Practical Zero-Knowledge Arguments from Structured Reference Strings

Mary Maller

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

2019

I, Mary Maller, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Zero-knowledge proofs have become an important tool for addressing privacy and scalability concerns in cryptographic protocols. For zero-knowledge proofs used in blockchain applications, it is desirable to have small proof sizes and fast verification. Yet by design, existing constructions with these properties such as zk-SNARKs also have a secret trapdoor embedded in a relation dependent structured reference string (SRS). Knowledge of this trapdoor suffices to break the security of these proofs.

The SRSs required by zero-knowledge proofs are usually constructed with multiparty computation protocols, but the resulting parameters are specific to each individual circuit. In this thesis, we propose a model for constructing zero-knowledge arguments (i.e. zero-knowledge proofs with computational soundness) in which the generation of the SRS is directly considered in the security analysis. In our model the same SRS can be used across multiple applications. Further, the model is updatable i.e. users can update the universal SRS and the SRS is considered secure provided at least one of these users is honest.

We propose two zero-knowledge arguments with updatable and universal SRSs, as well as a third which is neither updatable nor universal, but which through similar techniques achieves simulation extractability. The proposed arguments are practical, with proof sizes never more than a constant number of group elements. Verification for two of our constructions consist of a small number of pairing operations. For our other construction, which has the desirable property of a linear sized updatable and universal SRS, we describe efficient batching techniques so that verification is fast in the amortised setting.

Impact Statement

The results presented in this dissertation are likely to facilitate the development of privacy preserving technologies in distributed systems. Research and industry will have more choice in the features they acquire from the zero-knowledge arguments that they use in their designs. More specifically, this dissertation introduces novel techniques for handling trust and integrity issues around zero-knowledge arguments that are used in decentralised protocols. We have designed and developed a technique for generating a set of semi-trustless parameters for zero-knowledge arguments that can be used for any (bounded) system; we have also shown how efficient zero-knowledge arguments can be built under these parameters. Further, we show how to prevent malicious parties from using previously seen data to their advantage.

When designing one of our constructions, we collaborated with developers from Zcash (a privacy focussed cryptocurrency) who implemented and improved upon the design. Our protocol is suitable for systems like Zcash, where data is stored indefinitely and thus needs to be small, and verification is run by all members of the system and thus needs to be fast in the amortised setting.

Our work on preventing adversarial attacks that utilise previously seen zero-knowledge protocols has been implemented by both the SCIPR labs in C++ and by O(1) labs in ocaml. O(1) labs are currently attempting to build a constant sized blockchain by utilising simulation-extractable SNARKs. Their construction is recursive: the provers prove

that they know a valid proof for the verifiers equations. With simulation extractable SNARKs like the one in this thesis, they do not run into complications from the provers potentially knowing multiple solutions to the verifiers equations.

Finally, the content in this thesis has been published not only in conference proceedings but also in open access repositories; thus, it can facilitate impact from other researchers that are looking into using and improving zero-knowledge.

Acknowledgements

This thesis would not have been possible without the time, patience, and kindness of my wonderful supervisors: Sarah Meiklejohn, Jens Groth, and Markulf Kohlweiss. I am truly honoured to have had the opportunity to work with them. I also want to thank my previous teachers, in particular my former supervisor Yves Tourigny, for having confidence in me and for encouraging me to pursue this path.

To all the additional great minds that I have had the pleasure of collaborating with - Sean Bowe, Andrea Cerulli, Jonathan Bootle, Sune Jakobsen, Haaroon Yousaf, George Kappos, Sarah Azouvi, and Melissa Chase - thank you for being as excited by our research as I have been. To the UCL infosec department, thank you for creating such a vibrant and welcoming community, without which this PhD experience really would not have been the same.

To Microsoft Research Cambridge, thank you for funding my research, and for giving me the opportunity to work with your world-leading team as an intern.

To Jenny Galuschka, Steve Maller, and Kit Smeets, thank you for being brave enough to proofread this work. Last but not least I would like to thank Philip Allaker, who has been by my side every step of way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis looks into the design of zero-knowledge arguments of knowledge. Zero-knowledge arguments of knowledge are a tool that cryptographers use to build privacy enhancing applications, which are themselves used to secure our digital communications. They allow us to prove that we are who we say we are, while never meeting in person, and while giving away nothing that could be used to impersonate us in future. Combined with the internet, even simple forms of zero-knowledge already provide a means to send credit card details safely and securely to a trusted retail brand. As more versatile forms of zero-knowledge become ever more practical, previously unrealisable applications are beginning to emerge.

Introduced three decades ago by Goldwasser, Micali, and Rackoff [1], the purpose of a zero-knowledge proof is to prove the truth of a statement while revealing nothing except its truth. For example, one could prove that an encrypted vote contains either yes or no, thus proving integrity, without decrypting or otherwise compromising the privacy of the vote's contents. For the sole purpose of convincing the reader of the power of zero-knowledge, we list the following applications: verifiable outsourced computation, anonymous credentials, blacklists, range proofs, trusted platform modules, ring signatures, group signatures, online auctions, public key infrastructures, mix-nets, multi-party computation and many memorable others [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. In recent years, cryptocurrerencies have been one increasingly popular real-world application [15, 16, 17, 18], with zero-knowledge arguments now deployed in both Zcash and Ethereum.

The particular type of zero-knowledge arguments deployed in these cryptocurrencies is a (pre-processing) succinct non-interactive argument of knowledge, or zk-SNARK for short, which guarantees constant-size proofs and constant-time verification costs even for arbitrarily large arithmetic circuits. In comparison to other types of zero-knowledge arguments, these guarantees make zk-SNARKs a natural fit for the cryptocurrency setting, in which proofs are put, kept, and verified on a globally visible blockchain. While the efficiency guarantees of zk-SNARKs are crucial in this setting, they have two major downsides: (1) they require a trusted set of entities to generate the structured reference string (SRS), and (2) once generated, the SRS is not universal, meaning it can be used to prove only a single relation. While this second issue might not seem so bad, deployed protocols regularly undergo upgrades (to add features, fix bugs, etc.), which may result in changes to their underlying relation. If this results in the need to generate a new SRS, then it opens up a new opportunity for an adversary to subvert the trusted setup, which compounds any threat presented by the first issue. Indeed, both of these issues have been observed already in Zcash, which recently had to run a second trusted setup (the Powers of Tau ceremony [19]) due to the Sapling upgrade in their protocol, which changed the relation they use. If the parameters they use were compromised by an adversary (or set of adversaries) during the setup process, then that adversary could create counterfeit units of currency without detection. Other techniques such as Bulletproofs [20], which are forthcoming in Monero, do not require a trusted setup and have a universal reference string. Less desirably, they have linear-time verification costs.

This thesis is arranged as follows. In Section 1.1 we discuss the authors published works. In Chapter 2 we discuss related work and in Chapter 3 we introduce relevant background material and definitions.

Our first construction, in Chapter 4, addresses the issue of adversaries that have access to proofs that they did not themselves create [21]. Knowledge-sound NIZKs only ensure that the prover knows a witness if the prover cannot see previous proofs. In the case of cryptocurrencies, where all proofs are visibly available on the blockchain and where miners might see proofs before other users, this is not a realistic

threat model. Our zk-SNARK construction addresses these concerns because it is simulation-extractable (an SE-SNARK): even a prover that can see old proofs cannot create new proofs without knowledge of the witness. Our construction is competitive with the state-of-the-art, requiring only 3 proof elements and 2 verification equations. Even within this thesis we see the benefit of simulation-extractable proofs. When designing our update proofs, the updater has access to previous update proofs; if the update proofs are simulation extractable then we can ensure the updater cannot utilise the previous proofs to find an attack simply by checking uniqueness. On the other hand, it does rely on a trusted setup.

In Chapter 5 we introduce the concept of updatability, meaning an open set of participants can contribute secret randomness to the SRS. While this is not a fully trustless setup, it means that confidence in the security of the parameters can be increased as more and more participants contribute, as only one previous contributor must have destroyed their randomness in order for the SRS to be secure. In the updatable SRS model, any user can at any point choose to update the reference string, provided that they also prove they have done the update correctly. If the proof of correctness verifies, then the new SRS resulting from the update can be considered trustworthy (i.e., uncorrupted) as long as either the old SRS or the updater was honest. If multiple users participate in this process, then it is possible to get a sequence of updates by different people over a period of time. If any one update is honest at any point in the sequence, then the scheme is sound.

We then in Chapter 6 go on to construct an updatable QAP-based zk-SNARK that uses a quadratic-sized universal SRS, but allows for the derivation of linear-sized relation-dependent SRSs (and thus linear prover complexity) [22]. In terms of efficiency, however, while the construction does have constant-size proofs and constant-time verification, it requires an SRS that is quadratic with respect to the number of multiplication gates in the supported arithmetic circuits. Moreover, updating the SRS requires a quadratic number of group exponentiations, and verifying the updates requires a linear number of pairings. Finally, while the prover and verifier in any concrete usage need only a linear-size circuit-specific string (rather

than the whole SRS), deriving this from the SRS still requires an expensive gaussian elimination process. In a concrete usage such as in Zcash, which has a circuit with 2^{17} multiplication gates, the SRS would be on the order of terabytes and is thus prohibitively expensive.

To address these joint concerns of efficiency, trustlessness, and universality, we present in Chapter 7 a zero-knowledge argument, Sonic, with a linear-size updatable and universal structured reference string [23]. Our proofs are of constant size regardless of the circuit, but our construction is not a true SNARK as the verifier must perform a linear number of field operations (which is still preferable to the linear number of group exponentations required by Bulletproofs). We describe, however, ways to batch verification, so that the verifier need only perform a linear number of field operations for an entire batch (and a constant number of group operations per proof). Our batching is done via the use of a helper, who combines multiple proofs together in order to help the verifier. The helper is not trusted because the verifier checks for themselves that they have done the batching correctly, but the addition of this extra party does raise the natural question of who would be expected to play this role in a given application. In a blockchain setting, however, there is an equally natural answer: miners, who are responsible for sealing individual transactions into blocks, already have access to multiple proofs before they are given to the verifiers and already expend significant computational energy in order to produce blocks (at least in cryptocurrencies using proof-of-work as their consensus protocol).

1.1 Publications

This section discusses the author's published works, including those that are not included in this thesis. Publications are ordered chronologically. All papers are joint work unless otherwise stated.

[24] Melissa Chase, Mary Maller, and Sarah Meiklejohn. Déjà Q all over again: Tighter and broader reductions of q-type assumptions. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 655–681, 2016

This paper looks into assumptions in composite order groups. Specifically, it builds upon the *déja-q* framework of Chase and Meiklejohn [25] in order to expand on the range of q-type assumptions that can be implied by subgroup hiding. Chase and Meiklejohn showed how to cover certain decisional target group assumptions using the framework. The proof techniques involve hybrid jumping between games using parameter hiding and subgroup hiding, until one arrives at a game which is statistically impossible. The argument that the final game is statistically impossible was designed by Chase, and utilises the invertibility of the Vandermonde matrix. All authors showed how to get tighter bounds at the expense of an extra subgroup. The author applied a framework by Abe et. al. [26] to show that a number of schemes in symmetric groups can be converted into asymmmetric groups and thus covered by our framework.

[21] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 581–612, 2017

This paper introduces a pairing based simulation-extractable SNARK which it proves to have optimal proof sizes and number of verification equations. This paper is discussed in detail in Chapter 4. All sections of this paper are joint work, except the section discussing Square Arithmetic Programs, which is due to Groth.

[27] Sarah Azouvi, Mary Maller, and Sarah Meiklejohn. Egalitarian society or benevolent dictatorship: The state of cryptocurrency governance. In *22nd International Conference on Financial Cryptography and Data Security*, 2018

This paper considers concrete methods for measuring the level of decentralisation in cryptocurrencies. The primary author is Sarah Azouvi who scraped GitHub repositories to find the number of code contributers and commenters, plotted the graphs, and calculated their statistical significance. All authors contributed in choosing the decentrality metrics, in discussing their implications, and in the experiment writeup.

[28] George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in Zcash. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, pages 463–477, 2018

This paper looks into the anonymity guarantees in Zcash: it finds that although it is possible to use Zcash in an anonymous manner, many users have habits that can be used to deanonymise them. Meiklejohn noticed that non-trivial information was being leaked through Zcash's "shielded pool". Yousaf processed the blockchain data so that we could run our analysis. All authors helped develop the heuristics. Yousaf and Meiklejohn calculated the general blockchain statistics and applied clustering techniques. The author interacted with the exchanges so that we could tag the larger clusters. Kappos analysed the interations with the shielded pool and spotted a number of deanonymising patterns, including a method of categorising transactions from the company's founders. Yousaf and the author worked on a case study for transactions related to a hacker collective and the author identified a user that made 3 suspicious transactions within 3 months.

[22] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pages 698–728, 2018

This paper introduces the updatability framework and an updatable and universal zk-SNARK. This paper is discussed in detail in Chapter 6. Kohlweiss proposed the concept of updatability, which was formalised by Groth, Kohlweiss and Meiklejohn. Kohlweiss and the author considered how the updates would run in practice: Kohlweiss showed that an adversary that runs all the updates except the setup is equivalent to an adversary that can run many updates; the author showed that an adversary that runs updates can extract a trapdoor. Groth spent one (apparently long) weekend thinking on the problem, and came back with a null-space argument that was efficient enough for us to compete with non-updatable SNARKs. Groth and the author formalised this idea and proved it secure in the Knowledge-of-Exponent model. Miers and Meiklejohn discussed the implications of these results. The author found a (non-trivial) impossibility result, namely that any updatable scheme cannot contain hidden polynomials in the structured reference string, and was aided in formalising it by Kohlweiss and Meiklejohn.

[29] Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, pages 595–626, 2018

This paper allows for verifiable computation in zero-knowledge without overly large prover costs. To do this it processes TinyRAM programs into arithmetic circuits

so that prover efficient techniques by Bootle et al. [30] could be applied. The primary author was Andrea Cerulli who formalised the types of constraints that need checking including both memory constraints and instruction constraints defined by the program. Jakobsen designed a permutation argument for the memory constraints. Bootle designed and proved secure the protocols that use techniques from [30]. Groth found a hidden bits argument that could be used to batch the (expensive) boolean operation checks. Cerulli and the author linked the TinyRAM checks to the relavent proofs by Bootle by discussing how to commit to the transcript and then how to apply the subproofs.

[23] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. *IACR Cryptology ePrint Archive*, 2019:99, 2019

This paper constructs a zero-knowledge argument with a linear sized structured reference string, short proofs, and efficient verifier times for batched proofs. This work is discussed in more detail in Chapter 7. The author was the primary investigator of this work and was responsible for designing the scheme. Bowe implementated the scheme, helped debug earlier constructions, and found non-trivial efficiency improvements such as a way to upload the instance in time that depends only on the size of the instance. Kohlweiss and the author worked on the security proofs. Meiklejohn looked into the security definitions and worked jointly with the author in designing an actor that we call the "helper".

This work has been significantly improved since the writing of this thesis. Bowe found substantial improvements to the protocol, including a method to avoid having proof elements in the second source group, and the author proved that the improved protocol is secure in the algebraic group model. The author also found a method to get a fully succinct protocol in the unhelped setting, albeit at the cost of concrete efficiency. The fully succinct protocol was written up and proven by Bowe and the author. The improved version of this protocol is accepted at ACM Conference on Computer and Communications Security 2019.

Chapter 2

Literature Review

In the past decade, there has been an unexplained explosion of practical zero-knowledge protocols applicable to NP statements. It is difficult to explain the cause of progress because many of the techniques the community are using have been around for a considerable time. Nonetheless, we now have a multitude of practical schemes applicable to a wide selection of languages. In this section we begin by giving credit to the forefathers and foremothers of zero-knowledge with a brief historic section. After, we endeaver to explain the trade-offs between some popular zero-knowledge protocols, and to give an overview of the techniques. We shall then discuss subversion resistance, i.e. the level of security that can be achieved when adversaries corrupt the public parameters, due to its importance in this thesis. We finish by discussing simulation-extractability, again included due to its relevance.

2.1 Historic

The initial concept of zero-knowledge was introduced by Goldwasser, Micali, and Rackoff in the 1980s [1]. Their original idea considered an interactive proof between a computationally unbounded prover that has to convince a probabilistic verifier that an instance is in a language. Brassard et al. [31] suggested a weakening of soundness and their definition is termed as zero-knowledge arguments [32]; in zero-knowledge arguments a computationally bounded prover has negligible probability of cheating a probabilistic verifier. The question then arose as to which NP languages have zero-knowledge proof systems, to which the satisfying answer is that all of them

do provided that one way functions exist [31, 33]. A strengthening of the zero-knowledge concept, called zero-knowledge proofs/arguments of knowledge, was suggested in [34, 35, 36, 37, 38, 39]. The idea for a proof of knowledge is that the pure existence of witnesses does not suffice, and the successful prover should know at least one witness.

Blum, Feldman and Micali introduced non-interactive zero-knowledge (NIZK) protocols in the common reference string model [40]. This assumes that there are a set of parameters generated by an honest party that are known to all parties. In 2018 the Zero Knowledge Standards workshop recommended using the terms common random string (CRS) and structured reference string (SRS) to distinguish between common reference strings that have structure and ones that do not [41][1]. Groth et al. [42] designed a NIZK for NP in the CRS model from bilinear groups. Groth and Sahai [43] additionally designed a NIZK for pairing based languages that avoids NP reductions.

Without a CRS/SRS, Goldreich et al. showed that zero-knowledge arguments require at least 3 rounds [44]. There are constructions with four rounds such as [37, 45]. For interactive protocols, Fiat and Shamir [46] demonstrated that the interaction could be removed in the random oracle model [47]. Their transform replaces the verifier's randomness with a hash of the prover's first message. Using this transform Schnorr introduced a proof of knowledge of discrete logarithms widely used in identification protocols today [48]. Fischlin extended their methods to handle online extractors [49].

2.2 The State of the Art

We shall now compare our work with the state-of-the-art for zero-knowledge proofs. An efficiency comparison of all the schemes we discuss in this section is provided in Table 2.1. In Table 2.1 n is the number of gates, d is the depth of the circuit, h is the width of the subcircuits, c is the number of copies of the subcircuits, ℓ is the size of the instance, and w is the size of the witness. For our work we have given the verifier

[1] In the past sometimes papers make a distinction between URS (uniform random string) and CRS (common reference string, which may be structured).

Scheme	Runtime		Size		PQ?	Universal?	Untrusted setup?	Assumptions
	Prover	Verifier	CRS	Proof				
Hyrax	$d(hc+c\log c)+w$	$\ell+d(h+\log(hc))$	$\sqrt{w}$	$d\log(hc)+\sqrt{w}$	○	●	●	DL, ROM
Bulletproofs	$n\log(n)$	$n\log(n)$	n	$\log(n)$	○	●	●	DL, ROM
Ligero	$n\log(n)$	$c\log(c)+h\log(h)$	0	$\sqrt{n}$	◐	●	●	CRHF, ROM
STARKs	n polylog(n)	polylog(n)	0	$\log^2(n)$	◐	●	●	CRHF, ROM
Bootle et al. [30]	n	n	0	$\sqrt{n}$	◐	●	●	CRHF, ROM
Baum et. al. [50]	$n\log(n)$	n	$\sqrt{n}$	$\sqrt{n\log(n)}$	◐	●	●	SIS, ROM
ZK vSQL	$n\log(c)$	$\ell+d$ polylog(n)	$\log(n)$	$d\log(c)$	○	●	◐	q-type, KOE, ROM
SNARKs	$n\log(n)$	ℓ	n	1	○	○	○	q-type, KOE
Chapter 6	$n\log(n)$	ℓ	n^2	1	○	●	◐	q-type, KOE
Chapter 7	$n\log(n)$	n^*	n	1	○	●	◐	q-type, KOE, ROM

Table 2.1: Asymptotic efficiency comparison of state-of-the-art zero-knowledge arguments.

computation as n^* to represent that the verifier's costs can be efficiently batched. We refer to the works in this thesis by the chapters they appear in: Chapter 6 refers to the updatable and universal zk-SNARK and Chapter 7 refers to Sonic. Our other work in Chapter 4 for an SE-SNARK has the same asymtotic efficiency as other SNARKs in the literature, and is therefore grouped together with SNARKs. An empty circle denotes that the scheme does not have this property and a full circle denotes that the scheme does have this property. A half circle for post-quantum security denotes that the security depends on assumptions which have no known quantum attack, but that there is no formal proof that a quantum adversary cannot attack the scheme with other methods. A half circle for untrusted setup denotes that the scheme is updatable. DL stands for discrete log, CRHF stands for collision-resistant hash functions, ROM stands for random oracle model, and KOE stands for knowledge-of-exponent.

2.2.1 Quantum Resistant Protocols

Symmetric primitives such as Reed-Solomon codes have recently been gaining attention for their post-quantum potential, as there are no known quantum attacks on error-correcting codes and protocols that use them do not require expensive and trusted pre-processing phases. Schemes that use these techniques [51, 30, 52] are typically made non-interactive in the random oracle model, as opposed to the quantum random oracle model, and designing efficient zero-knowledge protocols in the quantum random oracle model [53] remains an open problem. The codes are typically cheap to compute for the prover. The downside to this style of proof is that they require very large circuits before the asymptotics can take effect, because the constants are relatively large.

Ligero [51] uses collision resistant hash functions. This work stems from the "MPC-in-the-head" paradigm [54, 55, 56]. The idea is to model the computation as being carried out by a multiparty computation, but then have the prover and verifier simulate multiple parties. A large part of its overhead comes from compiling the addition gates, and the authors observed that when there are many repetitions of the same addition gates in the same layer, it is possible to batch the compilation.

Bootle et al. [30] introduce a model that they call the ideal linear commitment (ILC) model in which a prover can commit to vectors by sending them to a channel, and a verifier can query the channel on linear combinations of the committed vectors. They then compile the ILC programs into proofs using an error-correcting code by Ishai et al. [57] which can be computed in linear time. As a result, they prove the possibility of zk-proofs that have linear prover overhead.

STARKs [52] look to simultaneously minimise proof size and verifier computation and they show, with an implementation, that protocols based on interactive oracle proofs [58] can be practical. Interactive oracle proofs optimise a technique introduced by Kilian [59] which format probabilistically checkable proofs into Merkle trees. A STARK prover, when applied to a circuit with 2^{27} gates, takes roughly 1 minute to run. However, proof sizes are still over 100KB, even for relatively small circuits.

Also renowned for its post-quantum potential, lattice based cryptography is a major research topic, and zero-knowledge protocols from lattice based assumptions are no exception. These schemes are built from assumptions such as the shortest integer solution and the closest vector problem. Baum et al. [50] introduced the first lattice based protocol with sublinear communication costs. They achieve this by designing a proof of knowledge for committed values using techniques by Cramer et al. [60]. The proof of knowledge is efficient in the amortised setting. They apply this proof of knowledge to circuits processed using Bulletproof [61] techniques. As a result, their verifier time is high. It would be interesting to see whether their verifier could also be batched if there were a helper available.

2.2.2 Discrete Logarithm Protocols

Bulletproofs [61, 20] are based on the discrete logarithm problem and have no trusted setup. The idea is to send a constant sized commitment to a larger vector and prove (not in zero-knowledge) that the committed vector satisfies verifier's equation. They call this construct an inner product argument - it inductively shows that, at each stage, a new committed vector of half the length satisfies a new equation if and only if the old committed vector satisfied the old equation. The inner product argument is logarithmically sized. On the downside the verification computation is high. Although Bulletproofs lend themselves well to batching, even batched proofs require a computation per proof that depends on the size of the circuit. For very small circuits, such as for range proofs, Bulletproofs have the advantage of having relatively low concrete overhead.

Hyrax [62] is a zero-knowledge protocol that processes circuits using a sum-check protocol originally introduced from the verifiable computation scheme by Goldwasser et al [63] and improved by Cormode et al. [64]. It is especially well-suited to circuits with a high level of parallelisation, such as showing that a committed value is included in a Merkle tree. Additionally, the protocol is ideal for circuits with small witnesses. This is because the protocol applies different variants of the sum-check protocol for instance wires and witness wires. It directly uses a parallelised sum-check protocol on the instance wires, and thus does not require the use of (expensive) public key cryptography. For the witness wires, it applies a zero-knowledge variant of the sum-check protocol. Their sum-check protocol uses an adaptation of the inner-product argument from Bulletproofs to check multiplication constraints.

2.2.3 Protocols using Verifiable Polynomial Delegation

Zhang et al.'s [65] zero-knowledge variant of vSQL was originally designed for handling SQL queries. They also process circuits using techniques by Cormode et al. [64]. This means that their techniques also have better efficiency for highly parallelised circuits. Like Sonic, they rely on an adapted polynomial commitment scheme, which they call a verifiable polynomial delegation scheme. However, rather

than use our technique of using Kate et al.'s [66] single variant scheme as a base, they use Papamanthou et al.'s multivariate scheme [67]. The reason this is available to them is because they can use multivariate polynomials where each variable has degree 1; for each round of their sum-check protocol, they include one extra variable in their polynomial equation. For our scheme, there are two variables of degree $\mathcal{O}(n)$, thus Papamanthou et al.'s scheme would result in a quadratic-sized reference string and quadratic prover computation.

2.2.4 SNARKs

Using knowledge assumptions, it is possible to build zk-SNARKs [68, 2, 69, 70, 71, 72]. These have constant-size proofs and verifier times that depend solely on the instance. However, they typically use circuit-specific quadratic span programs or quadratic arithmetic programs [73]. As such the structured reference strings are neither updatable nor universal [74]. The prover costs for zk-SNARKs are typically high due to the need for expensive cryptographic operations, although a recent work has looked into methods to distribute these costs [75].

We give a performance comparison of pairing-based zk-SNARKs (as well as Sonic) in Table 2.2, comparing the relative size of the SRS, the proof, and the computation required for the prover and verifier. In this table, there are ℓ known circuit inputs, m wires, and n gates; $\mathbb{G}$ means group elements in either source group, $\mathbb{F}$ means field elements, Ex means group exponentiations, $M_{\mathbb{G}}$ means group multiplications, $M_{\mathbb{F}}$ means field multiplications, and P means pairings. For [76] and Sonic, d relates to the maximum sized circuit that can be committed to. We compare Groth's original zk-SNARK [76], Pinocchio [2], Groth's 2016 zk-SNARK [76], our SE-SNARK (Chapter 4), our updatable and universal zk-SNARK (Chapter 6), and Sonic (Chapter 7). For the QAP-based SNARKs one could use Valiant's universal circuit construction [77, 78] to achieve universality but this would introduce a $\log n$ multiplicative overhead to the size of the circuit.

Scheme	Runtime		Size		
	Prover	Verifier	Universal SRS	Circuit SRS	Proof
Groth [76] ($\mathbb{F}_2$)	$\mathcal{O}(n^2)$ Ex	$36P+\mathcal{O}(n)$ $M_{\mathbb{G}}$	$\mathcal{O}(n^2)$ $\mathbb{G}$	—	42 $\mathbb{G}$
Pinocchio ($\mathbb{F}_q$)	$n+7m-\ell$ Ex	$12P+\ell$ Ex	—	$n+7m-\ell$ $\mathbb{G}$	8 $\mathbb{G}$
Groth [72] ($\mathbb{F}_q$)	$4n+m-\ell$ Ex	$3P+\ell$ Ex	—	$3n+m$ $\mathbb{G}$	3 $\mathbb{G}$
Chapter 4 ($\mathbb{F}_q$)	$m+6n-\ell$ Ex	$5P+\ell$ Ex	—	$m+6n$ $\mathbb{G}$	3 $\mathbb{G}$
Chapter 6 ($\mathbb{F}_q$)	$20n+3m-\ell$ Ex	$5P+\ell$ Ex	$42n^2+32n$ $\mathbb{G}$	$23n+3m-\ell$ $\mathbb{G}$	3 $\mathbb{G}$
Chapter 7 ($\mathbb{F}_q$)	$24n$ Ex	$24P+\ell$ Ex $+n$ $M_{\mathbb{F}}$	$4d$ $\mathbb{G}$	$6n$ $\mathbb{G}$	17 $\mathbb{G}$ +1 $\mathbb{F}_q$

Table 2.2: Comparison for pairing-based zk-SNARKs for boolean and arithmetic circuit satisfiability.

2.2.5 Designated Verifier

In the designated verifier setting, i.e. when the verifier holds a secret key unknown to the prover, one can often achieve results not possible in the publicly verifiable setting discussed above. Subversion resistance ceases to be a concern because the public parameters can be output by the verifier. While totally unsuitable for distributed systems, they are sometimes preferable for applications such as verifiable computation. Chaidos and Couteau [79] designed a designated verifier NIZK proof of knowledge in the standard model which covers a wide class of algebraic assumptions, and which they claim is competitive with NIZKs in the random oracle model. Gennaro et al. designed a lattice based zk-SNARK using circuit-specific quadratic span programs [80] (in fact they use a variant called square span programs introduced by Danezis et al. [71]). Designing a publically verifiable NIZK in the standard model from quantum resistant assumptions is to this day an open problem, although considering this thesis strays so far from both quantum resistant assumptions and the standard model, it is not one that we shall consider.

2.3 Subversion Resistance

Here we consider works that discuss subversion resistance i.e. works that consider the consequences of adversaries that can corrupt the generation of the SRS.

Bellare, Fuchsbauer and Scafuro [81] ask what security can be maintained for NIZK proofs when the SRS is subverted. They formalise the different notions of subversion resistance and then investigate their possibility. Using similar techniques to Goldreich et al. [82], they show that soundness in this setting cannot be achieved

at the same time as (standard) zero-knowledge. Building on the notions of Bellare et al., two recent papers [83, 84] discuss how to achieve subversion zero-knowledge for zk-SNARKs. None of these schemes, however, can avoid the impossibility result and they do not simultaneously preserve soundness and zero-knowledge under subversion.

The multi-string model by Groth and Ostrovsky [85] addresses the problem of subversion by designing protocols that require only the majority of the parties contributing multiple reference strings to be honest. Their construction gives statistically sound proofs but they are of linear size in both the number of reference strings and the size of the instance.

Two early zk-SNARKs by Groth [76] and Lipmaa [86] do use only monomials in the reference string, and therefore are updatable and universal. The main drawback of [76] is that it has a quadratic-sized SRS and quadratic prover computation, but it has a SRS that consists solely of monomials, and thus is updatable. Lipmaa still has quadratic prover computation, however he suggests the use of progression-free sets to construct NIZK arguments with an SRS consisting of $n^{\mathcal{O}(1)}$ group elements.

In concurrent work, Bowe et al. [19] propose a two-phase protocol for the generation of a zk-SNARK reference string that is player-replaceable [87]. Like our protocol, the first phase of their protocol also computes monomials with parties operating in a similar one-shot fashion. However, there are important differences. To create a full SRS which does not have quadratic prover time, Bowe et al. require a second phase. As one party in each phase must be honest and the second phase depends on the first, the final SRS is not updatable. There is no way to increase the number of parties in the first phase after the second phase has started and restarting the first phase means discarding the participants in the second phase. As a result, the protocol is a multi-party computation to produce a fixed SRS with a fixed set of participants, albeit with the set of participants fixed midway through the protocol instead of at the start.

To discuss the efficiency of generating structured reference strings using either MPC techniques or our updating techniques we shall considier two security factors:

the well-formedness of the final reference string; and the inclusion of each of the players randomness contributions in the final parameters. Ben-Sasson et al. [74] and subsequently Bowe et al. [88] examined the use of a four round multi-party computation to generate an SRS, where only one of the participating parties needs to be honest but the participants must be selected in advance. Their final reference string contains a linear number of group elements and thus costs linear work to verify its well-formedness. However, they require that all the players remain online until the completion of the MPC, which severely limits the number of players that can be included. Also their proofs that the players randomness was included uses the Forking Lemma [89] and thus their security proofs only hold when there are a logarithmic number of players. Bowe et al. [19] improved on these works with a two-phase protocol that is player replaceable in the sense that the players participating in the first and second rounds may be different. They can thus support more players. They require a linear number of proofs of knowledge in the number of players. Our updating process also contains a linear number of proofs of knowledge in the number of updaters. The final reference string for Sonic in Chapter 7 contains a linear number of group elements and thus costs linear work to verify its well-formedness. However, our zk-SNARK in Chapter 6 requires a quadratic sized global reference string thus will have quadratic work to verify its well-formeness. Additionally, we require a derivation process to obtain the circuit specific parameters that depends on Gaussian elimination.

2.4 Simulation Extractability

Sahai [90] introduced simulation-soundness of NIZK proofs as a notion to capture that even after seeing simulated proofs it is not possible to create a fake proof for a false instance unless copying a previous simulated proof. He showed that it is possible to construct simulation-sound NIZKs from an ordinary NIZKs and one way functions. Further, he showed that CCA2 encryption can be constructed from simulation-sound NIZKs.

Combining the notions of simulation soundness and proofs of knowledge,

Groth [91] defined the stronger security notion that we should be able to extract a witness from an adversary that creates a valid new proof, even if this adversary has seen many simulated proofs for arbitrary instances. He then showed how to adapt Groth-Sahai proofs to be simulation-extractable in the standard model. Groth-Sahai proofs are built over pairing based languages, therefore the witnesses extracted are group elements. In the quasi-adaptive setting (i.e. when there is a relation dependent SRS) where the pairing based languages have no quadratic component, Libert et al. [92] provided a constant sized simulation-sound NIZK that is secure in the quasi-adaptive setting and Abe et al. provided a simulation-sound NIZK with an (almost) tight reduction to SXDH [93].

Faust, Kohlweiss, Marson, and Venturi discuss how to achieve simulation soundness in the random oracle model [94]. It would be interesting to see whether their techniques could be adapted to prove the simulation-extractability of Sonic in Chapter 7. Bowe et al. [95] published a follow up paper to our SE-SNARK which demonstrates how to adapt Groth's zk-SNARK [72] to be simulation-extractable in the random oracle model. They require 5 proof elements (2 more than us), however they claim that their prover is more efficient because it requires a smaller multi-exponentiation for the proof element in the second source group[2].

[2]Asymmetric bilinear groups have two source groups, and current implementations have more efficient operations in the first source group than the second.

Chapter 3

Background and Definitions

In this Chapter, we introduce definitions which are relevant to this thesis, such as subversion zero-knowledge, updatable knowledge soundness, and simulation extractability. We discuss the notation for linear non-interactive proofs which is used in Chapter 5 when proving constraints on the format of updatable SRSs. Once our definitions have been specified, we discuss the assumptions that are used in this thesis. These are all either knowledge-of-exponent assumptions or computational q-type assumptions. Our simulation extractable zk-SNARK in Chapter 4 requires the strongest assumptions, and our Sonic construction in Chapter 7 requires the weakest assumptions (albeit Sonic is in the random oracle model). The final part of this chapter aims to provide some indication about how computational problems are specified in this thesis. All protocols are built on top of arithmetic circuits with fan-in 2 gates. We provide an example of such an arithmetic circuit, namely one for proving that given $y, s, a, b \in \mathbb{F}$, the prover possesses knowledge of a fourth value v such that $y = a^s b^v$.

3.0.1 Notation

If x is a binary string then $|x|$ denotes its bit length. If S is a finite set then $|S|$ denotes its size and $x \xleftarrow{\$} S$ denotes sampling a member uniformly from S and assigning it to x. We use $\lambda \in \mathbb{N}$ to denote the security parameter and 1^λ to denote its unary representation. We use ε to denote the empty string.

Algorithms are randomized unless explicitly noted otherwise. "PPT" stands for "probabilistic polynomial time" and "DPT" stands for "deterministic polynomial

time." We use $y \leftarrow \mathscr{A}(x;r)$ to denote running algorithm $\mathscr{A}$ on inputs x and random coins r and assigning its output to y. We write $y \xleftarrow{\$} \mathscr{A}(x)$ or $y \xleftarrow{r} \mathscr{A}(x)$ (when we want to refer to r later on). To specify that a randomised algorithm runs with randomness r we write $y \leftarrow \mathscr{A}(\cdot;r)$.

We use code-based games in security definitions and proofs [96]. A game $\mathsf{Sec}_{\mathscr{A}}(\lambda)$, played with respect to a security notion Sec and adversary $\mathscr{A}$, has a MAIN procedure whose output is the output of the game. The notation $\mathsf{Pr}[\mathsf{Sec}_{\mathscr{A}}(\lambda)]$ is used to denote the probability that this output is 1.

Matrix Notation: We denote matrices by capital letters $\hat{M}$ and column vectors by $\boldsymbol{x}$. We use the typical notation $\hat{M}\boldsymbol{x}$ for matrix multiplication, $\boldsymbol{x} \circ \boldsymbol{y}$ for an element-wise vector product, and $\boldsymbol{x} \cdot \boldsymbol{y}$ for a dot product. We use $\hat{M}_{i,j}$ to denote the entry in the i-th row and j-th column of $\hat{M}$. When indexing matrices in a list, we denote the jth matrix by $\hat{M}^j$.

3.0.2 Bilinear Groups

All constructions in this thesis are built on top of bilinear groups. Let $\mathsf{BGen}(1^\lambda)$ be a bilinear group generator[1] that given the security parameter 1^λ produces bilinear parameters $\mathsf{bp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$: $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups whose order is divisible by p with generators $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$; $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerative bilinear map. That is, $e(g^a, h^b) = e(g,h)^{ab}$ for all field elements a, b and $e(g,h)$ generates $\mathbb{G}_T$.

We further require our bilinear group generator to produce what Galbraith, Paterson and Smart [97] classify as Type III bilinear groups, such that no efficiently computable homomorphism exists between $\mathbb{G}_1$ and $\mathbb{G}_2$. These are currently the most efficient bilinear groups.

3.1 Definitions

In this section, we give the definitions relevant to updatable SRS schemes, in terms of defining properties of zero-knowledge proofs in the case in which the adversary

[1] Often the cryptographic literature allows for probabilistic bilinear group generation, but for our purpose it is useful to have *deterministic* parameter generation that cannot be influenced by the adversary.

may subvert or participate in the generation of the reference string. We also provide definitions of simulation extractability, which is a stronger notion of soundness for when adversaries can see proofs that they did not generate themselves. Given that our protocol in Chapter 7 is interactive (but made non-interactive in the random oracle model), we also present definitions for interactive protocols that take into account these alternative methods of SRS generation. The definitions of updatability and subversion zero-knowledge are not the authors own, but a joint effort between Groth, Kohlweiss and Meiklejohn. The definition of subversion zero-knowledge is motivated by Bellare et al. [81]. They are included for completeness.

3.1.1 SRS Correctness

Intuitively, the subvertible SRS model [81] allows the adversary to fully generate the reference string itself, and the updatable SRS model [22] allows the adversary to partially contribute to its generation by performing some update. Formally, an updatable SRS scheme is defined by two PPT algorithms Setup and Update, and a DPT algorithm VerifySRS. These behave as follows:

- $(\mathtt{srs}, \rho) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ takes as input the security parameter and returns an SRS and a proof of its correctness.

- $(\mathtt{srs}', \rho') \xleftarrow{\$} \mathsf{Update}(1^\lambda, \mathtt{srs}, (\rho_i)_{i=1}^n)$ takes as input the security parameter, an SRS, and a list of update proofs. It outputs an updated SRS and a proof of the correctness of the update.

- $b \leftarrow \mathsf{VerifySRS}(1^\lambda, \mathtt{srs}, (\rho_i)_{i=1}^n)$ takes as input the security parameter, an SRS, and a list of proofs. It outputs a bit indicating acceptance ($b = 1$), or rejection ($b = 0$).

Definition 3.1.1 (Correctness)**.** *An updatable SRS scheme is perfectly correct if*

$$\Pr\left[(\mathtt{srs}, \rho) \xleftarrow{\$} \mathsf{Setup}(1^\lambda) : \mathsf{VerifySRS}(1^\lambda, \mathtt{srs}, \rho) = 1\right] = 1,$$

and for all $(\lambda, \mathtt{srs}, (\rho_i)_{i=1}^n)$ *such that* $\mathsf{VerifySRS}(1^\lambda, \mathtt{srs}, (\rho)_{i=1}^n) = 1$*, we have that*

$$\Pr\left[\begin{array}{l}(\mathtt{srs}', \rho_{n+1}) \xleftarrow{\$} \mathsf{Update}(1^\lambda, \mathtt{srs}, (\rho_i)_{i=1}^n) : \\ \mathsf{VerifySRS}(1^\lambda, \mathtt{srs}', (\rho)_{i=1}^{n+1}) = 1\end{array}\right] = 1.$$

3.1.2 NIZK Arguments

A non-interactive zero-knowledge (NIZK) argument for a relation R is defined by PPT algorithms Setup, Prove, and a DPT algorithm Verify. These behave as follows:

- $\mathtt{srs} \xleftarrow{\$} \mathsf{Setup}(R)$ takes as input a relation R and outputs an SRS.
- $\pi \xleftarrow{\$} \mathsf{Prove}(\mathtt{srs}, \phi, w)$ takes as input an SRS, and an instance and witness included in the relation $(\phi, w) \in R$. It outputs a proof.
- $b \leftarrow \mathsf{Verify}(\mathtt{srs}, \phi, \pi)$ takes as input an SRS, an instance and a proof. It outputs a bit indicating acceptance ($b = 1$), or rejection ($b = 0$).

An updatable non-interactive zero-knowledge (NIZK) argument for a relation R a NIZK scheme together with an updatable SRS scheme, so that both schemes have the same setup algorithm.

3.1.3 Subversion Zero-Knowledge

In terms of the usage of these SRSs in NIZK arguments, it is known that a protocol cannot satisfy both zero-knowledge and subvertible soundness [81]. That is, assuming the adversary knows all the randomness used to generate the SRS, then they can either break zero-knowledge or they can break soundness. We thus recall here the two strongest properties we can hope to satisfy, which are subvertible zero-knowledge and updatable knowledge soundness. The definitions of these properties are simplified versions of the ones given by Groth et al. [22], with the addition of a random oracle H (which behaves as expected if the scheme is in the ROM and returns $\perp$ to any query if the scheme is not in the ROM).

Definition 3.1.2 (Subversion Zero-Knowledge)**.** *An updatable NIZK argument for the relation R is subversion zero-knowledge if for all probabilistic polynomial time*

(PPT) algorithms $\mathscr{A}$, there exists a PPT simulator $\mathsf{SimProve}$ *such that the advantage* $|2\Pr[\mathsf{S\text{-}ZK}_{\mathscr{A}}(1^\lambda) = 1] - 1|$ *is negligible in* λ*, where this game is defined as follows:*

$\underline{\text{MAIN } \mathsf{S\text{-}ZK}_{\mathscr{A}}(\lambda)}$
$b \xleftarrow{\$} \{0,1\}$
$(\mathtt{srs}, (\rho_i)_{i=1}^n) \xleftarrow{r} \mathscr{A}^H(1^\lambda)$
if $\mathsf{VerifySRS}(1^\lambda, \mathtt{srs}, (\rho_i)_{i=1}^n) = 0$:
  $b' \xleftarrow{\$} \{0,1\}$
else $b' \leftarrow \mathscr{A}^{H, \mathscr{O}_{\mathsf{pf}}}(r)$
return $b' = b$

$\underline{\mathscr{O}_{\mathsf{pf}}(\phi, w)}$
if $(\phi, w) \notin R$ *return* $\perp$
if $b = 0$:
  return $\mathsf{SimProve}(\mathtt{srs}, r, \phi)$
if $b = 1$:
  return $\mathsf{Prove}(\mathtt{srs}, \phi, w)$

3.1.4 Updatable Knowledge Soundness

We model updatable knowledge soundness as a game in which the adversary attempts to find a verifying proof, and they win if the extractor cannot use their transcript to output a valid witness. The adversay can influence the generation of the SRS, however at least one party that the adversary does not control must also influence the generation of the SRS. The final SRS can only be assigned through the call to an oracle. This oracle can perform three tasks: it can setup a local SRS, it can update a local SRS, and it can assign the final SRS. It will only assign the final SRS if the adversary can also provide a verifying string of update proofs such that at least one of the proofs is one of the oracles responses.

Definition 3.1.3 (Updatable Knowledge Soundness)**.** *A NIZK argument for the relation R is updatable knowledge sound if for all PPT algorithms $\mathscr{A}$ there exists a PPT extractor $\mathscr{X}_{\mathscr{A}}$ such that the probability* $\Pr[\mathsf{U\text{-}KSND}_{\mathscr{A},\mathscr{X}}(1^\lambda)$ *is negligible in* λ*, where this game is defined as follows:*

MAIN $\mathsf{U\text{-}KSND}_{\mathscr{A},\mathscr{X}}(\lambda)$

$\mathtt{srs} \leftarrow \bot,\ Q \leftarrow \emptyset$

$(\phi,\pi) \xleftarrow{r} \mathscr{A}^{\mathsf{U}\text{-}\mathscr{O}_{\mathsf{s}}}(1^\lambda)$

$w \xleftarrow{\$} \mathscr{X}(\mathtt{srs}, r)$

return $\mathsf{Verify}(\mathtt{srs},\phi,\pi) \wedge (\phi,w) \notin R$

$\mathsf{U}\text{-}\mathscr{O}_{\mathsf{s}}(\mathsf{intent}, \mathtt{srs}_n, (\rho_i)_{i=1}^n)$

if $\mathtt{srs} \neq \bot$ *return* $\bot$

if $\mathsf{intent} = \mathsf{setup}$

  $(\mathtt{srs}', \rho') \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$

  $Q \leftarrow Q \cup \{\rho'\}$

  return $(\mathtt{srs}', \rho')$

if $\mathsf{intent} = \mathsf{update}$

  $b \leftarrow \mathsf{VerifySRS}(1^\lambda, \mathtt{srs}_n, (\rho_i)_{i=1}^n)$

  if $b = 0$ *return* $\bot$

  $(\mathtt{srs}', \rho') \xleftarrow{\$} \mathsf{Update}(1^\lambda, \mathtt{srs}_n, (\rho_i)_{i=1}^n)$

  $Q \leftarrow Q \cup \{\rho'\}$

  return $(\mathtt{srs}', \rho')$

if $\mathsf{intent} = \mathsf{final}$

  $b \leftarrow \mathsf{VerifySRS}(1^\lambda, \mathtt{srs}_n, (\rho_i)_{i=1}^n)$

  if $b = 0$ *or* $Q \cap \{\rho_i\}_{i=1}^n = \emptyset$ *return* $\bot$

  $\mathtt{srs} \leftarrow \mathtt{srs}_n$

  return $\mathtt{srs}$

else return $\bot$

3.1.5 Updatable Witness-Extended Emulation

For soundness, we require interactive security definitions. In the arguments we consider a prover P and a verifier V, both of which are PPT interactive arguments. The view of the transcript produced by P and V when interacting on inputs s and t is denoted by $\mathsf{view} \leftarrow \langle \mathsf{P}^*(s), \mathsf{V}(t) \rangle$.

For zero knowledge, we do not need an interactive variant of the security definitions, as we can argue for the non-interactive definition directly. This is because the simulator does not program the random oracle, it instead uses the trapdoor in the SRS. For soundness, the extractor uses only a weak form of programmability, namely it assumes that if it calls the oracle on the same input, it will receive a different output (with high probability). To obtain a non-interactive protocol, we apply the Fiat-Shamir heuristic i.e. the verifiers messages are obtained as a hash the provers messages and we assume that the hash outputs are indistinguishable from random.

We do not use the standard definition of special soundness because our verifier

provides two challenges, but rather the generalized notion of *witness-extended emulation* [39]. In particular, we adapt the definition given by Bootle et al. [61] as follows:

Definition 3.1.4 (Updatable Witness-Extended Emulation)**.** *An argument for the relation R satisfies updatable witness-extended emulation if for all DPT* P^* *and for all PPT algorithms* $\mathscr{A}$ *there exists an expected PPT emulator* $\mathscr{E}$ *such that:*

$$\mathsf{Pr}\left[\begin{array}{l}(\mathtt{srs}_1,\rho_1)\xleftarrow{\$}\mathsf{Setup}(1^\lambda);\ (\mathtt{srs},(\rho_i)_{i=2}^n,\phi,w)\xleftarrow{\$}\mathscr{A}(\mathtt{srs}_1,\rho_1);\\ \mathsf{view}\leftarrow\langle\mathsf{P}^*(\mathtt{srs},\phi,w),\mathsf{V}(\mathtt{srs},\phi)\rangle\ :\\ \mathsf{VerifySRS}(1^\lambda,\mathtt{srs},(\rho_i)_{i=1}^n)\ \wedge\ \mathscr{A}(\mathsf{view})=1\end{array}\right]\approx$$
$$\mathsf{Pr}\left[\begin{array}{l}(\mathtt{srs}_1,\rho_1)\xleftarrow{\$}\mathsf{Setup}(1^\lambda);\ (\mathtt{srs},(\rho_i)_{i=2}^n,\phi,w)\xleftarrow{\$}\mathscr{A}(\mathtt{srs}_1,\rho_1)\\ (\mathsf{view},w)\leftarrow\mathscr{E}^{\langle\mathsf{P}^*(\mathtt{srs},\phi,w),\mathsf{V}(\mathtt{srs},\phi)\rangle}:\\ \mathsf{VerifySRS}(1^\lambda,\mathtt{srs},(\rho_i)_{i=1}^n)\ \wedge\ \mathscr{A}(\mathsf{view})=1\\ \ \wedge\ \textit{if}\ \mathsf{view}\ \textit{is accepting then}\ (\phi,w)\in R\end{array}\right]$$

where the oracle called by $\mathscr{E}^{\langle\mathsf{P}^*(\mathtt{srs},\phi,w),\mathsf{V}(\mathtt{srs},\phi)\rangle}$ *permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards.*

This definition uses a slightly different setup from the one in [22]: rather than interact arbitrarily with an update oracle to set the SRS, the adversary is instead given an initial one and is then allowed to update that in a one-shot fashion. Kohlweiss showed in [22, Lemma 6] that these two definitions are equivalent for the setup for Sonic, so we opt for the simpler one.

3.1.6 Simulation Extractability

Zero-knowledge and soundness are core security properties of NIZK arguments. However, it is conceivable that an adversary that sees a simulated proof might modify the proof into another proof whether or not they know a witness. This scenario is actually very common in security proofs for cryptographic schemes, so it is often desirable to have some form of non-malleability that prevents cheating in the presence of simulated proofs. Traditionally, simulation extractability is defined with respect to an extraction key associated with the reference string that allows the extraction of a

witness from a valid proof. However, in succinct NIZK arguments the proofs are too small to encode the full witness. We therefore instead define simulation extractable NIZK arguments using a non-black-box extractor that can deduce the witness from the internal data of the adversary.

Definition 3.1.5 (Simulation Extractability)**.** *A NIZK argument for the relation R satisfies simulation extractability if for all PPT algorithms $\mathscr{A}$, there exists a PPT extractor $\mathscr{X}$ such that the probability* $\Pr[\mathsf{SE\text{-}KSND}_{\mathscr{A},\mathscr{X}}(1^\lambda)]$ *is negligible in λ, where this game is defined as follows:*

$\underline{\text{MAIN } \mathsf{SE\text{-}KSND}_{\mathscr{A},\mathscr{X}}(\lambda)}$
$\mathtt{srs} \xleftarrow{\tau} \mathsf{Setup}(R)$
$Q \leftarrow \emptyset$
$(\phi,\pi) \xleftarrow{r} \mathscr{A}^{\mathscr{O}_{\mathsf{SimProve}}}(1^\lambda)$
$w \xleftarrow{\$} \mathscr{X}(\mathtt{srs}, r)$
return $\mathsf{Verify}(\mathtt{srs},\phi,\pi) \wedge (\phi,\pi) \notin Q \wedge (\phi,w) \notin R$

$\underline{\mathscr{O}_{\mathsf{SimProve}}(\phi)}$
$\pi \leftarrow \mathsf{SimProve}(\mathtt{srs},\tau,\phi)$
$Q \leftarrow Q \cup \{\phi,\pi\}$
return π

We observe that simulation extractability implies knowledge soundness, since knowledge soundness corresponds to simulation extractability where the adversary is not allowed to use the simulation oracle.

3.1.7 Linear Interative Proofs

We use Bitansky et al.'s [69] framework for Linear Interactive Proofs (LIP's) with Groth's [72] extension to Non-Interactive Linear Proofs (NILP's) in arguing our results about the format of updatable SRSs. LIPs and NILPs work over finite fields and the prover's and verifier's messages consist of vectors of field elements. The prover's messages are computed using only linear operations. Our NILP models the updater as an algorithm that can only compute messages using linear operations and is defined as follows:

- $(\boldsymbol{\sigma},\rho) \xleftarrow{\boldsymbol{\tau}} \mathsf{Setup}(1^\lambda)$: The setup generates a random vector $\boldsymbol{\tau}$ in $\mathbb{F}^q$ and returns $\boldsymbol{\sigma} \leftarrow f(\boldsymbol{\tau})$ for a fixed polynomial $f(X_1,\ldots,X_q) \in \mathbb{F}[X_1,\ldots,X_q]$. It returns the SRS $\boldsymbol{\sigma}$ and a proof of correctness ρ.

- $(\boldsymbol{\sigma}', \rho') \xleftarrow{\$} \mathsf{Update}(1^\lambda, \boldsymbol{\sigma}, (\rho_i)_{i=1}^n)$: The updater begins by running $\hat{U} \xleftarrow{\hat{T}} \mathsf{UpdateMatrix}(1^\lambda)$, where UpdateMatrix is a probabilistic algorithm that outputs a matrix. We require that $\mathsf{Setup}(1^\lambda; \hat{T}\boldsymbol{\tau}) = (\boldsymbol{\sigma}', \cdot)$. Then it computes the update as $\boldsymbol{\sigma}' = \hat{U}\boldsymbol{\sigma}$. It outputs an updated SRS $\boldsymbol{\sigma}'$ and a proof of the correctness ρ' of the update.

- $\boldsymbol{\pi} \xleftarrow{\$} \mathsf{Prove}(R, \boldsymbol{\sigma}, \phi, w)$: The prover runs $\hat{P} \xleftarrow{\$} \mathsf{ProveMatrix}(R, \phi, w)$ where ProveMatrix is a probabilistic algorithm that outputs a matrix. Then it outputs $\boldsymbol{\pi} \leftarrow \hat{P}\boldsymbol{\sigma}$. It returns $\boldsymbol{\pi}$.

- $0/1 \xleftarrow{\$} \mathsf{Verify}(R, \boldsymbol{\sigma}, \phi, \boldsymbol{\pi})$: The verifier runs a DPT algorithm $\boldsymbol{t}(X_1, \ldots, X_{q+\ell}) \leftarrow \mathsf{Test}(R, \phi)$ to get a vector of multi-variate polynomials. It returns 1 if $\boldsymbol{t}(\boldsymbol{\sigma}, \boldsymbol{\pi}) = \mathbf{0}$ and 0 otherwise.

We consider a NILP to be pairing based when the verifier's testing polynomial outputs a vector with maximum degree 2.

3.1.8 Disclosure Freeness

We require that the prover learns no useful information from the reference strings. Like Groth [72] we argue that this scenario is achieved by a disclosure free reference string, i.e. if an adversary outputs a polynomial and afterwards the setup outputs two reference strings, then the probability that the polynomial evaluates to zero on one string but not the other is negligible.

Definition 3.1.6 (Disclosure Free)**.** *A NILP is disclosure free if for all adversariers $\mathscr{A}$ we have that*

$$\Pr\left[\begin{array}{c} \boldsymbol{f}(X_1, \ldots, X_q) \xleftarrow{\$} \mathscr{A}(1^\lambda);\ (\boldsymbol{\sigma}_1, \rho_1), (\boldsymbol{\sigma}_2, \rho_2) \xleftarrow{\$} \mathsf{Setup}(1^\lambda) : \\ \boldsymbol{f}(\boldsymbol{\sigma}_1) = \mathbf{0} \ \textit{ if and only if } \ \boldsymbol{f}(\boldsymbol{\sigma}_2) = \mathbf{0} \end{array}\right] \approx 1$$

3.2 Assumptions

Like all other constant-sized NIZK schemes in the literature, we use so-called "knowledge-of-exponent" assumptions. These are non-falsifiable in the sense that proving them false would require proving the *non-existence* of an extractor. It remains

an interesting open question as to whether it is possible to build constant-sized NIZKs from more standard assumptions in the random oracle model, and indeed Bitansky et al. [98] demonstrated that this problem is equivalent to the construction of extractable collision-resistant hash functions.

3.2.1 Knowledge of Exponent Assumptions

The knowledge of exponent assumption (KEA) introduced by Damgård [99] assumes that given group elements $g_1, g_2 = g_1^{\alpha}$ it is infeasible to create A, B such that $B = A^{\alpha}$ without knowing an exponent c such that $A = g_1^c$ and $B = g_2^c$. Bellare and Palacio [100] extended this to the KEA3 assumption, which says that given $g, g^{\alpha}, g^s, g^{\alpha s}$ it is infeasible to create A, A^{α} without knowing c_0, c_1 such that $A = g^{c_0}(g^s)^{c_1}$. This assumption has also been used in symmetric bilinear groups by Abe and Fehr [101], who called it the extended knowledge-of-exponent assumption. The bilinear knowledge of exponent assumption (B-KEA), which Abdolmaleki et al. [83] refer to as the BDH-KE assumption, generalizes further to asymmetric groups. It states that it is infeasible to compute A, B such that $e(A,h) = e(g,B)$ without knowing s such that $(A,B) = (g^s, h^s)$. It corresponds to the special case of $q = 0$ of the q-power knowledge of exponent (q-PKE) assumption which we specify later.

Our strongest knowledge assumption is the extended power knowledge of exponent (XPKE) assumption, which we use to prove our SE-SNARK secure in Chapter 4. We consider an adversary with access to source group elements whose discrete logarithms are polynomials evaluated on secret random variables. The assumption says that the only way the adversary can produce group elements in the two source groups with matching discrete logarithms, i.e., $g^a \in \mathbb{G}_1$ and $h^a \in \mathbb{G}_2$, is if it knows a linear combination of polynomials that evaluates to a. We assume that g, h are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively (the assumption still holds if $\mathbb{G}_1 = \mathbb{G}_2$, provided the generators are sampled independently).

Assumption 3.2.1 ((d,q)-XPKE)**.** *Let $\mathscr{A}$ be an adversary and let $\mathscr{X}$ be an extractor. Define the advantage* $\mathsf{Adv}^{\mathsf{XPKE}}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A},\mathscr{X}_A}(\lambda) = \Pr[\mathsf{XPKE}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A},\mathscr{X}}(\lambda)]$ *where* $\mathsf{XPKE}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A},\mathscr{X}}$ *is defined as below.*

$\underline{\text{MAIN } \mathsf{XPKE}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A},\mathscr{X}}(\lambda)}$

$\mathsf{bp} \leftarrow \mathsf{BGen}(1^\lambda)$;

$\boldsymbol{x} \leftarrow \mathbb{Z}_p^q$; $Q \leftarrow \emptyset$

$(g^a, h^b) \xleftarrow{r} \mathscr{A}^{\mathscr{O}^1,\,\mathscr{O}^2}(\mathsf{bp})$

$\boldsymbol{\eta} \in \mathbb{Z}_p^{|Q|} \xleftarrow{\$} \mathscr{X}(\mathsf{bp}; r)$;

return 1 if $a = b$ *and* $b \neq \sum_{b_j \in Q} \eta_j b_j(\boldsymbol{x})$

else return 0

$\underline{\mathscr{O}^1(a_i(X_1,\ldots,X_q))}$

assert $\deg(a_i) \leq d$

return $g^{a_i(\boldsymbol{x})}$

$\underline{\mathscr{O}^2(b_j(X_1,\ldots,X_q))}$

assert $\deg(b_j) \leq d$

$Q = Q \cup \{b_j\}$

return $h^{b_j(\boldsymbol{x})}$

The $(d(\lambda), q(\lambda))$-XPKE *assumption holds relative to* BGen *if for all non-uniform PPT adversaries* $\mathscr{A}$*, there exists a non-uniform PPT algorithm* $\mathscr{X}$ *such that* $\mathsf{Adv}^{\mathsf{XPKE}}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A},\mathscr{X}}$ *is negligible in* λ.

In proving our updatable zk-SNARK secure in Chapter 6, we can weaken this assumption slightly in the sense that the adversary gets access to source group elements that have discrete logarithms that are monomials, as opposed to polynomials, evaluated on secret random variables.

Assumption 3.2.2 ((d,q)-MKE)**.** *Let* $\mathscr{A}$ *be an adversary and let* $\mathscr{X}$ *be an extractor. Define* $\mathsf{Adv}^{\mathsf{MKE}}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A},\mathscr{X}_A}(\lambda) = \Pr[\mathsf{MKE}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A},\mathscr{X}}(\lambda)]$ *where* $\mathsf{MKE}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A},\mathscr{X}}$ *is defined as below.*

$\underline{\text{MAIN } \mathsf{MKE}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A},\mathscr{X}}(\lambda)}$

$\mathsf{bp} \leftarrow \mathsf{BGen}(1^\lambda)$;

$\boldsymbol{x} \leftarrow \mathbb{Z}_p^q$; $Q \leftarrow \emptyset$

$(g^a, h^b) \xleftarrow{r} \mathscr{A}^{\mathscr{O}^1,\, \mathscr{O}^2}(\mathsf{bp})$

$\boldsymbol{\eta} \in \mathbb{Z}_p^{|Q|} \xleftarrow{\$} \mathscr{X}(\mathsf{bp}, r)$;

return 1 if $a = b$ *and* $b \neq \sum_{b_j \in Q_2} \eta_j b_j(\boldsymbol{x})$

else return 0

$\underline{\mathscr{O}^1(a_i(X_1,\ldots,X_q))}$

assert a_i *is a monomial*

assert $\deg(a_i) \leq d$

return $g^{a_i(\boldsymbol{x})}$

$\underline{\mathscr{O}^2(b_j(X_1,\ldots,X_q))}$

assert b_i *is a monomial*

assert $\deg(b_j) \leq d$

$Q = Q \cup \{b_j\}$

return $h^{b_j(\boldsymbol{x})}$

The $(d(\lambda), q(\lambda))$-MKE *assumption holds relative to* BGen *if for all non-uniform PPT adversaries* $\mathscr{A}$*, there exists a non-uniform PPT algorithm* $\mathscr{X}$ *such that* $\mathsf{Adv}^{\mathsf{MKE}}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A},\mathscr{X}}$ *is negligible in* λ.

Our Sonic construction in Chapter 7 uses our weakest extractor assumption; we require the q-power knowledge of exponent assumption introduced by Groth [102].

Assumption 3.2.3 (q-PKE assumption)**.** *Let* $\mathscr{A}$ *be an adversary and let* $\mathscr{X}$ *be an extractor. Define* $\mathsf{Adv}^{\mathsf{PKE}}_{\mathsf{BGen},q(\lambda),\mathscr{A},\mathscr{X}}(\lambda) = \Pr[\mathsf{PKE}_{\mathsf{BGen},q(\lambda),\mathscr{A},\mathscr{X}}(\lambda)]$ *where* $\mathsf{PKE}_{\mathsf{BGen},q(\lambda),\mathscr{A},\mathscr{X}}$ *is defined as below.*

$\underline{\text{MAIN } \mathsf{PKE}_{\mathsf{BGen},q(\lambda),\mathscr{A},\mathscr{X}}(\lambda)}$

$\mathsf{bp} \leftarrow \mathsf{BGen}(1^\lambda)$;

$\alpha, x \xleftarrow{\$} \mathbb{Z}_p$;

$(g^a, h^b) \xleftarrow{r} \mathscr{A}(bp, \{g^{x^i}, g^{\alpha x^i}, h^{x^i}, h^{\alpha x^i}\}_{i=-q}^{q})$

$(\boldsymbol{a}, \boldsymbol{b}) \leftarrow \mathscr{X}(\mathsf{bp}, \{g^{x^i}, g^{\alpha x^i}, h^{x^i}, h^{\alpha x^i}\}_{i=-q}^{q}, r)$

return 1 if $a = b$ *and* $b \neq \sum_{i=-q}^{q} a_i x^i + b_i \alpha x^i$

else return 0

The $q(\lambda)$-PKE assumption holds relative to BGen *if for all non-uniform PPT adversaries $\mathscr{A}$, there exists a non-uniform PPT extractor $\mathscr{X}$ such that $\mathsf{Adv}^{\mathsf{PKE}}_{\mathsf{BGen},q(\lambda),\mathscr{A},\mathscr{X}}$ is negligible in λ.*

Plausibility of the assumptions: To be plausible an assumption should not be trivial to break using generic group operations. There are various ways to formalize generic group models that restrict the adversary to such operations [103, 104, 105]. Using the framework from [106] it is easy to show that each of these knowledge assumptions hold in the generic group model. Our assumptions are proven in asymmetric bilinear groups only.

3.2.2 Computational Assumptions

In addition to the knowledge of exponent assumptions, our security depends on a number of computational q-type assumptions that are secure in the generic group model.

Our strongest computational assumption is the computational polynomial assumption (Poly) used to prove our SE-SNARK secure. The Poly assumption is related to the d-linear assumption of Escala, Herold, Kiltz, Ràfols and Villar [107]. In the univariate case, the Poly assumption says that for any $g \in \mathbb{G}_1^*$, given $g^{a_1(x)}, \ldots, g^{a_m(x)}$, an adversary cannot compute $g^{a(x)}$ for a polynomial g that is linearly independent from $a_1, \ldots, a_m$ - even if it knows $h^{a(x)}$ for $h \in \mathbb{G}_2^*$.

Assumption 3.2.4 ((d,q)-Poly)**.** *Let $\mathscr{A}$ be a PPT algorithm, and define the advantage $\mathsf{Adv}^{\mathsf{Poly}}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A}}(\lambda) = \Pr[\mathsf{Mono}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A}}(\lambda)]$ where $\mathsf{Poly}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A}}$ is defined below.*

$\underline{\text{MAIN } \mathsf{Poly}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A}}(\lambda)}$

$\mathsf{bp} \leftarrow \mathsf{BGen}(1^\lambda)$;

$\boldsymbol{x} \leftarrow \mathbb{Z}_p^q$; $Q \leftarrow \emptyset$

$(g^a, a(X_1,\ldots,X_q)) \xleftarrow{\$} \mathscr{A}^{\mathscr{O}^1,\,\mathscr{O}^2}(\mathsf{bp})$

return 1 *if* $a = a(\boldsymbol{x})$ *and* $a(X_1,\ldots,X_q) \notin span\{Q\}$

else return 0

$\underline{\mathscr{O}^1(a_i(X_1,\ldots,X_q))}$

assert $\deg(a_i) \leq d$

$Q \leftarrow Q \cup \{a_i\}$

return $g^{a_i(\boldsymbol{x})}$

$\underline{\mathscr{O}^2(b_j(X_1,\ldots,X_q))}$

assert $\deg(b_j) \leq d$

return $h^{b_j(\boldsymbol{x})}$

The $(d(\lambda),q(\lambda))$-Poly *assumption holds relative to* BGen *if for all non-uniform PPT adversaries* $\mathscr{A}$ *we have* $\mathsf{Adv}^{\mathsf{Poly}}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A}}$ *is negligible in* λ.

In proving our updatable zk-SNARK secure in Chapter 6, we can weaken this assumption slightly in the sense that the adversary gets access to source group elements that have discrete logarithms that are monomials, as opposed to polynomials, evaluated on secret random variables. The following multivariate computational assumption is closely related to the univariate q-bilinear gap assumption of Ghadafi and Groth [108].

Assumption 3.2.5 ((d,q)-Mono)**.** *Let* $\mathscr{A}$ *be a PPT algorithm, and define the advantage* $\mathsf{Adv}^{\mathsf{Mono}}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A}}(\lambda) = \Pr[\mathsf{Mono}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A}}(\lambda)]$ *where* $\mathsf{Mono}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A}}$ *is defined below.*

$\underline{\text{MAIN } \mathsf{Mono}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A}}(\lambda)}$

$\mathsf{bp} \leftarrow \mathsf{BGen}(1^\lambda)$;

$\boldsymbol{x} \leftarrow \mathbb{Z}_p^q$; $Q \leftarrow \emptyset$

$(g^a, a(X_1,\ldots,X_q)) \xleftarrow{\$} \mathscr{A}^{\mathscr{O}^1,\,\mathscr{O}^2}(\mathsf{bp})$

return 1 *if* $a = a(\boldsymbol{x})$ *and* $a(X_1,\ldots,X_q) \notin span\{Q\}$

else return 0

$\underline{\mathscr{O}^1(a_i(X_1,\ldots,X_q))}$

assert a_i *is a monomial*

assert $\deg(a_i) \leq d$

$Q \leftarrow Q \cup \{a_i\}$

return $g^{a_i(\boldsymbol{x})}$

$\underline{\mathscr{O}^2(b_j(X_1,\ldots,X_q))}$

assert b_j *is a monomial*

assert $\deg(b_j) \leq d$

return $h^{b_j(\boldsymbol{x})}$

The $(d(\lambda),q(\lambda))$-Mono *assumption holds relative to* BGen *if for all non-uniform PPT adversaries* $\mathscr{A}$ *we have* $\mathsf{Adv}^{\mathsf{Mono}}_{\mathsf{BGen},d(\lambda),q(\lambda),\mathscr{A}}$ *is negligible in* λ.

Our Sonic construction in Chapter 7 also depends on a weakening of the Mono assumption. This assumption states that the adversary cannot compute $\frac{f(x)}{\alpha}$ in the target group. We call it the q-Bilinear Target Polynomial Fraction assumption (q-BTPF).

Assumption 3.2.6 (q-BTPF)**.** *Let* $\mathscr{A}$ *be a PPT algorithm, and define the advantage* $\mathsf{Adv}^{\mathsf{BTPF}}_{\mathsf{BGen},q(\lambda),\mathscr{A}}(\lambda) = \Pr[\mathsf{BTPF}_{\mathsf{BGen},q(\lambda),\mathscr{A}}(\lambda)]$ *where* $\mathsf{BTPF}_{\mathsf{BGen},q(\lambda),\mathscr{A}}$ *is defined below.*

$\underline{\text{MAIN } \mathsf{BTPF}_{\mathsf{BGen},q(\lambda),\mathscr{A}}(\lambda)}$

$\mathsf{bp} \leftarrow \mathsf{BGen}(1^\lambda)$

$x \xleftarrow{\$} \mathbb{Z}_p; Q \leftarrow \emptyset$

$(T, f(X)) \leftarrow \mathscr{A}\left(\mathsf{bp}, \{g^{x^i}, g^{\alpha x^i}, h^{x^i}, h^{\alpha x^i}\}_{i=-q}^{q}\right)$

return 1 *if* $T = e(g,h)^{\frac{f(x)}{\alpha}} \;\wedge\; f(X) \neq 0 \wedge \deg(f(X)) \leq q$

else return 0

The $(q(\lambda))$-BTPF *assumption holds relative to* BGen *if for all non-uniform PPT adversaries* $\mathscr{A}$ *we have* $\mathsf{Adv}^{\mathsf{BTPF}}_{\mathsf{BGen},q(\lambda),\mathscr{A}}$ *is negligible in* λ.

The remaining assumptions provided in this section are all weaker computational assumptions that are used to prove our Sonic construction secure.

Ghadafi and Groth [108] showed that a large class of computation assumptions in bilinear groups are implied by the q-bilinear generalised Diffie-Hellman Exponent (q-BGDHE) assumption. We use their results to base the security of our scheme on their assumption. The assumption we provide is a slight strengthening: we give all powers of x in both groups, but we do not give αx^q in the first group, and this is the component that the adversary is required to compute.

Assumption 3.2.7 (q-BGDHE)**.** *Let $\mathscr{A}$ be a PPT algorithm, and define the advantage* $\mathsf{Adv}^{\mathsf{BGDHE}}_{\mathsf{BGen},q(\lambda),\mathscr{A}}(\lambda) = \Pr[\mathsf{BGDHE}_{\mathsf{BGen},q(\lambda),\mathscr{A}}(\lambda)]$ *where* $\mathsf{BGDHE}_{\mathsf{BGen},q(\lambda),\mathscr{A}}$ *is defined below.*

$$\underline{\text{MAIN } \mathsf{BGDHE}_{\mathsf{BGen},q(\lambda),\mathscr{A}}(\lambda)}$$
$$\mathsf{bp} \leftarrow \mathsf{BGen}(1^\lambda)$$
$$x \xleftarrow{\$} \mathbb{Z}_p;\ Q \leftarrow \emptyset$$
$$A \leftarrow \mathscr{A}(\mathsf{bp}, \{g^{x^i}, h^{x^i}\}_{i=0}^{2q}, \{g^{\alpha x^i}, h^{\alpha x^i}\}_{i=0, i\neq q}^{2q})$$
$$\textit{return } 1 \textit{ if } A = g^{\alpha x^q}$$
$$\textit{else return } 0$$

The $(q(\lambda))$-BGDHE *assumption holds relative to* BGen *if for all non-uniform PPT adversaries $\mathscr{A}$ we have* $\mathsf{Adv}^{\mathsf{BGDHE}}_{\mathsf{BGen},q(\lambda),\mathscr{A}}$ *is negligible in λ.*

Our next assumption is an adaptation of the q-SDH assumption due to Boneh and Boyen [109] which we call the q Bilinear Generalised Strong Diffie-Hellman (q-BGSDH) assumption. We require the assumption to hold in bilinear groups and we additionally give the adversary the negative powers of x. We justify this assumption in the generic group model.

Assumption 3.2.8 (q-BGSDH)**.** *Let $\mathscr{A}$ be a PPT algorithm, and define the advantage* $\mathsf{Adv}^{\mathsf{BGSDH}}_{\mathsf{BGen},q(\lambda),\mathscr{A}}(\lambda) = \Pr[\mathsf{BGSDH}_{\mathsf{BGen},q(\lambda),\mathscr{A}}(\lambda)]$ *where* $\mathsf{BGSDH}_{\mathsf{BGen},q(\lambda),\mathscr{A}}$ *is defined below.*

$\underline{\text{MAIN } \mathsf{BGSDH}_{\mathsf{BGen},q(\lambda),\mathscr{A}}(\lambda)}$

$\mathsf{bp} \leftarrow \mathsf{BGen}(1^\lambda)$

$x \xleftarrow{\$} \mathbb{Z}_p;\ Q \leftarrow \emptyset$

$(A,c) \leftarrow \mathscr{A}(\mathsf{bp}, \{g^{x^i}, g^{\alpha x^i}, h^{x^i}, h^{\alpha x^i}\}_{i=-q}^{q})$

return 1 *if* $A = g^{\frac{\alpha}{x+c}} \wedge c \neq 0$

else return 0

The $(q(\lambda))$-BGSDH *assumption holds relative to* BGen *if for all non-uniform PPT adversaries* $\mathscr{A}$ *we have* $\mathsf{Adv}^{\mathsf{BGSDH}}_{\mathsf{BGen},q(\lambda),\mathscr{A}}$ *is negligible in* λ.

Since it does not trivially hold, we prove the q-BGSDH assumption in the generic group model.

Lemma 3.2.9. *The* q-BGSDH *assumption holds in the generic group model.*

Proof. Suppose that an adversary outputs $g^{\frac{\alpha}{x+c}}, c$. Then there exists an extractor that outputs $a_{-q} \ldots a_q$ such that

$$a_{-q}X^{-q} + \ldots + a_qX^q = \frac{1}{X+c}$$

implying that

$$ca_{-q}X^{-q} + (a_{-q} + ca_{-q+1})X^{-q+1} + \ldots + (a_{-1} + ca_0) + \ldots + (a_{q-1} + ca_qX^q) + a_qX^{q+1} = 1.$$

Then $a_q = 0$, which implies that $a_{q-1} = 0$, which implies that $a_{q-2} = 0$. Continuing in this fashion we get that $a_q, \ldots, a_0 = 0$. Thus $a_{-1} = 1$. Also, $ca_{-q} = 0$ and since $c \neq 0$, $a_{-q} = 0$. This implies that $a_{-q+1} = 0$, which implies that $a_{q+2} = 0$. Continuing in this fashion we get that $a_{-q}, \ldots, a_{-1} = 0$, contradicting our previous result that $a_{-1} = 1$. □

3.3 Arithmetic Circuits

Arithmetic circuits are a means to describe computations that consist solely of field additions and multiplications. An arithmetic circuit is described over a field $\mathbb{F}$ and

consists of gates connected together by wires. The gates specify an operation (either addition or multiplication) and the wires contain values in $\mathbb{F}$. We say that the gates are fan-in 2, meaning that each gate has two wires leading into it. Each gate has a left input wire and a right input wire leading into it, and an output wire leading from it. The circuit can have split wires i.e. the same wire leads into multiple gates. The circuit is satisfied if for every gate, the operation applied to the input wires is equal to the output wire.

Any NP relation can be described with a family of arithmetic circuits that decide which statement and witness pairs are included. In a relation described by an arithmetic circuit, an instance is defined by a value assignment to ℓ fixed input/output wires. The witness is the values of the remaining $m-\ell$ wires such that the arithmetic circuit is satisfied. For example, to encode a relation $x^2 - y^2 = 1$, The circuit would split the first wire a_1. It would use a_1 as both a left input and a right input into a multiplication gate to get the output wire a_2. The circuit would also split the third wire a_3. and would use it as both a left input and a right input into a multiplication gate to get the output wire a_4. Finally, the circuit would use a_2 and a_4 as inputs into an addition gate to get the output wire a_5. If $a_5 = 1$ then the circuit is satisfied.

As an example, consider the arithmetic circuit in Figure 3.1. It checks that a value is equal to $a^s b^v$ for known s and unknown v. This problem is used to prove ownership of a coin in Zerocoin [110]. The blue values are known inputs/outputs and the red values are unknown inputs/outputs. In this circuit, the instance is always the same except for the last input wire a^s. The red values should not be revealed from a zero-knowledge proof. First the value v is decomposed into its binary form $v = \sum_{i=0}^{N-1} v_i 2^i$. The bits v_i are used as input to the circuit. Arithmetic circuits accept field elements as opposed to bits, thus we check that each of the bits v_i are equal to 0 or 1 by checking that $v_i(v_i - 1) = 0$. For each v_i, we then find $b^{v_i 2^i}$ by calculating $v_i(b^{2^i} - 1) + 1$. Observe that this value is equal to 1 when $v_i = 0$ and b^{2^i} when $v_i = 1$. Further observe that $b^{0 \times 2^i} = 1$ and that $b^{1 \times 2^i} = b^{2^i}$. By multiplying these values together we get $b^{\sum_{i=0}^{N-1} v_i 2^i}$ which equals b^v. Finally we multiply the result b^v by a^s to get our final output.

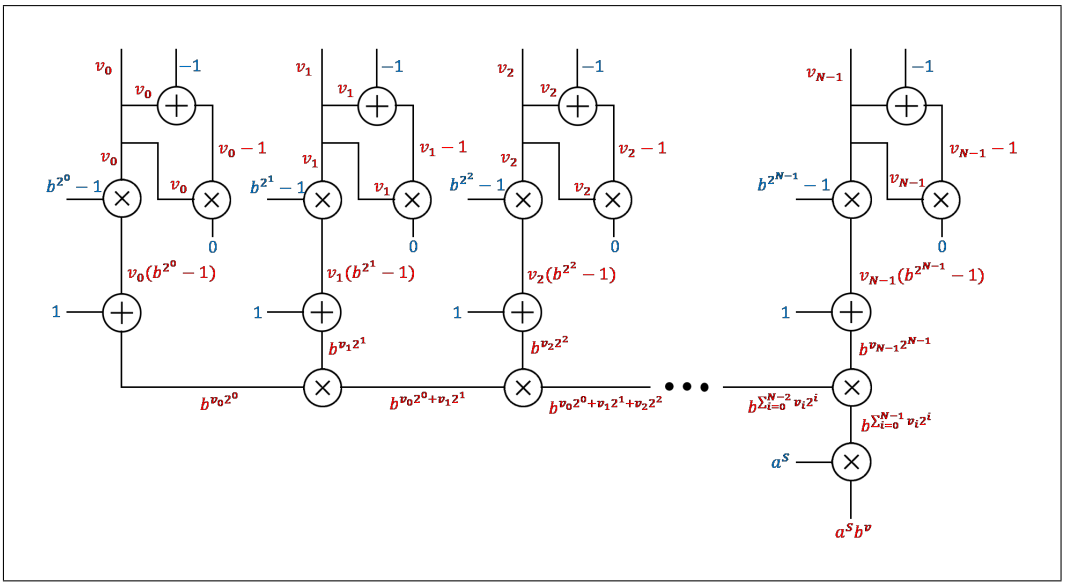

Figure 3.1: Arithmetic circuit for calculating $a^s b^v$ where a, b, s are known and v is unknown.

Chapter 4

Snarky Signatures

This chapter presents a simulation-extractable SNARK published at Crypto 2017 [21] together with Jens Groth. The security of the construction in [21] relies upon not giving the generator of the first source group, which is implicitly possible because the setup algorithm can scale the generators with hidden randomness. In this thesis the generators are explicitly given and scaled, which should help avoid mistakes at the implementation level. New to this thesis is an algorithm for verifying the structure of the reference string output by the setup, which is used to prove subversion zero-knowledge. The construction is not updatable, indeed it is demonstrated how the monomial extractor from Chapter 5 could be used to break security if an update algorithm existed.

4.1 Our Techniques

Our SE-SNARK takes inspitation from that of Groth [72], which itself optimises standard techniques for building SE-SNARKs. First, there is a trusted party that outputs a relation specific SRS. Then the prover outputs an instance and a proof consisting of group elements. Then the verifier checks that the proof satisfies a pairing equation determined by the instance; the prover can only find verifying group elements if it knows a witness to the instance.

Let us provide some intuition as to why pairing-based zk-SNARKs are, typically speaking, not simulation-extractable. The problem is that an adversary that sees a proof is often able to modify it into a different proof for the same instance. Such

modifications do not violate standard zk-SNARKs, however, for SE-SNARKs an adversary may request a simulated proof for a false instance, and then modify it into a different proof for the same false instance, which breaks simulation-extractability.

In the case of Groth's zk-SNARK, suppose for an instance ϕ that (A,B,C) are three group elements in a proof that satisfy the verification equations. The verification equation is then given by

$$e(A,B) = e(g^{\alpha},h^{\beta})e(g^{f(\phi)},h)e(C,h^{\delta}) \tag{4.1}$$

for a known polynomial f in ϕ and some secret α,β,δ.

There are two methods to generically randomise a proof A,B,C that satisfy (4.1). An adversary can either set

$$A' = A^{r};\ B' = B^{\frac{1}{r}};\ C' = C$$

or they can set

$$A' = A;\ B' = Bh^{r\delta};\ C' = A^{r}C$$

for any field element r.

To neutralise the first attack we add the verification equation

$$e(A,h) = e(g,B).$$

However this still leaves the case where $r = -1$. So we further take the elements $g^{\alpha\delta},h^{\beta\delta}$ into the quadratic constraint i.e. rather than using $e(A,B)$ in the verification equation we use $e(Ag^{\alpha\delta},Bh^{\beta\delta})$.

To neutralise the second attack our SRS is designed to contain h^{δ}, $g^{\gamma\delta}$ and $h^{\gamma\delta}$ but not g^{δ}. That way, if the adversary sets $B' = Bh^{r\delta}$, then the only possible value for A' is $Ag^{r\delta}$ (which the adversary cannot compute). This is a simplication of the attack - in the full proof we are concerned about $B' = Bh^{\psi(\tau)\delta}$ for ψ a polynomial in the trapdoor.

4.1.1 Square Arithmetic Programs

Formally, we will be working with square arithmetic programs R that have the following description

$$R = (\mathsf{bp}, \ell, \{u_i(X), w_i(X)\}_{i=0}^{m}, t(X)),$$

where the bilinear group defines the finite field $\mathbb{Z}_p$ we will be working over, $1 \leq \ell \leq m$, $u_i(X), w_i(X), t(X) \in \mathbb{Z}_p[X]$ and $u_i(X), w_i(X)$ have strictly lower degree than n, the degree of $t(X)$. Furthermore, we require that the set $S = \{u_i(X) : 0 \leq i \leq \ell\}$ is linearly independent and that any $u_i(X) \in S$ is also linearly independent from the set $\{u_j(X) : \ell < j \leq m\}$. A square arithmetic program with such a description defines the following binary relation, where we define $a_0 = 1$,

$$R = \left\{ (\phi, w) \;\middle|\; \begin{array}{l} \phi = (a_1, \ldots, a_\ell) \in \mathbb{Z}_p^\ell \\ w = (a_{\ell+1}, \ldots, a_m) \in \mathbb{Z}_p^{m-\ell} \\ \exists\, q(X) \in \mathbb{Z}_p[X], \deg(q) \leq n-2 : \\ \qquad (\sum_{i=0}^{m} a_i u_i(X))^2 = \sum_{i=0}^{m} a_i w_i(X) + q(X)t(X) \end{array} \right\}$$

We say $\mathscr{R}$ is a bilinear group and square arithmetic program generator if it generates relations of the form given above with prime $p > 2^{\lambda-1}$.

Any arithmetic circuit can be converted into an SAP using techniques by Groth which are explained in [21]. One downside is that these techniqes require that we double the number of multiplicative constraints and this is taken into account in our efficiency comparison in Chapter 2.

4.2 Derivation of a Relation Dependent SRS

We require a reference string that depends on an SAP (we denote the SAP by `sap`). In Figure 4.2 we provide algorithm for generating the SRS and for verifying its structure. The verification algorithm is new to this thesis and we use it to prove subversion zero-knowledge. It is essential that the proof of knowledge algorithm (POK) does not reveal g^{δ} and we are implicitly assuming that the verifier checks

$\underline{\mathsf{Setup}(1^\lambda, \mathtt{sap}) \mapsto (\mathtt{srs}_{\mathtt{sap}}, \rho)\text{:}}$

$\alpha, \beta, \gamma, \delta, x \xleftarrow{\$} \mathbb{Z}_p$ such that $t(x) \neq 0$

$$\mathtt{srs}_{\mathtt{sap}} \leftarrow \left(\begin{array}{l} g^\alpha, g^\gamma, g^x, g^{\alpha\delta}, g^{\gamma\delta t(x)}, g^{\gamma^2\delta t^2(x)}, g^{(\alpha+\beta)\gamma\delta t(x)}, \\ h^\beta, h^\delta, h^{\beta\delta}, h^{\gamma\delta t(x)}, h^{\delta^2} \\ \left\{ g^{\gamma\delta x^i}, h^{\gamma\delta x^i}, g^{\gamma^2\delta t(x) x^i} \right\}_{i=0}^{n-1}, \left\{ g^{\gamma\delta w_i(x)+\delta(\alpha+\beta)u_i(x)} \right\}_{i=0}^{\ell}, \\ \left\{ g^{\gamma^2\delta w_i(x)+\gamma\delta(\alpha+\beta)u_i(x))} \right\}_{i=\ell+1}^{m}, \end{array} \right)$$

$\rho \xleftarrow{\$} \mathsf{POK}\left\{ g^\alpha, h^\beta, g^\gamma, h^\delta, g^x : \quad \alpha, \beta, \gamma, \delta, x \right\}$

return $(\mathtt{srs}, \rho)$

$\underline{\mathsf{VerifySRS}(1^\lambda, \mathtt{sap}, \mathtt{srs}_{\mathtt{sap}}, \rho) \mapsto 0/1}$

check $\mathsf{POKVerify}(\rho)$

$$\left(\begin{array}{l} A_\alpha, A_\gamma, A_x, A_{\alpha\delta}, A_{\gamma\delta t(x)}, A_{\gamma^2\delta t^2(x)}, A_{(\alpha+\beta)\gamma\delta t(x)}, \\ B_\beta, B_\delta, B_{\beta\delta}, B_{\gamma\delta t(x)}, B_{\delta^2} \\ \left\{ A_{\gamma\delta x^i}, B_{\gamma\delta x^i}, A_{\gamma^2\delta t(x) x^i} \right\}_{i=0}^{n-1}, \left\{ A_{\gamma\delta w_i(x)+\delta(\alpha+\beta)u_i(x)} \right\}_{i=0}^{\ell}, \\ \left\{ A_{\gamma^2\delta w_i(x)+\gamma\delta(\alpha+\beta)u_i(x))} \right\}_{i=\ell+1}^{m}, \end{array} \right) \leftarrow \mathsf{parse}(\mathtt{srs}_{\mathtt{qap}})$$

check that the following hold:

$$\begin{array}{lcllcl} e(A_{\alpha\delta}, h) & = & e(A_\alpha, B_\delta) & e(A_{\gamma\delta}, B_\beta) & = & e(A_\gamma, B_{\beta\delta}) \\ e(A_{\gamma\delta}, h) & = & e(A_\gamma, B_\delta) & e(A_{\gamma\delta}, B_\delta) & = & e(A_\gamma, B_{\delta^2}) \\ f(X) = \frac{t(X)-t_0}{X} & & & & & \\ e(A_{\gamma\delta t(x)}, B_{\gamma\delta}) & = & e(\prod_{i=1}^{n-1} A_{\gamma\delta x^i}^{f_i}, B_{\gamma\delta x}) e(A_{\gamma\delta}^{-t_0}, B_{\gamma\delta}) & e(A_{\gamma\delta t(x)}, B_{\gamma\delta}) & = & e(A_{\gamma\delta}, B_{\gamma\delta t(x)}) \\ e(A_{\gamma^2\delta t(x)^2}, B_\delta) & = & e(A_{\gamma\delta t(x)}, B_{\gamma\delta t(x)}) & e(A_{(\alpha+\beta)\gamma\delta t(x)}, B_\delta) & = & e(A_{\alpha\delta}, B_{\gamma\delta t(x)}) e(A_{\gamma\delta t(x)}, B_{\beta\delta}) \\ e(A_\gamma, B_\delta) & = & e(g, B_{\gamma\delta}) & & & \end{array}$$

for $0 \leq i \leq n-2$:
$e(A_x, B_{\gamma x^i}) = e(g, B_{\gamma x^{i+1}})$ and $e(A_{\gamma\delta x^i}, B_{\gamma\delta x}) = e(A_{\gamma\delta x^{i+1}}, B_{\gamma\delta})$

for $1 \leq i \leq n-1$:
$e(A_{\gamma^2\delta t(x) x^i}, B_{\gamma\delta}) = e(A_{\gamma\delta t(x)}, B_{\gamma\delta x^i})$

for $0 \leq i \leq \ell$:
$e(A_{\gamma\delta w_i(x)+\delta(\alpha+\beta)u_i(x)}, B_{\gamma\delta}) = e(A_{\gamma\delta}, \prod_{j=0}^{n-1} B_{\gamma\delta x^j}^{w_{i,j}}) e(A_{\alpha\delta}, \prod_{j=0}^{n-1} B_{\gamma\delta x^j}^{u_{i,j}}) e(\prod_{j=0}^{n-1} A_{\gamma\delta x^j}^{u_{i,j}}, B_{\beta\delta})$

for $\ell+1 \leq i \leq m$:
$e(A_{\gamma^2\delta w_i(x)+\gamma\delta(\alpha+\beta)u_i(x)}, B_\delta) = e(A_{\gamma\delta}, \prod_{j=0}^{n-1} B_{\gamma\delta x^j}^{w_{i,j}}) e(A_{\alpha\delta}, \prod_{j=0}^{n-1} B_{\gamma\delta x^j}^{u_{i,j}}) e(\prod_{j=0}^{n-1} A_{\gamma\delta x^j}^{u_{i,j}}, B_{\beta\delta})$

return 1 if all checks pass, else return 0

Figure 4.1: Algorithm for verifying the structure of the SRS in our SE-SNARK construction

that the elements in the proof of knowledge are consistent with the SRS.

4.3 Our SE-SNARK Construction

Our construction of a simulation-extractable SNARK is given in Figure 4.2. The prover parses the wires of the circuit as $(1, a_1, \ldots, a_m)$, and then embeds the SAP polynomials $a_i u_i(X)$ evaluated at the unknown point x into one proof element in the first source group and one proof element in the second source group. They provide a

Common Input: $\mathsf{info} = \mathsf{bp}, \mathtt{sap}, \mathtt{srs}_{\mathtt{sap}}, e(g^{\alpha\delta}, h^{\beta\delta})$

$\underline{\mathsf{Prove}(\mathsf{info}, \phi, w) \mapsto \pi}$:
$(a_0, a_1, \ldots, a_m) \leftarrow \mathsf{parse}(1, \phi, w)$
$q(X) \leftarrow \frac{\left(\sum_{i=0}^{m} a_i u_i(X)\right)^2 - \sum_{i=0}^{m} a_i w_i(X)}{t(X)}$
$r \xleftarrow{\$} \mathbb{Z}_p$
$A \leftarrow g^{\gamma\delta\left(rt(x) + \sum_{i=0}^{m} a_i u_i(x)\right)}$
$B \leftarrow h^{\gamma\delta\left(rt(x) + \sum_{i=0}^{m} a_i u_i(x)\right)}$
$C \leftarrow g^{\gamma\delta\left(\sum_{i=l+1}^{m} a_i(\gamma w_i(x) + (\alpha+\beta)u_i(x)) + r(\alpha+\beta)t(x) + \gamma t(x)\left[r^2 t(x) + q(x) + 2r\sum_{i=0}^{m} a_i u_i(x)\right]\right)}$
return (A, B, C)

$\underline{\mathsf{Verify}(\mathsf{info}, \phi, \pi) \mapsto 0/1}$:
$(a_0, a_1, \ldots, a_\ell) \in \mathbb{Z}_p^\ell \leftarrow \mathsf{parse}(1, \phi)$
$(A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \leftarrow \mathsf{parse}(\pi)$
check $e(Ag^{\alpha\delta}, Bh^{\beta\delta}) = e(g^{\alpha\delta}, h^{\beta\delta}) e(g^{\sum_{i=0}^{\ell} a_i \delta(\gamma w_i(x) + (\alpha+\beta)u_i(x))}, h^{\gamma\delta}) e(C, h^{\delta})$
check $e(A, h) = e(g, B)$
return 1 if both checks pass, else return 0.

Figure 4.2: Our construction of a Simulation-Extractable SNARK.

third proof element whose exponent is the product of the exponents of the first two proof elements. The verifier checks with a pairing that the first and second proof elements share common exponents. They then check, also with a pairing, that the third proof element's exponent is indeed the product of the exponents of the first two proof elements.

4.3.1 Security Proof

Theorem 4.3.1. *The construction in Figure 4.2 has subversion zero-knowledge.*

Proof. To prove subversion zero-knowledge, we need to both show the existence of a an extractor $\mathscr{X}_{\mathscr{A}}$ that can compute a trapdoor from the reference string, and describe a SimProve algorithm that produced indistinguishable proofs when provided with the extracted trapdoor. It can be seen that an adversary that outputs a verifying reference string must know $\alpha, \beta, \gamma, \delta, x$ in the exponents of $g^{\alpha}, h^{\beta}, g^{\gamma}, h^{\delta}, g^{x}$ because the proof of knowledge verifies. Furthermore, if each of the verifier's pairing checks verify then the adversary's outputted reference string has the same structure as one output

by the setup algorithm. So the $(\alpha, \beta, \gamma, \delta, x)$ that the extractor computes from the proof of knowledge is a valid trapdoor for the reference string.

A simulator is given a trapdoor $\tau = (\alpha, \beta, \gamma, \delta, x)$ and behaves as follows.

$$\underline{\mathsf{SimProve}(\mathsf{bp}, \mathtt{sap}, \mathtt{srs}_{\mathtt{sap}}, \tau, \phi) \mapsto \pi}$$
$$(a_0, a_1, \ldots, a_\ell) \leftarrow \mathsf{parse}(1, \phi)$$
$$\mu \xleftarrow{\$} \mathbb{Z}_p$$
$$A, B \leftarrow g^{\mu\delta}, h^{\mu\delta}$$
$$C \leftarrow g^{\left(\mu^2\delta + (\alpha+\beta)\mu\delta - \gamma\delta\sum_{i=0}^{\ell} a_i(\gamma w_i(x) + (\alpha+\beta)u_i(x))\right)}$$
$$\text{return } (A, B, C)$$

To see that the simulated proofs are indistinguishable from the real proofs, first observe that the simulation procedure always produces verifying proofs. Next, observe that for a given instance and proof $\pi = (A, B, C)$ the element A uniquely determines B through the second verification equation, and the elements A, B uniquely determine C through the first verification equation. In a real proof the random choice of r makes A uniformly random, and in a simulated proof the random choice of μ makes A uniformly random. So in both cases, we get the same probability distribution over proofs with uniformly random A and the unique matching B, C. □

Theorem 4.3.2. *The protocol in Figure 4.2 is simulation-extractable (implying it is knowledge sound) provided that the* $(n, q+5)$-$\mathsf{XPKE}(\lambda)$ *and* $(n, q+5)$-$\mathsf{Poly}(\lambda)$ *assumptions hold, where n is the number of squaring constraints and q the number of simulation queries the adversary asks.*

Proof. Suppose that an adversary $\mathscr{A}$ is given an $\mathtt{srs}$. It accesses its simulation oracle on the instances $(\phi_1, \ldots, \phi_q)$ to obtain the responses $(\pi_1, \ldots, \pi_q)$. We show that if $\mathscr{A}$ outputs verifying (ϕ, π) then either (ϕ, π) is one of the oracle queries and responses (ϕ_j, π_j) or there exists an extractor $\mathscr{X}_{\mathscr{A}}$ that outputs w such that $(\phi, w) \in R$.

From the second verification equation we have that $e(A, h) = e(g, B)$. From the $(n, q+5)$-XPKE assumption there exists an extractor that outputs

$$(\eta_0, \eta_\beta, \eta_\delta, \eta_{\beta\delta}, \eta_{\delta^2}, \eta_{\gamma,\delta,t}, \eta_{\gamma\delta}(X), \eta_{b,j})$$

such that

$$\log(B) = \eta_0 + \eta_\beta\beta + \eta_\delta\delta + \eta_{\beta\delta}\beta\delta + \eta_{\delta^2}\delta^2 + \eta_{\gamma\delta t}\gamma\delta t(x) + \gamma\delta\eta_{\gamma\delta}(x) + \delta\sum_j \eta_{b,j}\mu_j.$$

Taking the adversary and the extractor together, we can see them as a combined algorithm that outputs A, B, C and the formal polynomial $\eta(X, X_\beta, X_\gamma, X_x, X_\delta, X_{\mu_1}, \ldots, X_{\mu_q})$ such that $A = g^{\eta(x,\beta,\gamma,\delta,\mu_1,\ldots,\mu_q)}$. By the $(n, q+5)$-Poly assumption this has negligible probability unless η is in the span of

$$\begin{aligned}
&X_0,\ X_\alpha,\ X_\gamma,\ X_x,\ X_\alpha X_\delta,\ X_\gamma X_\delta t(X),\ X_\gamma X_\delta (t(X))^2,\ (X_\alpha + X_\beta) X_\gamma X_\delta t(X),\\
&\left\{X_\gamma X_\delta X^i,\ X_\gamma^2 X_\delta t(X) X^i\right\}_{i=0}^{n-1}, \left\{X_\gamma X_\delta w_i(X) + X_\delta(X_\alpha + X_\beta)u_i(X)\right\}_{i=0}^{\ell},\\
&\left\{X_\gamma^2 X_\delta w_i(X) + X_\gamma X_\delta (X_\alpha + X_\beta)u_i(X)\right\}_{i=\ell+1}^{m},\ \left\{X_{\mu_j} X_\delta\right\}_{j=1}^{q}\\
&\left\{\left(X_{\mu_j}^2 X_\delta + (X_\alpha + X_\beta)X_{\mu_j} X_\delta - X_\gamma X_\delta \textstyle\sum_{i=0}^{\ell} a_{\mu_j,i}(X_\gamma w_i(X) + (X_\alpha + X_\beta)u_i(X))\right)\right\}_{j=1}^{q}.
\end{aligned} \tag{4.2}$$

This means that

$$\eta(x, \beta, \gamma, \delta, \mu_1, \ldots, \mu_q) = \eta_0 + \eta_{\gamma\delta t}\gamma\delta t(x) + \gamma\delta\eta_{\gamma\delta}(x) + \delta\sum \eta_{b,j}\mu_j.$$

From the first verification equation we get that $C = g^{f(x,\beta,\gamma,\delta,\mu_1,\ldots,\mu_q)}$ where f is given by

$$\begin{aligned}
\frac{1}{\delta}\left(\eta(x, \beta, \gamma, \delta, \mu_1, \ldots, \mu_q) + \alpha\delta\right) \cdot \Big(&\eta(x, \beta, \gamma, \delta, \mu_1, \ldots, \mu_q) + \beta\delta\Big)\\
&- \alpha\beta\delta - \sum_{i=0}^{\ell} a_i\gamma\delta(\gamma w_i(x) + (\alpha + \beta)u_i(x))
\end{aligned}$$

By the $(n, q+5)$-Poly assumption this means that

$$\begin{aligned}
\frac{1}{X_\delta}\left(\eta(X, X_\beta, X_\gamma, X_\delta, X_{\mu_1}, \ldots, X_{\mu_q}) + X_\alpha X_\delta\right) \cdot \Big(&\eta(X, X_\beta, X_\gamma, X_\delta, X_{\mu_1}, \ldots, X_{\mu_q}) + X_\beta X_\delta\Big)\\
&- X_\alpha X_\beta X_\delta - \sum_{i=0}^{\ell} a_i X_\gamma X_\delta (X_\gamma w_i(X) + (X_\alpha + X_\beta)u_i(X))
\end{aligned}$$

also belongs to the span in (4.2).

The span has no polynomials of the form $\frac{1}{\delta}$ thus $\eta_0 = 0$. The span has no polynomials of the form $X_\delta X_{\mu_j} X_{\mu_k}$ for $j \neq k$ thus at most one $\eta_{b,j}$ is uncancelled. Suppose without loss of generality that $\eta_{b,j}$ are cancelled for $j \geq 2$ and rename $\eta_{b,1}$ by η_μ and μ_1 by μ.

We are now left with

$$\delta\left(\eta_{\gamma\delta t}\gamma t(x) + \gamma\eta_{\gamma\delta}(x) + \eta_\mu\mu + \alpha\right)\cdot\left(\eta_{\gamma\delta t}\gamma t(x) + \gamma\eta_{\gamma\delta}(x) + \eta_\mu\mu + \beta\right) = \alpha\beta\delta + \sum_{i=0}^{\ell} a_i\gamma\delta(\gamma w_i(x) + (\alpha+\beta)u_i(x)) + \log(C).$$

If $\eta_\mu \neq 0$ then $\eta_{\gamma\delta t}$ and $\eta_{\gamma\delta}(X)$ both cancel because the span has no polynomials of the form $X_{\mu_{b,1}} X_\gamma X_\delta$. As a result, the polynomial $f(x,\beta,\gamma,\delta,\mu_1,\ldots,\mu_q)$ extracted from C is given by

$$(\eta_\mu\mu)^2 + \eta_\mu\mu(\alpha+\beta) - \sum_{i=0}^{\ell} a_i\gamma\delta(\gamma w_i(x) + (\alpha+\beta)u_i(x)).$$

The only way to obtain the $(\eta_\mu\mu)^2$ term and the $\eta_\mu\mu(\alpha+\beta)$ term is if f contains a non-trivial linear combination of the term

$$\left(X_{\mu_1}^2 X_\delta + (X_\alpha + X_\beta)X_{\mu_1}X_\delta - X_\gamma X_\delta \sum_{i=0}^{\ell} a_{\mu_1,i}(X_\gamma w_i(X) + (X_\alpha + X_\beta)u_i(X))\right).$$

Thus $\eta_\mu^2 = \eta_\mu$ i.e. $\eta_{b,1} = 1$ and f contains exactly one of the above term. There are no polynomials in the span that can be used to balance $\gamma\delta(\alpha+\beta)u_i(x)$ because: there are no $X_\beta X_\gamma X_\delta X^i$ terms in $v(\boldsymbol{X})$; $t(X)$ has degree n which is strictly greater than the degree of the other polynomials; and the set $S = \{u_i(X) : 0 \leq i \leq \ell\}$ is linearly independent from the set $\{u_j(X) : \ell < j \leq m\}$. Hence $a_{\mu_1,i} = a_i$, i.e. $(\phi,\pi) = (\phi_1,\pi_1)$ and the adversary has regurgitated a simulated proof. Thus for all j, $\eta_{\mu_j} = 0$.

We are now left with

$$\delta\left(\eta_{\gamma\delta t}\gamma t(x)+\gamma\eta_{\gamma\delta}(x)+\alpha\right)\cdot\left(\eta_{\gamma\delta t}\gamma t(x)+\gamma\eta_{\gamma\delta}(x)+\beta\right) = \alpha\beta\delta+\sum_{i=0}^{\ell}a_i\gamma\delta(\gamma w_i(x)+(\alpha+\beta)u_i(x))+\log(C).$$

Looking at the terms involving α, we get that

$$\eta_{\gamma\delta t}\alpha\gamma\delta t(x)+\alpha\gamma\delta\eta_{\gamma\delta}(x)=\sum_{i=0}^{\ell}a_i\gamma\delta\alpha u_i(x)+\sum_{i=\ell+1}^{m}a_i\gamma\delta\alpha u_i(x)+a_{(\alpha+\beta)\gamma\delta t}\alpha\gamma\delta t(x)$$

where $a_{\ell+1},\ldots,a_m$ are the coefficients in f relating to the terms

$$\left\{X_\gamma^2X_\delta w_i(X)+X_\gamma X_\delta(X_\alpha+X_\beta)u_i(X)\right\}_{i=\ell+1}^{m}$$

and $a_{(\alpha+\beta)\gamma\delta t}$ relates to the term $(X_\alpha+X_\beta)X_\gamma X_\delta t(X)$. We see that $\eta_{\gamma\delta t}=a_{(\alpha+\beta)\gamma\delta t}$ because the degree of $t(X)$ is strictly greater than the degree of the other polynomials.

Looking at the terms involving $\gamma^2\delta$, we get that

$$\eta_{\gamma\delta t}\gamma^2\delta t^2(x)+\gamma^2\delta\eta_{\gamma\delta}^2(x)=\sum_{i=0}^{\ell}a_i\gamma^2\delta w_i(x)+\sum_{i=\ell+1}^{m}a_i\gamma^2\delta w_i(X)+\gamma^2\delta q(x)t(x)+a_{\gamma^2\delta t}t^2(X).$$

for some polynomial $q(X)$ relating to the coefficients in f that refer to the terms $X_\gamma^2X_\delta t(X)X^i$ and where $a_{\gamma^2\delta t}$ relates to the term $X_\gamma^2X_\delta t^2(X)$. We see that $\eta_{\gamma\delta t}=a_{\gamma^2\delta t}$ because the degree of $t^2(X)$ is strictly greater than the degree of the other polynomials.

Putting these two expressions for $\eta_{\gamma\delta}(X)$ together gives us that

$$\left(\sum_{i=0}^{m}a_iu_i(X)\right)^2=\sum_{i=0}^{m}a_iw_i(X)+q(X)t(X)$$

which gives us that $a_{\ell+1},\ldots,a_m$ is a valid witness for ϕ, completing our proof. □

4.3.2 Efficiency

The proof size is 2 elements in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$. Counting group elements in the SRS, we find there are $7+2n+(\ell+1)+(m-\ell)$ $\mathbb{G}_1$ elements and $5+n$ $\mathbb{G}_2$ elements (recall m is the number of wires, n is the number of gates, and ℓ is the size of the instance). Accounting for the fact that we have doubled the number of gates to obtain squaring constraints, the SRS thus contains, $8+4n+2m$ $\mathbb{G}_1$ and $5+2n$ $\mathbb{G}_2$ elements.

The verifier can work with a reduced reference string that only contains $\ell+2$ elements from $\mathbb{G}_1$, 3 elements from $\mathbb{G}_2$, and 1 element from $\mathbb{G}_T$ of the form

$$\left(g^{\alpha\delta}, h^{\delta}, h^{\beta\delta}, h^{\gamma\delta}, \{g^{\delta(\gamma w_i(x)+(\alpha+\beta)u_i(x))}\}_{i=0}^{\ell}, e(g^{\alpha\delta}, h^{\beta\delta})\right).$$

The verification consists of checking that the proof contains 3 appropriate group elements and checking 2 pairing product equations. The verifier's computation is dominated by a multi-exponentiation $\mathbb{G}_1$ to ℓ exponents (noting that $a_0 = 1$) and 5 pairings (assuming $e(g^{\alpha\delta}, h^{\beta\delta})$ is precomputed).

The prover has to compute the polynomial $q(X)$ and it depends on the relation how long this computation takes. If we construct the SAP from an arithmetic circuit where each multiplication gate connects to a constant number of wires, there is a set of distinct points $r_1, \ldots, r_n$ where the polynomials are non-zero only in a few places. In this case we can use fast polynomial manipulation techniques to compute $q(X)$ in $\tilde{O}(n)$ operations in $\mathbb{Z}_p$. The prover also computes the coefficients of $\sum_{i=0}^{m} a_i u_i(X)$, which again can be done in $\tilde{O}(n)$ operations in $\mathbb{Z}_p$ for polynomials arising from arithmetic circuits where each multiplication gate connects to a constant number of wires. Having all the coefficients of relevant polynomials, the prover's cost is dominated by $m+2n-\ell$ exponentiations in $\mathbb{G}_1$ and n exponentiations in $\mathbb{G}_2$.

Chapter 5

Updatability

This section discusses the types of reference strings that can and cannot be updated. First methods for updating universal reference strings that contain only monomials are discussed. These methods are joint work with Markulf Kohlweiss and are based on work published at Crypto 2018 [22]. Next this section discusses an impossibility result: updating a reference string reveals the monomials. This impossibility result was also presented in [22]; the author is the primary investigator and has been aided by Markulf Kohlweiss and Sarah Meiklejohn.

5.1 Simulation Sound Proofs of Knowledge

A key property of a reference string that contains only monomials is that it is possible to update it and prove that the update was applied correctly. As long as one updater behaves according to the protocol, the resulting parameters are secure. Further, if all of the updaters are colluding then we can build a simulator that knows the trapdoor. In order to ensure trapdoor extraction we utilise proofs of knowledge (POKs) that can be build from trustless setups.

One suggestion for POKs is Fischlin transformed Sigma protocols [49] which are zero-knowldge in the ROM and have straight line extractors. Fischlin's trick is to enforce that any verifying proof must have a number of lower order bits to be zero in the hash of the proof. This extra constraint makes it highly probable that a prover queries their random oracle at least twice, thus removing the need for the forking lemma. Bernhard, Fischlin and Warinschi showed that a Fischlin transformed

Sigma protocol is a simulation-sound adaptive proof in the random oracle model (see Theorem 1 [111]). These properties are helpful for our updatability proofs, because by disallowing an adversary to output a previously seen update proof, we ensure that an adversary cannot utilise previous update proofs. Fischlin transformed proofs have high prover costs. However, our language is very simple (knowledge of a small number of exponents), so they are practical enough for our purposes.

A second suggestion for POKs is to work directly with a KEA assumption and have the prover provide A and B such that

$$e(A,h) = e(g,B).$$

In [22] we show how, when this is completed together with a computational requirement, it is possible to prevent the adversary from gaining any advantage from old proofs. In this thesis, we assume Fischlin transformed Sigma protocols are used for the POK.

5.2 Updating Reference Strings with Monomials

Our structured reference strings are sampled from specified distributions and soundness is only guaranteed if the SRSs are sampled from the correct distribution. In this sense we sometimes refer to the SRS being "well-formed". Ideally we sould like to verify that the SRS is sampled correctly using only pairing equations with elements in the SRS. We formalise this notion with progression full sets. The idea is that given a handful of base elements, the verifier can inductively check that higher degree monomials are consistent with lower degree monomials.

5.2.1 Progression Full Sets

Consider a reference string of monomials. Then the reference string is updatable if the monomials can be described by progression-full sets, a notion which we provide below.

The sets $\mathscr{U}$ and $\mathscr{V}$ contains vectors of integers with length q, where q is the number of variables required by the setup algorithm and the vector represents the

exponents of the variables. The degree of elements of the form $(a_1,\ldots,a_q)$ in the sets $\mathscr{U}$ or $\mathscr{V}$ is equal to $d=\sum_{i=1}^{q}|a_i|$.

Let U_i, V_i be the subsets of $\mathscr{U}$, $\mathscr{V}$ that contains degree i elements. Elements in $\mathscr{U}$, $\mathscr{V}$ that have degree 1 and whose non-zero entry is $+1$ (as opposed to -1) are the base elements. Let U_b, V_b denote the sets of base elements. Then $\mathscr{U}$ and $\mathscr{V}$ are progression-full if:

1. for every $\boldsymbol{u}\in U_1\setminus U_b$ and for every $\boldsymbol{v}\in V_1\setminus V_b$, we have that $-\boldsymbol{u}\in V_1$ and $-\boldsymbol{v}\in U_1$;

2. for every $\boldsymbol{u}\in U_i$ and for every $\boldsymbol{v}\in V_i$, there exists some $\boldsymbol{a}_1,\boldsymbol{a}_2\in\bigcup_{j=1}^{i-1}\mathscr{U}_j$ and $\boldsymbol{b}_1,\boldsymbol{b}_2\in\bigcup_{j=1}^{i-1}\mathscr{V}_j$ such that

$$\boldsymbol{u}=\boldsymbol{a}_1+\boldsymbol{b}_1 \quad \boldsymbol{v}=\boldsymbol{a}_2+\boldsymbol{b}_2.$$

5.2.2 Chain Proofs

The verifier would like to be assured that all the randomness added in each update is included in the final reference string. To ensure this, the update proofs include a "chain proof", which consists of a handful of group elements. By including the chain proofs, the update string does not need to include the (sizeable) intermediary SRSs.

The chain of proofs is updated step by step, and our chain prover only requires the previous link in the chain. The ith link in the chain is denoted by chain_i. The chain considers each element in U_b and V_b separately, and distinguishes between those in $U_b\cap V_b$ and those in V_b/U_b. The ith chain proof for an element $\boldsymbol{u}$ is denoted by $\mathsf{chain}_i[\boldsymbol{u}]$. The verifier requires the entire chain. The element $\mathsf{chain}_0[\boldsymbol{u}]$ is taken by both prover and verifier to be (g,h) if $\boldsymbol{u}\in U_b\cap V_b$ and (g,g,h) if $\boldsymbol{u}\in V_b\setminus U_b$.

For our reference strings we are assuming that all degree 1 elements are either of the form $\boldsymbol{u}\in U_b\cap V_b$; or are of the form $\boldsymbol{v}\in V_b/U_b$. By progression fullness there exists $\boldsymbol{u}\in U_1$ such that $\boldsymbol{u}+\boldsymbol{v}\in U_2$. This covers all the updatable reference strings required for this thesis. We are implicitely assuming that there will be a check in the verification that ensures that the final elements in the chain are consistent with the base elements in the final `srs`.

$\underline{\mathsf{ChainProve}(1^\lambda, \mathscr{U}, \mathscr{V}, \mathsf{chain}_i, \boldsymbol{x}) \mapsto \mathsf{chain}_{i+1}}$
for $\boldsymbol{u} \in U_b \cap V_b$:
  $(X, A) \leftarrow \mathsf{parse}(\mathsf{chain}_i[\boldsymbol{u}])$
  $s \leftarrow \boldsymbol{u} \cdot \boldsymbol{x}$
  $\mathsf{chain}_{i+1}[\boldsymbol{u}] \leftarrow (X^s, h^s)$

for $\boldsymbol{v} \in V_b / U_b$:
  $(W, Z, C) \leftarrow \mathsf{parse}(\mathsf{chain}_i[\boldsymbol{v}])$
  $y \leftarrow \boldsymbol{v} \cdot \boldsymbol{x}$
  $\boldsymbol{u} \leftarrow$ vector in U_1 such that $\boldsymbol{u} + \boldsymbol{v} \in U_2$
  $s \leftarrow \boldsymbol{u} \cdot \boldsymbol{x}$
  $\mathsf{chain}_{i+1}[\boldsymbol{v}] \leftarrow (Z^s, Z^{sy}, h^y)$

return chain

$\underline{\mathsf{ChainVerify}(1^\lambda, \mathscr{U}, \mathscr{V}, \mathsf{chain}) \mapsto 0/1}$
for $\boldsymbol{u} \in U_1 \cap V_1$:
  for $1 \leq i \leq \mathsf{len}(\mathsf{chain})$:
    $(X_{i-1}, A_{i-1}) \leftarrow \mathsf{parse}(\mathsf{chain}_{i-1}[\boldsymbol{u}])$
    $(X_i, A_i) \leftarrow \mathsf{parse}(\mathsf{chain}_i[\boldsymbol{u}])$
    check $e(X_i, h) = e(X_{i-1}, A_i)$

for $\boldsymbol{v} \in V_1 / U_1$
  for $1 \leq i \leq \mathsf{len}(\mathsf{chain})$:
    $(W_{i-1}, Z_{i-1}, C_{i-1}) \leftarrow \mathsf{parse}(\mathsf{chain}_{i-1}[\boldsymbol{v}])$
    $(W_i, Z_i, C_i) \leftarrow \mathsf{parse}(\mathsf{chain}_i[\boldsymbol{v}])$
    $\boldsymbol{u} \leftarrow$ vector in U_1 such that $\boldsymbol{u} + \boldsymbol{v} \in U_2$
    $(X_i, A_i) \leftarrow \mathsf{parse}(\mathsf{chain}_i[\boldsymbol{u}])$
    check $e(Z_{i-1}, A_i) = e(W_i, h)$
    check $e(W_i, C_i) = e(Z_i, h)$

return 1 if all checks pass, else return 0

Figure 5.1: Algorithms for creating and verifying chains of update proofs.

5.2.3 Update Algorithm

We now give a lemma used to prove the update security of our construction. This lemmas proves that even a dishonest updater needs to know their contribution to the trapdoor.

Lemma 5.2.1 (Trapdoor extraction for subvertible SRSs)**.** *Suppose there exists a PPT adversary $\mathscr{A}$ that outputs a* $\mathtt{srs}, (\rho_i)_{i=1}^m$ *such that* $\mathsf{VerifySRS}(1^\lambda, \mathtt{srs}, (\rho_i)_{i=1}^m) = 1$ *with non-negligible probability. Then there exists a PPT extractor $\mathscr{X}$ that, given the random tape of $\mathscr{A}$ as input, outputs τ such that* $(\mathtt{srs}, \cdot) \leftarrow \mathsf{Setup}(1^\lambda; \tau)$.

Proof. By the extractability of the proof of knowledge, for every element $(X, A) = \mathsf{chain}_i[\boldsymbol{u}]$ (or $(W, Z, C) = \mathsf{chain}_i[\boldsymbol{v}]$), there exists a PPT extractor that outputs s such that $A = h^s$ (or $C = h^s$). For each variable in V_b, the chain verification ensures that the final element in the chain is equal to the product of the contributions in each link. The remaining checks ensure that the reference string is of the correct form with respect to the extracted variables. Thus setting τ as the hadamard product of all the extracted elements, we have that $(\mathtt{srs}, \cdot) = \mathsf{Setup}(1^\lambda; \tau)$. □

From the simulation-sound proof of knowledge, we get that even when given an honestly generated SRS as input, updaters need to know their contribution to the

$\underline{\mathsf{Setup}(1^\lambda) \mapsto (\mathtt{srs}, \rho_1)}$
$x_1, \ldots x_q \xleftarrow{\$} \mathbb{F}$
$\kappa_1 \leftarrow \mathsf{POK}\{h^{x_j} : x_j\}_{j \in V_1}$
$\mathsf{chain}_1 \leftarrow \mathsf{ChainProve}(1^\lambda, \mathscr{U}, \mathscr{V}, \mathsf{chain}_0, (x_1, \ldots, x_q))$
$\mathtt{srs} \leftarrow \{g^{x_1^{a_1} \ldots x_q^{a_q}}\}_{(a_1,\ldots,a_1) \in \mathscr{U}},\ \{h^{x_1^{b_1} \ldots x_q^{b_q}}\}_{(b_1,\ldots,b_q) \in \mathscr{V}}$
$\rho_1 \leftarrow (\mathsf{chain}_1, \kappa_1)$
return $(\mathtt{srs}, (\rho_1))$

$\underline{\mathsf{Update}(1^\lambda, \mathtt{srs}, (\rho)_{i=1}^m) \mapsto (\mathtt{srs}', \rho_{m+1})}$
$\{g_{a_1,\ldots,a_q}\}_{(a_1,\ldots,a_q) \in \mathscr{U}},\ \{h_{b_1,\ldots,b_q}\}_{(b_1,\ldots,b_1) \in \mathscr{V}} \leftarrow \mathsf{parse}(\mathtt{srs})$
$x_1, \ldots x_q \xleftarrow{\$} \mathbb{F}$
$\kappa_{m+1} \leftarrow \mathsf{POK}\{h^{x_j} : x_j\}_{j \in V_1}$
$\mathsf{chain}_{m+1} \leftarrow \mathsf{ChainProve}(1^\lambda, \mathscr{U}, \mathscr{V}, \mathsf{chain}_n, (x_1, \ldots, x_q))$
$\mathtt{srs}' \leftarrow \{g_{a_1,\ldots,a_q}^{x_1^{a_1} \ldots x_q^{a_q}}\}_{(a_1,\ldots,a_q) \in \mathscr{U}},\ \{h_{b_1,\ldots,b_q}^{x_1^{b_1} \ldots x_q^{b_q}}\}_{(b_1,\ldots,b_q) \in \mathscr{V}}$
$\rho_{m+1} \leftarrow (\mathsf{chain}_{m+1}, \kappa_{m+1})$
return $(\mathtt{srs}', \rho_{n+1})$

$\underline{\mathsf{VerifySRS}(1^\lambda, \mathtt{srs}, (\rho_i)_{i=1}^m) \mapsto 0/1}$
$\{g_{a_1,\ldots,a_q}\}_{(a_1,\ldots,a_q) \in \mathscr{U}},\ \{h_{b_1,\ldots,b_q}\}_{(b_1,\ldots,b_1) \in \mathscr{V}} \leftarrow \mathsf{parse}(\mathtt{srs})$
$(\mathsf{chain}_i, \kappa_i)_{i=1}^m \leftarrow \mathsf{parse}((\rho_i)_{i=1}^m)$
for $\boldsymbol{u} \in U_1 \setminus U_b$:
  check $e(g_{\boldsymbol{u}}, h_{-\boldsymbol{u}}) = e(g, h)$

for $\boldsymbol{v} \in V_1 \setminus V_b$:
  check $e(g_{-\boldsymbol{v}}, h_{\boldsymbol{v}}) = e(g, h)$

for $\boldsymbol{u} \in U_i$ of degree $i \geq 2$:
  find $\boldsymbol{a} \in \bigcup_{j=1}^{i-1} U_j$ and $\boldsymbol{b} \in \bigcup_{j=1}^{i-1} V_j$ such that $\boldsymbol{u} = \boldsymbol{a} + \boldsymbol{b}$
  check $e(g_{\boldsymbol{u}}, h) = e(g_{\boldsymbol{a}}, h_{\boldsymbol{b}})$

for $\boldsymbol{v} \in V_i$ of degree $i \geq 2$:
  find $\boldsymbol{a} \in \bigcup_{j=1}^{i-1} U_j$ and $\boldsymbol{b} \in \bigcup_{j=1}^{i-1} V_j$ such that $\boldsymbol{v} = \boldsymbol{a} + \boldsymbol{b}$
  check $e(g, h_{\boldsymbol{v}}) = e(g_{\boldsymbol{a}}, h_{\boldsymbol{b}})$

for $1 \leq i \leq m$:
  check $\kappa_i \neq \kappa_j$ for $j < i$
  check $\mathsf{POKVerify}(\kappa_i)$

check $\mathsf{ChainVerify}(1^\lambda, \mathscr{U}, \mathscr{V}, (\mathsf{chain}_i)_{i=1}^m)$

return 1 if all checks pass, else return 0.

Figure 5.2: Algorithms for updating structured reference strings with monomials.

trapdoor. In this way security against the updater is linked to an honest SRS.

5.3 Simpler Update Security implies Update Security

In this section we discuss how – for our constructions – proving security for an adversary that makes one update to a freshly generated SRS is equivalent to proving the full version of updatable security, in which an adversary makes all but one update in the sequence. The proof for Lemma 5.3.1 is due to Kohlweiss and is given in [22].

The following lemma relates updatable security to a model in which the adversary can make only a single update after an honest setup. This is because it is much cleaner to prove the security of our construction in this latter model, but we would still like to capture the generality of the former. We already know from Lemma 5.2.1 that it is possible to extract the adversary's contribution to the trapdoor when the adversary generates the SRS itself, and from the simulation-soundness of the proof of knowledge that it is possible to extract it when the adversary updates an honest SRS.

To collapse chains of honest updates into an honest setup it is convenient that the trapdoor contributions of Setup and Update commute in our scheme. Trapdoor contributions cannot just be commuted but also combined; that is, for τ, τ' and τ'', $\mathsf{Update}(1^\lambda, \mathsf{Update}(1^\lambda, \mathsf{Setup}(1^\lambda; \tau); \tau'); \tau'') = \mathsf{Setup}(1^\lambda; \tau \otimes \tau' \otimes \tau'') = \mathsf{Update}(1^\lambda, \mathsf{Update}'(1^\lambda, \mathsf{Setup}'(1^\lambda; \tau''); \tau'); \tau)$. Moreover, in our constructions the proof ρ depends only on the relation and the randomness of the update algorithm. In particular it is independent of the reference string being updated. This enables the following simulation: Given the trapdoor of $\mathtt{srs}$, and the degree 1 elements of $\mathtt{srs}'$ we can simulate a proof ρ_2 of $\mathtt{srs}'$ being an update of $\mathtt{srs}$ using τ^{-1}.

Lemma 5.3.1 (Single adversarial updates imply full updatable knowledge soundness)**.** *If our construction is* U-KSND *secure for adversaries that can query on the update only once and then on the final oracle for a set S such that $|S| \leq 2$, then assuming any adversary that produces verifying setup proofs and update proofs can extract the setup and update randomness, the construction (fully)* U-KSND*-secure.*

As the trapdoor in our scheme consists of all the randomness used by the setup

and the updaters, we shall oft refer to chains of honest updates and (single) honest setups interchangeably.

5.4 Updatable SRSs Contain Only Monomials

In this section we show a negative result; namely, that for any updatable NILP with polynomials encoded into the reference string, it must also be allowed (which often it is not) for an adversary to know encodings of the monomials that make up the polynomials. The reason is that from the encodings of the polynomials, we can construct an adversary that uses the update algorithm in order to extract the monomials. In Chapter 4 we show how our monomial extractor could be used to break our SE-SNARK assuming that it was updatable. Due to the similarity in the approaches, we believe that the same techniques could be used to show that most other QSP/QAP-based zk-SNARKs in the literature also cannot be made updatable. As our universal SRSs do consist of monomials, we can avoid this impossibility result.

5.4.1 The Monomial Extractor

Suppose that a NILP scheme has an update algorithm Update, and that its structured reference strings $\boldsymbol{\sigma}$ are sampled from the distribution $\hat{X}\boldsymbol{\tau}$ for $\hat{X}$ a matrix of known field elements and $\boldsymbol{\tau}$ an unknown vector of (known) monomials. Suppose that for each j there exists an i such that $\hat{X}_{i,j} \neq 0$ (otherwise $\boldsymbol{\sigma}$ is independent of $\boldsymbol{\tau}$). Then there exists an algorithm MonoExtract that can extract $\boldsymbol{\tau}$ from the reference string $\boldsymbol{\sigma}$.

Without loss of generality assume that $\hat{X}$ is in reduced row echelon form (if not the algorithm can apply elementary matrices to $\hat{X}$ and $\boldsymbol{\sigma}$). Also without loss of generality, assume $\hat{X}$ is a square matrix (by adding some all-zero rows if necessary). Write $\hat{X} = \sum_{j=1}^{r} \hat{X}^j$ where $\hat{X}^j$ has at most 1 non-zero element in each column and

row. For example

$$\begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

If $r > 1$ then we will show that it is possible to extract encodings of monomials that are not given in the reference string. This means that for any updatable NIZK with polynomials encoded into the reference string, soundness must hold even for an adversary that knows encodings of the monomials that make up the polynomials. For our constructions in Chapters 6 and 7 we have that $r = 1$; i.e., the SRS already contains all the monomials.

We use an inductive algorithm. We start by showing that there is a probabilistic algorithm BaseMono that can compute vectors $\boldsymbol{u}$, along with a corresponding set of matrices $\left\{\hat{A}^j\right\}_{j=2}^{r}$, that satisfy the equation

$$\hat{X}^1\boldsymbol{\tau} = \boldsymbol{u} + \sum_{j=2}^{r} \hat{A}^j \hat{X}^j \boldsymbol{\tau}.$$

We give a second algorithm InductMono that, upon input of $r-i$ vectors $\{\boldsymbol{u}_\ell\}_{\ell=1}^{r-i}$, and $(r-i)^2$ matrices $\left\{\hat{A}^{\ell,j}\right\}_{j=i+1,\ell=1}^{r,r-i}$, such that

$$\hat{X}^i\boldsymbol{\tau} = \boldsymbol{u}_\ell + \sum_{j=i+1}^{r} \hat{A}^{\ell,j} \hat{X}^j \boldsymbol{\tau}$$

outputs $r-i-1$ vectors $\{\boldsymbol{z}\}_{\ell=1}^{r-i-1}$ and $(r-i-1)^2$ matrices $\left\{\hat{D}^{\ell,j}\right\}_{\ell=1,j=i+2}^{r-i-1,r}$ such that

$$\hat{X}^{i+1}\boldsymbol{\tau} = \boldsymbol{z}_\ell + \sum_{j=i+2}^{r} \hat{D}^{\ell,j} \hat{X}^j \boldsymbol{\tau}.$$

We will then inductively find $\hat{X}^r\boldsymbol{\tau}$ in the algorithm FinalMonoExtract. We can then backwards compute in the algorithm MonoExtract to find $\hat{X}^j\boldsymbol{\tau}$ for $1 \leq j \leq r$.

Theorem 5.4.1. *Suppose that a NILP scheme has structured reference strings $\boldsymbol{\sigma}$ that*

are sampled from the distribution $\hat{X}\boldsymbol{\tau}$ for $\hat{X}$ a matrix of known field elements with no empty columns and $\boldsymbol{\tau}$ an unknown vector of (known) monomials. Suppose there exists an update algorithm

$$(\boldsymbol{\sigma}', \rho') \xleftarrow{\hat{T}} \mathsf{Update}(1^\lambda, \boldsymbol{\sigma}, \rho)$$

such that $\mathsf{VerifySRS}(1^\lambda, \boldsymbol{\sigma}', (\rho, \rho')) = 1$ *and a corresponding extractor that outputs a linear transformation $\hat{T}$ such that $\boldsymbol{\sigma}' = \hat{T}\boldsymbol{\sigma}$. Then there exists an algorithm* MonoExtract *that can extract $\boldsymbol{\tau}$ from the reference string $\boldsymbol{\sigma}$.*

The Base Case: Here we describe the base case algorithm BaseMono. First note that in a generic scheme which is updatably zero-knowledge we require that that for an accepting updated reference string $\boldsymbol{\sigma}_1$ the updater knows their personal contribution to the trapdoor. In other words they know a matrix $\hat{T}$ such that $\boldsymbol{\sigma}_1 = \hat{X}\hat{T}\boldsymbol{\tau}$ (but they do not necessarily know $\boldsymbol{\tau}$).

$\underline{\mathsf{BaseMono}(\boldsymbol{\sigma}, \hat{X}^1, \ldots, \hat{X}^r)}$
$\boldsymbol{\sigma}_1 \xleftarrow{\hat{T}} \mathsf{Update}(\boldsymbol{\sigma})$
$\hat{T}^i \xleftarrow{\$}$ matrices with $\hat{X}\hat{T} = \sum_i \hat{T}^i \hat{X}^i$
$\mathbf{u} \leftarrow (\hat{T}^1)^{-1}\boldsymbol{\sigma}_1$
$\hat{A}^i \leftarrow -(\hat{T}^1)^{-1}(\hat{T}^i)$
return $\mathbf{u}, \hat{A}^2, \ldots, \hat{A}^r$

It can be seen that

$$\begin{aligned}
\boldsymbol{u} + \textstyle\sum_{j=2}^r \hat{A}^j \hat{X}^j \boldsymbol{\tau} &= (\hat{T}^1)^{-1}\boldsymbol{\sigma}_1 - \textstyle\sum_{j=2}^r (\hat{T}^1)^{-1}\hat{T}^j\hat{X}^j\boldsymbol{\tau} \\
&= (\hat{T}^1)^{-1}\left[\hat{X}\hat{T}\boldsymbol{\tau} - \textstyle\sum_{j=2}^r \hat{X}^j\hat{T}\boldsymbol{\tau}\right] \\
&= (\hat{T}^1)^{-1}\left[\hat{X} - \textstyle\sum_{j=2}^r \hat{X}^j\right]\hat{T}\boldsymbol{\tau} \\
&= (\hat{T}^1)^{-1}\hat{X}^1\hat{T}\boldsymbol{\tau} \\
&= (\hat{T}^1)^{-1}\hat{T}^1\hat{X}^1\boldsymbol{\tau} \\
&= \hat{X}^1\boldsymbol{\tau}
\end{aligned}$$

so BaseMono outputs correct values of $\boldsymbol{u}$ and $\hat{A}^j$.

We wish to argue the existence of $\hat{T}^1, \ldots, \hat{T}^r$. It can be assumed that $\hat{X}$ does not have any empty columns, otherwise the setup can select $\boldsymbol{\tau}$ from a smaller distribution. Thus for

$$a_{ij} = (\hat{X}\hat{T})_{ij}$$

there exists $\hat{X}^k$ and ℓ such that $\hat{X}^k_{\ell j} = x$. The BaseMono can set $T^k_{i\ell} = \frac{a_{ij}}{x}$.

If $\hat{X}^k_{i\ell} = 0$ then choosing $\hat{T}^k_{ji} \xleftarrow{\$} \mathbb{F}$ will not affect the result. Now for $k < r$ the matrices $\hat{T}^k$ contain at least one column chosen at random. Further, $\hat{X}\hat{T}$ cannot have a zero-column (else $\boldsymbol{\tau}$ and $\hat{T}\boldsymbol{\tau}$ are sampled from different distributions), and thus $\hat{T}^k$ cannot contain a zero-row. This means that the determinant

$$\det(\hat{T}^k) = \sum_{\sigma \in S_n} \mathsf{sgn}(\sigma) \prod_{i=1}^{n} T^k_{i,\sigma(i)}$$

is non-zero with overwhelming probability, so $\hat{T}^k$ is overwhelmingly invertible.

The Induction: The inductive algorithm InductMono works as follows.

$\underline{\mathsf{InductMono}(U, A, r, i)}$

$\mathbf{u}_1, \ldots, \mathbf{u}_{r-i} \leftarrow \mathsf{parse}(U)$

$\{\hat{A}^{1,j}\}_{j=i+1}^{r}, \ldots, \{\hat{A}^{r-i,j}\}_{j=i+1}^{r} \leftarrow \mathsf{parse}(A)$

for $1 \le \ell \le r-i-1$:

$\quad \hat{M}^\ell \leftarrow \left[\hat{A}^{1,1} - \hat{A}^{\ell+1,1}\right]^{-1}$

$\quad \mathbf{z}_\ell = \hat{M}^\ell \left[\mathbf{u}_{\ell+1} - \mathbf{u}_1\right]$

for $1 \le \ell \le r-i-1$:

$\quad$ for $i+2 \le j \le r$:

$\quad\quad \hat{D}^{\ell,j} = \hat{M}^\ell \left[\hat{A}^{\ell+1,j} - \hat{A}^{1,j}\right]$

return $\{\mathbf{z}_\ell\}_{\ell=1}^{r-i-1}$, $\{\hat{D}^{\ell,j}\}_{\ell=1,j=i+2}^{r-i-1,r}$

For each $1 \le \ell \le r-i-1$, we have that $\mathbf{u}_1, \mathbf{u}_{\ell+1}$ are such that

$$\hat{X}^i\boldsymbol{\tau} = \mathbf{u}_1 + \sum_{j=i+1}^{r} \hat{A}^{1,j}\hat{X}^j\boldsymbol{\tau} \text{ and } \hat{X}^i\boldsymbol{\tau} = \mathbf{u}_{\ell+1} + \sum_{j=i+1}^{r} \hat{A}^{\ell+1,j}\hat{X}^j\boldsymbol{\tau}.$$

Putting the two equations together yields

$$\mathbf{u}_1 + \sum_{j=i+1}^{r} \hat{A}^{1,j}\hat{X}^j\boldsymbol{\tau} = \mathbf{u}_{\ell+1} + \sum_{j=i+1}^{r} \hat{A}^{\ell+1,j}\hat{X}^j\boldsymbol{\tau}.$$

We then rearrange to get

$$\left(\hat{A}^{1,i+1} - \hat{A}^{\ell,i+1}\right)\hat{X}^{i+1}\boldsymbol{\tau} = \mathbf{u}_{\ell+1} - \mathbf{u}_1 + \sum_{j=i+2}^{r} \left(\hat{A}^{\ell+1,j} - \hat{A}^{1,j}\right)\hat{X}^j\boldsymbol{\tau}.$$

With the matrix $\hat{M}^\ell = \left(\hat{A}^{1,i+1} - \hat{A}^{\ell+1,i+1}\right)^{-1}$ we have that

$$\hat{X}^{i+1}\boldsymbol{\tau} = \hat{M}\left[\mathbf{u}_{\ell+1} - \mathbf{u}_1 + \sum_{j=i+2}^{r} \left(\hat{A}^{\ell+1,j} - \hat{A}^{1,j}\right)\hat{X}^j\boldsymbol{\tau}\right].$$

If $\hat{M}^\ell$ does not exist then the probabilistic algorithm BaseMono can be rerun. Hence we have that

$$X^{i+1}\boldsymbol{\tau} = \mathbf{z}_\ell + \sum_{j=i+2}^{r} \hat{D}^{\ell,j}\hat{X}^j\boldsymbol{x}$$

as required.

Extractor of the Final Monomial Our monomial extractor first runs an algorithm to extract the final monomial, and from there backwards computes. We use the notation $U[i]$ to denote sampling the ith component from the set U.

$\underline{\mathsf{FinalMonoExtract}(\boldsymbol{\sigma}, \hat{X}^1, \ldots, \hat{X}^r)}$
$U = \emptyset$; $A = \emptyset$; $B = \emptyset$
for $1 \leq i \leq r$:
  $\mathbf{u}, \{\hat{A}^j\}_{j=2}^r \xleftarrow{\$} \mathsf{BaseMono}(\boldsymbol{\sigma}, \hat{X}^1, \ldots, \hat{X}^r)$
  $U = U \cup \{\mathbf{u}\}$; $A = A \cup \{\{\hat{A}^j\}_{j=2}^r\}$
$B = B \cup \{U[1], A[1]\}$
for $1 \leq i \leq r-1$
  $(U, A) \leftarrow \mathsf{InductMono}(U, A, r, i)$
  $B = B \cup \{U[1], A[1]\}$
return B

This algorithm outputs a set B consisting of pairs, where each pair contains a vector

and a sets of matrices. The i-th entry is $B[i] = \{\mathbf{u}, \{\hat{A}^j\}_{j=i+1}^{r}\}$ such that

$$\hat{X}^i \boldsymbol{\tau} = \mathbf{u} + \sum_{j=i+1}^{r} \hat{A}^j \hat{X}^j \boldsymbol{\tau}.$$

For the final entry $B[r]$, the set of matrices is empty, so the right-hand side of the above equation is simply $\mathbf{u}$. Hence this final $\mathbf{u}$ is equal to $\hat{X}^r \boldsymbol{\tau}$.

The Monomial Extractor We are now in a position to define an algorithm that takes the output of FinalMonoExtract and then backwards computes to find $\{\hat{X}^i \boldsymbol{\tau}\}_{i=1}^{r}$. This algorithm is our monomial extractor.

$\underline{\mathsf{MonoExtract}(\boldsymbol{\sigma}, \hat{X}^1, \ldots, \hat{X}^r)}$
$B \xleftarrow{\$} \mathsf{FinalMonoExtract}(\boldsymbol{\sigma}, \hat{X}^1, \ldots, \hat{X}^r)$
$\{\mathbf{v}_r, \emptyset\} \leftarrow \mathsf{parse}\ B[r]$
for $r-1 \geq i \geq 1$:
  $\{\mathbf{u}, \{\hat{A}^j\}_{j=i+1}^{r}\} \leftarrow \mathsf{parse}\ B[i]$
  $\mathbf{v}_{r-1} \leftarrow \mathbf{u} + \sum_{j=i+1}^{r} \hat{A}^j \mathbf{v}_j$
return $\mathbf{v}_1, \ldots, \mathbf{v}_r$

5.5 Our SE-SNARK is not Updatable

Intuitively, the existence of the monomial extractor would break most pairing-based NIZK proofs using QAPs or QSPs. This is because these arguments typically depend on the instance polynomials and the witness polynomials being linearly independent from each other. Here it is demonstrated how the existence of a monomial extractor would break the knowledge soundness of the SE-SNARK in Chapter 4. Our results are based of a similar proof by the author showing that Pinocchio [2] is not updatable published in [22].

The SRS is given by

$$\left(\begin{array}{l} g^{\alpha}, g^{\gamma}, g^{x}, g^{\alpha\delta}, g^{\gamma\delta t(x)}, g^{\gamma^2 \delta t(x)^2}, g^{(\alpha+\beta)\gamma\delta t(x)}, \\ h, h^{\beta}, h^{\delta}, h^{\beta\delta}, h^{\gamma\delta t(x)}, h^{\delta^2} \\ \left\{ g^{\gamma\delta x^i}, h^{\gamma\delta x^i}, g^{\gamma^2 \delta t(x) x^i} \right\}_{i=0}^{n-1}, \left\{ g^{\gamma\delta w_i(x) + \delta(\alpha+\beta) u_i(x))} \right\}_{i=0}^{\ell}, \\ \left\{ g^{\gamma^2 \delta w_i(x) + (\alpha+\beta)\gamma\delta u_i(x))} \right\}_{i=\ell+1}^{m}, \end{array} \right)$$

where $\alpha, \beta, \gamma, \delta, x$ are random field elements. For our attack, it suffices to just consider the exponents of g and not h. There exists a matrix $\hat{X}$ such that $\boldsymbol{\sigma} = \hat{X}\boldsymbol{\tau}$ for

$$\boldsymbol{\tau} = \left(\alpha, \gamma, x, \{\alpha\delta x^i, \beta\delta x^i\}_{i=0}^{n-1}, \{\gamma\delta x^i, \alpha\gamma\delta x^i, \beta\gamma\delta x^i\}_{i=0}^{n}, \{\gamma^2\delta x^i\}_{i=0}^{2n} \right). \quad (5.1)$$

Lemma 5.5.1. *If there exists a monomial extractor for Construction 4.2, then there exists an adversary that can find a verifying proof for any instance* $(a_1, ..., a_\ell) \in \mathbb{Z}_p$.

Proof. The verifier returns 1 if and only the following equations are satisfied

$$e(Ag^{\alpha\delta}, Bh^{\beta\delta}) = e(g^{\alpha\delta}, h^{\beta\delta}) e(g^{\sum_{i=0}^{\ell} a_i \delta(\gamma w_i(x) + (\alpha+\beta)u_i(x))}, h^{\gamma\delta}) e(C, h^{\delta})$$
$$e(A, h) = e(g, B)$$

Suppose the adversary $\mathscr{A}$ chooses $\mu \xleftarrow{\$} \mathbb{Z}_p$ and sets the components A, B, C by

$$A = g^{\mu\gamma\delta},\ B = h^{\mu\gamma\delta},\ C = g^{\mu^2\gamma^2\delta + \mu\alpha\gamma\delta + \mu\beta\gamma\delta - \sum_{i=0}^{\ell} a_i(\delta\gamma^2 w_i(x) + \alpha\gamma\delta u_i(x) + \beta\gamma\delta u_i(x))}.$$

The component $g^{\gamma\delta}$ is in the extracted monomials in (5.1) thus $\mathscr{A}$ can compute A. The component $h^{\gamma\delta}$ is in $\mathtt{srs}$ thus $\mathscr{A}$ can compute B. The components

$$\gamma^2\delta, \alpha\gamma\delta, \beta\gamma\delta, \{\delta\gamma^2 x^i, \alpha\gamma\delta x^i, \beta\gamma\delta x^i\}_{i=0}^{n-1}$$

are in the extracted monomials in (5.1) thus $\mathscr{A}$ can compute C. Direct verification shows that the verifier's equations are satisfied. □

Theorem 5.5.2. *If there exists an update algorithm for Construction 4.2, then either the relation is easy or the scheme is not knowledge-sound.*

Proof. Suppose that $\mathtt{srs} \leftarrow \mathsf{Setup}(1^\lambda)$; i.e., $\mathtt{srs} = \hat{X} g^{\boldsymbol{\tau}}$ for $\boldsymbol{\tau}$ as in Equation 5.1. Suppose that $(a_1, \ldots, a_\ell) \in \mathbb{Z}_p$. By running $\mathsf{MonoExtract}$, an adversary $\mathscr{A}$ can calculate $g^{\boldsymbol{\tau}}$. By Lemma 5.5.1, the adversary $\mathscr{A}$ can continue, and calculate a verifying proof for $(a_1, \ldots, a_\ell)$. Hence either there is a PPT extractor that can output a valid witness for any instance (meaning the language is easy), or there is no extractor and $\mathscr{A}$ breaks knowledge-soundness. □

Chapter 6

Updatable and Universal zk-SNARKs

This chapter presents a construction of an updatable and universal zk-SNARK from a work published at Crypto 2018 [22] together with Jens Groth, Markulf Kohlweiss, Sarah Meiklejohn, and Ian Miers. The construction is QAP-based and makes use of a universal reference string. A proof of subversion zero knowledge and updatable knowledge soundness is given under the knowledge-of-exponent assumptions introduced in Chapter 3. The content in this chapter is joint work with Jens Groth and Markulf Kohlweiss.

6.0.1 Quadratic Arithmetic Programs

We let the security parameter 1^λ (deterministically) determine parameters $(n, m, \ell, \mathsf{bp})$, where $\mathsf{bp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$, n is the maximum degree of the QAP, and m is the maximum number of input variables.

Formally, we will be working with quadratic arithmetic programs R due to Gennero et al. [73] that have the following description

$$R = (\mathsf{bp}, \ell, \{u_i(X), v_i(X), w_i(X)\}_{i=0}^{m}, t(X)),$$

where the bp defines the finite field $\mathbb{Z}_p$ we will be working over, $1 \leq \ell \leq m$ is the instance formed of public field elements to a QAP, $u_i(X), v_i(X), w_i(X), t(X) \in \mathbb{Z}_p[X]$ and $u_i(X), v_i(X), w_i(X)$ have strictly lower degree than n, the degree of $t(X)$.

Furthermore, we require that the set $S = \{u_i(X) : 0 \leq i \leq \ell\}$ is linearly independent and that any $u_i(X) \in S$ is also linearly independent from the set $\{u_j(X) : \ell < j \leq m\}$. A quadratic arithmetic program with such a description defines the following binary relation, where we define $a_0 = 1$,

$$R = \left\{ (\phi, w) \;\middle|\; \begin{array}{l} \phi = (a_1, \ldots, a_\ell) \in \mathbb{Z}_p^\ell \\ w = (a_{\ell+1}, \ldots, a_m) \in \mathbb{Z}_p^{m-\ell} \\ \\ \exists\, q(X) \in \mathbb{Z}_p[X], \deg(q) \leq n-2 : \\ \qquad (\sum_{i=0}^m a_i u_i(X))\,(\sum_{i=0}^m a_i v_i(X)) = \sum_{i=0}^m a_i w_i(X) + q(X)t(X) \end{array} \right\}$$

We say $\mathscr{R}$ is a bilinear group and quadratic arithmetic program generator if it generates relations of the form given above with prime $p > 2^{\lambda-1}$. The sequence of parameters indexed by the security parameter define a universal relation R consisting of all pairs of QAPs and instances as described above that have a matching witness.

6.1 Reworking the QAP recipe

Our final scheme is formally given in Figures 5.2 and 6.2. In this section we describe some of the technical ideas behind it. Due to the existence of monomial extractors for updatable SRSs (Chapter 5), many of the usual tricks behind the QAP-based approach are not available to us. Instead we first switch to a multi-variate scheme, where the proof elements need to satisfy equations in the indeterminates X, Y, Z. We can then prove the well-formedness of our proof elements using a subspace argument for our chosen sums of witness QAP polynomials. Once we have that the proof elements are well formed, we show that the exponents of two of them multiply to get an exponent in the third proof element such that (1) the sum of all the terms where Y has given power j is equal to the QAP expression in the X indeterminate, and (2) the value Y^j is not given in the universal SRS. For our final scheme, we use $j = 7$.

Fix the circuit: The circuit need only be fixed upon running the SRS derivation algorithm. At this point, the circuit is described as a QAP; i.e., for $a_0 = 1$, the field

elements $(a_1,\ldots,a_m) \in R$ if and only if

$$\left(\sum_{i=0}^{m} a_i u_i(X)\right) \cdot \left(\sum_{i=0}^{m} a_i v_i(X)\right) = \sum_{i=0}^{m} a_i w_i(X) + q(X)t(X)$$

for some degree $(n-2)$ polynomial $q(X)$.

Prove the commitments are well formed: In our scheme an honest prover outputs group elements (A,B,C) such that

$$\log(A) = \log(B) = q(x)y + \sum_{i=0}^{m} a_i(w_i(x)y^2 + u_i(x)y^3 + v_i(x)y^4) - y^5 - t(x)y^6.$$

Ensuring that $\log(A) = \log(B)$ can be achieved with a pairing equation of the form $e(A,h) = e(g,B)$. Thus we need to show only that A is of the correct form.

Usually, proving an element is of the correct form is achieved by encoding relation dependent polynomials in the SRS and forcing the prover to use linear combinations of these polynomials; this is the approach we take for constructing a subversion-extractable zk-SNARK in Chapter 4. Since we cannot take this approach and allow updates, we instead construct a new subspace argument. First the verifier subtracts out the known elements in the instance and obtains a new group element with the exponent

$$q(x)y + \sum_{i=\ell+1}^{m} a_i(w_i(x)y^2 + u_i(x)y^3 + v_i(x)y^4).$$

A matrix $\hat{M}$ is set to be the $(m+d-\ell) \times 4n$ matrix that contains the coefficients of $\{(w_i(x)y^2 + u_i(x)y^3 + v_i(x)y^4)\}_{i=\ell+1}^{m}, \{x^i y\}_{i=0}^{n-2}$ with respect to monomials $\{x^i y^j\}_{(i,j)=(0,1)}^{(n-1,4)}$. The corresponding null-matrix $\hat{N}$ is mapped by $\hat{M}$ to zero, i.e. $\hat{M}\hat{N} = \mathbf{0}$. If we introduce the variable z, then the verifier can scale the columns of $\hat{N}$ by different powers of z, and thus can check that $\hat{M}$ maps all the columns of $\hat{N}$ to $\mathbf{0}$. This should hold if and only if A is chosen from the correct subspace.

Prove that the QAP is satisfied: Assuming that A and B are of the correct form, we have that $\log(A) \cdot \log(B)$ is equal to

$$2\left(q(x)y + \sum_{i=0}^{m} a_i(w_i(x)y^2 + u_i(x)y^3 + v_i(x)y^4) - y^5 - t(x)y^6\right)^2.$$

which, for terms involving y^7, yields

$$t(x)q(x) - \sum_{i=0}^{m} a_i w_i(x) + \left(\sum_{i=0}^{m} a_i u_i(x)\right) \cdot \left(\sum_{i=0}^{m} a_i v_i(x)\right).$$

The terms in other powers of y can be considered as computable garbage and are cancelled out in other proof components. The equation above is satisfied for some polynomial $q(X)$ if and only if the QAP is satisfied. Thus, given an SRS that does not contain any y^7 terms, and a verification equation that checks $\log A \cdot \log B = \log C$, we ensure that the proof element C is computable if and only if the QAP is satisfied.

6.2 A Null Space Argument

The primary investigator in designing the following null space argument is Jens Groth, and the more detailed description in this section is new to this thesis. The argument utilises the Rank-Nullity theorem, which states that the null space of a set of vectors is orthogonal to the span of the set of vectors.

Consider a set of vectors $S = \{\boldsymbol{u}_i\}_{i \in I}$ which is a subset of a vector space $S \subset V$. The null space of S is the largest possible set of linearly independent vectors in V, $N = \{\boldsymbol{\eta}_j\}_{j \in J}$ such that for all $i \in I$ and $j \in J$,

$$\boldsymbol{u}_i \cdot \boldsymbol{\eta}_j = 0.$$

If we wish to show that a vector $\boldsymbol{a}$ is included in the span of S, then it suffices to show that for all $j \in J$,

$$\boldsymbol{a} \cdot \boldsymbol{\eta}_j = 0.$$

In our case the vector space and the set of vectors are given by

$$\begin{aligned} V &= \{X^i Y^j\}_{i=0,j=1}^{n,4} \\ S &= \{X^i Y\}_{i=0}^{n-2} \cup \{w_i(X)Y^2 + u_i(X)Y^3 + v_i(X)Y^4\}_{i=\ell+1}^{m} \cup \{t(X)Y^j\}_{j=2}^{4} \end{aligned}$$

The Rank-Nullity theorem tells us that

$$\dim(V) = \mathsf{rank}(S) + \mathsf{nullity}(S) \quad \text{i.e.} \quad |V| = \mathsf{rank}(S) + |N|.$$

The dimension of V is equal to $4n+4$. The rank of S is equal to the number of linear independent vectors in S, which is bounded by the size of S. Thus the rank of S is bounded by $n+m-\ell+2$. Now m is equal to the number of wires in the circuit, and because we are working with fan-in 2 gates, this means that m is bounded by $2n$. The nullity of S is equal to the size of the null space. Therefore

$$\begin{aligned} &|V| = \mathsf{rank}(S) + |N| \\ &\Rightarrow |N| = 4n+4-\mathsf{rank}(S) \geq 4n+4-(n+m-\ell+2) \geq 4n+4-(n+2n-\ell+2) \\ &\Rightarrow |N| \geq n+2+\ell \end{aligned}$$

i.e. when the instance is small, there are approximately n vectors in the null space.

We represent our null space in a polynomial of the form

$$\eta(X,Y,Z) = \sum_{i=0}^{n} \sum_{j=1}^{4} \sum_{k=1}^{|N|} \eta_{k,i,j} X^{4n-i} Y^{4-j} Z^k$$

where $\eta_{k,i,j}$ is the $i(n+1)+j$th entry of the kth vector in the null space. We represent elements $\boldsymbol{a}$ that are in the span of S by polynomials of the form

$$a(X,Y) = \sum_{i=0}^{n} \sum_{j=1}^{4} a_{i,j} X^i Y^j$$

where $a_{i,j}$ is the $i(n+1)+j$th entry of $\boldsymbol{a}$. Then

$$\begin{aligned} a(X,Y)\eta(X,Y,Z) &= \sum_{i,s=0}^{n}\sum_{j,t=1}^{4}\sum_{k=1}^{|N|} a_{i,j}\eta_{k,s,t}X^{i+4n-k}Y^{j+4-t}Z^{k} \\ &= \sum_{i=0}^{n}\sum_{j=1}^{4}\sum_{k=1}^{|N|} a_{i,j}\eta_{k,i,j}X^{4n}Y^{4}Z^{k} + \sum_{(i,j)\neq(n,4)}\sum_{k=1}^{|N|} r_{i,j}X^{i}Y^{j}Z^{k} \\ &= \sum_{(i,j)\neq(n,4)}\sum_{k=1}^{|N|} r_{i,j}X^{i}Y^{j}Z^{k} \end{aligned}$$

for $r_{i,j}$ known field elements.

The SRS does not contain the elements $\{g^{x^n y^4 z^k}\}_{k=1}^{|N|}$. The prover provides an element $A = g^{a(x,y)}$ and a second element $C = g^{a(x,y)\eta(x,y,z)}$. The verifier checks that

$$e(g^{a(x,y)}, h^{\eta(x,y,z)}) = e(C,h)$$

If $a(X,Y)$ is not in the span of S, then $a(x,y)\eta(x,y,z)$ will have non-zero terms $X^nY^4Z^k$. By the Mono assumption the prover cannot compute $g^{x^{4n}y^4z^k}$ and therefore it can only compute C if $a(X,Y)$ is in the correct span.

6.3 Derivation of a Linear SRS

Astute readers may note that our null space argument requires the SRS to contain a quadratic sized set of monomials. We tackle this issue by providing an untrusted derive function (which can be seen as a form of precomputation) in order to find a linear SRS for a fixed relation. The output of the derive algorithm can be considered equivalent to the output of a trusted setup for a zk-SNARK with a relation specific SRS. Using the linear reference string, our prover computation consists of a linear number of group exponentiations in the circuit size.

The universal SRS contains base g exponents $\{x^i y^j z^k\}_{(i,j,k)\in\mathscr{U}}$ where

$$\mathscr{U} = \left(\begin{array}{c} \{(1,0,0),(0,1,0),(0,0,1)\} \\ \cup\{(i,j,0) : i \in [0,2n], j \in [1,12], j \neq 7\} \\ \cup\{(i,j,k) : i \in [0,2n], j \in [1,6], k \in [1,3n], (i,j) \neq (n,4)\} \\ \cup\{(i,j,6n) : i \in [0,n], j \in [1,4]\} \end{array} \right)$$

and base h exponents $\{x^i y^j z^k\}_{(i,j,k)\in\mathscr{V}}$ where

$$\mathscr{V} = \left(\begin{array}{c} \{(1,0,0),(0,1,0),(0,0,1),(0,0,6n)\} \\ \cup\{(i,j,0) : i \in [0,n], j \in [1,6]\} \\ \cup\{(i,j,k) : i \in [0,n], j \in [0,2], k \in [1,3n]\} \end{array} \right).$$

This SRS is updatable using techniques described in Chapter 5.

We provide an algorithm for deriving a `qap` specific SRS from the universal SRS in Figure 6.1. With the derived SRS we can construct efficient prove and verify algorithms. The derive algorithm is trustless in the sense that it can be ran by any party. On the other hand, the derived `srs` cannot be updated. Computing the null space requires Gaussian elimination over a sparse matrix.

6.4 Our Construction

In this section we construct a zk-SNARK for QAP satisfiability given the derived structured reference string from the previous section. The construction is given in Figure 6.2. The prover is given $(\phi, w) \in R$ and must convince the verifier that ϕ is in the language.

Theorem 6.4.1. *The proof system described in Figures 6.1&6.2 has subversion zero-knowledge against all PPT adversaries that generate a verifying global* `srs`.

Proof. To prove subversion zero-knowledge, we need to both show the existence of an extractor $\mathscr{X}_{\mathscr{A}}$, and describe a SimProve algorithm that produces indistinguishable proofs when provided the extracted trapdoor (which it can compute given the randomness of both $\mathscr{A}$ and the honest algorithms). The

$\underline{\mathsf{Derive}(\mathsf{bp},\mathtt{qap},\mathtt{srs}) \mapsto \mathtt{srs}_{\mathtt{qap}}}$
parse $(\ell,\{u_i(X),v_i(X),w_i(X)\}_{i=0}^{m},t(X)) \leftarrow \mathtt{qap}$
assert $g^{y-t(x)y^2} \neq 1$
for $0 \leq i \leq n-w$:
  $s_i(X,Y) \leftarrow X^iY$
for $1 \leq j \leq 3$:
  $s_{n-1+j}(X,Y) \leftarrow t(X)Y^{j+1}$
for $n+3 \leq i \leq n+3+m-\ell$:
  $s_i(X,Y) \leftarrow w_i(X)Y^2+u_i(X)Y^3+v_i(X)Y^4$

compute a basis of the polynomials $\eta_1(X,Y),\ldots,\eta_{3n-m+\ell}(X,Y)$ such that
  for all $\ell+1 \leq i \leq m+3$ *and* $1 \leq k \leq 3n-m+\ell$, *the product*
  $s_i(X,Y)\cdot\eta_k(X,Y)$ *has coefficient* 0 *for the term* X^nY^4

$\eta(X,Y,Z) \leftarrow Z^{6n}+\sum_{k=1}^{3n-m+\ell}\eta_k(X,Y)Z^k$

$$\mathtt{srs}_{\mathtt{qap}} \leftarrow \left(\begin{array}{l} \{g^{x^iy^j}\}_{i=0,j=1,j\neq 7}^{2n,12}, g^{y-t(x)y^2}, \{g^{w_i(x)y^2+u_i(x)y^3+v_i(x)y^4}\}_{i=0}^{m}, \\ g^{y^5+t(x)y^6}, \{g^{(w_i(x)y^2+u_i(x)y^3+v_i(x)y^4)\cdot\eta(x,y,z)}\}_{i=\ell+1}^{m} \\ \{h^{x^iy}\}_{i=0}^{n}, h^{y-t(x)y^2}, \{h^{w_i(x)y^2+u_i(x)y^3+v_i(x)y^4}\}_{i=0}^{m}, \\ h^{y^5}, h^{t(x)y^6}, h^{\eta(x,y,z)} \end{array}\right)$$

return $\mathtt{srs}_{\mathtt{qap}}$

Figure 6.1: The derive algorithm for generating circuit specific reference strings for our updatable and universal zk-SNARK.

simulator knows x,y,z and picks $r \leftarrow \mathbb{Z}_p$ and sets $A = g^r, B = h^r$ and $C = g^{r^2+(r+y^5+t(x)y^6-\sum_{i=0}^{\ell}a_i(w_i(x)y^2+u_i(x)y^3+v_i(x)y^4))\cdot\eta(x,y,z)}$. The simulated proof have the same distribution as the real proofs because A includes the random term $r(y-t(x)y^2)$ and $y \neq 0$ and $t(x) \neq y^{-1}$. Given A the verification equations uniquely determine B and C. Both real and simulated proofs have uniformly random A and satisfy the equations. Consequently, subversion zero-knowledge follows from the extraction of the trapdoor, which can be extracted by Lemma 5.2.1. □

Theorem 6.4.2. *The protocol in Figure 6.2 is update knowledge sound provided that the* $(2n,36n^2)$-MKE *and the* $(2n,36n^2)$-Mono *assumptions hold for n the number of multiplicative constraints.*

Proof. To prove this it suffices, by the results in Chapter 5, to prove security in

Common Input: $\mathsf{info} = \mathsf{bp}, \mathtt{qap}, \mathtt{srs}_{\mathtt{qap}}$

$\underline{\mathsf{Prove}(\mathsf{info}, \phi, w) \mapsto \pi}$
$(a_0, a_1, \ldots, a_m) \leftarrow \mathsf{parse}(1, \phi, w)$
$r \xleftarrow{\$} \mathbb{Z}_p$

$$q(X) \leftarrow \frac{\sum_{i=0}^{m} a_i u_i(X) \cdot \sum_{i=0}^{m} a_i v_i(X) - \sum_{i=0}^{m} a_i w_i(X)}{t(X)}$$

$$a(X,Y) \leftarrow \begin{pmatrix} q(X)Y + r(Y - t(X)Y^2) + \sum_{i=0}^{m} a_i (w_i(X)Y^2 + u_i(X)Y^3 + v_i(X)Y^4) \\ -Y^5 - t(X)Y^6 \end{pmatrix}$$
$$c(X,Y,Z) \leftarrow \begin{pmatrix} a^2(X,Y) + (q(X)Y + r(Y - t(X)Y^2)) \cdot n(X,Y,Z) \\ \left(\sum_{i=\ell+1}^{m} a_i (w_i(X)Y^2 + u_i(X)Y^3 + v_i(X)Y^4)\right) \cdot n(X,Y,Z) \end{pmatrix}$$

$A, B, C \leftarrow g^{a(x,y)}, h^{a(x,y)}, g^{c(x,y,z)}$
return $\pi = (A, B, C)$

$\underline{\mathsf{Verify}(\mathsf{info}, \phi, \pi) \mapsto 0/1}$
$(a_0, a_1, \ldots, a_\ell) \in \mathbb{Z}_p^{\ell+1} \leftarrow \mathsf{parse}(1, \phi)$
$(A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \leftarrow \mathsf{parse}(\pi)$
$f(X,Y) \leftarrow Y^5 + t(X)Y^6 - \sum_{i=0}^{\ell} a_i (w_i(X)Y^2 + u_i(X)Y^3 + v_i(X)Y^4)$
check $e(A,B)e(Ag^{f(x,y)}, h^{\eta(x,y,z)}) = e(C,h)$
check $e(A,h) = e(g,B)$
return 1 if both checks pass, else return 0.

Figure 6.2: An updatable and specialisable zk-SNARK for QAP

the setting in which the adversary makes only one update to the SRS. Imagine we have a PPT adversary $\mathscr{A}^{\mathsf{U}\text{-}\mathscr{O}_{\mathtt{s}}}$ that after querying $\mathsf{U}\text{-}\mathscr{O}_{\mathtt{s}}$ on $(\mathsf{Setup}, \emptyset)$ to get $\mathtt{srs}$, then queries on verifying $(\mathsf{final}, \mathtt{srs}', \{\rho, \rho'\})$, and then outputs verifying (ϕ, π); i.e., such that $\mathsf{VerifySRS}(R, \mathtt{srs}', \{\rho, \rho'\}) = 1$, $\mathtt{srs}_{\mathtt{qap}} \leftarrow \mathsf{Derive}(\mathtt{srs}', \mathtt{qap})$, and $\mathsf{Verify}(\mathtt{srs}_{\mathtt{qap}}, \phi, \pi) = 1$. Set $a_0 = 1$ and parse the instance as $\phi = (a_1, \ldots, a_\ell)$ and the proof as (A, B, C). The updated SRS verifies, thus there exists an extractor $\mathscr{X}_{\mathscr{A}}$ that outputs $\tau = (\alpha, \beta, \gamma)$ such that $\mathsf{Update}(1^\lambda, \mathtt{srs}, \{\rho\}; \tau) = (\mathtt{srs}', \rho')$.

From the second verification equation we have $e(A,h) = e(g,B)$. From the q-MKE assumption there exists a PPT extractor $\mathscr{X}_{\mathscr{A}}$ for $\mathscr{A}$ that outputs field elements $\{a_{i,j,k}\}_{(i,j,k) \in \{(0,0,0)\} \cup \mathscr{W}}$ defining a formal polynomial $a(X,Y,Z)$ equal to

$$a_{0,0,0} + a_{1,0,0}X + \sum_{i=0,j=1}^{n,6} a_{i,j,0} X^i Y^j + \sum_{i=0,j=0,k=1}^{2n,3,3d} a_{i,j,k} X^i Y^j Z^k + a_{0,0,6n} Z^{6n}$$

such that $B = h^{a(x,y,z)}$.

Taking the adversary and extractor together, we can see them as a combined algorithm that outputs A, B, C and the formal polynomial $a(X,Y,Z)$ such that $A = g^{a(x,y,z)}$. By the q-Mono assumption this has negligible probability of happening unless $a(X,Y,Z)$ is in the span of $\{0,0,0\} \cup \mathscr{U} \cap \mathscr{V}$:

$$\left\{1, X, Z, \{X^i Y^j\}_{i=0,j=1,j\neq 7}^{2n,12}, \{X^i Y^j Z^k\}_{i=0,j=1,k=1,(i,j)\neq(n,4)}^{2n,6,3n}, \{X^i Y^j Z^{6n}\}_{i=0,j=1}^{n,4}\right\}.$$

This means that

$$a(X,Y,Z) = a_{0,0,0} + a_{1,0,0}X + \sum_{i=0,j=1}^{n,6} a_{i,j,0} X^i Y^j + \sum_{i=0,j=1,k=1}^{n,3,3n} a_{i,j,k} X^i Y^j Z^k.$$

From the first verification equation we get $C = g^{f(x,y,z)}$ where $f(x,y,z)$ is given by

$$a(x,y,z)^2 + \Big(a(x,y,z) + \beta^5 y^5 + t(\alpha x)\beta^6 y^6 - \sum_{i=0}^{\ell} a_i (w_i(\alpha x)\beta^2 y^2 + u_i(\alpha x)\beta^3 y^3 + v_i(\alpha x)\beta^4 y^4)\Big) \cdot n(\alpha x, \beta y, \gamma z).$$

By the q-Mono assumption this implies

$$a(X,Y,Z)^2 + \Big(- \sum_{i=0}^{\ell} a_i (w_i(\alpha X)\beta^2 Y^2 + u_i(\alpha X)\beta^3 Y^3 + v_i(\alpha X)\beta^4 Y^4) + a(X,Y,Z) + \beta^5 Y^5 + t(\alpha X)\beta^6 Y^6\Big) \times \Big(\gamma^{6n} Z^{6n} + \sum_{k=1}^{3n-m+\ell} n_k(\alpha X, \beta Y)\gamma^k Z^k\Big)$$

also belongs to the span of

$$\left\{1, X, Z, \{X^i Y^j\}_{i=0,j=1,j\neq 7}^{2n,12}, \{X^i Y^j Z^k\}_{i=0,j=1,k=1,(i,j)\neq(n,4)}^{2n,6,3n}, \{X^i Y^j Z^{6n}\}_{i=0,j=1}^{n,4}\right\}.$$

Set $a'_{i,j,k} = \frac{a_{i,j,0}}{\alpha^i \beta^j \gamma^k}$ and observe that

$$a(X,Y,Z) = \sum_{i,j,k} a_{i,j,k} X^i Y^j Z^k = \sum_{i,j,k} a'_{i,j,k} (\alpha X)^i (\beta Y)^j (\gamma Z)^k = a'(\alpha X, \beta Y, \gamma Z).$$

Wlog. we can then rename the variables αX, βY, γZ by X,Y,Z to get that

$$a'(X,Y,Z)^2 + \Big(a'(X,Y,Z) + Y^5 + t(X)Y^6 \\ - \sum_{i=0}^{\ell} a_i(w_i(X)Y^2 + u_i(X)Y^3 + v_i(X)Y^4)\Big) \cdot (Z^{6n} + \sum_{k=1}^{3n-m+\ell} n_k(X,Y)Z^k)$$

The span has no monomials of the form $X^iY^jZ^k$ for $k > 6n$. Looking at the sub-part $a'(X,Y,Z)Z^{6n}$ we deduce that $a'_{i,j,k} = 0$ for all $k \neq 0$, which means

$$a'(X,Y,Z) = a'_{0,0,0} + a'_{1,0,0}X + \sum_{i=0,j=1}^{n,6} a'_{i,j,0}X^iY^j.$$

There are also no Z^{6n} or XZ^{6n} monomials in the span, so we get $a'_{0,0,0} = 0$ and $a'_{1,0,0} = 0$. We are now left with

$$a'(X,Y,Z) = \sum_{i=0,j=1}^{n,6} a'_{i,j,0}X^iY^j.$$

Define $p(X,Y)$ such that

$$p(X,Y) = \sum_{i=0,j=1}^{n,6} a'_{i,j,0}X^iY^j + Y^5 + t(X)Y^6 - \sum_{i=0}^{\ell} a_i(w_i(X)Y^2 + u_i(X)Y^3 + v_i(X)Y^4).$$

Looking at the remaining terms of the form $X^iY^jZ^k$ we see that for $k = 0,\ldots,3n-m+\ell$

$$p(X,Y)\eta_k(X,Y) \in \mathsf{span}\{X^iY^j\}_{i=0,j=1,(i,j)\neq(n,4)}^{2n,6}.$$

This implies $p(X,Y)\eta_k(X,Y)$ has coefficient 0 for the term X^dY^4. Recall the $n_k(X,Y)$ polynomials are constructed such that this is only possible if $p(X,Y)$ can be written as

$$Yq(X) + \sum_{i=\ell+1}^{m} a_i(w_i(X)Y^2 + u_i(X)Y^3 + v_i(X)Y^4) + r_1t(X)Y^2 + r_2t(X)Y^3 + r_3t(X)Y^4.$$

Finally, we look at terms of the form X^iY^7. These do not exist in the span, so

all the terms of that form in $a(X,Y,Z)^2$ should sum to zero. This implies

$$\left(\begin{array}{l} q(X)\cdot Y + \sum_{i=0}^{m} a_i(w_i(X)Y^2 + u_i(X)Y^3 + v_i(X)Y^4) \\ +r_1 t(X)Y^2 + r_2 t(X)Y^3 + r_3 t(X)Y^4 - Y^5 - t(X)Y^6 \end{array} \right)^2$$

should have no X^iY^7 terms. This in turn implies

$$2\left(\begin{array}{c} (r_3 \sum_{i=0}^{m} a_i u_i(X) + r_2 \sum_{i=0}^{m} a_i v_i(X) - r_1 - q(X))\cdot t(X) \\ -\sum_{i=0}^{m} a_i w_i(X) + \sum_{i=0}^{m} a_i u_i(X) \cdot \sum_{i=0}^{m} a_i v_i(X) \end{array} \right) = 0$$

By definition of qap we now have that $(a_{\ell+1},\ldots,a_m)$ is a witness for the instance $(a_1,\ldots,a_\ell)$. □

Chapter 7

Sonic: Zero-Knowledge Arguments from Linear Sized SRSs

This Chapter presents work [23] together with Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. The author was the primary investigator in this work. The construction, which we call Sonic, is of a NIZK with an updatable and universal SRS of linear size (as opposed the quadratic SRS in Chapter 6). The proof sizes are succinct and the verifier's computation is succinct in the "helper" setting. By this we mean that there is an untrusted helper that produces succinct arguments that a batch of proofs verify, and a verifier can use this argument to check the entire batch with a one off linear cost and a succinct cost per proof.

7.0.1 Building Blocks

Sonic uses two main primitives as building blocks: a polynomial exponent commitment scheme and a signature of correct computation [67]. A polynomial commitment scheme is defined by three DPT protocols:

- $(\mathsf{ComR}, \mathsf{ComR}') \leftarrow \mathsf{Commit}(bp, \mathtt{srs}, f(X))$ takes as input the bilinear group, the structured reference string, a maximum degree, and a Laurent polynomial with powers between $-d$ and d. It returns a commitment ComR, and a proof of knowledge of the contents of the commitment ComR'.

- $b \leftarrow \mathsf{pcV}(bp, \mathtt{srs}, \mathsf{ComR}, \mathsf{ComR}', R, \pi)$ takes as input the bilinear group, the SRS, a commitment, an evaluation and a proof. It outputs a bit indicating

acceptance ($b = 1$), or rejection ($b = 0$).

We require that this scheme is *evaluation binding*; i.e., given a commitment ComR, an adversary cannot open ComR to two different evaluations R_1 and R_2 (more formally, that it cannot output a tuple $(\mathsf{ComR}, R_1, \pi_1, R_2, \pi_2)$ such that pcV returns 1 on both sets of evaluations and proofs). We also require that it is *polynomial extractable*; i.e., any adversary that can provide a valid evaluation opening also knows an opening $f(X)$ with powers $-d \leq i \leq d, i \neq 0$ (more formally, that this is true for any adversary that outputs a tuple $(\mathsf{ComR}, \mathsf{ComR}')$ that passes verification). For both properties, we require that they hold with respect to an adversary that can update the SRS; i.e., that has access initially to the oracle in Definition 3.1.3.

In Section 7.4 we provide a polynomial exponent commitment scheme satisfying these two properties. We prove its security in Lemma 7.4.1.

A signature of correct computation is defined by two DPT protocols:

- $(s(z,y), \mathsf{sc}) \leftarrow \mathsf{scP}(bp, \mathtt{srs}, s(X,Y), y)$ takes as input the bilinear group, the SRS, a two-variate polynomial $s(X,Y)$, and a point in the field y. It returns a commitment $S = \mathsf{Commit}(bp, \mathtt{srs}, s(X,y))$ and a proof sc.

- $b \leftarrow \mathsf{scV}(bp, \mathtt{srs}, s(X,Y), (z,y), S, \mathsf{sc})$ takes as input the same parameters as the scP algorithm in addition to an evaluation and a proof. It outputs a bit indicating acceptance ($b = 1$), or rejection ($b = 0$).

We require that this scheme is sound; i.e., given y and S, an adversary can convince the verifier only if $S = \mathsf{Commit}(bp, \mathtt{srs}, s(X,y))$.

In Section 7.4 we provide a signature of correct computation satisfying this property. We prove its security in Lemma 7.4.2.

7.0.2 Our Techniques

At a high level, our protocol combines techniques from both Bulletproofs and zk-SNARKs to obtain a scheme with many of the benefits of both. In order to achieve this, we make heavy use of a pairing-based polynomial commitment scheme due to Kate et al. [66].

In a bit more detail, the goal of Sonic is to provide zero-knowledge arguments for the satisfiability of arithmetic circuits. The key idea behind our protocol is to start with a two-variate polynomial equation used in Bulletproofs that was designed by Bootle et al. [61]. The polynomial equation is satisfied if and only if the circuit is satisfied. This two-variate polynomial depends only on monomials and not on polynomials. By designing our scheme around Bulletproofs, as opposed to around the quadratic arithmetic programs that are typically used by zk-SNARKs, we acquire an SRS that naturally depends only on monomials. Thus we derive a scheme that can be built from our universal and updatable structured reference string. The prover is required to calculate polynomials *in the exponent* at unknown points x and known points y. The prover does not reveal its polynomials, but rather provides commitments to them using Kate et al.'s polynomial commitment scheme. This scheme has constant size and verification time, but is designed for single-variate polynomials, whereas our polynomials are two-variate.

There are three key observations that allow us to use a single-variate polynomial commitment scheme in order to commit to two-variate polynomials.

The first observation is that for polynomials of the form $f(X,Y) = \sum_{i=0}^{d} a_i X^i Y^i$, we have that for all z and y in $\mathbb{F}$, $f(zy,1) = f(z,y)$. We use this to have our prover first commit to $f(X,1)$ and later commit to $f(X,y)$ at a random point y. The verifier sends the prover a random point z. By opening the first commitment at zy and the second at z and checking that the results are equal, the verifier learns that the two commitments are consistent. Our prover hides their witness in the unknown polynomial before they learn y, and are later required to scale their committed polynomial by y.

The second observation is that when the prover and verifier both know a two-variate polynomial to which the verifier wants to calculate a commitment, this work can be unloaded onto the prover. We utilise this observation by placing the work of uploading a polynomial specifying the circuit onto the prover. The prover can commit to the polynomial, and the verifier can check that the commitment is correct by asking the prover to open the commitment at a random point, and then checking that the result is the evaluation of the polynomial at that point. This shifts the required

work for the verifier from computing a polynomial in the exponent to computing a value in the field.

Our third observation is that verifiers can sometimes be helped by entities other than the prover, which we call "helpers". In particular, when off-loading its computation to the prover, the verifier asks each individual prover to open their commitments at different random points. Using a helper, however, it can instead ask the helper to open all of the commitments at the *same* random point. Thus the verifier only needs to compute a polynomial in the field once. Therefore, they can be convinced that all of the commitments have been computed correctly provided that the helper's argument verifies.

7.1 Our Techniques

Like all the constructions in this thesis, Sonic is used to show satisfiability of arithmetic circuits. There are a number of techniques for parsing from an arithmetic circuit to a set of polynomial constraints that are compatible with zero-knowledge proofs such as the quadratic arithmetic programs discussed in other chapters [73]. Sonic uses the polynomial constraints system described in [61]. The system in [61] was designed not with amortisation in mind, but with the goal of being practical even if only used once. Hence we obtain a system in which it is relatively cheap to derive the circuit specific constraints.

Pairing based SNARKs require the prover to show that the exponents in the group elements are in the correct span of the polynomial constraints. Our updatable and universal zk-SNARK does this with a null-space argument and our SE-SNARK with a relation dependent SRS. This is where Bulletproofs are truly ideal for Sonic. Bulletproofs use a polynomial constraint system where the exponents of the group elements must only be linearly independent from a single monomial and this can be shown with a relation independent SRS.

To use the constraint system in Bulletproofs that was originally introduced in [61] we require a polynomial commitment scheme. Here we can take advantage of our structured reference string and the fact that we are allowing pairings to use an

adaption of a scheme by Kate, Zaverucha, and Goldberg [66]. They have constant sized proofs for any sized polynomial (up to a given bound) and verification consists of checking a pairing.

The Sonic prover first computes three polynomials that depend on the witness and satisfy the verifier's constraints. Second, the prover commits to one of these polynomials. The other two polynomials are entirely determined by the first polynomial. Third, the verifier sends a challenge. Fourth, the prover evaluates their polynomials at the challenge point. They send the evaluated polynomials and a proof of correct evaluation to the verifier. The verifier checks that the polynomials satisfy the polynomial constraints. If they do, then it knows that the evaluated polynomials were computed by a prover that knows a valid witness.

Our linear verification cost comes in checking that the polynomials satisfy the polynomial constraints. In particular, it is required to calculate a two-variate (but sparse!) polynomial in the field. Note that this is more efficient than Bulletproofs in which the verifier is required to calculate a two-variate polynomial in the exponent. In the batched setting, a Bulletproofs verifier can calculate a polynomial in the exponent once, but still needs to calculate a two-variate polynomial for every proof in the batch. Sonic, on the otherhand, uses a helper (described in Section 7.3) to reduce the verifier work per proof to constant.

7.1.1 Structured Reference String

In all of the following we require a structured reference string with unknowns x and α of the following form

$$\left\{ \ \{g^{x^i}\}_{i=-d}^{d},\ \{g^{\alpha x^i}\}_{i=-d,i\neq 0}^{d},\ \{h^{x^i}, h^{\alpha x^i}\}_{i=-d}^{d},\ e(g,h^{\alpha}) \ \right\}$$

for some large enough d to support the circuit with n multiplication gates. By omitting g^{α} from the reference string we can, when necessary, force the prover to demonstrate that a committed polynomial (in x) has a zero constant term. We can generate such a reference string using techniques described in Chapter 5.

7.1.2 Circuit Processing Techniques from Bulletproofs

We process our circuits using techniques by Bootle et al. [61]. Let **a**, **b**, and **c** be n-length vectors over $\mathbb{Z}_p$ such that the i-th entries correspond to the left, right and output wires, respectively, of the i-th multiplication gate of an arithmetic circuit. The goal of our protocol is to allow a prover to demonstrate knowledge of a satisfying witness $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ for some fixed circuit.

We will be using a quadratic polynomial equation by Bootle et al. [61], with some minor tweaks. We start by expressing our circuit as a system of arithmetic constraints so that the prover can equivalently demonstrate that the system is satisfied.

Our constraint system consists of n multiplication constraints corresponding to each multiplication gate, where the i-th multiplication constraint is of the form

$$a_i \cdot b_i = c_i.$$

We will also have Q linear constraints in order to simulate the effects of addition gates, where the q-th linear constraint is of the form

$$\sum_{i=1}^{n} a_i u_{i,q} + \sum_{i=1}^{n} b_i v_{i,q} + \sum_{i=1}^{n} c_i w_{i,q} = k_q$$

for **k** that depends on the instance and for constant coefficients **u**, **v**, and **w**.

For example, to represent the constraint $x^2 + y^2 = z$, one would set

- $\boldsymbol{a} = (x, y)$, $\boldsymbol{b} = (x, y)$, $\boldsymbol{c} = (x^2, y^2)$
- $\boldsymbol{u}_1 = (1,0), \boldsymbol{v}_1 = (-1,0), \boldsymbol{w}_1 = (0,0), k_1 = 0$
- $\boldsymbol{u}_2 = (0,1), \boldsymbol{v}_2 = (0,-1), \boldsymbol{w}_2 = (0,0), k_2 = 0$
- $\boldsymbol{u}_3 = (0,0), \boldsymbol{v}_3 = (0,0), \boldsymbol{w}_3 = (1,1), k_3 = z$

Any arithmetic circuit can be represented with our constraint system by using the multiplication constraints to determine the multiplication gates and the linear constraints to determine the wiring of the circuit and the addition gates. Thus the constraint system covers NP.

We follow the general logic of Bootle et al. to embed each equation into the coefficient of a polynomial in formal indeterminate Y, producing the equation

$$\sum_{i=1}^{n} a_i u_i(Y) + \sum_{i=1}^{n} b_i v_i(Y) + \sum_{i=1}^{n} c_i w_i(Y) + \sum_{i=1}^{n} Y^i a_i b_i - k(Y) = 0 \tag{7.1}$$

where

$$\begin{aligned} u_i(Y) &= \sum_{q=1}^{Q} Y^{q+n} u_{i,q} & v_i(Y) &= \sum_{q=1}^{Q} Y^{q+n} v_{i,q} \\ w_i(Y) &= -Y^i + \sum_{q=1}^{Q} Y^{q+n} w_{i,q} & k(Y) &= \sum_{q=1}^{Q} Y^{q+n} k_i \end{aligned} .$$

This equation will hold at all Y for valid wire assignments and has negligible probability of holding for invalid assignments at most Y given a large enough field.

Bootle et al. argue that (7.1) holds by constructing a special Laurent polynomial $t(X,Y)$ in a second formal indeterminate X which has $2k(Y)$ as the constant term. We use a modification of this polynomial, swapping exponents to simplify polynomial commitments later. Where Bootle et al. set

$$r(X,Y) = \sum_{i=1}^{n} a_i Y^i X^i + \sum_{i=1}^{n} b_i X^{-i} + \sum_{i=1}^{n} c_i X^{i+n}$$

we set

$$r(X,Y) = \sum_{i=1}^{n} a_i Y^i X^i + \sum_{i=1}^{n} b_i X^{-i} + \sum_{i=1}^{n} c_i X^{-i-n}$$

i.e. we use negative exponents for the output wires rather than positive exponents. We then modify our other polynomials to accommodate this change. We also append 6 dummy multiplication gates so that the additional wire values can act as blinding factors for the polynomials. This is what gives our scheme its zero-knowledge properties.

$$\begin{aligned} r(X,Y) &= \sum_{i=1}^{n+6} a_i Y^i X^i + \sum_{i=1}^{n+6} b_i X^{-i} + \sum_{i=1}^{n+6} c_i X^{-i-n-6} \\ s(X,Y) &= \sum_{i=1}^{n+6} u_i(Y) Y^{-i} X^{-i} + \sum_{i=1}^{n+6} v_i(Y) X^i + \sum_{i=1}^{n+6} w_i(Y) X^{i+n+6} \\ r'(X,Y) &= r(X,Y) + 2s(X,Y) \\ t(X,Y) &= r(X,Y) r'(X,Y) - 2k(Y) \end{aligned} \tag{7.2}$$

By demonstrating that the constant term of $t(X,Y)$ is zero, we demonstrate that (7.1) is satisfied.

7.2 Our Sonic Construction

The general approach behind Sonic is to have the prover commit to the polynomial $r(X,Y)$ with their choice of witness $(\mathbf{a},\mathbf{b},\mathbf{c})$, and then evaluate $r(x,y)$ and $s(x,y)$ such that the verifier can apply pairings to ensure that $t(X,Y)$ has a constant term of zero.

We split the polynomial $r(X,Y)$ by expressing it as the sum of two polynomials

$$\begin{aligned} r_1(X,Y) &= \sum_{i=1}^{n+6} a_i X^i Y^i \\ r_2(X) &= \sum_{i=1}^{n+6} b_i X^{-i} + \sum_{i=1}^{n+6} c_i X^{-i-n-6} \end{aligned}$$

The prover begins by sending the verifier commitments $\mathsf{ComR} = g^{\alpha r_1(x,1)}$ and $R_2 = g^{\alpha r_2(x)}$. The verifier chooses $y \xleftarrow{\$} \mathbb{Z}_p$ so that the prover can send $R_1 = g^{\alpha r_1(x,y)}$ and $S = g^{\alpha s(x,y)}$ to the verifier.

The subprotocol $(\mathsf{scP},\mathsf{scV})$ is run over S to ensure it is correct. Because $s(X,Y)$ is a Laurent polynomial, the subprotocol scales the commitment by h^{x^n} within the pairing equation so that the prover and verifier can work over a polynomial in X.

The subprotocol $(\mathsf{peP},\mathsf{peV})$ is ran over ComR and R_1 to ensure R_1 was computed consistently with y. The verifier is concerned that R_1 may be a commitment to a polynomial in X of degree larger than n, and so the prover sends $R_x = g^{\alpha x^{d-n-6} r_1(x,y)}$ to demonstrate that this is not the case, which the verifier checks with a pairing.

The prover shows that R_1, R_2 are commitments to polynomials in the variables X, X^{-1} respectively. It does this by providing elements $A_1 = R_1^{x^{-d}}$ and $A_2 = R_2^{x^d}$, which the verifier checks.

The prover sends $R' = h^{\alpha(2s(x,y)+r(x,y))}$ which the verifier checks. Finally, the prover sends $T = g^{\alpha t(x,y)}$ and the verifier is convinced if

$$e(R_1 R_2 (2S), R') e(g,h)^{-2\alpha^2 k(y)} = e(T, h^\alpha)$$

holds. If T has a term that is not scaled by α, then the verifier's equation is not satisfied as all of R_1, R_2, S only have terms scaled by α. This demonstrates that $t(X,Y)$ has a constant term of zero.

Common Input: $\mathsf{info} = \ \mathsf{bp}, \mathtt{srs}, s(X,Y), k(Y), \phi, e(g^{x\alpha}, h^{\alpha x^{-1}})$

$\underline{\mathsf{zkP}_1(\mathsf{info}, \mathbf{a}, \mathbf{b}, \mathbf{c}) \mapsto (\mathsf{zk.msg}_1, \mathsf{zk.st}_1)}$:
$a_{n+1}, \ldots, a_{n+6}, b_{n+1}, \ldots, b_{n+6} \xleftarrow{\$} \mathbb{Z}_p$
$c_{n+1}, \ldots, c_{n+6} \leftarrow a_{n+1} \cdot b_{n+1}, \ldots, a_{n+6} \cdot b_{n+6}$
$r_1(X) \leftarrow \sum_{i=1}^{n+6} a_i X^i$
$r_2(X) \leftarrow \sum_{i=1}^{n+6} b_i X^{-i} + c_i X^{-i-n-6}$
$(\mathsf{pe.msg}_1, \mathsf{pe.st}_1) \leftarrow \mathsf{peP}_1(\mathtt{srs}, r_1(X))$
$R_2 \leftarrow g^{\alpha r_2(x)}$
$A_2 \leftarrow g^{r_2(x)x^d}$
$\mathsf{zk.msg}_1 \leftarrow \{R_2, A_2, \mathsf{pe.msg}_1\}$
$\mathsf{zk.st}_1 \leftarrow \{\mathsf{info}, r_1(X), r_2(X), \mathsf{pe.st}_1\}$
return $(\mathsf{zk.msg}_1, \mathsf{zk.st}_1)$

$\underline{\mathsf{zkV} \mapsto \mathsf{zkP}}$: Send $y \xleftarrow{\$} \mathbb{Z}_p$ to prover

$\underline{\mathsf{zkP}_2(\mathsf{st}_1, y) \mapsto (\mathsf{zk.msg}_2, \mathsf{zk.st}_2)}$:
$(\mathsf{pe.msg}_2, \mathsf{pe.st}_2) \leftarrow \mathsf{peP}_2(\mathsf{pe.st}_1, y)$
$(\mathsf{sc.msg}_1, \mathsf{sc.st}_1) \leftarrow \mathsf{scP}_1(\mathtt{srs}, X^{n+6}s(X,Y), y)$
$A_1 \leftarrow g^{r_1(xy)x^{-d}}$
$S \leftarrow g^{\alpha s(x,y)}$
 # The value $R_1 = \mathsf{pe.msg}_2$ is equal to $g^{r_1(xy)}$.
 # The value $S_x = \mathsf{sc.msg}_1$ is equal to $g^{x^{n+6}s(x,y)}$.
$\mathsf{zk.msg}_2 = \{S, A_1, \mathsf{sc.msg}_1, \mathsf{pe.msg}_2\}$
$\mathsf{zk.st}_2 = \{\mathsf{pe.st}_2, \mathsf{sc.st}_1\}$
return $(\mathsf{zk.msg}_2, \mathsf{zk.st}_2)$

$\underline{\mathsf{zkV} \mapsto \mathsf{zkP}}$: Send $z \xleftarrow{\$} \mathbb{Z}_p$ to prover

$\underline{\mathsf{zkP}_3(\mathsf{zk.st}_2, i) \mapsto (\mathsf{zk.msg}_3)}$:
$R' \leftarrow h^{\alpha r(x,y) + 2\alpha s(x,y)}$
$R_x \leftarrow R_1^{\alpha x^{d-n-6}}$
$T \leftarrow g^{\alpha t(x,y)}$
$\mathsf{pe.msg}_3 \leftarrow \mathsf{peP}_3(\mathsf{peP.st}_2, z)$
$\mathsf{sc.msg}_2 \leftarrow \mathsf{scP}_2(\mathsf{sc.st}_1, z)$
$\mathsf{zk.msg}_3 \leftarrow \{R', R_x, T, \mathsf{pc.msg}_2, \mathsf{pe.msg}_3, \mathsf{sc.msg}_2\}$
return $\mathsf{zk.msg}_3$

$\underline{\mathsf{zkV}(\mathsf{info}, \mathsf{zk.msg}_1, \mathsf{zk.msg}_2, \mathsf{zk.msg}_3, y, z) \mapsto 0/1}$:
set $R = R_1 R_2$
check $\mathsf{peV}(\mathtt{srs}, \mathsf{pe.msg}_1, y, \mathsf{pe.msg}_2, \mathsf{pe.msg}_3, z)$
check $\mathsf{scV}(\mathtt{srs}, X^{n+6}s(X,Y), y, \mathsf{sc.msg}_1, \mathsf{sc.msg}_2, z)$
check $e(A_1, g^{\alpha x^{-d}}) = e(R_1, h)$
check $e(A_2, h^{\alpha}) = e(R_2, h^{x^d})$
check $e(g, R') = e(RS^2, h)$
check $e(R_x, h) = e(R_1, h^{x^{d-n-6}})$
check $e(S, h^{x^{n+6}}) = e(S_x, h^{\alpha})$
check $e(T, h^{\alpha}) = e(R, R')e(g^{\alpha x}, h^{\alpha x^{-1}})^{-2k(y)}$
return 1 if all checks pass, else return 0.

Figure 7.1: Sonic protocol to check that prover knows a valid assignment of the wires in the circuit.

Theorem 7.2.1. *Sonic satisfies subversion zero-knowledge.*

Proof. To prove subversion zero-knowledge, we need to both show the existence of an extractor $\mathscr{X}_{\mathscr{A}}$ that can compute a trapdoor from the updated proofs, and describe a SimProve algorithm that produces indistinguishable proofs when provided with the extracted trapdoor.

The simulator is given the trapdoor g^{α} and chooses random vectors $\boldsymbol{a}$, $\boldsymbol{b}$ from $\mathbb{Z}_p$ of length $n+6$ and sets $\boldsymbol{c} = \boldsymbol{a} \cdot \boldsymbol{b}$. It computes $r(X,Y)$, $r'(X,Y)$, $t(X,Y)$ as in (7.2) where (unlike for the prover) $t(X,Y)$ can have a non-zero coefficient in X^0. The simulator then behaves exactly as the prover in Figure 7.1 with its random

polynomials. Observe that the simulator can compute $g^{\alpha t(x,y)}$ for all $y \in \mathbb{Z}_p$.

Both the prover and the simulator utilise a polynomial exponent commitment scheme for $r_1(X,Y)$ and they also both provide $r_2(X)$ in the exponent. The former reveals the values $r_1(x,1)$, $r_1(x,y)$, $r_1(z,y)$, $w_0(x)$, $w_1(x)$ i.e. 5 evaluations (some of these are in the exponent). The latter simply reveals the value $r_2(x)$. The prover has six blinders for $r_1(X)$, $(a_{n+1},\cdots,a_{n+6})$ and six blinders for $r_2(X)$, $(b_{n+1},\cdots,b_{n+6})$. Thus for a verifier that obtains less than six evaluations, the prover's polynomial is indistinguishable from the simulators random polynomial. All other components in the proofs are either uniquely determined given the previous components for both prover and simulator, or are calculated independently from the witness (and are chosen in the same method by both prover and simulator). Consequently, subversion zero-knowledge follows from the extraction of the trapdoor, which we show below.

A reference string and proof $(\mathtt{srs}, \{\rho\}_{i=1}^{m})$ that passes verification is structured as if it were computed by $\mathsf{Setup}(1^\lambda)$ i.e. there exist values $(x,\alpha) \in \mathbb{Z}_p^2$ such that $\mathtt{srs} = \left\{ \{g^{x^i}\}_{i=-d}^{d}, \{g^{\alpha x^i}\}_{i=-d,i\neq 0}^{d}, \{h^{x^i}, h^{\alpha x^i}\}_{i=-d}^{d} \right\}$. The ith verifying update includes a proof of knowledge for α_i such that $\alpha = (\alpha_1 \cdots \alpha_{i-1}) \times \alpha_i$. Combining the extractors for these proofs of knowledge gives $\alpha = \alpha_1 \cdots \alpha_n$. The combined extractors run in polynomial time because, where the proofs of knowledge use Fischlin transforms, the extractors are straight-line. □

Theorem 7.2.2. *Sonic has updatable witness extended emulation assuming the d-PKE and the 3d-BGDHE asssumptions hold.*

Proof. By Lemma 5.3.1, it suffices to consider an adversary that performs all the updates except the original setup. Suppose that $\mathscr{A}$ is an adversary that receives a structured-reference string from the setup algorithm

$$\left\{ \{g^{x_1^i}\}_{i=-d}^{d}, \{g^{\alpha_1 x_1^i}\}_{i=-d,i\neq 0}^{d}, \{h^{x_1^i}, h^{\alpha_1 x_1^i}\}_{i=-d}^{d} \right\}.$$

Further suppose that $\mathscr{A}$ performs m verifying updates to set the `srs` to

$$\left\{ \begin{array}{l} \{g^{(x_1 \ldots x_m)^i}\}_{i=-d}^{d}, \\ \{g^{(\alpha_1 \ldots \alpha_m)\ (x_1 \ldots x_m)^i}\}_{i=-d, i\neq 0}^{d} \\ \{h^{(x_1 \ldots x_m)^i}, h^{(\alpha_1 \ldots \alpha_m)(x_1 \ldots x_m)^i}\}_{i=-d}^{d} \end{array} \right\}.$$

Since the adversary's updates verify, there exists an extractor that outputs $\alpha_2, \ldots, \alpha_m$ and $x_2, \ldots, x_m$. For ease of notation, we set $x = x_1 \ldots x_m$ and $\alpha = \alpha_1 \ldots \alpha_m$.

Suppose that $\mathscr{A}$ can convince a verifier with non-negligible probability. Our extractor runs the first step of $\mathscr{A}$ to get $\mathsf{zk.msg}_1$ with respect to the updated `srs` and an instance ϕ. It then runs the second two stages of $\mathscr{A}$, with the random challenges y and z.

The verifier checks that that $e(g, R') = e(RS^2, h)$. Therefore, by the q-PKE assumption there exists an extractor $\mathscr{X}_{\mathscr{A}}$ that outputs u_i, v_i such that

$$\log(R') = 2\alpha s(x,y) + \textstyle\sum_{i=-d}^{d} u_i x_1^i + v_i \alpha_1 x_1^i \ .$$

By the soundness of the signature of correct computation (see Lemma 7.4.2), $\log(S) = \alpha s(x,y)$ and

$$\log(R) = u(x_1) + \alpha_1 v(x_1).$$

By the soundness of the polynomial exponent commitment scheme (see Lemma 7.4.1), the extractor can output $r_1(X)$ such that $\mathsf{ComR} = g^{\alpha_1 r_1(x_1)}$ and $R_1 = g^{\alpha_1 r_1(x_1)}$. This means that the extractor also learns $r_2(X) = \log(R) - \alpha r_1(X)$.

The verifier checks that

$$e(A_1, h^{\alpha}) = e(g^{\alpha r_1(x_1)}, h^{x^{-d}}) \quad \text{and} \quad e(A_2, h^{\alpha}) = e(g^{\alpha r_2(x_1)}, h^{x^d}).$$

This implies that

$$e(A_1^{\alpha_2 \ldots \alpha_m}, h^{\alpha_1}) = e(g^{\alpha r_1(x_1) x_2 \ldots x_m}, h^{x_1^{-d}})$$
$$e(A_2^{\alpha_2 \ldots \alpha_m}, h^{\alpha_1}) = e(g^{\alpha r_2(x_1) x_2 \ldots x_m}, h^{x_1^{d}}).$$

If $r_{1,-1},\ldots,r_{1,-d},r_{2,1},\ldots,r_{2,d} \neq 0$ then $\mathscr{A}$ obtains non-trivial linear combinations of $x_1^{d+1},\ldots,x_1^{2d}$, breaking the $(2d+i)$-BGDHE assumption for $0 \leq i \leq d$. If $r_{1,0},r_{2,0} \neq 0$ then the adversary breaks the d-BGDHE assumption. Where $e(R_x,h) = e(g^{r_1(x_1)},h^{x^{d-n-6}})$, we get that $r_{1,n+7},\ldots,r_{1,d}$ must equal zero. If not, then the adversary could obtain non-trivial linear combinations of $x_1^{d+1},\ldots,x_1^{2n-n-6}$, breaking the $(2d+i)$-BGDHE assumption.

As a result, the extractor learns that $\log(R)$ is given by

$$r(x,y) = \alpha_1 \sum_{i=1}^{n+6} r_{1,i}x^i y^i + \alpha_1 \sum_{i=1}^{d} r_{2,i}x^{-i}.$$

They also learn that $R' = h^{2\alpha s(x,y)+\alpha_1 r(x_1,y))}$. This means that

$$\log(T) = \alpha_1 r(x_1,y)\cdot(2\alpha s(x,y)+\alpha_1 r(x_1,y)) - 2\alpha k(y).$$

Now $\log(T)$ cannot have a zero-term in $\alpha_1 x_1$ by the d-BGDHE assumption. Set

$$\begin{array}{rcccc} (\alpha_2\ldots\alpha_m)\boldsymbol{a} & = & \frac{r_{1,1}}{x_2\ldots x_m}, & \ldots, & \frac{r_{1,n+6}}{(x_2\ldots x_m)^{n+6}} \\ (\alpha_2\ldots\alpha_m)\boldsymbol{b} & = & \frac{r_{2,1}}{(x_2\ldots x_m)^{-1}}, & \ldots, & \frac{r_{2,n+6}}{(x_2\ldots x_m)^{-n-6}} \\ (\alpha_2\ldots\alpha_m)\boldsymbol{c} & = & \frac{r_{2,n+7}}{(x_2\ldots x_m)^{-n-7}}, & \ldots, & \frac{r_{2,2n+12}}{(x_2\ldots x_m)^{-2n-12}} \end{array}.$$

This means that $\sum_{i=1}^{n+6} y^i a_i b_i + a_i u_i(y) + b_i v_i(y) + c_i w_i(y) - 2k(y) = 0$. Finally, suppose that this holds for $n+Q+1$ different challenges $y \in \mathbb{Z}_p$. Then, we have equality of polynomials in (7.1), since a non-zero polynomial of degree $n+Q+1$ cannot have $N+Q$ roots. This means that the circuit is satisfied. □

7.2.1 Efficiency

Our system uses 1 polynomial exponent commitment and 1 signature of correct computation in addition to the explicit checks. Overall there are 13 elements in $\mathbb{G}_1$, 4 elements in $\mathbb{G}_2$, and 1 element in $\mathbb{Z}_p$. The verifier is required to compute 24 pairings in total, or 12 pairing equations (of these 11 can be batched). Thus the batched verifier needs to compute $2m+22$ pairings where m is the number of proofs in each batch.

7.2.1.1 Circuit Processing Adaptations for Sonic

Our aim is to amortise computation over many proofs, thus we now differ from [61] in the two following fashions.

First Adaptation: In order to run our "helper" to amortise the verifier costs, we require that the polynomial $s(X,Y)$ is unaffected by the instance, i.e., we require that the vectors $\boldsymbol{u}_q, \boldsymbol{v}_q, \boldsymbol{w}_q$ are unaffected by the instance. Recall that the instance is denoted by ϕ and has size ℓ. We use between ℓ and 2ℓ additional constraints to include the instance. These are designed so that the instance only affects the scalars k.

- When ϕ_m is a wire that leads into a multiplication gate we can simply include an equality check. For example, suppose that $\mathbf{a}_i = \phi_m$. Then we would set $\mathbf{u}_q$ to equal 1 in the ith entry and zero everywhere else, $\mathbf{v}_q$, $\mathbf{w}_q$ to equal 0 in all entries, and $k_q = \phi_m$.

- In the more complicated scenario where ϕ_j is used to scale an addition constraint e.g. $a_i u_{q,i} + b_j \phi_m = k_q$ we do the following: (1) add a multiplication gate to the circuit of the form $a_{n+1} b_j = c_{n+1}$: (2) check $\phi_m = a_{n+1}$ as per the previous example; (3) set $w_{q,n+1} = 1$ in the ith entry and zero everywhere else, and replace our original constraint with one of the form $a_i u_{q,i} + c_{n+1} w_{q,n+1} = k_q$. This ensures that only k_{q+1} is affected by the instance.

Second Adaptation: The second adaptation was designed by Sean Bowe. The verifier is expected to calculate a polynomial $k(Y)$, and the sparseness of this polynomial depends on the number of non-zero k_q values. To keep to verifier costs down, we use the following trick to ensure that all "non-instance" values k_q are equal to zero. First we add three instance wires to the circuit, a_1, b_1, c_1, all of which are set to equal 1 (the circuit only needs to check that $a_1 = 1$). Then when

$$\mathbf{a} \cdot \mathbf{u}_q + \mathbf{b} \cdot \mathbf{v}_q + \mathbf{c} \cdot \mathbf{w}_q = k_q$$

Scenario	Helper time	Verifier time	Helper proof size
No helper	0	$\mathcal{O}(m \times n)$	0
Helper	$\mathcal{O}(mn\log(n))$	$\mathcal{O}(m)+\mathcal{O}(n)$	$2m+1$ $\mathbb{G}_1$, m $\mathbb{Z}_p$

Table 7.1: Efficiency comparison for the helper and the (un)-helped verifier for Sonic.

if we reset $u_{q,1}$ from 0 to $-k_q$ we have that

$$\mathbf{a}\cdot\mathbf{u}_q+\mathbf{b}\cdot\mathbf{v}_q+\mathbf{c}\cdot\mathbf{w}_q=0.$$

7.3 A Helping Hand for the Verifier

As currently described, the zero-knowledge proofs in Sonic have linear verification time, because the verifier has to check the signature of correct computation. However, in the amortised setting, where one is proving the same thing many times, we can use what we refer to "helpers" in order to make the linear cost a one-off payment, and then we have constant verification costs per proof. The proofs provided by the helper are succinct, and the helper can be run by anyone (i.e., they do not need any secret information from the prover). The natural candidate for this role in the setting of blockchains is a miner, as they are already investing computational energy into the system.

The key observation is that if the helper knows the evaluations $S_1,\ldots,S_m$ in advance, then they can run the protocol for scP where all m iterations receive the same challenge. Then, the expensive part of the scV verifier, namely checking that $s(z,y)=v$, need only be run once.

When a helper's services have been provided, the verifier does not check that the scV verifier is satisfied. Nonetheless, for the challenges $y_1,\ldots,y_m$, the verifier still needs to be assured that $S_1,\ldots,S_m=g^{s(x,y_1)},\ldots,s^{s(x,y_m)}$ for a public polynomial $s(X,Y)$. The helper should not be able to convince a verifier to accept their help unless this is the case. A summary of the efficiency tradeoffs in both the helped and the unhelped scenarios is in Table 7.1. Here n is the number of multiplication gates, m is the number of proofs for the same circuit, and k is the number of times

the chained helpers update the chain.

7.3.1 The Helper

The algorithm for our helper is given in Figure 7.2. The helper is denoted by hscP and the verifier is denoted by hscV. Roughly the idea is as follows. The helper is given a random challenge by the verifier. First the helper computes an element C which contains the polynomial $s(X,Y)$ in the exponent, but it flips the unknown coefficient. So rather than computing $s(x,y)$ in the known coefficient y and the unknown coefficient x, it computes $s(z,x)$ in the known challenge u and the unknown challenge x. It sends a compact proof that both S_j and C open to the same point at y_j. Provided that C has been calculated correctly, the helper can open C only to $s(z,y_j)$. Thus if S_j can be opened to $s(z,y_j)$ for any challenge z, then S_j must equal $g^{s(x,y_j)}$. Overall, the verifier is required to check the proofs that S_j and C open to the same values, and that C has been computed correctly.

Common Input: $\mathsf{info} = \quad \mathsf{bp},\ \mathtt{srs},\ \{S_j, \bar{S}_j, y_j\}_{j=1}^m,\ s(X,Y) = \sum_{i,j=0}^d s_{i,j} X^i Y^j$

$\underline{\mathsf{hscV} \mapsto \mathsf{hscP}}$: Send $u \xleftarrow{\$} \mathbb{Z}_p$ to prover

$\underline{\mathsf{hscP}_1(\mathsf{info}, u) \mapsto (\mathsf{hsc.msg}_1, \mathsf{hsc.st}_1)}$:

$C, \bar{C} \leftarrow g^{s(u,x)}, h^{s(u,x)}$

for $1 \le j \le m$: $r_j = s(u, y_j)$

$w_j(X) \leftarrow \frac{s(X,y) - r_j}{X - u}$

$q_j(X) \leftarrow \frac{s(u,X) - r_j}{X - y_j}$

$W_j, Q_j \leftarrow g^{w_j(x)}, g^{q_j(x)}$

$\mathsf{hsc.msg}_1 \leftarrow (C, \{W_j, Q_j, r_j\}_{j=1}^m)$

$\mathsf{hsc.st}_1 \leftarrow (C, u, \{W_j, Q_j, r_j\}_{j=1}^m)$

return $(\mathsf{hsc.msg}_1, \mathsf{hsc.st}_1)$

$\underline{\mathsf{hscV} \mapsto \mathsf{hscP}}$: Send $z \xleftarrow{\$} \mathbb{Z}_p$ to prover

$\underline{\mathsf{hscP}_2(\mathsf{hsc.st}_1, z) \mapsto (\mathsf{hsc.msg}_2)}$:

$c \leftarrow s(u,z)$

$q_0(X) \leftarrow \frac{s(u,X) - c}{X - z}$

$Q_0 \leftarrow g^{q_0(x)}$

$\mathsf{hsc.msg}_2 \leftarrow (Q_0)$

return $(\mathsf{hsc.msg}_2)$

$\underline{\mathsf{hscV}(\mathsf{info}, \mathsf{hsc.msg}_1, \mathsf{hsc.msg}_2, u, z) \mapsto 0/1}$:

$c \leftarrow s(u,z)$

check $e(Cg^{-c}, h) = e(Q_0, h^{x-z})$

check $e(C, h) = e(g, \bar{C})$

for $1 \le j \le m$, check $e(S_j g^{-r_j}, h) = e(W_j, h^{x-u})$

check $e(S_j, h) = e(g, \bar{S}_j)$

check $e(Cg^{-r_j}, h) = e(Q_j, h^{x-y_j})$

return 1 if all checks pass, else return 0

Figure 7.2: The helper protocol for aggregating signatures of correct computation.

Lemma 7.3.1. *Suppose that for a set of elements $S_1, \bar{S}_1, \ldots, S_m, \bar{S}_m$, a corresponding set of points $(y_1, \ldots, y_m)$, and a polynomial $s(X,Y)$, an adversarial helper*

$\mathscr{A}$ convinces a hscV *verifier. Then by the d-PKE and the d-BGSDH assumption,* $S_j = g^{s(x,y_j)}$.

Proof. The verifier checks that $e(C,h) = e(g,\bar{C})$ and $e(S_j,h) = e(g,\bar{S}_j)$. Thus by the d-PKE assumption, there exists an extractor that can output $c(X)$, $f_j(X)$ such that $C = g^{c(x)}$ and $S_j = g^{f_j(x)}$.

The verifier opens C to $s(u,z)$ at z. If $c(X) \neq s(u,X)$ then the verifier can find $g^{\frac{s(u,z)-c(z)}{x-z}}$, breaking the d-BGSDH assumption. The verifier opens C to r_j at y_j. If $r_j \neq s(z,y_j)$ then the verifier can find $g^{\frac{r_j-s(z,y_j)}{x-y_j}}$, breaking the d-BGSDH assumption. The verifier opens S_j to $r_j = s(z,y_j)$. If $f_j(z) \neq s(z,y_j)$ then the verifier can find $g^{\frac{r_j-f_j(z)}{x-z}}$ breaking the d-BGSDH assumption. Suppose we have $2d+1$ challenges such that $f_j(z_i) = s(z_i,y_j)$. Then $f_j(X)$ and $s(X,y_j)$ are degree d Laurent polynomials that are equal on $2d+1$ points, meaning that $f_j(X) = s(X,y_j)$. □

7.4 A Polynomial Commitment Scheme

Kate, Zaverucha, and Goldberg designed a constant-size, pairing-based polynomial commitment scheme [66]. The idea is to commit to a polynomial $f(X)$ by evaluating $C = g^{f(x)}$ at some unknown point x. They then use that for any $z \in \mathbb{F}$, $X - z$ divides $f(X) - f(z)$ perfectly to find $W = g^{\frac{f(x)-f(z)}{x-z}}$. To verify, they use a pairing to show that

$$W^{x-z} = Cg^{-f(z)}.$$

We take inspiration from their scheme in the design of our zero-knowledge argument.

In order to get knowledge extraction we wish to use the q-PKE assumption. This can either go through by putting the proof component W in the second source group $\mathbb{G}_2$ or by including a replicated version of C in $\mathbb{G}_2$ and checking a second pairing. We choose to use the second pairing. This is because when a verifier is checking many proofs, they are able to batch the proofs. In all of the schemes below, we assume that x is a fixed but unknown value hidden in the structured reference string.

7.4.1 Polynomial Exponent Commitment Scheme

We wish to commit to polynomials of the form

$$f(X,Y) = \sum_{i=1}^{m} a_i X^i Y^i$$

and to evaluate them at and unknown point X and a known point y. To do this, the prover commits to $f(X,1)$ and to $f(X,y)$ separately. Then, at a random point z, they open $f(X,1)$ to $f(yz,1)$ and $f(X,y)$ to $f(z,y)$. If the prover behaves honestly then these two values will be equal. If not, then we show that they have negligible probability of being equal.

Common Input: $\mathsf{info} = \mathsf{bp}, \mathtt{srs}, e(g,h^\alpha)$

$\underline{\mathsf{peP}_1(\mathsf{info}, f(X,Y)) \mapsto (\mathsf{pe.msg}_1, \mathsf{pe.st}_1)}$:
$\mathsf{ComR} \leftarrow g^{\alpha f(x,1)}$, $\mathsf{ComR}' \leftarrow h^{\alpha f(x,1)}$,
$\mathsf{pe.msg}_1 \leftarrow \{\mathsf{ComR}, \mathsf{ComR}'\}$,
$\mathsf{pe.st}_1 \leftarrow \{\mathsf{info}, f(X,Y)\}$
return $(\mathsf{pe.msg}_1, \mathsf{pe.st}_1)$

$\underline{\mathsf{peV} \mapsto \mathsf{peP}}$: Send $y \xleftarrow{\$} \mathbb{Z}_p$ to prover

$\underline{\mathsf{peP}_2(\mathsf{pe.st}_1, y) \mapsto (\mathsf{pe.msg}_2, \mathsf{pe.st}_2)}$:
$R \leftarrow g^{\alpha f(x,y)}$, $\bar{R} \leftarrow h^{\alpha f(x,y)}$,
$\mathsf{pe.msg}_2 \leftarrow \{R, \bar{R}\}$, $\mathsf{pe.st}_2 \leftarrow \{\mathsf{info}, f(X,Y), y\}$
return $(\mathsf{pe.msg}_2, \mathsf{pe.st}_2)$

$\underline{\mathsf{peV} \mapsto \mathsf{peP}}$: Send $z \xleftarrow{\$} \mathbb{Z}_p$ to prover

$\underline{\mathsf{peP}_3(\mathsf{st}_2, z) \mapsto \mathsf{pe.msg}_3}$:
$r \leftarrow f(y,z)$,
$w_0(X) \leftarrow \frac{f(X,1)-f(yz,1)}{X-yz}$, $w_1(X) \leftarrow \frac{f(X,y)-f(z,y)}{X-z}$
$W_0, W_1 \leftarrow g^{w_0(x)}, g^{w_1(x)}$, $\mathsf{pe.msg3} \leftarrow \{r, W_0, W_1\}$
return $\mathsf{pe.msg}_3$

$\underline{\mathsf{peV}(\mathsf{info}, \mathsf{pe.msg}_1, \mathsf{pe.msg}_2, \mathsf{pe.msg}_3, y, z)}$:
check $e(\mathsf{ComR}, h) = e(g, \mathsf{ComR}')$
check $e(R,h) = e(g, \bar{R})$
check $e(\mathsf{ComR}, h)e(g,h^\alpha)^{-r} = e(W_0, h^{\alpha x - zy\alpha})$
check $e(R,h)e(g,h^\alpha)^{-r} = e(W_1, h^{\alpha x - z\alpha})$
return 1 if both checks pass, else return 0

Figure 7.3: Polynomial exponent commitment scheme.

Lemma 7.4.1. *The polynomial exponent commitment scheme in Figure 7.3 is evaluation binding and polynomial extractable by the d-PKE, d-BTPF, and d-BGSDH assumptions.*

Proof. Suppose that an adversary outputs ComR, R in the polynomial exponent commitment scheme and convinces a verifier. We show that there exists an extractor that can output $f(X)$ such that $\mathsf{ComR} = g^{f(x)}$ and $R = g^{f(xy)}$.

By Lemma 6 from [22], we only need to consider an adversary that receives an `srs` from the setup, and performs all other updates. Then for an `srs` containing

$g^{x_1 \ldots x_m}$, the adversary knows $x_2, \ldots, x_m$.

From the first pairing equation, the extractor can compute $a(X), b(X)$ such that $\mathsf{ComR} = g^{a(x_1)+\alpha_1 b(x_1)}$ or else the adversary breaks the q-PKE assumption. From the second pairing equation, the extractor can compute $p(X), q(X)$ such that $R = g^{p(x_1)+\alpha_1 q(x_1)}$ or the adversary breaks the d-PKE assumption.

The adversary outputs r, W_0, W_1 such that

$$e(g^{a(x_1)+\alpha b(x_1)-r\alpha}, h) = e(W_0, h^{\alpha x - zy\alpha})$$
$$e(g^{p(x_1)+\alpha_1 q(x_1)-r\alpha}, h) = e(W_1, h^{\alpha x - z\alpha})$$

with $x = x_1 \ldots x_m$ and $\alpha = \alpha_1 \ldots \alpha_m$. If $a(X) \neq 0$ then the adversary can compute

$$e(g,h)^{\frac{a(x)}{\alpha_1}} = e(g^{\alpha_2 \ldots \alpha_m (r - b(x_1))} (W_0)^{\alpha_2 \ldots \alpha_m}, h^{x - zy}).$$

Hence it can break the d-BTPF assumption. Likewise $p(x_1) = 0$.

If $r \neq b(zy)$ then the adversary can find another value $r' = b(zy), W_0'$ that satisfies the verification equation. Hence they can find $g^{\frac{(r'-r)}{\alpha x - z}}$. By multiplying through by $\frac{\alpha_2 \ldots \alpha_m}{r'-r}$ then the adversary breaks the d-BGSDH assumption with $c = \frac{z}{x_2 \ldots x_m \alpha_2 \ldots \alpha_m}$. Likewise, r is equal to $p(z)$.

Suppose this holds for $2d+1$ different values $z_j \in \mathbb{Z}_p$. Then $q(z_j) = q_0 + q_1(z_j) + \ldots + q_n(z_j)^n$ and hence the polynomial $q_0 + \frac{q_1}{y} X + \ldots + \frac{q_n}{y^n} X^n$ evaluates to r_j at $z_j y$. By uniqueness of interpolated polynomials, this scaled polynomial equals $b(X)$. This completes the proof. □

7.4.2 Signature of Correct Computation

Our signature of correct computation is checking the correct evaluation in the exponent rather than in the field. The scheme is given for proving correct evaluation of a known multivariate polynomial $s(X,Y)$ at an unknown point x and a known point y; $s(X,Y) = \sum_{i,j=0}^{n} s_{i,j} X^i Y^j$. The prover calculates $g^{s(x,y)}$. The verifier responds with a challenge z. The prover sends a proof that $s(z,y)$ is a correct evaluation of the committed polynomial at z. This suffices to show that the commitment was calculated correctly. The verifier is required to compute $s(z,y)$, so there is still a

linear overhead (recall $s(X,Y)$ is sparse). However, this linear overhead is in terms of field operations as opposed to group exponentiations, so there is less work for the verifier.

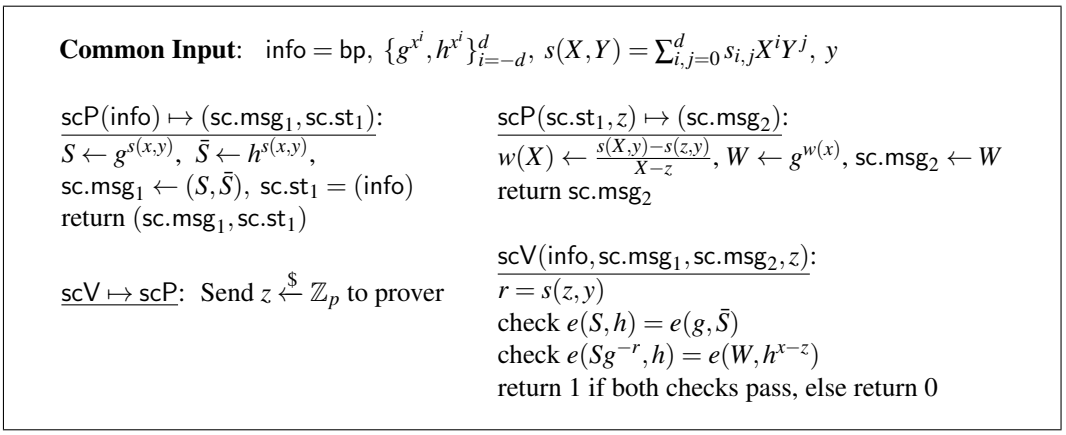

Figure 7.4: Signature of correct computation.

Lemma 7.4.2. *The signature of correct computation in Figure 7.4 is sound by the d-PKE and the d-BGSDH assumptions.*

Proof. We show that if an adversary, including one that can perform updates, outputs a verifying transcript then either $S = g^{s(x,y)}$ or the adversary breaks the d-PKE or the d-BGSDH assumptions.

By Lemma 6 from [22], we only need to consider an adversary that receives an `srs` from the setup, and performs all other updates. Suppose that an adversary has non-negigible probability of outputting a verifying transcript. Then by the d-PKE assumption, the adversary can output a polynomial $f(X)$ such that $S = g^{f(x_1)}$. Hence the adversary can also output

$$p(X) = f_0 + \frac{f_1}{x_1 \dots x_m} X + \dots + \frac{f_n}{(x_1 \dots x_m)^n} X^n$$

such that $S = g^{p(x_1 \dots x_m)}$.

Suppose this holds for $2d+1$ challenges, z_j. If $f(z_j) \neq s(z_j, y)$ then $\mathscr{A}$ can break the d-BGSDH assumption as in Lemma 7.4.1. Since $f(X)$ and $s(X,y)$ are

Scheme	Verifier Costs			Proof Size
	No. Pairings	No. Eqs	Asymp	
$(\mathsf{peP}, \mathsf{peV})$	8	4	$\mathscr{O}(1)$	$4\mathbb{G}_1$, $2\mathbb{G}_2$, $1\mathbb{Z}_p$
$(\mathsf{scP}, \mathsf{scV})$	4	2	$\mathscr{O}(n)$	$3\mathbb{G}_1$, $1\mathbb{G}_2$

Table 7.2: Efficiency costs for the different polynomial commitment schemes used by Sonic.

(Laurent) polynomials of degree d, and they evaluate to the same values at $2d+1$ different points, $f(X) = s(X, y)$. □

7.4.3 Efficiency

Table 7.2 gives the size of the proofs and the number of pairings a verifier needs to check for the polynomial commitment schemes given in this section. Here n is the degree of the polynomial that is being committed to. The prover costs for each of these schemes is $\mathscr{O}(n\log(n))$ assuming that the provers use multi-exponentiation techniques [112].

Observe that if many pairings are being computed then the verifier can batch the checks to reduce the total cost using techniques similar to those in [113]. To see this, note that

$$e(Cg^r, h) = e(W, h^{x-z}) \Leftrightarrow e(CW^z g^r, h) = e(W, h^x).$$

Thus, to check whether (C_0, W_0, r_0, z_0) and (C_1, W_1, r_1, z_1) both hold, the verifier picks $(\gamma_0, \gamma_1) \xleftarrow{\$} \mathbb{Z}_p$ and checks that

$$e((C_0 W_0^{z_0} g^{r_0})^{\gamma_0} (C_1 W_1^{z_1} g^{r_1})^{\gamma_1}, h) = e(W_0^{z_0} W_1^{z_1}, h^x).$$

Chapter 8

Conclusions and Future Work

In this thesis we have looked into mitigating two security issues for zero-knowledge arguments: trusted setup and malleability. Our constructions focus on maintaining efficiency in the amortised setting. We have designed: one SE-SNARK that requires trusted setup; one updatable and universal zk-SNARK with quadratic global parameters; and one updatable and universal zero-knowledge argument with linear global parameters and efficient "helped" verification. We show that updatable schemes are possible if the SRS contains only monomials, but not if it contains polynomials.

Compared to current zk-SNARKs and Bulletproofs, our updatable zk-SNARK has competitive proof sizes and prover computation. However, the quadratic global parameters are potentially unrealistic in practice. The storage requirements for circuits with 2^{17} gates is in the order of terabytes, running the updates scales linearly in the size of the SRS, and verifying the updates requires a linear number of pairings. Although there are supercomputers that potentially have the resources to run this computation, the implication is that the only parties that could verify the computations are those that also have large computational power at their disposal. Overall, it appears that to move this approach from theoretical to practical, one would need to find a means to reduce the size of the null space.

Sonic is realistic in practice provided that there is a party available to run the helper. Unlike our updatable zk-SNARK, Sonic has a linear sized reference string and the storage requirements for circuits with 2^{17} gates is in the order of megabytes. Verifying the updates requires a linear number of group exponentiations and a

constant number of pairings; verification could be ran from a personal computer. In blockchain settings the helper can be ran by miners and stakers, thus it is a natural alternative to zk-SNARKs: the proof sizes, prover computation and verifier computation in the batched setting are all competitive with the zk-SNARKs in use. To make Sonic attractive for other settings, one would need to look into methods for verifying the calculation of bivariate polynomials efficiently.

Our SE-SNARK has competitive proof sizes, SRS sizes, prover computation, and verifier computation, which all asymptotically match the most efficient zk-SNARKs in the literature. Unlike other zk-SNARKs it is simulation extractable. However, the SRS's are not updatable nor universal, therefore the trusted setup concerns are not mitigated.

Our trio of schemes are constructed over bilinear groups, but it would be interesting to see whether some of our techniques used might be transferable to other settings such as the lattice world. By allowing updatable SRSs rather than untrusted setup, we are able to amortise our verifier's costs, and it is possible that hidden monomial evaluations in other schemes might help improve efficiency. Furthermore, our method for employing a helper to aggregate proofs to reduce the verifier computation may be applicable to other schemes.

In defining simulation-extractability, we are strict about what the adversary is not allowed to do. We say that the scheme is only secure if any adversary that produces any unseen instance-proof pair must know a witness. Yet in the case of digital signatures, it is not always the case that malleating a proof is the fundamental security concern; sometimes one is only concerned if the adversary can change the message. It would be interesting to investigate how zk-SNARKs can be constructed with respect to an adversary that can change the proof but not the instance, and whether this would have consequences for receipt-freeness [11].

On the negative side, all our protocols depend on knowledge-of-exponent assumptions. Due to an impossibility result by Gentry and Wichs [114], NIZKs with sublinear proof sizes are not possible in the standard model, and this result is often used to justify the use of such strong assumptions. However, their result does

not rule out the possibility of succinct proofs in the random oracle model. While often slated for its unrealisability [115], the random oracle model (ROM) allows for practical and non-interactive schemes. Further, proofs of knowledge in the ROM show the existence of an extractor that computes a valid witness from any adversary as opposed to an adversary dependent extractor. We argue that this simple change in the order of quantifiers is preferable for security. In the ROM one provides an extractor in the security proof and soundness can be shown to be broken from the *existence* of an adversary against which the extractor does not succeed. In KEA protocols the extractor is assumed to exist and soundness can only be provably broken from the *non-existence* of an extractor for some adversary. We finish this thesis with the question:

Can zk-SNARKs in the random oracle model match the efficiency of zk-SNARKs built from knowledge assumptions?

Bibliography

[1] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304, 1985.

[2] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *Proceedings of the IEEE Symposium on Security & Privacy*, 2013.

[3] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 90–108, 2013.

[4] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a Von Neumann architecture. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 781–796, 2014.

[5] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 345–356, 2008.

[6] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 108–125, 2009.

[7] Christina Garman, Matthew Green, and Ian Miers. Decentralized anonymous credentials. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*, 2014.

[8] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: New definitions and delegatable anonymous credentials. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 199–213, 2014.

[9] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 262–288, 2015.

[10] David Bernhard, Georg Fuchsbauer, and Essam Ghadafi. Efficient signatures of knowledge and DAA in the standard model. In *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, pages 518–533, 2013.

[11] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. BeleniosRF: A non-interactive receipt-free electronic voting scheme. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1614–1625, 2016.

[12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Succinct malleable NIZKs and an application to compact shuffles. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 100–119, 2013.

[13] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 757–788, 2018.

[14] Jonathan Frankle, Sunoo Park, Daniel Shaar, Shafi Goldwasser, and Daniel J. Weitzner. Practical accountability of secret processes. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, pages 657–674, 2018.

[15] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *Proceedings of the IEEE Symposium on Security & Privacy*, 2014.

[16] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 839–858, 2016.

[17] Matthew Green and Ian Miers. Bolt: Anonymous payment channels for decentralized currencies. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 473–489, 2017.

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

[19] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology

ePrint Archive, Report 2017/1050, 2017. https://eprint.iacr.org/2017/1050.

[20] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *Proceedings of the IEEE Symposium on Security & Privacy*, 2018.

[21] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 581–612, 2017.

[22] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pages 698–728, 2018.

[23] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. *IACR Cryptology ePrint Archive*, 2019:99, 2019.

[24] Melissa Chase, Mary Maller, and Sarah Meiklejohn. Déjà Q all over again: Tighter and broader reductions of q-type assumptions. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 655–681, 2016.

[25] Melissa Chase and Sarah Meiklejohn. Déjà Q: using dual systems to revisit q-type assumptions. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 622–639, 2014.

[26] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Takeya Tango. Converting cryptographic schemes from symmetric to asymmetric bilinear groups. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 241–260, 2014.

[27] Sarah Azouvi, Mary Maller, and Sarah Meiklejohn. Egalitarian society or benevolent dictatorship: The state of cryptocurrency governance. In *22nd International Conference on Financial Cryptography and Data Security*, 2018.

[28] George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in Zcash. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, pages 463–477, 2018.

[29] Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, pages 595–626, 2018.

[30] Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. 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*, pages 336–365, 2017.

[31] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[32] Gilles Brassard, Claude Crépeau, and Moti Yung. Constant-round perfect zero-knowledge computationally convincing protocols. *Theor. Comput. Sci.*, 84(1):23–52, 1991.

[33] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[34] Martin Tompa and Heather Woll. Random self-reducibility and zero knowledge interactive proofs of possession of information. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 472–482, 1987.

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

[36] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *J. Cryptology*, 1(2):77–94, 1988.

[37] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 526–544, 1989.

[38] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 390–420, 1992.

[39] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *J. Cryptology*, 16(3):143–184, 2003.

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

[41] ZKProof Standards Workshop, 2018. https://zkproof.org/proceedings-snapshots/zkproof-implementation-20180801.pdf.

[42] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012.

[43] Jens Groth and Amit Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput.*, 41(5):1193–1232, 2012.

[44] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.

[45] Mihir Bellare, Markus Jakobsson, and Moti Yung. Round-optimal zero-knowledge arguments based on any one-way function. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 280–305, 1997.

[46] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986.

[47] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73, 1993.

[48] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 239–252, 1989.

[49] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Advances in Cryptology - CRYPTO 2005: 25th*

Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings, pages 152–168, 2005.

[50] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 669–699, 2018.

[51] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of ACM CCS*, 2017.

[52] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. `https://eprint.iacr.org/2018/046`.

[53] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 41–69, 2011.

[54] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[55] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for boolean circuits. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 1069–1083, 2016.

[56] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-

quantum zero-knowledge and signatures from symmetric-key primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1825–1842, 2017.

[57] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 433–442, 2008.

[58] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 31–60, 2016.

[59] Joe Kilian. Improved efficient arguments (preliminary version). In *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, pages 311–324, 1995.

[60] Ronald Cramer, Ivan Damgård, and Marcel Keller. On the amortized complexity of zero-knowledge protocols. *J. Cryptology*, 27(2):284–316, 2014.

[61] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT*, 2016.

[62] Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 926–943, 2018.

[63] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th*

Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008, pages 113–122, 2008.

[64] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 90–112, 2012.

[65] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. A zero-knowledge version of vSQL. *IACR Cryptology ePrint Archive*, 2017:1146, 2017.

[66] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 177–194. Springer, 2010.

[67] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 222–242, 2013.

[68] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, pages 41–60, 2013.

[69] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 315–333, 2013.

[70] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In *CRYPTO*, 2014.

[71] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In *ASIACRYPT*, pages 532–550, 2014.

[72] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*, pages 305–326, 2016.

[73] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT*, pages 626–645, 2013.

[74] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *Proceedings of the IEEE Symposium on Security & Privacy*, 2015.

[75] Howard Wu, Wenting Zheng, Alessandro Chiesa, Raluca Ada Popa, and Ion Stoica. DIZK: A distributed zero knowledge proof system. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, pages 675–692, 2018.

[76] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, pages 321–340, 2010.

[77] Leslie G. Valiant. Universal circuits (preliminary report). In *Proceedings of the 8th Annual ACM Symposium on Theory of Computing*, pages 196–203, 1976.

[78] Helger Lipmaa, Payman Mohassel, and Seyed Saeed Sadeghian. Valiant's universal circuit: Improvements, implementation, and applications. *IACR Cryptology ePrint Archive*, 2016:17, 2016.

[79] Pyrros Chaidos and Geoffroy Couteau. Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In *Advances in Cryptology*

- EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III, pages 193–221, 2018.

[80] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-SNARKs from square span programs. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 556–573, 2018.

[81] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: security in the face of parameter subversion. In *ASIACRYPT*, pages 777–804, 2016.

[82] Oded Goldreich, Rafail Ostrovsky, and Erez Petrank. Computational complexity and knowledge complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 1(7), 1994.

[83] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In *Proceedings of Asiacrypt 2017*, 2017.

[84] Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. Cryptology ePrint Archive, Report 2017/587, 2017.

[85] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. *J. Cryptology*, 27(3):506–543, 2014.

[86] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, pages 169–189, 2012.

[87] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 51–68, 2017.

[88] Sean Bowe, Ariel Gabizon, and Mathew Green. A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK. Cryptology ePrint Archive, Report 2017/602, 2017.

[89] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 390–399, 2006.

[90] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 543–553, 1999.

[91] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, pages 444–459, 2006.

[92] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 514–532, 2014.

[93] Masayuki Abe, Charanjit S. Jutla, Miyako Ohkubo, and Arnab Roy. Improved (almost) tightly-secure simulation-sound QA-NIZK with applications. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, pages 627–656, 2018.

[94] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 60–79, 2012.

[95] Sean Bowe and Ariel Gabizon. Making groth's zk-snark simulation extractable in the random oracle model. *IACR Cryptology ePrint Archive*, 2018:187, 2018.

[96] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT*, pages 409–426, 2006.

[97] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[98] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *J. Cryptology*, 30(4):989–1066, 2017.

[99] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, pages 445–456, 1991.

[100] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In *ASIACRYPT*, pages 48–62, 2004.

[101] Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In *TCC*, 2007.

[102] Jens Groth. Short non-interactive zero-knowledge proofs. In *ASIACRYPT*, pages 341–358, 2010.

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

[104] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 256–266, 1997.

[105] Ueli M. Maurer and Stefan Wolf. Lower bounds on generic algorithms in groups. In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, pages 72–84, 1998.

[106] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. *IACR Cryptology ePrint Archive*, 2005:15, 2005.

[107] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Luis Villar. An algebraic framework for Diffie-Hellman assumptions. *J. Cryptology*, 30(1):242–288, 2017.

[108] Essam Ghadafi and Jens Groth. Towards a classification of non-interactive computational assumptions in cyclic groups. 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 II*, pages 66–96, 2017.

[109] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 56–73, 2004.

[110] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed E-Cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 397–411, 2013.

[111] David Bernhard, Marc Fischlin, and Bogdan Warinschi. Adaptive proofs of knowledge in the random oracle model. *IET Information Security*, 10(6):319–331, 2016.

[112] Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM J. Comput.*, 9(2):230–250, 1980.

[113] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, pages 236–250, 1998.

[114] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.

[115] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.