality under the BDH-KE knowledge assumption.*

Proof. (*Sketch*). As we did not change the key ck and committing procedure Com, so the proof of binding is as in the original scheme which is done in [96] under the Γ-PSDL assumption in the group $\mathbb{G}_t$ for $t \in \{1,2\}$. To prove sub-equivocality, in [96], it is shown that the original scheme is equivocal under a trusted setup, namely the setup phase is simulatable, and the algorithms Com^* and Equiv that can generate a fake commitment and valid opening are shown in Fig. 16. The algorithms Com^* and Equiv get the honestly generated trapdoor tk, but in this,

our case the tk is not trustable anymore. To deal with this issue, instead of getting tk directly from a malicious key generator $\mathscr{A}$, we construct a knowledge assumption-based extraction algorithm $\mathsf{Ext}_{\mathscr{A}}$ (expressed in Fig. 17) that can extract the trapdoor tk from $\mathscr{A}$ and provide it to the algorithms Com^* and Equiv.

Next using the extracted trapdoor tk by the extraction algorithm $\mathsf{Ext}_{\mathscr{A}}$, one can simulate a malicious setup phase by the algorithm $\mathsf{Sim}_{\mathscr{A}}$ described in Fig. 18. Finally, after extracting the trapdoor tk, the rest of proof will be as in the standard equivocality given in [96]. □

Remark 1. *In the manuscript [11], we use batching techniques [23, 89] and present a batched version of the original* CKVer *algorithm (described in Fig. 16) that can be more efficient and practical.*

5.7. Commitment Schemes with Sub-hiding and Sub-binding

Next, we discuss the second positive result by constructing commitment schemes that achieve sub-hiding and sub-binding at the same time, but not equivocality. Let $\Pi_{\mathsf{Com}}^{2-party} = (\mathsf{KGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Ver})$ be a commitment scheme that does not require a particular setup phase and the committer and verifier can ignore the output of $\mathsf{KGen}_{\mathsf{Com}}$. Namely, the classical commitments that guarantee hiding and binding and do not require a setup phase. We show that such a hiding and binding commitment scheme, also guarantees sub-hiding and sub-binding. Roughly speaking, this includes all commitment schemes that do not require a (particular) setup phase, except choosing some public parameters that can be agreed between both parties, e.g., agreeing on the order and generator of the underlying group or a particular secure and collision resistant hash function family. Intuitively, one can see that in such setting (e.g. $\mathsf{ck} = \{\}$), there is no risk of subverting.

The following lemma and theorem establish the second positive result, and their complete proof is presented in the manuscript [11].

Lemma 3. *Let* $\Pi_{\mathsf{Com}}^{2-party} = (\mathsf{KGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Ver})$ *be a commitment scheme that does not require a particular setup phase. If* $\Pi_{\mathsf{Com}}^{2-party}$ *satisfies binding and hiding, it also guarantees sub-binding and sub-hiding.*

Proof. The proof is provided in the manuscript [11]. □

Theorem 6 (Sub-hiding and Sub-binding Commitment Schemes)**.** *In the CRS model, under some standard assumptions, there exist commitment schemes that achieve sub-hiding and sub-binding.*

Proof. The commitment schemes that do not require a particular setup phase and guarantee hiding and binding are a $\Pi_{\mathsf{Com}}^{2-party}$ commitment scheme. For instance, a commitment scheme built using a family of collision-resistant hash functions [1].

[1] E.g. `https://cs.nyu.edu/courses/fall08/G22.3210-001/lect/lecture14.pdf`

Therefore, by considering the result of Lemma 3, all of them can also guarantee sub-hiding and sub-binding. More discussion can be found in [11]. □

5.8. Commitment Schemes with Sub-hiding, Equivocality and binding

In the first positive result, we showed that under a *knowledge assumption* (a non-falsifiable assumption) one can construct a commitment scheme that will guarantee sub-equivocality and binding at the same time. In this section, we show that one can still achieve sub-hiding under standard assumptions (falsifiable assumption) by requiring that there exist hiding, binding, and equivocal commitment schemes.

Pedersen Commitment Scheme Can Achieve Sub-hiding. The Pedersen commitment scheme [106] can guarantee sub-hiding with minimal checking. The committer only needs to run the CKVer algorithm to verify ck before using the key for committing, and the check for this scheme is quite simple. Basically a committer needs to check whether both $g \neq 0$ and $h \neq 0$ before using $ck = (g,h)$.

The following theorem establishes the third positive result, and its complete proof is presented in the manuscript [11].

Theorem 7 (Subversion-Resistant Pedersen Commitment)**.** *The Pedersen commitment scheme with checking $g \neq 0$ and $h \neq 0$, satisfies hiding, equivocal, binding and sub-hiding under the discrete logarithm assumption in $\mathbb{G}$.*

Proof. Proof is presented in [11]. □

5.9. Summary of Results

To sum up, we summarize the results of the section in Tab. 7. Each row considers constructing commitment schemes that simultaneously can achieve the indicated notions (by checkmark, ✓). The last column lists the theorems that established the results.

Table 7. A summary of presented negative and positive results. In each row we refer to simultaneously achieving all selected notions. HD: Hiding, EQ: Equivocality, BD: Binding, Sub-HD: Sub-hiding, Sub-EQ: Sub-Equivocality, Sub-BD: Sub-binding.

	Committer's Security				Verifier's Security		
	HD	EQ	Sub-HD	Sub-EQ	BD	Sub-BD	result in
negative		✓				✓	Thm. 4
positive 1	✓	✓	✓	✓	✓		Thm. 5
positive 2	✓		✓		✓	✓	Thm. 6
positive 3	✓	✓	✓		✓		Thm. 7

6. EFFICIENT ZK-SNARKS FOR UC-SECURE PROTOCOLS

6.1. Motivation

In Section 3 and Section 4, we showed how to decrease the required trust and improve the achievable security in zk-SNARKs to achieve nBB simulation knowledge-soundness (defined in Def. 13). But still, in some practical applications, nBB simulation knowledge-soundness is not enough and one needs to amplify the security of zk-SNARKs even more. For instance, there have been recently some blockchain applications that are intended to achieve UC-security [44] and also use zk-SNARKs in their systems. Privacy-preserving smart contract systems such as Hawk [93] and Gyges [90], or the private Proof-of-Stake (PoS) system like Ouroboros Crypsinous [92] are some known examples.

But as the default security of zk-SNARKs is too weak to achieve UC-security, none of them can be deployed directly in those UC-secure applications. The technical reason is that the security of all efficient zk-SNARKs relies on some knowledge assumptions and non-black-box extraction algorithms that depend on the source code of a particular adversary and there is no universal extractor for all adversaries which is required in the UC framework [44]. In other words, in the UC framework, the UC-simulator should be able to simulate all honest and corrupted parties, with only black-box access to them (either honest or corrupted). Particularly, the simulator should be able to extract witnesses from the corrupted parties without being dependent on their source code.

Already it was shown that to be able to use NIZK arguments in the UC framework, a NIZK argument needs to guarantee black-box simulation knowledge-soundness defined in Def. 14 (a.k.a. black-box simulation extractability) [45, 76, 85]. By considering this, in 2015, Kosba et al. [94] proposed the C∅C∅ framework which allows lifting NIZK arguments that guarantee soundness to a NIZK argument that will guarantee black-box simulation knowledge-soundness, which is sufficient to achieve UC-security. We reviewed a construction of C∅C∅ framework which is used in the mentioned applications in Section 2.4.5. Later, in 2016, Kosba et al. [93] and Juels et al. [90] proposed two UC-secure privacy-preserving smart contract systems called Hawk and Gyges that both use a variation of zk-SNARK used in Zcash [27] that is lifted by C∅C∅ framework. In both works [90, 93], it is shown that efficiency of the whole systems are dominated by the efficiency of the lifted zk-SNARK as lifting zk-SNARKs to achieve black-box simulation extractability leads to proofs linear in the witness size (but still circuit succinct), unlike the common zk-SNARKs that have (witness and circuit) succinct proofs. As we observed in Section 2.4.5, to lift a zk-SNARK using the C∅C∅ framework one would need to construct arithmetic circuits for a commitment scheme, a PRF and a public-key encryption scheme.

So, it is obvious that constructing simpler and more efficient zk-SNARKs that

can guarantee black-box simulation extractability can help us to simplify the construction of UC-secure protocols that need to use zk-SNARKs (e.g., the UC-secure protocols in [90, 92, 93]) and also improve their efficiency in some cases.

6.2. Problem Statement

Following the motivation discussed above, can we construct zk-SNARKs that have simpler constructions and are more efficient than the one that is used in UC-secure protocols like Hawk [93] and Gyges [90]? From a different perspective, as recently there is an increasing number of UC-secure protocols that aim to use zk-SNARKs as a building block, especially in the block-chain applications [92], can we construct black-box simulation extractable zk-SNARKs with simpler constructions and better efficiency?

6.3. A New Approach to Construct Black-Box SE zk-SNARKs

As discussed before, the main goal of the C∅C∅ framework is to use some practically efficient primitives and construct NIZK arguments that can guarantee black-box simulation extractability, which allows to use them in UC-secure protocols and applications.

By considering recent progress in the construction of zk-SNARKs, we revisited their solution by having the same goal in mind. We observe that recent progresses and constructions proposed for zk-SNARKs allows one to achieve the same goal with simpler constructions and fewer extra primitives that sometimes can even lead to more efficient NIZKs or zk-SNARKs.

In the rest, we review our results and the used technique, but a full detailed description of the results can be found in the full version of the paper [9] which is available in [10].

6.3.1. Our Technique and Construction

Recall that given a sound NIZK argument Π_{NIZK} with PPT algorithms $(\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{CV}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for language $\mathscr{L}$ and the corresponding NP relation $\mathbf{R}_{\mathscr{L}}$, the strongest construction of C∅C∅ framework defines a new language $\mathscr{L}''$ such that $((\mathsf{x}, c, \mu, \mathsf{vk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Enc}}, \rho), (r, r_0, \mathsf{w}, s_0)) \in \mathbf{R}_{\mathscr{L}''}$ iff:

$$c = \mathsf{Enc}(\mathsf{pk}_{\mathsf{Enc}}, \mathsf{w}; r) \wedge \big((\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathscr{L}} \vee (\mu = f_{s_0}(\mathsf{vk}_{\mathsf{Sig}}) \wedge \rho = \mathsf{Com}(s_0; r_0))\big),$$

where $\{f_s : \{0,1\}^* \to \{0,1\}^{\lambda}\}_{s \in \{0,1\}^{\lambda}}$ is a PRF family; Com is a perfectly binding commitment scheme and Enc is a semantically secure public-key encryption scheme (namely, IND-CPA secure). By considering recent developments on the construction of zk-SNARKs, we show that using currently available NIZK arguments that guarantee *nBB* simulation (knowledge) soundness (defined in Def. 13), e.g., [82, 99], we can construct a *BB simulation extractable* NIZK argument by

CRS generator, $\mathsf{crs}' \leftarrow \mathsf{KGen}'_{\mathsf{NIZK}}(1^\lambda, \mathbf{R}_{\mathscr{L}})$**:** Given the security parameter λ and description of language $\mathscr{L}$, sample $(\mathsf{crs} \parallel \mathsf{tc}) \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda, \mathbf{R}_{\mathscr{L}''})$; $(\mathsf{pk}_{\mathsf{Enc}}, \mathsf{sk}_{\mathsf{Enc}}) \leftarrow \mathsf{KGen}_{\mathsf{Enc}}(1^\lambda)$; and output $(\mathsf{crs}' \parallel \mathsf{tc}' \parallel \mathsf{tx}') := ((\mathsf{crs}, \mathsf{pk}_{\mathsf{Enc}}) \parallel \mathsf{tc} \parallel \mathsf{sk}_{\mathsf{Enc}})$.

Prover, $\pi' \leftarrow \mathsf{P}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}, \mathsf{x}, \mathsf{w})$**:** Parse $\mathsf{crs}' := (\mathsf{crs}, \mathsf{pk}_{\mathsf{Enc}})$; Abort if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}_{\mathscr{L}}$; sample $r \leftarrow_{\$} \{0,1\}^\lambda$; compute $c = \mathsf{Enc}(\mathsf{pk}_{\mathsf{Enc}}, \mathsf{w}; r)$; execute the prover of input NIZK and generate $\pi \leftarrow \mathsf{P}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, c, \mathsf{pk}_{\mathsf{Enc}}), (r, \mathsf{w}))$; output $\pi' := (c, \pi)$.

Verifier, $0/1 \leftarrow \mathsf{V}'(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}', \mathsf{x}, \pi')$**:** Parse $\mathsf{crs}' := (\mathsf{crs}, \mathsf{pk}_{\mathsf{Enc}})$ and $\pi' := (c, \pi)$; call the verifier of input NIZK and return 0 if $\mathsf{V}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, \mathsf{pk}_{\mathsf{Enc}}, c), \pi) = 0$.

Simulator, $\pi' \leftarrow \mathsf{Sim}'(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}', \mathsf{tc}', \mathsf{x})$**:** Parse $\mathsf{crs}' := (\mathsf{crs}, \mathsf{pk}_{\mathsf{Enc}})$ and $\mathsf{tc}' := \mathsf{tc}$; sample $z, r \leftarrow_{\$} \{0,1\}^\lambda$; compute $c = \mathsf{Enc}(\mathsf{pk}_{\mathsf{Enc}}, z; r)$; execute simulator of the input NIZK argument and generate $\pi \leftarrow \mathsf{Sim}(\mathbf{R}_{\mathscr{L}''}, \mathsf{crs}, (\mathsf{x}, c, \mathsf{pk}_{\mathsf{Enc}}), \mathsf{tc})$; and output a simulated proof $\pi' := (c, \pi)$.

Extractor, $\mathsf{w} \leftarrow \mathsf{Ext}'(\mathbf{R}_{\mathscr{L}}, \mathsf{crs}', \mathsf{tx}', \mathsf{x}, \pi')$**:** Parse $\pi' := (c, \pi)$, $\mathsf{tx}' := \mathsf{sk}_{\mathsf{Enc}}$; extract $\mathsf{w} \leftarrow \mathsf{Dec}(\mathsf{sk}_{\mathsf{Enc}}, c)$; output w.

Figure 19. Lifting a nBB simulation (knowledge) sound NIZK to a BB simulation knowledge-sound NIZK.

adding a linear size commitment and a NIZK proof for a new language which is achieved by embedding an encryption of the witness in the old language. Technically speaking, we show that given a nBB simulation (knowledge) sound NIZK argument with language $\mathscr{L}$ and the corresponding NP relation $\mathbf{R}_{\mathscr{L}}$, we can define a new and simpler language $\mathscr{L}''$ such that $((\mathsf{x}, c, \mathsf{pk}_{\mathsf{Enc}}), (r, \mathsf{w})) \in \mathbf{R}_{\mathscr{L}''}$ iff:

$$(c = \mathsf{Enc}(\mathsf{pk}_{\mathsf{Enc}}, \mathsf{w}; r)) \wedge ((\mathsf{x}, \mathsf{w}) \in \mathbf{R}_{\mathscr{L}}),$$

where $(\mathsf{KGen}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ is a set of algorithms for a semantically secure encryption scheme with keys $(\mathsf{pk}, \mathsf{sk})$.

Accordingly, a nBB simulation (knowledge) sound NIZK argument system Π_{NIZK} for $\mathscr{R}$ constructed from PPT algorithms $(\mathsf{KGen}_{\mathsf{NIZK}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ can be lifted to BB simulation knowledge-sound NIZK Π'_{NIZK} with PPT algorithms $(\mathsf{KGen}'_{\mathsf{NIZK}}, \mathsf{P}', \mathsf{V}', \mathsf{Sim}', \mathsf{Ext}')$ described in Fig. 19.

Regard to the new constructions, shown in Fig. 19, we note that in the simplified language $\mathscr{L}''$, all verifications will be done inside the verification of the original NIZK argument, but prover will generate some new public outputs (ciphertexts) in the extended circuit which increase the size of communication (statement) to linear in the witness size but still succinct in the circuit size. The definition of a new language $\mathscr{L}''$ enforces a prover P to encrypt its witness with a public key given in the CRS and send the ciphertext along with the proof. In this scenario, in the security proof of BB simulation extractability, the secret key of the

encryption scheme is given to the Ext which allows extracting witnesses in a BB manner. This is an already known technique to achieve BB extraction that is used in the C∅C∅ framework as well. The intuition behind our simplifications while defining $\mathscr{L}''$ is that the input NIZK already guarantees simulation (knowledge) soundness and one does not need to use PRF and commitment schemes that are used in C∅C∅ to achieve non-malleability in the proofs.

The following theorem is proven in [9], which shows that given a NIZK argument that guarantees completeness, ZK, and simulation (knowledge) soundness, the described construction in Fig. 19 results in a construction that achieves completeness, ZK and *black-box* simulation knowledge-soundness (a.k.a. black-box simulation extractability).

Theorem 8 (Completeness, ZK, Black-Box Simulation Extractability)**.** *Assuming the encryption scheme Π_{Enc} is semantically secure and perfectly correct, and the input NIZK argument Π_{NIZK} guarantees (non-black box) simulation (knowledge) soundness, the NIZK argument Π'_{NIZK} constructed in Figure 19, satisfies completeness, zero-knowledge and black-box simulation extractability.*

Proof. The proof is given in [9] that is included in the thesis. □

6.3.2. Efficiency of New Constructions

Proof size and Prover: As can be seen in Fig. 19, the proof size of new constructions will be equal to the proof size of the input NIZK argument plus the size of the ciphertext c which gives us proofs linear in the witness size (but succinct in circuit size). However, the prover needs to generate a proof for an arithmetic circuit which encodes the new language $\mathscr{L}''$ and has a larger number of gates.

Remark 2. *Note that the proposed construction also allows one to build a commit-and-proof system. In that case, c can be considered as the commitment, as it can be sent in advance. But the rest of proof will be the proof of opening and will be succinct.*

Verifier: As in the new constructions, the verifier is unchanged, so the verification of the new constructions will be the same as the input NIZK but with a larger statement. By considering verification of current paring-based nBB simulation extractable zk-SNARKs [82, 99], verification of the new constructions will be dominated by m_0 exponentiations, where m_0 is the length of the statement.

Setup phase: By considering the simplified language $\mathscr{L}''$, in new constructions (shown in Fig. 19), the setup phase will be done for a new arithmetic circuit which is extended with a sub-circuit for encrypting the witness.

6.3.3. Two Black-Box SE zk-SNARKs

Next, we instantiate the proposed construction in Fig. 19 with two instances and construct two NIZK arguments (precisely zk-SNARKs) that will guarantee black-

box simulation knowledge-soundness. Recall that the proposed construction takes a NIZK argument that can achieve nBB simulation (knowledge) soundness, defined in Def. 13.

First instantiation. In 2017, Groth and Maller [82] proposed the first SAP-based zk-SNARK in the CRS model that can achieve nBB simulation (knowledge) soundness and consequently guarantees non-malleability of the proofs. The scheme is constructed using asymmetric bilinear groups and works for Square Arithmetic Programs; this implies encoding the target language $\mathscr{L}''$ to an arithmetic circuit which has only squaring and addition gates. In their scheme the proof is 2 elements in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$.

Second instantiation. Recently Lipmaa [99] proposed several zk-SNARKs based on different NP characterizations QAPs, SAPs, Quadratic Span Programs (QSPs) and Square Span Programs (SSPs) that all guarantee nBB simulation knowledge-soundness. All his proposed constructions are instances of our construction, but here we focus on his QAP-based construction, as the QAP-based and SAP-based constructions are more convenient when one works with arithmetic circuits.

Tab. 8 compares the efficiency of two instances that guarantee nBB simulation knowledge-soundness from different perspectives. Lifting either of them will result in a zk-SNARK (which are only circuit succinct) that guarantees BB simulation knowledge-soundness, and consequently sufficient for UC-secure protocols [45, 76, 85]. As we observed in Tab. 4, in practice QAP-based zk-SNARKs are more efficient than the SAP-based zk-SNARKs, as the latter works with arithmetic circuits with squaring gates that doubles the number of multiplication gates (by considering $ab = ((a+b)^2 - (a-b)^2)/4$). Due to this fact, we expect that instantiating the construction with Lipmaa's QAP-based zk-SNARK [99] will result in a more efficient black-box simulation knowledge-sound zk-SNARK in comparison with one instantiated with the Groth-Maller scheme [82].

Table 8. A comparison of Groth-Maller [82] and Lipmaa's [99] (QAP-based) zk-SNARKs for arithmetic circuit satisfiability with m_0 element instance (input values), m wires, n multiplication gates. As Groth-Maller [82] is constructed for SAPs, so n multiplication gates translate to $2n$ squaring gates in comparison with QAP-based schemes. $\mathbb{G}_1$ and $\mathbb{G}_2$: group elements, E: exponentiations, P: pairings and # VE: number of verification equations.

SNARK	CRS Size	Proof	Prover	Verifier	# VE
GM [82]	$2m+4n+5$ $\mathbb{G}_1$	2 $\mathbb{G}_1$	$2m+4n-m_0$ E_1	m_0 E_1	2
For SAPs	$2n+3$ $\mathbb{G}_2$	1 $\mathbb{G}_2$	$2n$ E_2	5 P	
Lip [99]	$m+3n+5$ $\mathbb{G}_1$	3 $\mathbb{G}_1$	$m+4n-m_0$ E_1	m_0 +1 E_1	2
For QAPs	$n+4$ $\mathbb{G}_2$	1 $\mathbb{G}_2$	n E_2	5 P	

6.4. On the Efficiency of Privacy Preserving Smart Contracts

Two privacy-preserving smart contract systems Hawk [93] and Gyges [90] frequently use a zk-SNARK to prove different statements in each smart contract. As both systems are intended to achieve UC-security, they both use a particular zk-SNARK which is obtained by applying the C∅C∅ framework on a variation of the Pinocchio [105] non-UC-secure zk-SNARK that is proposed in [31]. In Hawk (and similarly Gyges) the authors discuss that the efficiency of their system is dominated by the efficiency of the underlying lifted zk-SNARK. Next, we discuss how the zk-SNARK constructed in Section 6.3.3 can improve efficiency of both smart contract systems. We focus precisely on Hawk, but as Gyges have also used the C∅C∅ framework and the same zk-SNARK, the discussion also holds for Gyges.

As mentioned before, the designers of Hawk (and similarly Gyges) lifted Ben Sasson et al.'s zk-SNARK [31] using the C∅C∅ framework to achieve black-box simulation knowledge-soundness. As shown in Section 2.4.5, using the C∅C∅ framework requires a more extended language $\mathscr{L}''$, consequently more changes on the input NIZK or zk-SNARK. On the other hand, as we showed in Section 6.3.1 and Fig. 19, our extended language $\mathscr{L}''$ requires fewer changes to achieve the same goal. We should mention that the reason is that we use NIZKs (particularity zk-SNARKs) that already guarantee simulation (knowledge) soundness. In fact, our changes are a subset of the changes required by the C∅C∅ framework. So in the case of using zk-SNARKs that are constructed for the same NP characterization (e.g., QAPs), the overhead in our case would be smaller than the case where one uses the C∅C∅ framework. In fact, if one use a QAP-based (nBB simulation extractable) zk-SNARK (for instance [99]) which has better efficiency than the QAP-based zk-SNARK used in Hawk and Gyges, our construction can simplify their protocol and improve their practical efficiency.

Tab. 9 compares the asymptotic and practical performance of the zk-SNARK used in HAWK and Gyges [31] with Groth and Maller's [82] and Lipmaa's [99] zk-SNARKs (before applying any changes). Empirical performance of [31] and [82] based on their implementations in the libsnark library. The experiments are done on a machine equipped with 3.40 GHz Intel Core i7-4770 CPU, in single-threaded mode, using the BN128 curve. The performance of Lipmaa's QAP-based scheme is estimated based on similar existing QAP-based (e.g., Groth's scheme [79]) implementations on the same machine[1].

Tab. 9 shows that both Lipmaa's and the GM zk-SNARK outperform Ben Sasson et al.'s zk-SNARK in all metrics. Beside faster running times in all algorithms, Lipmaa's and GM zk-SNARKs has only 2 verification equations, instead of 5 in [31]. As already mentioned in Section 6.3.3, Lipmaa's scheme [99] is con-

[1]Based on reported implementations on `https://github.com/scipr-lab/libsnark/tree/master/libsnark/zk_proof_systems/ppzksnark`

Table 9. A comparison of Ben Sasson et al.'s [31] (BCTV), Groth-Maller [82] (GM) and Lipmaa's [99] (Lip) zk-SNARKs for arithmetic circuit satisfiability with m_0 element instance, m wires, n multiplication gates. $\mathbb{G}_1$ and $\mathbb{G}_2$: group elements, E: exponentiations, P: pairings, B: byte, sec: second, ms: millisecond. Implementations are reported for an R1CS instance with $n = 10^6$ constraints (input values) and $m = 10^6$ variables, of which $m_0 = 10$ are input variables. Practical performance of [99] is estimated based on asymptotic performance and current similar implementations in libsnark library.

SNARK	CRS	Proof	Prover	Verifier	#VE
BCTV [31]	$6m+n-m_0$ $\mathbb{G}_1$	7 $\mathbb{G}_1$	$6m+n-m_0$ E_1	m_0 E_1	5
For QAPs	m $\mathbb{G}_2$	1 $\mathbb{G}_2$	m E_2	12 P	
in libsnark	104.8 sec	287 B	128.6 sec	4.2 ms	—
GM [82]	$2m+4n+5$ $\mathbb{G}_1$	2 $\mathbb{G}_1$	$2m+4n-m_0$ E_1	m_0 E_1	2
For SAPs	$2n+3$ $\mathbb{G}_2$	1 $\mathbb{G}_2$	$2n$ E_2	5 P	
in libsnark	100.4 sec	127 B	116.4 sec	2.3 ms	—
Lip [99]	$m+3n+5$ $\mathbb{G}_1$	3 $\mathbb{G}_1$	$m+4n-m_0$ E_1	m_0 +1 E_1	2
For QAPs	$n+4$ $\mathbb{G}_2$	1 $\mathbb{G}_2$	n E_2	5 P	
Estimation	$\approx$ 82 sec	160 B	$\approx$ 94 sec	$\approx$ 2.3 ms	—

structed for QAPs and appending a new sub-circuit for a particular computation to Groth-Maller zk-SNARK is more costly than appending a sub-circuit for the same computation to Lipmaa's scheme. So, by considering efficiency report in Tab. 9, and the fact that our changes (Shown in Fig. 19) are a subset of the changes applied on Ben Sasson et al.'s zk-SNARK before deployment in the Hawk system, one can observe that a lifted version of Lipmaa's zk-SNARK can simplify protocols of both smart contract systems and also improve their efficiency.

The designers of both systems (especially Hawk) proposed various effective optimizations to maximize the efficiency of underlying lifted zk-SNARKs (Section V in [93]). The same techniques can work with the new constructions. For instance, it is shown that in the *Finalize* operation of a smart contract in Hawk, one may use non-UC-secure zk-SNARK, whereas similarly in the new case one can use non-UC-secure version of Lipmaa's QAP-based [99] or GM SAP-based [82] zk-SNARKs that are more efficient than the one that currently used (as can be seen in Tab. 9) and additionally they ensure nBB simulation extractability.

7. CONCLUSIONS AND FUTURE WORK

7.1. Conclusions

In this thesis, we focused on two following research questions related to zk-SNARKs and commitment schemes. (i) How much can we reduce the required trust in the setup phase of zk-SNARKs and equivocal commitment schemes in the CRS model? (ii) How to efficiently improve the security of zk-SNARKs in different (either UC-secure or non-UC-secure) practical applications?

To answer the first question, we first in Chapter 3 constructed a subversion-resistant SNARK that guarantees Subversion ZK, ZK without trusting the third party, and knowledge-soundness. The result showed that we can construct zk-SNARKs where the prover does not need to trust the CRS generators to achieve ZK, but as before the verifier needs to trust the CRS generators to achieve knowledge-soundness. Then, in Section 4.4, we showed that we can lift our presented construction in Chapter 3 to achieve simulation knowledge-soundness while keeping Sub-ZK. Roughly speaking, this showed that similar to previous constructions the prover can achieve ZK without trusting the CRS generators, while the protocol will also guarantee simulation (knowledge) soundness, which allows to have non-malleable proofs. In Chapter 5, we showed that with similar techniques one can construct subversion-resistant equivocal commitment schemes that require less trust than before. Indeed, we showed that one can construct equivocal commitment schemes where the committer will achieve equivocality without trusting the key generators, while the scheme will guarantee binding as before, under a trusted setup. Such constructions will allow mitigating the trust in the bigger cryptographic protocols that aim to use an equivocal commitment scheme, e.g., [96].

In order to answer the second question, in Section 4.3, we showed that the folklore OR technique [24] along with a proper instantiation allows us to lift a zk-SNARK with nBB knowledge-sound to zk-SNARKs with nBB *simulation* knowledge-soundness efficiently. As an instance, we presented a variation of Groth's QAP-based zk-SNARK that can achieve nBB *simulation* extractability and considerably outperforms Groth and Maller's [82] SAP-based zk-SNARK, which similarly guarantees nBB simulation extractability. Recall that compared to (nBB) knowledge-soundness, the notion of (nBB) simulation extractability additionally guarantees that the proofs are non-malleable, which is a necessary requirement in practical applications but still is not sufficient for applications that aim to achieve UC-security.

In Chapter 6, we observed that considering recent developments in building zk-SNARKs, namely due to existence of nBB simulation extractable zk-SNARKs (e.g. [82]), one can simplify the construction of the C∅C∅ framework and more simply build NIZK arguments that will achieve BB simulation extractability. We showed that our proposed technique allows one to simplify the construction and improve the efficiency of privacy-preserving smart contract systems such as Hawk

and Gyges [90, 93].

7.2. Future Directions and Ongoing works

Next, we discuss some open and ongoing research topics that can be considered as future directions for our studied topics.

7.2.1. Beyond Subversion-Resistance in the CRS model

In this study, we showed that one can construct SNARKs that can achieve Sub-ZK and simulation knowledge-soundness (Chapters 3 and Section 4.4). From the trust point of view, by considering the setting where the CRS is generated by an MPC protocol [2, 29], in new constructions the prover does not need to trust any of CRS generators but the verifier needs to trust only 1 out of n parties in the MPC protocol. After this result, an interesting question arose: can the the *single* party among the n parties that the verifier needs to trust, be the verifier himself? In other words, can the verifier join the MPC protocol whenever he/she likes, such that after joining the CRS generation MPC protocol, similar to the prover, the verifier also will not need to trust any of the CRS generators? In 2018, this question was answered positively by Groth et al. [80], by presenting a zk-SNARK with universal and updatable CRS that allows the verifier to update the CRS and avoid trusting the third party. Following this result there have been some efficient zk-SNARKs with updatable CRS [5, 16, 80, 100].

Following the above constructions for NIZK arguments with updatable parameters, an interesting question is to extend the idea to different NIZK arguments (e.g. Quasi-Adaptive NIZK arguments) or other cryptographical primitives. Particularly, by considering our presented subversion-resistant commitment scheme in Section 5.6, a possible research question is can we construct commitment schemes with update commitment keys? Such that the verifier will be able to update the commitment keys instead of trusting the key generators.

7.2.2. More Efficient zk-SNARKs for UC-Protocols

In Chapter 6, we focused on constructing zk-SNARKs with black-box simulation knowledge-soundness which is shown to be a sufficient property to achieve UC-security in NIZK arguments, consequently sufficient to use NIZKs in UC-secure protocols. The key idea behind those constructions was that one can use the *simulation* property of nBB simulation (knowledge) sound NIZK arguments to simplify constructing black-box simulation extractable NIZK arguments that the C∅C∅ framework constructs. We observed that depending on the underlying language (e.g., QAP-based constructions are more efficient than the SAP-based ones), new constructions can be more efficient.

By the above observation, constructing more efficient zk-SNARKs that guarantee nBB simulation knowledge-soundness will allow constructing more effi-

cient zk-SNARKs with black-box simulation extractability.

From a different perspective, as zk-SNARKs are constructed for QAPs, SAPs, QSPs or SSPs characterizations, their size (order) depends on the number of multiplication gates of the circuit that encodes the language. Therefore, constructing circuits that require a smaller number of multiplication gates to encode the language can directly improve the efficiency of the constructions. For instance, to compute SHA-256 the smart contract system Hawk [93] uses an arithmetic circuit which has around 25500 multiplication gates, while the digital coin Zcash [27] uses an arithmetic circuit that has around 28000 multiplication gates. Therefore, using the first circuit leads to have a better efficiency in CRS size and prover's computations.

7.2.3. UC-Secure Parameter Generation for BB SE zk-SNARKs

In Chapter 6, we presented two constructions that achieve ZK and BB simulation knowledge-soundness, which can be deployed in UC-secure protocols. In those constructions, both the prover and verifier need to trust the CRS generators. To mitigate the trust in such constructions, one can use MPC protocols for CRS generation which will distribute the trust to several parties. This can be considered an interesting research question to mitigate trust in those systems. More clearly, it would be interesting to consider how one can construct a UC-secure MPC protocol similar to the one proposed in [2], but for either of the presented constructions in Section 6.3.3. Recently, in [16], we showed how to build such constructions with updatable parameters which allows the parties to bypass the need for a trusted third party. But still such constructions cannot achieve UC-security in the setup phase. So, it would be interesting to consider if one can achieve UC-security in the setup of such constructions.

Finally, an important research direction for all the studied topics is to consider if similar protocols can be constructed based on cryptographic assumptions that are secure against quantum adversaries. There are some valuable attempts in this direction [40, 66, 104], but still they are not deployable in most of the applications where zk-SNARKs are currently deployed in.

BIBLIOGRAPHY

[1] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michal Zajac. DL-extractable UC-commitment schemes. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 385–405. Springer, Heidelberg, June 2019.

[2] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michal Zajac. UC-secure CRS generation for SNARKs. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 99–117. Springer, Heidelberg, July 2019.

[3] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III, the full version is avalable on* `https://eprint.iacr.org/2017/599`, pages 3–33, 2017.

[4] Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michal Zajac. On QA-NIZK in the BPK model. *IACR Cryptology ePrint Archive*, 2018:877, 2018.

[5] Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. Technical report, Cryptology ePrint Archive, Report 2020/062, 2019. http://eprint. iacr. org/2020/062, 2020.

[6] Shahla Atapoor and Karim Baghery. Simulation extractability in Groth's zk-SNARK. In Cristina Perez-Sola, Guillermo Navarro-Arribas, Alex Biryukov, and Joaquin Garcia-Alfaro, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26-27, 2019, Proceedings*, volume 11737 of *Lecture Notes in Computer Science*, pages 336–354. Springer, 2019.

[7] Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signature schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 364–375. ACM Press, October 2015.

[8] Benedikt Auerbach, Mihir Bellare, and Eike Kiltz. Public-key encryption resistant to parameter subversion and its realization from efficiently-embeddable groups. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 348–377. Springer, Heidelberg, March 2018.

[9] Karim Baghery. On the efficiency of privacy-preserving smart contract systems. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 118–136. Springer, Heidelberg, July 2019.

[10] Karim Baghery. On the efficiency of privacy-preserving smart contract systems. Cryptology ePrint Archive, Report 2019/480, 2019. `https://eprint.iacr.org/2019/480`.

[11] Karim Baghery. Subversion-resistant commitment schemes: Definitions and constructions. Cryptology ePrint Archive, Report 2019/1065, 2019. `http://eprint.iacr.org/2019/1065`.

[12] Karim Baghery. Subversion-resistant simulation (knowledge) sound NIZKs. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, pages 42–63. Springer, Heidelberg, December 2019.

[13] Karim Baghery and Behzad Abdolmaleki. An AKARI-based secure communication scheme for EPC tags. In *2017 Advances in Wireless and Optical Communications (RTUWO)*, pages 208–213, Nov 2017.

[14] Karim Baghery, Behzad Abdolmaleki, Shahram Khazaei, and Mohammad Reza Aref. Breaking anonymity of some recent lightweight RFID authentication protocols. *Wireless Networks*, 25(3):1235–1252, 2019.

[15] Karim Baghery, Alonso González, Zaira Pindado, and Carla Ràfols. Signatures of knowledge for boolean circuits under standard assumptions. In Ashraf Badaw, Abderrahmane Nitaj, and Amr Youssef, editors, *AFRICACRYPT*, LNCS. Springer, 2020.

[16] Karim Baghery and Mahdi Sedaghat. TIRAMISU: Black-box simulation extractable NIZKs in the updatable CRS model. Cryptology ePrint Archive, Report 2020/474, 2020. `http://eprint.iacr.org/2020/474`.

[17] Razvan Barbulescu and Sylvain Duquesne. Updating Key Size Estimations for Pairings. Technical Report 2017/334, IACR, April 14, 2017. Available from `http://eprint.iacr.org/2017/334`, revision from April 26, 2017.

[18] Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. Cryptology ePrint Archive, Report 2017/334, 2017. `http://eprint.iacr.org/2017/334`.

[19] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 171–188. Springer, Heidelberg, May 2004.

[20] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee

Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016.

[21] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an Untrusted CRS: Security in the Face of Parameter Subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016 (2)*, volume 10032 of *LNCS*, pages 777–804, Hanoi, Vietnam, December 4–8, 2016. Springer, Cham.

[22] Mihir Bellare, Juan A. Garay, and Tal Rabin. Batch Verification with Applications to Cryptography and Checking. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN 1998*, volume 1380 of *LNCS*, pages 170–191, Campinas, Brazil, April 20–24, 1998. Springer, Heidelberg.

[23] Mihir Bellare, Juan A. Garay, and Tal Rabin. Batch verification with applications to cryptography and checking. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN 1998*, volume 1380 of *LNCS*, pages 170–191. Springer, Heidelberg, April 1998.

[24] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211. Springer, Heidelberg, August 1990.

[25] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 1–19. Springer, Heidelberg, August 2014.

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

[27] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[28] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In Ran Canetti and Juan Garay, editors, *CRYPTO (2) 2013*, volume 8043 of *LNCS*, pages 90–108, Santa Barbara, California, USA, August 18–22, 2013. Springer, Heidelberg.

[29] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE Computer Society Press, May 2015.

[30] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure Sampling of Public Parameters for Succinct Zero

Knowledge Proofs. In *IEEE SP 2015*, pages 287–304, San Jose, CA, USA, May 17–21, 2015. IEEE Computer Society.

[31] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive arguments for a von neumann architecture. Cryptology ePrint Archive, Report 2013/879, 2013. `http://eprint.iacr.org/2013/879`.

[32] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. In Kevin Fu and Jaeyeon Jung, editors, *USENIX 2014*, pages 781–796, San Jose, CA, USA, August 20–22, 2014. USENIX Association.

[33] David Bernhard, Georg Fuchsbauer, and Essam Ghadafi. Efficient signatures of knowledge and DAA in the standard model. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 518–533. Springer, Heidelberg, June 2013.

[34] Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *CRYPTO'81*, volume ECE Report 82-04, pages 11–15. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981.

[35] Manuel Blum, Paul Feldman, and Silvio Micali. Non-Interactive Zero-Knowledge and Its Applications. In *STOC 1988*, pages 103–112, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.

[36] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

[37] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd FOCS*, pages 112–117. IEEE Computer Society Press, November 1982.

[38] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[39] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

[40] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 247–277. Springer, Heidelberg, April / May 2017.

[41] Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK.

Technical Report 2017/602, IACR, June 21, 2017. http://eprint.iacr.org/2017/602, last accessed version from 25 Jun 2017. Accepted to BITCOIN 2018.

[42] Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *FC 2018 Workshops*, volume 10958 of *LNCS*, pages 64–77. Springer, Heidelberg, March 2019.

[43] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-Knowledge Contingent Payments Revisited: Attacks and Payments for Services. Technical Report 2017/566, IACR, June 9, 2017. Available from http://eprint.iacr.org/2017/566.

[44] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[45] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[46] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, August 2006.

[47] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-Interactive and Non-Malleable Commitment. In Jeffrey Scott Vitter, editor, *STOC 1998*, pages 141–150, Dallas, Texas, USA, May 23–26, 1998.

[48] Ivan Damgård. On the existence of bit commitment schemes and zero-knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 17–27. Springer, Heidelberg, August 1990.

[49] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992.

[50] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 418–430. Springer, Heidelberg, May 2000.

[51] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142. Springer, Heidelberg, December 2002.

[52] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014.

[53] Vanesa Daza, Alonso González, Zaira Pindado, Carla Ràfols, and Javier Silva. Shorter quadratic QA-NIZK proofs. In *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part I*, pages 314–343, 2019.

[54] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001.

[55] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[56] Shimon Even, Oded Goldreich, and Abraham Lempel. A Randomized Protocol for Signing Contracts. *Communications of the ACM*, 28(6):637–647, June 1985.

[57] Prastudy Fauzi, Helger Lipmaa, and Michal Zajac. A shuffle argument secure in the generic model. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 841–872. Springer, Heidelberg, December 2016.

[58] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. *IACR Cryptology ePrint Archive*, 2018:990, 2018.

[59] Marc Fischlin and Roger Fischlin. Efficient non-malleable commitment schemes. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 413–431. Springer, Heidelberg, August 2000.

[60] Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018.

[61] Georg Fuchsbauer. WI is not enough: Zero-knowledge contingent (service) payments revisited. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 49–62. ACM Press, November 2019.

[62] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

[63] Georg Fuchsbauer and Michele Orrù. Non-interactive zaps of knowledge. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 44–62. Springer, Heidelberg, July 2018.

[64] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Bringing people of different beliefs together to do UC. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 311–328. Springer, Heidelberg, March 2011.

[65] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

[66] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-SNARKs from square span programs. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 556–573. ACM Press, October 2018.

[67] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions (extended abstract). In G. R. Blakley and David Chaum, editors, *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *LNCS*, pages 276–288, Santa Barbara, California, USA, August 19–22, 1984. Springer, Heidelberg, 1985.

[68] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC 1987*, pages 218–229, New York City, 25–27 May 1987.

[69] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, 38(3):691–729, 1991.

[70] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[71] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[72] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[73] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[74] Shafi Goldwasser, Silvio Micali, and Andrew Chi-Chih Yao. On signatures and authentication. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 211–215. Plenum Press, New York, USA, 1982.

[75] Jens Groth. Non-interactive zero-knowledge arguments for voting. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 467–482. Springer, Heidelberg, June 2005.

[76] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006.

[77] Jens Groth. Homomorphic trapdoor commitments to group elements. Cryptology ePrint Archive, Report 2009/007, 2009. http://eprint.iacr.org/2009/007.

[78] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010.

[79] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

[80] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018.

[81] Jens Groth and Steve Lu. Verifiable shuffle of large size ciphertexts. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 377–392. Springer, Heidelberg, April 2007.

[82] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, August 2017.

[83] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 323–341. Springer, Heidelberg, August 2007.

[84] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, August 2006.

[85] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006.

[86] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[87] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 123–128. Springer, Heidelberg, May 1988.

[88] Rolf Haenni. Swiss post public intrusion test: Undetectable attack againstvote integrity and secrecy https://e-voting.bfh.ch/app/download/7833162361/PIT2.pdf?t=1552395691. 2019.

[89] Gottfried Herold, Max Hoffmann, Michael Klooß, Carla Ràfols, and Andy Rupp. New techniques for structural batch verification in bilinear groups with applications to groth-sahai proofs. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1547–1564. ACM Press, October / November 2017.

[90] Ari Juels, Ahmed E. Kosba, and Elaine Shi. The ring of Gyges: Investigating the future of criminal smart contracts. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 283–295. ACM Press, October 2016.

[91] Jonathan Katz, Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Distributing the setup in universally composable multi-party computation. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM PODC*, pages 20–29. ACM, July 2014.

[92] Thomas Kerber, Aggelos Kiayias, Markulf Kohlweiss, and Vassilis Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *2019 IEEE Symposium on Security and Privacy*, pages 157–174. IEEE Computer Society Press, May 2019.

[93] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858. IEEE Computer Society Press, May 2016.

[94] Ahmed E. Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, T.-H. Hubert Chan, Charalampos Papamanthou, Rafael Pass, Abhi Shelat, and Elaine Shi. C∅C∅: A Framework for Building Composable Zero-Knowledge Proofs. Technical Report 2015/1093, November 10, 2015. `http://eprint.iacr.org/2015/1093`, last accessed version from 9 Apr 2017.

[95] Sarah Jamie Lewis, Olivier Pereira, and Vanessa Teague. Trapdoor commitments in the swisspost e-voting shuffle proof, `https://people.eng.unimelb.edu.au/vjteague/SwissVote`. 2019.

[96] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012.

[97] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, December 2013.

[98] Helger Lipmaa. Prover-Efficient Commit-And-Prove Zero-Knowledge SNARKs. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 2016*, volume 9646 of *LNCS*, pages 185–206, Fes, Morocco, April 13–15, 2016. Springer, Heidelberg.

[99] Helger Lipmaa. Simulation-extractable SNARKs revisited. Cryptology ePrint Archive, Report 2019/612, 2019. `http://eprint.iacr.org/2019/612`.

[100] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

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

[102] Izaak Meckler and Evan Shapiro. Coda: Decentralized cryptocurrency at scale. 2018.

[103] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003.

[104] Anca Nitulescu. Lattice-based zero-knowledge SNARGs for arithmetic circuits. In Peter Schwabe and Nicolas Thériault, editors, *LATINCRYPT 2019*, volume 11774 of *LNCS*, pages 217–236. Springer, Heidelberg, 2019.

[105] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

[106] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

[107] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Heidelberg, August 1992.

[108] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

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

[110] Douglas Wikström. A commitment-consistent proof of a shuffle. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 09*, volume 5594 of *LNCS*, pages 407–421. Springer, Heidelberg, July 2009.

ACKNOWLEDGEMENT

First of all, I would like to thank my supervisor Helger Lipmaa, for the opportunities, his supervision and support from the preliminary to the concluding stages which led to writing this dissertation. Helger created a pleasant working atmosphere for us at Tartu and gave me great freedom to work on the topics of my interest. I am also grateful to all my (ex-)advisors and co-authors Mohammad Reza Aref, Nigel Smart, Shahram Khazaei, Carla Ràfols, Michał Zając, Behzad Abdolmaleki, Janno Siim, Shahla Atapoor, Alonso González, Zaira Pindado, Mahdi Sedaghat, and my colleagues Toomas Krips and Prastudy Fauzi for their valuable advice and interesting discussions. I learned a lot from them.

I believe the thesis quality is improved significantly by having excellent comments and feedback given by internal reviewer Sven Laur and external pre-reviewers Georg Fuchsbauer and Markulf Kohlzeiss; I am very thankful to each of them for their valuable time and comments.

Studying at the University of Tartu (UT) has been a pleasure. I had the opportunity to learn from great professors, lecturers and met wonderful friends, particularity Dominique Unruh, Jaak Vilo, Sven Laur, Arnis Paršovs and Liina Tammekänd. I am also thankful to all administrative or academic affairs staff for making a lot of things easier for me, specially Anneli Vainumäe, Heisi Kurig, Reili Liiver, Martin Kaljula and Katren Oper. I would like to express my sincere gratitude to Janno Siim for his kind helps during my studies at the University of Tartu, including editing Estonian abstract of the thesis.

Next, I would like to thank Carla Ràfols and Vanesa Daza, who supported my visit from their group in Universitat Pompeu Fabra. We had great discussions there with their group, Zaira Pindado and Alonso González.

I do not think words can describe the gratitude I owe for my parents and family, as I could not be here without their love and support. Above all, I would love to thank my wife Shahla for her love and constant support, for all the late nights and early mornings, and for keeping me sane over the past few years. Thank you for being my best friend. I owe you everything, my love.

Finally, despite my love for cryptography, the research done in this thesis would not have been possible without financial supports. Throughout my studies at UT, I was funded or employed by the Estonian Research Council under research grant IUT2-1, the EU's Horizon 2020 research and innovation program under grant agreement No 780477 (project PRIViLEDGE), the Estonian Research Council grant (PRG49), and the Estonian ICT Doctoral School.

SUMMARY IN ESTONIAN

Usalduse vähendamine ja turvalisuse parandamine zk-SNARK-ides ja kinnitusskeemides

Üks olulisi tööriistu tänapäevases krüptograafias on nullteadmustõestus, mis võimaldab osapoolel tõestada väite paikapidavust ilma, et tema salajane info lekiks. Tänu nendele suurepärastele omadustele on mitteinteraktiivsed nullteadmustõestused võimaldanud paljusi uudseid rakendusi: kontrollitavad arvutussüsteemid nagu Pinocchio, privaatsust säilitavad krüptovaluutad nagu Zcash, privaatsust säilitavad nutilepingusüsteemid Hawk ja Gyges ning privaatne panusetõendus süsteem Ouroboros Crypsinous on mõned praktilistest rakendustest, mis kasutavad tõhusaid nullteadmustõestusi, mida nimetatakse zk-SNARK-ideks. zk-SNARK-id on tõhusad ja praktilised mitteinteraktiivsed tõestussüsteemid, mis on konstrueeritud viitestringi mudelis ning tänu *kompaktsetele* tõestustele ja väga tõhusale verifitseeritavusele on need laialdaselt kasutusele võetud suuremahulistes praktilistes rakendustes. Viitestringi mudelis tugineb zk-SNARK-ide konstruktsioon seadistusfaasile ja kasutajad peavad usaldama seadistusfaasi väljundit, mille peaks arvutama usaldatud kolmas osapool või hajutatud süsteem. Lisaks on teada, et osades rakendustes zk-SNARK-de tavaline turvalisuses ei ole piisav otseseks kasutamiseks protokollides, mille eesmärkiks on saavutada mõni tugevam turvalisuse vorm nagu näiteks tõestuste mitte deformeeritavus või veelgi tugevam täielik koosluskindlus (TK). Selles töös uurime zk-SNARK-e kahest vaatenurgast: nende *usalduse vähendamine* ja *turvalisuse tugevdamine*. Esimeses suunas uurime kui palju saab vähendada usaldust paaristuspõhiste zk-SNARK-ide puhul ilma nende tõhusust ohverdamata niiviisi, et kasutajad saavad teatud turvataseme ka siis kui seadistusfaas tehti pahatahtlikult või kui avalikustati seadistusfaasi salajane teave. Me pakume välja mõned tõhusad konstruktsioonid, mis suudavad takistada zk-SNARK-i seadistusfaasi ründeid ja mis saavutavad senisest tugevama turvataseme. Näitame ka seda, et sarnased tehnikad võimaldavad leevendada usaldust tagauksega kinnitusskeemides, mis on krüptograafiliste primitiivide veel üks silmapaistev perekond ja mis samuti nõub usaldatud seadistusfaasi. Teises suunas esitame mõned tõhusad konstruktsioonid, mis tagavad parema turvalisuse minimaalsete lisakuludega. Mõned esitatud konstruktsioonidest võimaldavad lihtsustada praegusi TK-turvalisi protokolle. Nimelt privaatsust säilitavate nutilepingusüsteemide Hawk ja Gyges konstruktsiooni ja parandada nende tõhusust. Uusi konstruktsioone saab aga otse kasutada uutes (mitte) TK-turvalistes protokollides, mis soovivad kasutada zk-SNARK-e. Osa väljapakutud zk-SNARK-e on implementeeritud teegis Libsnark ja empiirilised tulemused kinnitavad, et usalduse vähendamiseks või suurema turvalisuse saavutamiseks on arvutuslikud lisakulud väikesed.

PUBLICATIONS

CURRICULUM VITAE

Personal data

Name:	Karim Baghery
Birth:	July 27th, 1988
Citizenship:	Iranian
Languages:	Azerbaijani, English, Persian, Turkish
Contact:	baghery.karim@gmail.com

Education

2016 –	Ph.D. candidate in Computer Science University of Tartu, Estonia.
2011 – 2014	M. Sc. in Electrical Engineering - Communications Shahed University, Iran.
2006 – 2010	B. Sc. in Electrical Engineering - Telecommunications IAU University, Iran.

Employment

09.2016 – 12.2020	University of Tartu, Institute of Computer Science, Junior Research Fellow in Cryptography (0,60)
02.2018 – 08.2018	University of Tartu, Institute of Computer Science, Junior Research Fellow in Cryptography (0,62)
04.2016 – 05.2018	University of Tartu, Institute of Computer Science, Junior Research Fellow in Cryptography (1,00)
06.2013 – 03.2016	Sharif University of Technology, Research Assistant

Scientific work

Main fields of interest:

- Cryptography (Zero-knowledge proofs, UC-security, Authentication)
- Blockchains (Privacy-preserving coins and smart contracts, ...)
- Wireless communications and error correction codes

Recent key scientific talks:

2019: Subversion-Resistant Simulation (Knowledge) Sound NIZKs. IMACC 2019, University of Oxford, Oxford, UK.

2019: Reducing Trust in zk-SNARKs and Efficient zk-SNARKs for UC-secure Applications. Universitat Pompeu Fabra, Barcelona, Spain.

2019: Reducing Trust and Improving Security in zk-SNARKs, COSIC Group, KU Leuven University, Leuven, Belgium.

2019: On the Efficiency of Privacy-Preserving Smart Contract Systems. Joint Estonian-Latvian Theory Days 2019, Parnu, Estonia.

2019: DL-Extractable UC-Commitment Schemes. ACNS 2019, Universidad del Rosario, Bogota, Colombia.

2018: A Subversion-Resistant zk-SNARK. ISSL Lab., Sharif University of Technology, Tehran, Iran.

2018: Introduction to ZK proofs and SNARKs. Sharif Blockchain Lab., Sharif University of Technology, Tehran, Iran.

2018: The Bitcoin Lightning Network. Iran Telcommunication Reseach Center, Tehran, Iran.

2017: An AKARI-based Secure Communication Scheme for EPC Tags. RTUWO 2017, Riga, Latvia.

ELULOOKIRJELDUS

Isikuandmed

Nimi:	Karim Baghery
Sünniaeg ja -koht:	27. juuli 1988, Qoshachay (Miandoab), Iraan
Kodakondsus:	Iraan
Keelteoskus:	aserbaidžaani, inglise, pärsia, türgi
Kontaktandmed:	baghery.karim@gmail.com

Haridus

2016 –	Tartu Ülikool, informaatika doktorant
2011 – 2014	M. Sc. in Electrical Engineering - Communications Shahed University, Iran.
2006 – 2010	B. Sc. in Electrical Engineering - Telecommunications IAU University, Iran.

Teenistuskäik

09.2018 – 12.2020	Tartu Ülikool, loodus- ja täppisteaduste valdkond, arvuti-teaduse instituut, krüptograafia nooremteadur (0,60)
02.2018 – 08.2018	Tartu Ülikool, loodus- ja täppisteaduste valdkond, arvuti-teaduse instituut, krüptograafia nooremteadur (0,62)
04.2016 – 05.2018	Tartu Ülikool, loodus- ja täppisteaduste valdkond, arvuti-teaduse instituut, krüptograafia nooremteadur (1,00)
06.2013 – 03.2016	Sharif University of Technology, teadur

Teadustegevus

Peamised huvialad:

- krüptograafia (nullteadmustõestused, täielik koosluskindlus, autentimine)
- plokiahelad (turvalised krüptorahad ja ja nutilepingud, ...)
- traadita suhtlus ja veaparandus koodid

Viimased teadusettekanded:

2019: Subversion-Resistant Simulation (Knowledge) Sound NIZKs. IMACC 2019, University of Oxford, Oxford, UK.

2019: Reducing Trust in zk-SNARKs and Efficient zk-SNARKs for UC-secure Applications. Universitat Pompeu Fabra, Barcelona, Spain.

2019: Reducing Trust and Improving Security in zk-SNARKs, COSIC Group, KU Leuven University, Leuven, Belgium.

2019: On the Efficiency of Privacy-Preserving Smart Contract Systems. Joint Estonian-Latvian Theory Days 2019, Pärnu, Estonia.

2019: DL-Extractable UC-Commitment Schemes. ACNS 2019, Universidad del Rosario, Bogota, Colombia.

2018: A Subversion-Resistant zk-SNARK. ISSL Lab., Sharif University of Technology, Tehran, Iran.

2018: Introduction to ZK Proofs and SNARKs. Sharif Blockchain Lab., Sharif University of Technology, Tehran, Iran.

2018: The Bitcoin Lightning Network. Iran Telcommunication Reseach Center - ITRC, Tehran, Iran.

2017: An AKARI-based Secure Communication Scheme for EPC Tags. RTUWO 2017, Riga, Latvia.

DISSERTATIONES INFORMATICAE PREVIOUSLY PUBLISHED IN DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS

19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 lk.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
45. **Kristo Heero.** Path planning and learning strategies for mobile robots in dynamic partially unknown environments. Tartu 2006, 123 p.
49. **Härmel Nestra.** Iteratively defined transfinite trace semantics and program slicing with respect to them. Tartu 2006, 116 p.
53. **Marina Issakova.** Solving of linear equations, linear inequalities and systems of linear equations in interactive learning environment. Tartu 2007, 170 p.
55. **Kaarel Kaljurand.** Attempto controlled English as a Semantic Web language. Tartu 2007, 162 p.
56. **Mart Anton.** Mechanical modeling of IPMC actuators at large deformations. Tartu 2008, 123 p.
59. **Reimo Palm.** Numerical Comparison of Regularization Algorithms for Solving Ill-Posed Problems. Tartu 2010, 105 p.
61. **Jüri Reimand.** Functional analysis of gene lists, networks and regulatory systems. Tartu 2010, 153 p.
62. **Ahti Peder.** Superpositional Graphs and Finding the Description of Structure by Counting Method. Tartu 2010, 87 p.
64. **Vesal Vojdani.** Static Data Race Analysis of Heap-Manipulating C Programs. Tartu 2010, 137 p.
66. **Mark Fišel.** Optimizing Statistical Machine Translation via Input Modification. Tartu 2011, 104 p.
67. **Margus Niitsoo**. Black-box Oracle Separation Techniques with Applications in Time-stamping. Tartu 2011, 174 p.
71. **Siim Karus.** Maintainability of XML Transformations. Tartu 2011, 142 p.
72. **Margus Treumuth.** A Framework for Asynchronous Dialogue Systems: Concepts, Issues and Design Aspects. Tartu 2011, 95 p.
73. **Dmitri Lepp.** Solving simplification problems in the domain of exponents, monomials and polynomials in interactive learning environment T-algebra. Tartu 2011, 202 p.

74. **Meelis Kull.** Statistical enrichment analysis in algorithms for studying gene regulation. Tartu 2011, 151 p.
77. **Bingsheng Zhang.** Efficient cryptographic protocols for secure and private remote databases. Tartu 2011, 206 p.
78. **Reina Uba.** Merging business process models. Tartu 2011, 166 p.
79. **Uuno Puus.** Structural performance as a success factor in software development projects – Estonian experience. Tartu 2012, 106 p.
81. **Georg Singer.** Web search engines and complex information needs. Tartu 2012, 218 p.
83. **Dan Bogdanov.** Sharemind: programmable secure computations with practical applications. Tartu 2013, 191 p.
84. **Jevgeni Kabanov.** Towards a more productive Java EE ecosystem. Tartu 2013, 151 p.
87. **Margus Freudenthal.** Simpl: A toolkit for Domain-Specific Language development in enterprise information systems. Tartu, 2013, 151 p.
90. **Raivo Kolde.** Methods for re-using public gene expression data. Tartu, 2014, 121 p.
91. **Vladimir Šor.** Statistical Approach for Memory Leak Detection in Java Applications. Tartu, 2014, 155 p.
92. **Naved Ahmed.** Deriving Security Requirements from Business Process Models. Tartu, 2014, 171 p.
94. **Liina Kamm.** Privacy-preserving statistical analysis using secure multi-party computation. Tartu, 2015, 201 p.
100. **Abel Armas Cervantes.** Diagnosing Behavioral Differences between Business Process Models. Tartu, 2015, 193 p.
101. **Fredrik Milani.** On Sub-Processes, Process Variation and their Interplay: An Integrated Divide-and-Conquer Method for Modeling Business Processes with Variation. Tartu, 2015, 164 p.
102. **Huber Raul Flores Macario.** Service-Oriented and Evidence-aware Mobile Cloud Computing. Tartu, 2015, 163 p.
103. **Tauno Metsalu.** Statistical analysis of multivariate data in bioinformatics. Tartu, 2016, 197 p.
104. **Riivo Talviste.** Applying Secure Multi-party Computation in Practice. Tartu, 2016, 144 p.
108. **Siim Orasmaa.** Explorations of the Problem of Broad-coverage and General Domain Event Analysis: The Estonian Experience. Tartu, 2016, 186 p.
109. **Prastudy Mungkas Fauzi.** Efficient Non-interactive Zero-knowledge Protocols in the CRS Model. Tartu, 2017, 193 p.
110. **Pelle Jakovits.** Adapting Scientific Computing Algorithms to Distributed Computing Frameworks. Tartu, 2017, 168 p.
111. **Anna Leontjeva.** Using Generative Models to Combine Static and Sequential Features for Classification. Tartu, 2017, 167 p.
112. **Mozhgan Pourmoradnasseri.** Some Problems Related to Extensions of Polytopes. Tartu, 2017, 168 p.

113. **Jaak Randmets.** Programming Languages for Secure Multi-party Computation Application Development. Tartu, 2017, 172 p.
114. **Alisa Pankova.** Efficient Multiparty Computation Secure against Covert and Active Adversaries. Tartu, 2017, 316 p.
116. **Toomas Saarsen.** On the Structure and Use of Process Models and Their Interplay. Tartu, 2017, 123 p.
121. **Kristjan Korjus.** Analyzing EEG Data and Improving Data Partitioning for Machine Learning Algorithms. Tartu, 2017, 106 p.
122. **Eno Tõnisson.** Differences between Expected Answers and the Answers Offered by Computer Algebra Systems to School Mathematics Equations. Tartu, 2017, 195 p.

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

1. **Abdullah Makkeh**. Applications of Optimization in Some Complex Systems. Tartu 2018, 179 p.
2. **Riivo Kikas**. Analysis of Issue and Dependency Management in Open-Source Software Projects. Tartu 2018, 115 p.
3. **Ehsan Ebrahimi**. Post-Quantum Security in the Presence of Superposition Queries. Tartu 2018, 200 p.
4. **Ilya Verenich**. Explainable Predictive Monitoring of Temporal Measures of Business Processes. Tartu 2019, 151 p.
5. **Yauhen Yakimenka**. Failure Structures of Message-Passing Algorithms in Erasure Decoding and Compressed Sensing. Tartu 2019, 134 p.
6. **Irene Teinemaa**. Predictive and Prescriptive Monitoring of Business Process Outcomes. Tartu 2019, 196 p.
7. **Mohan Liyanage.** A Framework for Mobile Web of Things. Tartu 2019, 131 p.
8. **Toomas Krips.** Improving performance of secure real-number operations. Tartu 2019, 146 p.
9. **Vijayachitra Modhukur.** Profiling of DNA methylation patterns as biomarkers of human disease. Tartu 2019, 134 p.
10. **Elena Sügis.** Integration Methods for Heterogeneous Biological Data. Tartu 2019, 250 p.
11. **Tõnis Tasa.** Bioinformatics Approaches in Personalised Pharmacotherapy. Tartu 2019, 150 p.
12. **Sulev Reisberg.** Developing Computational Solutions for Personalized Medicine. Tartu 2019, 126 p.
13. **Huishi Yin.** Using a Kano-like Model to Facilitate Open Innovation in Requirements Engineering. Tartu 2019, 129 p.
14. **Faiz Ali Shah.** Extracting Information from App Reviews to Facilitate Software Development Activities. Tartu 2020, 149 p.
15. **Adriano Augusto**. Accurate and Efficient Discovery of Process Models from Event Logs. Tartu 2020, 194 p.