Designing Efficient Zero-Knowledge Proofs in the Ideal Linear Commitment Model

Jonathan Bootle

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Computer Science Department

University College London

June 15, 2019

I, Jonathan Bootle, 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 are cryptographic protocols where a prover convinces a verifier that a statement is true, without revealing why it is true or leaking any of the prover's secret information. Since the introduction of zero-knowledge proofs, researchers have found numerous applications to other cryptographic schemes, such as electronic voting, group signatures, and verifiable computation. Zero-knowledge proofs have also become an integral part of blockchain-based cryptocurrencies.

Thus, designing efficient zero-knowledge proofs is an important goal. Recently, the design space has become extremely large. To simplify protocol design, designers have begun to separate the process into modular steps. Information theoretic protocols are designed in idealised communication models and compiled into real protocols secure under cryptographic assumptions.

In this thesis, we investigate the Ideal Linear Commitment model, which characterises interactive zero-knowledge protocols where the prover and verifier use homomorphic commitment schemes. We demonstrate the model's power by exhibiting efficient protocols for useful tasks including NP-Complete problems and other more specialised problems. We demonstrate the model's versatility by compiling the idealised protocols into real protocols under two completely different cryptographic assumptions; the discrete logarithm assumption, and the existence of collision-resistant hash functions.

We show that the Ideal Linear Commitment model is a useful and effective abstraction for producing zero-knowledge protocols. Furthermore, by identifying the limitations of the model and finding protocols outside these constraints, we display special techniques which result in more efficient zero-knowledge proofs than ever.

The results are novel and highly efficient protocols. Results include the first ever discrete-logarithm argument for general statements with logarithmic communication cost, the first ever three-move discrete-logarithm argument for arithmetic circuit satisfiability with sub-linear communication costs, and an argument for list membership with sub-logarithmic communication, less than the number of bits required to specify a list index. Every single one of our protocols improves the theoretical state-of-the-art.

Impact Statement

This thesis demonstrates how real cryptographic protocols for a variety of tasks can be designed using a special communication model. It shows that one can separate the algebraic machinery used to design the protocols from the cryptographic assumptions and commitment schemes used to prove that the protocols are secure, and still exhibit highly efficient protocols which improve asymptotically on the prior state-of-the-art. This is likely to benefit the discipline considerably as it lowers the barriers to understanding and producing secure protocols of this type. Cryptographic assumptions can be quite specialised, and it can be difficult to understand the mathematics behind them. Having a framework within which one can prove idealised protocols secure and knowing that the result can be made into a real protocol drastically simplifies the task of protocol designers. Several subtle changes to the Ideal Linear Commitment model were also introduced to make it more realistic and effective.

Furthermore, by showing that certain proofs all fit into a framework, it becomes easier to understand their limitations. Linear algebra is an important part of security proofs in the Ideal Linear Commitment model. Therefore, in future, zero-knowledge protocols can be analysed through the lens of linear algebra. Linear algebra is very well studied, and powerful techniques from this discipline may lead to strong results about zero-knowledge protocols, such as lower bounds on the communication complexity of certain types of interactive zero-knowledge protocol.

The protocols presented in this work led to the creation of the zero-knowledge argument Bulletproofs [1]. Bulletproofs has been implemented by cryptocurrencies including Monero and PIVX. PIVX plans to bring Bulletproofs implementations into common use later in 2018. Following a first successful code audit, Bulletproofs is also set to enter widespread use on Monero's blockchain later in 2018, subject to further successful audits. As a result, the author's work will soon have a sizeable impact on the efficiency of payment systems in the real world, which at the time of writing, amount to a market capitalisation value of roughly two billion dollars and a daily trade volume of roughly thirty four million dollars [1]. The addition of Bulletproofs will lead to much smaller amounts of proof data being stored on

the blockchains for these cryptocurrencies, which means better performance and functionality. This may have a measurable impact on cryptocurrency adoption and pricing.

Zero-knowledge proofs are becoming better known, not only among research scientists, but increasingly among companies and technology enthusiasts [2]. There are also ongoing standardisation efforts. The implementation of cryptographic protocols is a notoriously difficult task even for experts, and errors can have disastrous consequences. Making protocols easier to design and understand will be of great benefit to interested, non-expert parties who might try to use protocols in the future, promote their usage among wider user communities, or implement them in software or hardware.

[1]Data taken from `coinmarketcap.com`, on the 17th of September 2018.
[2]See `https://zkproof.org/`

Acknowledgements

During the four years of study which led to this thesis, I have been incredibly lucky to have the chance to interact with not only my colleagues from the information security group at UCL, but researchers from NTT Secure Platform Laboratory, Université Rennes 1, Microsoft Research Redmond, and IBM Research Zurich. Our discussions always served to open my mind and help me consider things in new ways.

Thanks to my coauthors, collaborators and friends Pyrros Chaidos, Christophe Petit, Essam Ghadafi, Mohammed Hajiabadi, Sune Jakobsen, Mary Maller, Mehdi Tibouchi, Keita Xagawa, Benedikt Bunz, Dan Boneh, Andrew Poelstra, Pieter Wuille, Greg Maxwell, Carsten Baum, Vadim Lyubashevsky, Rafael del Pino, Claire Delaplace, Thomas Espitau and Pierre-Alain Fouque.

Thanks to my supervisor Jens Groth who was a great mentor and provided me with a better model for how to be an academic researcher than I could ever have hoped for. Thanks to Andrea Cerulli who was always willing to help and discuss things, probably at the expense of his own work. Thanks to Vasilios Mavroudis and David Kohan Marzgao for providing wonderful distractions in the form of exciting mathematical puzzles. Thanks to my mother, whose motivation when I was a child made all of this possible.

Contents

1 Introductory Material **16**

1.1 Introduction 16

1.2 Contributions 19

1.3 Published Work 28

1.4 Recipes 37

2 Background and Related Work **39**

3 Formal Definitions **54**

3.1 Notation and Computational Model 54

3.2 Arithmetic Circuits 55

3.2.1 Preprocessing Arithmetic Circuits for Arguments 56

3.2.2 Reduction of Circuit Satisfiability Problem to a Hadamard Matrix Product and Linear Constraints. 58

3.3 Commitment Schemes 59

3.4 Error-Correcting Codes 61

4 Zero-Knowledge Proofs and the Ideal Linear Commitment Model **62**

4.1 Zero-Knowledge Proofs 62

4.2 Introduction to the Ideal Linear Commitment Model 67

4.2.1 Definitions 69

4.2.2 Complexity Measures 71

4.2.3 Public Coin Protocols 72

4.2.4 Remarks on the Interaction Model 73

4.2.5 Comparison with Other Models and Types of Protocol . . . 75

5 Cryptographic Assumptions and Concrete Commitment Schemes 78

5.1 The Discrete Logarithm Assumption 78

5.2 The Collision Problem for Hash Functions 81

6 Lemmata for Security Proofs 84

6.1 Variant of the Schwarz-Zippel Lemma 84

6.2 A general forking lemma. 86

6.3 Extractable Algebraic Queries 90

7 Simple Protocols through the ILC Lens 94

7.1 Basic Proof-of-Knowledge 94

7.2 Proof-of-Knowledge of a Committed Bit 97

7.2.1 Query Matrices 101

8 Generic ILC Protocols 105

8.1 Remarks 105

8.2 Polynomial Commitment Sub-Protocol 106

8.2.1 Definitions 107

8.2.2 Idea 109

8.2.3 Protocol 111

8.2.4 Security Proof 112

8.2.5 Efficiency 113

8.3 3-Move Low Depth Circuit Protocol 113

8.3.1 Intuition behind Protocol 115

8.3.2 Communication 120

8.3.3 Computation 121

8.3.4 Applications 121

8.4 3-Move Square-Root Protocol 126

8.4.1 Idea 128

8.4.2 Efficiency 133

8.5 5-Move Square-Root Protocol 134
8.5.1 Idea 136
8.5.2 Protocol Description 138
8.5.3 Security Analysis. 140
8.5.4 Efficiency 142

9 Compiling Ideal Linear Commitment Protocols into Standard Zero-Knowledge Proofs 143
9.1 Compiling into Hash-Based Proofs 143
9.1.1 Reed-Solomon Codes 146
9.1.2 Construction 147
9.1.3 Security Analysis 148
9.1.4 Efficiency 157
9.1.5 Optimisations 158
9.2 Compiling into Discrete-Logarithm-Based Proofs 160
9.2.1 Construction 161
9.2.2 Security Analysis 161
9.2.3 Efficiency 164

10 Optimisations outside the ILC Model 169
10.1 Recursive Argument for Inner Product Evaluation 169
10.1.1 Main Idea 170
10.1.2 Formal description 171
10.1.3 Efficiency. 174
10.1.4 How to use this Argument 175
10.1.5 Generalisations 175
10.2 Field Extension Techniques 177
10.2.1 Reduction of Circuit Satisfiability to a Hadamard Matrix Product and Linear Constraints over $GF(p^{2k})$. 179
10.2.2 How to Apply This Idea 181

11 Conclusion 182

Appendices **185**

A Security Proofs for Optimised Hash and Error-Correcting-Code Compilation **185**

B Stand-alone Argument for Arithmetic Circuit Satisfiability with Logarithmic Communication Complexity **189**

B.0.1 Security Analysis. 192

B.0.2 Efficiency 194

C Exposition of an Efficient Multi-Exponentiation Algorithm **196**

C.1 Goal 196

C.2 Reduction to Multi-Products 196

C.2.1 Visualisation 197

C.2.2 Efficiency 198

C.3 Computing the Multi-Products 199

C.3.1 Visualisation 199

C.3.2 Efficiency 199

C.4 Recombining Inputs 200

C.5 Efficiency Analysis 200

C.5.1 Generalisation 200

D Computing Sums from [2] and [3] for Low-Depth Circuit Protocols **201**

chapters/Bibliography **203**

List of Figures

1.1 Different ways to combine the results in this thesis. 38

3.1 A simple arithmetic circuit, and the corresponding consistency equations. The first sub-circuit contains the wires $a_1, b_1, c_1, a_2, b_2, c_2, a_3, a_3, c_3$. The second sub-circuit contains the wires $c_1, a_4, c_2, b_4, c_4, c_5, c_6$. The third sub-circuit B contains the wires c_3, c_4, a_5, b_5, a_6. 58

4.1 Description of the ILC interaction environment. The $\mathbf{v}_i$ here are vectors over a field $\mathbb{F}$, x is a value from $\mathbb{F}$ and V, V' and Q are matrices over $\mathbb{F}$. 68

4.2 Description of how the parties in an ILC protocol interact. 71

4.3 Description of how the parties interact in an ILC protocol with several vector lengths. 74

5.1 Pedersen Commitments 80

5.2 Hash-Based Commitments 83

9.1 Vectors $\mathbf{v}_\tau$ organized in matrix V are encoded row-wise as matrix $E = \tilde{\mathsf{E}}_{\mathscr{C}}(V;R)$. The vertical line in the right matrix and vector denotes concatenation of matrices respectively vectors. The prover commits to each column of E. When the prover, given $\mathbf{q}$ and $\mathbf{q}'$, wants to reveal the linear combination $\mathbf{v}_{(\mathbf{q})} = \mathbf{q}V$ she also reveals $\mathbf{r}_{(\mathbf{q})} = \mathbf{q}R$ and $\mathbf{r}'_{(\mathbf{q})} = \mathbf{q}'R$. The verifier now asks for openings of 2λ columns $J = \{j_1,\dots,j_{2\lambda}\}$ in E and verifies for these columns that $\mathbf{q}E|_J = \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\mathbf{q})};\mathbf{r}_{(\mathbf{q})})|_J$ and $\mathbf{q}'E|_J = \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}'_{(\mathbf{q})};\mathbf{r}'_{(\mathbf{q})})|_J$, where the verifier computed $\mathbf{v}'_{(\mathbf{q})}$ by themselves. To avoid revealing any information about $\mathsf{E}_{\mathscr{C}}(V)$, we must ensure that $\forall j \in [n] : j \in J \Rightarrow j+n \notin J$. If the spot checks pass, the verifier believes that $\mathbf{v}_{(\mathbf{q})} = \mathbf{q}V$ and $\mathbf{v}'_{(\mathbf{q})} = \mathbf{q}'V$. 146

9.2 Construction of $(\mathscr{K},\mathscr{P},\mathscr{V})$ from $(\mathscr{K}_{\mathsf{ILC}},\mathscr{P}_{\mathsf{ILC}},\mathscr{V}_{\mathsf{ILC}})$, commitment scheme $(\mathsf{Setup},\mathsf{Commit})$ and error-correcting code $\mathscr{C}$. 165

9.3 Efficiency of a compiled proof of knowledge $(\mathscr{K},\mathscr{P},\mathscr{V})$ for $(\sigma,u,w) \in \mathscr{R}$. Communication is measured in field elements and computation in field operations. 166

9.4 Efficiency of a compiled proof of knowledge $(\mathscr{K},\mathscr{P},\mathscr{V})$ for $(\sigma,u,w) \in \mathscr{R}$ with optimisations made to the compiler. Communication is measured in field elements and computation in field operations. 166

9.5 Construction of $(\mathscr{K},\mathscr{P},\mathscr{V})$ from $(\mathscr{K}_{\mathsf{ILC}},\mathscr{P}_{\mathsf{ILC}},\mathscr{V}_{\mathsf{ILC}})$, and homomorphic commitment scheme $(\mathsf{Setup},\mathsf{Commit})$. 167

9.6 Efficiency of a compiled proof of knowledge $(\mathscr{K},\mathscr{P},\mathscr{V})$ for $(\sigma,u,w) \in \mathscr{R}$ based on homomorphic commitments. Communication is measured in field elements and computation in field operations. Here κ is the cost of computing a group exponentiation in the commitment space. 168

List of Tables

1.1 Efficiency comparisons when our low-depth circuit protocol is instantiated with hash functions. Here N is the number of items in the list for a membership proof, or the degree of the polynomial for a polynomial evaluation proof, or the width of the range in a range proof. Then t indicates the number of statements proved at the same time in a batch proof. 23

1.2 Efficiency comparisons when our low-depth circuit protocol is instantiated in the discrete logarithm setting. Here N is the number of items in the list for a membership proof. Then t indicates the number of statements proved at the same time in a batch proof. 24

1.3 Efficiency comparisons when our low-depth circuit protocol is instantiated in the discrete logarithm setting. Here N is the degree of the polynomial for a polynomial evaluation proof, or the width of the range in a range proof. Then t indicates the number of statements proved at the same time in a batch proof. 25

1.4 Efficiency comparison between our 5-move argument in the discrete logarithm setting and the most efficient constant-move interactive zero-knowledge arguments relying on discrete logarithms and hash-functions. We express communication in terms of numbers of group elements $\mathbb{G}$ and field elements $\mathbb{Z}_p$. We express computational costs in terms of numbers of exponentiations over $\mathbb{G}$ and multiplications over $\mathbb{Z}_p$. The efficiency displayed is for a circuit with N multiplication gates. 27

1.5 Efficiency of our hash-based arguments for arithmetic circuits. The efficiency displayed is for a circuit with N multiplication gates. . . . 28

Chapter 1

Introductory Material

1.1 Introduction

A zero-knowledge proof [4] is a protocol between two parties: a prover and a verifier. The prover may want to convince the verifier that an instance u belongs to a specific language $\mathscr{L}$ in NP. She wants to convince the verifier that the instance $u \in \mathscr{L}$ is a true statement without revealing any confidential information. Naturally, the protocol should not leak any secret information which is unknown to the verifier and could make the statement easier to prove.

Zero-knowledge proofs are widely used in cryptography since it is often useful to verify that a party is following a protocol without requiring her to divulge secret keys or other private information. Applications range from digital signatures and public-key encryption to secure multi-party computation schemes with strong security guarantees, anonymous credentials, and verifiable cloud computing.

More formally, a zero-knowledge proof consists of a triple of algorithms $(\mathscr{G}, \mathscr{P}, \mathscr{V})$. These are the prover $\mathscr{P}$ and the verifier $\mathscr{V}$, and they engage in the protocol. The common-reference-string generator $\mathscr{G}$ produces the necessary setup information for $\mathscr{P}$ and $\mathscr{V}$ to run the protocol. In this thesis, the prover and verifier are interactive algorithms. Based on the setup information produced by the generator, the prover and the verifier exchange messages. At the end of the protocol, the verifier decides whether the proof was convincing or not, and accepts or rejects the proof.

Zero-knowledge proofs satisfy three basic requirements.

Completeness. When the statement is true, the prover always succeeds in convincing the verifier. In other words, when $u \in \mathscr{L}$, and the prover has a valid witness w, the verifier will always accept at the end of the protocol. Completeness can be viewed as more of a functionality requirement than a security requirement, and this property guarantees that the protocol works properly when the prover and verifier are honest.

Soundness. When the statement is false, or the prover does not know a valid witness, the prover can practically never convince the verifier, and the verifier will almost always reject the proof. Viewed another way, if the verifier accepted the proof, then $u \in \mathscr{L}$, and the prover has a valid witness w. Soundness can be seen as a converse to completeness. It is a security requirement which protects the verifier from being fooled into accepting by a malicious prover. As such, it can be seen as a property of the verification algorithm.

Zero-Knowledge. Despite taking part in the protocol with the prover, the verifier can never learn anything from the interaction except that the statement is true. This is modelled by showing that the entire interaction between the prover and the verifier can be simulated without any knowledge of the witness. If convincing-looking proofs can be simulated without any secret information, then it follows that nothing secret can be learned from real proofs. Zero-knowledge is a security requirement which protects the prover against a malicious verifier who is trying to learn the prover's secrets. As such, it can be seen as a property of the proving algorithm.

Many efficient zero-knowledge proofs are based on the discrete logarithm assumption [5, 6, 7, 8, 9, 1, 2, 10, 3, 11]. Although efficiency has improved over time, it looks as though all of these protocols draw on the same small collection of techniques, some used for improving efficiency, and some for proving security.

For instance, all of the protocols above use commitment schemes. These allow either party to 'commit' to a message, so that the message stays hidden but is fixed relative to the commitment, and can be revealed later. A party can commit to a message m by applying a commitment algorithm with m as input, and obtain c, a commitment to m. The commitment c can then be sent to other parties. Later, the party can 'open' c by revealing m. The other parties may then check that c

corresponds to m.

Commitment schemes satisfy two basic requirements.

Hiding. Like a sealed envelope, a commitment should hide the message inside. The commitment c should not leak any information about m.

Binding. Once an envelope is sealed, it is not possible to change the message inside without breaking open the envelope. Given a commitment c to a message m, it should be impossible to open c to a different message.

The protocols all use homomorphic commitments, where one can combine two commitments together to obtain a commitment to the sum of the messages.

These protocols also follow a similar pattern of interaction between the prover and the verifier. The prover sends an initial message a_1 to the verifier. The verifier responds with a random challenge value e_1. The prover sends another message a_2 to the verifier, who responds with another random challenge e_2. This process continues for n rounds, until messages a_n and e_n have been exchanged. The prover sends a final message a_{n+1} to the verifier, and then the verifier decides whether to accept or reject.

In fact, there are even more similarities to be observed. For many protocols, the prover's messages a_1, a_2 up to a_n are actually single or possibly multiple commitments to secret values obtained by the prover, computed using the witness or the random challenge values that the prover has seen up to that point. The prover's final message a_{n+1} is often a specially chosen sum or linear combination of the secret values that the prover committed to earlier. Then, the verification algorithm involves checking that a_{n+1} really is the correct sum or linear combination of the prover's secret values, using the commitments to achieve this.

In their security proofs, the protocols all seem to use the same techniques, based on polynomial algebra, polynomial identity testing and linear algebra. In fact, on the whole, it does not seem important that the protocols are based on the discrete logarithm assumption, just that they use homomorphic commitment schemes to commit to the elements of finite fields.

One can ask whether it is possible to find the techniques common to all interactive zero-knowledge protocols based on the discrete logarithm assumption, and place them into a single framework. This would drastically simplify the tasks of designing protocols, and proving mathematically that they are secure. Having fully understood or implemented one protocol of this type, it would be much easier to understand or implement any other. In this thesis, we will explore the question of whether there is an information-theoretic framework which provides a good abstraction for discrete-logarithm based interactive zero-knowledge protocols.

However, the framework is useless unless it can be used to create efficient protocols for a wide range of tasks. Thus, we can extend our question in several directions, and propose the following hypothesis for investigation.

There is an information-theoretic framework which provides a good abstraction for discrete-logarithm based interactive zero-knowledge protocols, and further;

1. The efficiency of information-theoretic protocols in the framework accurately models the efficiency of real protocols.
2. There are efficient protocols in the framework for many different languages.
3. The framework is useful for designing protocols outside the discrete-logarithm setting.

1.2 Contributions

As a building block in our arguments, we present an adaptation of the polynomial commitment sub-protocol appearing in [11], which allows the prover to commit to a polynomial so that the verifier can learn an evaluation of the polynomial in a secure manner. The sub-protocol has square-root communication complexity in the degree of the polynomial.

Zero-Knowledge Proofs for Low-Depth Circuits While very efficient, arguments for general statements, like arithmetic circuit satisfiability, often make use of generic reductions and complex machinery, and fail to be as efficient as arguments specialised for a particular language. We give a zero-knowledge proof for low-depth circuits. In doing so, we bridge the gap between general and simple languages in three ways.

Firstly, we provide a framework to describe the types of languages commonly encountered. Protocols such as the 1-out-of-N membership argument of [2], and the polynomial evaluation argument of [3] prove membership in languages where the witnesses are zeroes of low-degree polynomial relations. In other words, the statement is an arithmetic circuit of low degree, and part of the witness is a satisfying assignment for the circuit. We give a general relation which allows us to recover specific protocols by instantiating with concrete polynomial relations. By separating the task of developing more efficient ways to perform the zero knowledge proof, and the task of designing better relations to describe a given language, we can explain the logic behind past optimisations of membership proofs in [2, 11], and produce new optimisations for membership proofs and polynomial evaluation proofs.

Secondly, we unify the approaches used in [2, 3, 10] to construct zero-knowledge proofs for membership and polynomial evaluation, which can all be viewed as employing the same construction method. The constructions of zero-knowledge arguments for low degree polynomial relations in these works proceed by masking an input variable u as $f_u = ux + u_b$, using a random challenge x and a random blinder u_b. During the proof, the polynomial or circuit from the statement is computed with f_u in place of u, so that the original relation appears in the leading x coefficient. The communication and computational complexity of the resulting arguments is determined by the degree of the polynomial relation and the number of inputs. By contrast, the complexity of general arithmetic circuit protocols is determined by the number of gates. In the case of [11], the authors embed a polynomial evaluation argument for a polynomial of degree N into a low degree polynomial with $\log N$ inputs and degree $\log N$, obtaining a protocol with $O(\log N)$ communication using 3 moves, and requiring $O(\log N)$ operations to form cryptographic commitments. On the other hand, a polynomial of degree N requires N multiplication gates to evaluate in general, so the best arithmetic circuit protocol [1] can only achieve $O(\log N)$ communication in $O(\log N)$ moves, and uses $O(N)$ operations to form cryptographic commitments. In particular, in some settings, like the discrete logarithm setting, forming cryptographic commitments is based on computing exponentiations in a group of prime order. The cost of computing exponentiations is usually much higher than that of computing finite-field multiplications. Computing $O(\log N)$ group exponentiations rather than $O(N)$ leads to a significant performance advantage when considering implementation on constrained devices.

Bayer [12] gives two efficient batch proofs for multiplication and polynomial evaluation, which achieve a square-root communication overhead in the number of proofs to be batched. The key to achieving square-root overhead in [12] is to use Lagrange interpolation to embed many instances of the same relation into a single field element. This technique can be applied more generally to produce efficient batch proofs for the low-degree relations described above. Furthermore, by combining this with the polynomial commitment subprotocol in section 8.2, we improve the communication cost of the batched proof from $\sqrt{t}c$ to $\sqrt{tc}$, where c is the communication cost of the original non-batched proof, and t is a large number representing the number of proofs to be batched together.

Thirdly, we exhibit a general protocol in our framework and show how to recover protocols of previous works with some optimisation. More specifically, we give new 1-out-of-N membership arguments and polynomial evaluation arguments. Our new instantiations simultaneously decrease communication costs and reduce prover and verifier computation, while retaining the conceptual clarity and simple 3-move structure of the originals. As an example, we obtain the most communication efficient Σ-protocols for membership or non-membership of a committed value in a public list, in the discrete logarithm setting. We also include an argument for range proofs. Our argument captures the folklore method for performing range proofs, where the prover commits to an integer and the bits of the integer and then proves in zero-knowledge that each committed bit is indeed a bit, and that the bit decomposition corresponds to the integer. This demonstrates the expressivity of our general relation. Incorporating the batching ideas already mentioned, this gives an efficient batch protocol for proving and verifying t instances of the same relation simultaneously.

One notable place where we improve communication efficiency over previous proofs is in our membership and polynomial evaluation proofs instantiated in the discrete logarithm setting, which use a constant number of group elements, but have better communication efficiency regardless of whether the proofs are instantiated in elliptic curve groups or multiplicative subgroups of finite fields. Another is the polynomial evaluation argument which communicates $O(\frac{\log N}{\log\log N})$ commitments and field elements, which is an asymptotic improvement over the previous state-of-the-art, $O(\log N)$ commitments and field elements. Finally, our batch polynomial evaluation argument improves on [12] as the cost is proportional to $\sqrt{\log N}$ rather than $\log N$.

See Tables 1.2 and 1.3 for our results in the discrete logarithm setting. N is the instance-size, t is the number of batched instances, $\mathbb{G}$ means the number of group elements transmitted, $\mathbb{Z}_p$ means the number of field elements transmitted, $(\mathbb{G}, exp)$ means the number of group exponentiations and $(\mathbb{Z}_p, \times)$ means the number of field multiplications. In the membership proofs, N is the number of items in the list for which we prove membership. In the polynomial evaluation proofs, N is the degree of the polynomial. In the range proofs, N is the width of the range that we consider.

See Table 1.1 for our results based on hash-functions.

Zero-Knowledge Proofs for Arithmetic Circuits. An arithmetic circuit is a circuit that consists of addition and multiplication gates over a finite field $\mathbb{F}$, whose wires take values from the field. Arithmetic circuits are an attractive target for zero-knowledge protocol design, where the goal is to build an efficient argument system to prove that an arithmetic circuit is satisfiable, meaning that there exist a set of input values to the circuit that produce the specified output value when the circuit is evaluated. The statement is the arithmetic circuit and some specified values for the circuit outputs. The prover's witness is a collection of input values for the arithmetic circuit which give the correct output. There are several reasons why this is useful.

- Given an arithmetic circuit and outputs, the problem of deciding whether there exist input wire values satisfying the circuit is NP-Complete. Therefore, if we

[1]We compare against the efficiency when [2] is instantiated using Pedersen commitments, and the prover and verifier know the openings of the list of commitments.

[2]We compare against the efficiency when [10] is instantiated using Pedersen commitments rather than Elgamal ciphertexts.

Protocol	Reference	Communication		Prover Computation		Verifier Computation	
		Hashes	$\mathbb{Z}_p$	Hashes	$(\mathbb{Z}_p, \times)$	Hashes	$(\mathbb{Z}_p, \times)$
Membership Proof	This Work, 8.3.4.1	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(N \log N)$	$O(\lambda)$	$O(N)$
Batch Membership Proof	This Work, 8.3.4.1	$O(\sqrt{t} \log N)$	$O(\sqrt{t} \log N)$	$O(t \log tN)$	$O(tN \log tN)$	$O(\lambda)$	$O(tN)$
Polynomial Evaluation	This Work, 8.3.4.2	$O(\frac{\log N}{\log \log N})$	$O(\frac{\log N}{\log \log N})$	$O(\frac{\log N}{\log \log N})$	$O(N \log N)$	$O(\lambda)$	$O(N)$
Batch Polynomial Evaluation	This Work, 8.3.4.2	$O(\sqrt{t} \log N)$	$O(\sqrt{t} \log N)$	$O(t \log tN)$	$O(tN \log tN)$	$O(\lambda)$	$O(tN)$
Range Proof	This Work, 8.3.4.3	$O(\frac{\log N}{\log \log N})$	$O(\frac{\log N}{\log \log N})$	$O(\log N)$	$O(\log N)$	$O(\lambda)$	$O(\log N)$
Batch Range Proof	This Work, 8.3.4.3	$O(\sqrt{t} \log N)$	$O(\sqrt{t} \log N)$	$O(t \log N)$	$O(t \log N)$	$O(\lambda)$	$O(t \log N)$

Table 1.1: Efficiency comparisons when our low-depth circuit protocol is instantiated with hash functions. Here N is the number of items in the list for a membership proof, or the degree of the polynomial for a polynomial evaluation proof, or the width of the range in a range proof. Then t indicates the number of statements proved at the same time in a batch proof.

Protocol	Reference	Communication		Prover Computation		Verifier Computation	
		$\mathbb{G}$	$\mathbb{Z}_p$	$(\mathbb{G}, exp)$	$(\mathbb{Z}_p, \times)$	$(\mathbb{G}, exp)$	$(\mathbb{Z}_p, \times)$
Membership Proof	[9]	$4\log N + 8$	$2\log N + 7$	$12N$	$O(N)$	$4N$	$O(N)$
Membership Proof [1]	[2]	$4\log N$	$3\log N + 1$	$O(\log N)$	$O(N \log N)$	$O(\log N)$	$O(N)$
Membership Proof [2]	[10]	$\log N + 12$	$\frac{3}{2}\log N + 6$	$O(\log N)$	$O(N \log N)$	$O(\log N)$	$O(N)$
Membership Proof	This Work, 8.3.4.1	7	$4\log N + 4$	$O(\frac{\log N}{\log\log N})$	$O(N \log N)$	$O(\frac{\log N}{\log\log N})$	$O(N)$
Membership Proof	This Work, 8.3.4.1	$2.7\sqrt{\log N}$ $+5$	$1.9\log N+$ $2.7\sqrt{\log N} + 4$	$O(\frac{\log N}{\log\log N})$	$O(N \log N)$	$O(\frac{\log N}{\log\log N})$	$O(N)$
Batch Membership Proof	This Work, 8.3.4.1	$4.1\sqrt{t \log N}$	$4.1\sqrt{t \log N}$	$O(t \log tN)$	$O(tN \log tN)$	$O(\sqrt{t} \log tN)$	$O(tN)$

Table 1.2: Efficiency comparisons when our low-depth circuit protocol is instantiated in the discrete logarithm setting. Here N is the number of items in the list for a membership proof. Then t indicates the number of statements proved at the same time in a batch proof.

Protocol	Reference	Communication		Prover Computation		Verifier Computation	
		$\mathbb{G}$	$\mathbb{Z}_p$	$(\mathbb{G}, exp)$	$(\mathbb{Z}_p, \times)$	$(\mathbb{G}, exp)$	$(\mathbb{Z}_p, \times)$
Polynomial Evaluation	[9]	$4\log N + 8$	$2\log N + 7$	$12N$	$O(N)$	$4N$	$O(N)$
Polynomial Evaluation	[3]	$4\log N + 2$	$3\log N + 3$	$O(\log N)$	$O(N\log N)$	$O(\log N)$	$O(N)$
Polynomial Evaluation	This Work, 8.3.4.2	7	$3\log N + 4$	$O(\frac{\log N}{\log\log N})$	$O(N\log N)$	$O(\frac{\log N}{\log\log N})$	$O(N)$
Polynomial Evaluation	This Work, 8.3.4.2	$O(\frac{\log N}{\log\log N})$	$O(\frac{\log N}{\log\log N})$	$O(\frac{\log N}{\log\log N})$	$O(N\log N)$	$O(\frac{\log N}{\log\log N})$	$O(N)$
Batch Polynomial Evaluation	[12]	$O(\sqrt{t}\log N)$	$O(\sqrt{t}\log N)$	$O(t\log N)$	$O(tN\log N)$	$O(\sqrt{t}\log N)$	$O(tN)$
Batch Polynomial Evaluation	This Work, 8.3.4.2	$2.8\sqrt{t\log N}$	$2.8\sqrt{t\log N}$	$O(t\log tN)$	$O(tN\log tN)$	$O(\sqrt{t}\log tN)$	$O(tN)$
Range Proof	This Work, 8.3.4.3	7	$3\log N + 4$	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$
Range Proof	This Work, 8.3.4.3	$O(\frac{\log N}{\log\log N})$	$O(\frac{\log N}{\log\log N})$	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$
Batch Range Proof	This Work, 8.3.4.3	$2.8\sqrt{t\log N}$	$2.8\sqrt{t\log N}$	$O(t\log N)$	$O(t\log N)$	$O(t\log N)$	$O(t\log N)$

Table 1.3: Efficiency comparisons when our low-depth circuit protocol is instantiated in the discrete logarithm setting. Here N is the degree of the polynomial for a polynomial evaluation proof, or the width of the range in a range proof. Then t indicates the number of statements proved at the same time in a batch proof.

can design zero-knowledge proofs for arithmetic circuit satisfiability, then this implies that we can give zero-knowledge proofs for all NP languages.

- Many cryptographic systems can be expressed in terms of arithmetic over finite fields of prime order. Given a zero-knowledge proof system for arithmetic circuit satisfiability, we can give zero-knowledge proofs which reason about other cryptographic schemes, often in order to provide stronger security guarantees.

- There exist compilers which take computer programs written in C [13, 14] (avoiding certain commands) and convert them into arithmetic circuits. Then, a zero-knowledge proof for arithmetic circuit satisfiability can become a zero-knowledge proof that the C program was executed correctly.

We provide two honest verifier zero-knowledge arguments for arithmetic circuit satisfiability. Results for discrete-logarithm arguments are given in Table 1.4 and results for hash-based arguments in Table 1.5.

In general, the arguments have a square-root communication complexity. The arguments work by reducing the problem of verifying arithmetic circuit satisfiability to the problem of checking that the prover knows that for three commitments, two correspond to committed vectors of values, and the third contains the scalar product of the two vectors.

3-Move Protocol for Arithmetic Circuit Satisfiability We give a 3-move protocol for arithmetic circuit satisfiability. The argument has sublinear communication costs and fewer rounds of interaction than any previously published arguments with sublinear communication in the discrete logarithm setting, and highlights some interesting subtleties in our communication model. When instantiated using Pedersen commitments, this gives the first arithmetic circuit satisfiability protocol with a square-root communication complexity in only three moves. Unfortunately, the argument has a large (superlinear) computational cost for both the prover and the verifier, so is presented mostly for theoretical interest.

We start from the circuit satisfiability argument of Groth [7], which requires

Reference	Moves	Communication		Prover Complexity		Verifier Complexity	
		$\mathbb{G}$	$\mathbb{Z}_p$	exp.	mult.	exp.	mult.
[6]	3	$6N$	$5N+2$	$6N$	$6N$	$6N$	0
[7]	7	$9\sqrt{N}+4$	$7\sqrt{N}+6$	$\frac{6N}{\log N}$	$O(N\log N)$	$\frac{39\sqrt{N}}{\log N}$	$O(N)$
[7]	$O(\log N)$	$2\sqrt{N}$	$7\sqrt{N}$	$\frac{6N}{\log N}$	$O(N)$	$\frac{18\sqrt{N}}{\log N}$	$O(N)$
[8]	5	$30\sqrt{N}$	$7\sqrt{N}$	$\frac{6N}{\log N}$	$O(N\log N)$	$\frac{77\sqrt{N}}{\log N}$	$O(N)$
[1]	$O(\log N)$	$2\log N$	$O(1)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$
This work	3	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\frac{N^{3/2}}{\log N})$	$O(N^2)$	$\frac{N}{\log N}$	$O(N^{3/2})$
This work	5	$2\sqrt{N}$	$2\sqrt{N}$	$\frac{6N}{\log N}$	$3N\log N$	$\frac{8\sqrt{3N}}{\log N}$	$O(N)$
This work	$O(\log N)$	$4\log N$	$2\log N$	$12N$	$O(N)$	$4N$	$O(N)$

Table 1.4: Efficiency comparison between our 5-move argument in the discrete logarithm setting and the most efficient constant-move interactive zero-knowledge arguments relying on discrete logarithms and hash-functions. We express communication in terms of numbers of group elements $\mathbb{G}$ and field elements $\mathbb{Z}_p$. We express computational costs in terms of numbers of exponentiations over $\mathbb{G}$ and multiplications over $\mathbb{Z}_p$. The efficiency displayed is for a circuit with N multiplication gates.

7 moves and has square root communication complexity in the *total* number of gates. In this argument the prover commits to all the wires using homomorphic multicommitments, verifies addition gates using the homomorphic properties, and uses a product argument to show that the multiplication gates are satisfied.

We first improve Groth's argument into a 5-move argument with square root communication complexity in the number of *multiplication gates* only. We achieve fewer moves compared to [7] by avoiding generic reductions to linear algebra statements. We remove the communication cost of the addition gates in the argument by providing a technique that can directly handle a set of Hadamard products and linear relations together.

Logarithmic Complexity Argument. In spite of all these improvements, the above argument still requires square root communication complexity with respect to multiplication gates. In the first move the prover commits to all circuit wires using $3m$ commitments to n elements each, where $mn = N$ is a bound on the number of multiplication gates, and in the last move after receiving a challenge he opens commitments that can be constructed from the previous ones and the challenge. The communication complexity is $O(m+n)$. This is minimised when $m,n = O(\sqrt{N})$,

Reference	Moves	Communication		Prover Complexity		Verifier Complexity	
		Hashes	$\mathbb{Z}_p$	Hashes	mult.	Hashes	mult.
[15]	$O(1)$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(N\log N)$	$O(\sqrt{N})$	$O(\sqrt{N})$
[16]	$O(\log\log N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(N\log N)$	$O(\lambda)$	$o(N)$
[17]	$O(1)$	$O(N)$	$O(N\log N)$	$O(N)$	$O(N\log N)$	$O(\log N)$	$O(\log N)$
This work	5	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(N\log N)$	$O(\lambda)$	$O(N)$

Table 1.5: Efficiency of our hash-based arguments for arithmetic circuits. The efficiency displayed is for a circuit with N multiplication gates.

which gives square-root communication complexity.

Our key idea to break this square root communication complexity barrier is to replace the last opening step in this protocol with a special protocol for scalar products. In Section 10.1 we provide an argument system for this problem, which only requires a logarithmic communication with respect to the vector sizes. The argument is built in a recursive way, reducing the size and complexity of the statement further in each recursion step. This uses a special property of commitments, namely homomorphic properties with respect to the keys. Pedersen commitments, based on the discrete logarithm assumption, satisfy this property. As a result, using this inner product argument as a subroutine in our main argument, and instantiating with Pedersen commitments, we obtain an arithmetic circuit satisfiability argument with logarithmic communication complexity based on the discrete logarithm assumption. This argument was the first of its kind. The constants in the complexities of the protocol have since been improved in [1].

When using a logarithmic number of moves and applying a reduction similar to [18], our scheme dramatically improves the communication costs with respect to all previous work without incurring any significant overhead. We note that [18] uses a similar reduction to reduce computation whereas we use it to reduce communication.

1.3 Published Work

In this section, we discuss the author's published works.

Group and Ring Signatures

- Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*, pages 243–265, 2015

This work proposed a new security model for a variant of ring-signatures called accountable ring signatures, and provided a construction of the cryptosystem. Ring signatures allow a single user to create a signature on behalf of a group of users, formed in an ad-hoc fashion. However, the original security model for ring signatures has no mechanism for revoking anonymity and tracing the origin of a signature in case a user misbehaves. Accountable ring signatures include a tracing mechanism, where the signer can designate an opener of their choice who can reveal the signer's identity in case of misbehaviour. The construction of accountable ring signatures given in the paper relies on a zero-knowledge proof that a committed value is a member of a list of values, which are provided in encrypted form. My contribution was to check that the construction, notation, security proof and efficiency calculations for the zero-knowledge proof were correct.

- Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, pages 117–136, 2016

This work proposed a new model for the security and functionality of group signatures, in the case where the group of users can be updated over time by adding new users and removing some users from the system. These were named 'Fully Dynamic Group Signatures', where 'dynamic' refers to the group of users. The paper shows that given any construction of a fully dynamic group signature, one can easily obtain constructions of group signatures in older security models, namely the partially dynamic group signatures of [20] and [21], and points to some subtle attacks that can arise in the other models since the attacks are not prohibited by the security definitions. In this paper, my contribution was to show that our model for fully dynamic group models was all-encompassing and that a construction of a fully dynamic group signature allowed one to easily build a construction of group signatures for the other security models, and prove that the constructions satisfied the appropriate definitions.

Lattice Cryptanalysis

- Jonathan Bootle, Mehdi Tibouchi, and Keita Xagawa. Cryptanalysis of Compact-LWE. In *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, pages 80–97, 2018

Compact-LWE [23] was a novel, lattice-based encryption scheme presented in [23]. It was proposed as a secure scheme for the post-quantum setting, as part of the recent NIST call for efficient post-quantum key-encapsulation mechanisms and signature schemes. The scheme was also based on a new assumption called Compact LWE, whose hardness was justified with a reduction to the more standard LWE problem, showing that solving the Compact LWE problem was at least as hard as solving the LWE problem. In our work, we showed that the encryption scheme was easily broken for the concrete parameters given in the original paper, since the secret key could always be recovered efficiently. Furthermore, we showed that the hardness reduction to the LWE problem was flawed, and gave an algorithm for solving the Compact-LWE problem which essentially showed that solving Compact-LWE was *no harder* than solving LWE. These arguments presented a strong case against the use of Compact-LWE. My own contribution to this paper was quite small, trying to find the best way to explain the details of the various attacks presented.

- Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi. LWE without modular reduction and improved side-channel attacks against BLISS. 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 494–524, 2018

BLISS [25] is an efficient, lattice-based signature scheme. However, previous work [26] shows that certain variable-time implementations of the signature's rejection sampling algorithm lead to side-channel attacks. Two quantities derived from the signature's secret key are leaked, related to the norm of the secret key and a noisy scalar product of the secret key with another public value. Previous work [26] demonstrates that for a small subset of weak secret keys, one can use the norm leakage to recover the secret at a high computational cost, and dismisses the scalar product leakage, as one would have to solve a problem akin to LWE in order to recover the secret. However, our work observes that the new LWE-like problem does not feature modular reduction, and so can be efficiently solved using linear regression. We measure the number of signatures and the time required to recover the secret key for different BLISS parameter settings. We also formalise the problem of LWE without modular reduction and give theoretical upper and lower bounds for the number of signatures required to solve the new problem, relating these to the BLISS parameter choices. In this paper, my contribution was to spot an idea from another source which used regression algorithms to solve a similar problem, implement the side-channel attack, and investigate the attack for different parameter settings.

Surveys

- Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, and Jens Groth. Efficient zero-knowledge proof systems. In *Foundations of Security Analysis and Design VIII - FOSAD 2014/2015/2016 Tutorial Lectures*, pages 1–31, 2016

This work was a tutorial on zero-knowledge proof systems for the International School on Foundations of Security Analysis and Design (FOSAD). It did not present any new techniques, but was split into three parts. The first was an explanation of the properties of zero-knowledge proofs. The second gave details on the design of some simple interactive zero-knowledge proofs, and the third section did the same for some basic non-interactive zero-knowledge protocols. I was responsible for writing the third section of the tutorial, where I explained simplified examples of techniques from the hidden-bits model featured in [28] and a proof [29] based on the Boneh-Goh-Nissim public-key encryption scheme [30], as well as giving some information on pairing-based SNARKs [31, 32, 33, 34, 35, 13, 36, 37].

Prover-Efficient Zero-Knowledge and Hash-based arguments

- 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

This work gave the first zero-knowledge proofs for arithmetic circuit satisfiability with sub-linear communication complexity, linear computational cost, or constant overhead, for the prover, and a slightly sub-linear verification cost. The argument works by introducing a new commitment scheme based on hash-functions and error-correcting codes, both of which are computable in linear time and make use of expander graphs. Then, a collection of techniques used in other works such as [7] are abstracted into a new idealised communication model called the Ideal Linear Commitment model (ILC). The paper presents a proof of arithmetic circuit satisfiability in the Ideal Linear Commitment model, and a compilation converting zero-knowledge proofs in the idealised model into real zero-knowledge proof with perfect zero-knowledge and soundness based on the existence of suitable collision-resistant hash-functions and error-correcting codes. My personal contribution to this paper was to design all of the ILC protocols for arithmetic circuit satisfiability, provide security proofs for them, and calculate their efficiency.

- Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology –*

ASIACRYPT 2018, pages 595–626, Cham, 2018. Springer International Publishing

This work considers a RAM machine specification called TinyRAM, and the problem of verifying, in zero-knowledge, that a given TinyRAM program was executed correctly. The authors solve the problem using the techniques from [16] and obtain zero-knowledge proofs with sub-linear communication complexity and close to constant computational overhead. The extra overhead arises due to computational costs associated with verifying the RAM machine model of computation that were not present with arithmetic circuits. The methodology is very similar to that of [16]; providing ILC protocols to verify the correctness of a RAM computation, and then using a compiler to produce real zero-knowledge proofs based on hash-functions and error-correcting codes. Again, my personal contribution to the paper was to design all of the extra ILC protocols required to verify correct program execution, as extra arguments, such as a verifiable shuffle, were required. I was also responsible for their security proofs and efficiency calculations.

Discrete-Logarithm-based arguments

- Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 327–357, 2016

In this work, the authors propose two new arguments for arithmetic circuit satisfiability, and sub-protocols for particular tasks, based on the discrete logarithm assumption. The first protocol is a 5-move interactive argument with a square-root communication complexity in the size of the arithmetic circuit, using a sub-protocol which commits to polynomials and then reveals the evaluation of the polynomial at a given point in a verifiable manner. Using a recursive sub-protocol with logarithmic communication and round complexity, which verifies that two values committed using Pedersen commitments have a given scalar product, the 5-move argument can be converted into a new arithmetic circuit argument with similar computation costs. My contribution to this paper was the formalisation of new security definitions required for the polynomial commitment sub-protocol and optimising that protocol, a description of how to pre-process an arithmetic circuit to convert it into the format required by the main zero-knowledge arguments, and notation and part of the proof of a generalised forking lemma used to prove the knowledge soundness of the logarithmic move arguments in the paper.

- Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334, 2018

The proof-system in this work is known as Bulletproofs. In this work, the authors optimise the logarithmic-communication argument of [9] to reduce communication costs by a factor of three. They also present a simplified argument for the special task of range proofs, which demonstrate that a committed value lies in a particular interval, and provide an implementation and concrete performance measurements for the new argument. They also give a secure multi-party computation protocol allowing various parties to compute their own zero-knowledge proofs in parallel and them aggregate them securely later on. Having discovered the techniques to cut communication costs by a factor of three in parallel with the rest of the other authors and joined the paper write-up at a later stage when almost complete, I was responsible for choosing the correct definitions of zero-knowledge proofs for the paper and helping to choose good notation for the arguments in the paper.

- Jonathan Bootle and Jens Groth. Efficient batch zero-knowledge arguments for low degree polynomials. In *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*, pages 561–588, 2018

In this work, the authors identify the techniques used to give zero-knowledge proofs in previous works such as [3, 2, 10], which are all statements encoded into low-depth circuits, or low-degree polynomials. They specify a relation-framework which encompasses all of the statements proved in those zero-knowledge proofs. They then give a zero-knowledge protocol for single instances of the relation, and a batched protocol which builds on techniques from [12]. They show that for particular choices of relation, one can obtain zero-knowledge membership proofs and polynomial evaluation arguments with better concrete and asymptotic efficiency than previously known, and capture folklore range-proofs based on the discrete logarithm assumption. My contributions to this paper were the security definitions for the polynomial commitment argument and optimisations to the argument itself, which are similar to my contributions in [9]. I also identified the method of generalising from arguments for single statement to batched arguments, and I discovered choices of relation within the framework which led to arguments with improved asymptotic properties.

Lattice-based arguments

- 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

In this work, the authors give zero-knowledge arguments for arithmetic circuit satisfiability based on cryptographic assumptions in lattices. The arguments have a constant number of moves, quasilinear computational complexity, and sub-linear communication complexity. In many respects, this argument is closely related to the square-root communication argument of [9], with modifications to reflect the change from discrete-logarithm groups to lattices. As a crucial step in the argument, the authors provide a zero-knowledge proof of knowledge of values committed using commitments based on the hardness of the Short-Integer-Solution problem. This proof-of-knowledge was a big improvement over prior work, as it proves that the prover knows openings to stated commitments, rather than some multiple of those commitments, which is a weaker security guarantee. Furthermore, this is the first lattice-based zero-knowledge argument for large and general statements which has a sub-linear communication complexity. My contributions in this paper were the adaptations of the 5-move argument from [9] to the new lattice-based setting, new security proofs for the protocol, and a novel technique boosting the soundness of the zero-knowledge protocol by simulating operations in finite field extensions over integer modules, building on work in [40] and [41].

Work in this Thesis This thesis focusses on contributions from the following papers.

1. [9], which contains the square-root and logarithmic communication arguments for arithmetic circuit satisfiability based on the discrete logarithm assumption. We use all of the arguments but the polynomial commitment sub-protocol from this paper.

2. [11], which contains the relation-framework for low-degree polynomials, and efficient batched protocols for low-degree polynomial relations based on the discrete logarithm assumption. We use all of the arguments from this paper.

3. [16], which defines the Ideal Linear Commitment model, and gives linear-time zero-knowledge protocols for arithmetic circuit satisfiability based on hash-functions and error-correcting codes. From this paper, we use the ILC model and compilation of ILC protocols into real protocols using hashes and error-correcting codes.

4. [39], which contains a square-root communication argument for arithmetic circuit satisfiability based on the Short Integer Solution problem. From this paper, we use the soundness-boosting techniques over finite field extensions.

We also include the following pieces of unpublished work.

1. A novel argument for arithmetic circuit satisfiability with three moves and a sub-linear communication complexity, which highlights issues with the ILC model as given in previous work.

2. Small modifications to the ILC model and compiler [16], for efficiency reasons.
3. A compiler from ILC protocols to real protocols based on the discrete logarithm assumption.

1.4 Recipes

The results in this thesis are fairly modular. That is, if one is only interested in a particular type of zero-knowledge argument, it is possible to restrict attention to particular parts of the thesis.

To construct a hash-based argument for arithmetic circuits, one can use either the three-move arithmetic circuit argument (Section 8.4) or the five-move arithmetic circuit argument (Section 8.5), with the compiler based on hash functions and error-correcting codes (Section 9.1). With the three-move argument, one can use the argument for small fields (Section) in order to boost the soundness of the resulting protocol.

To construct discrete-logarithm based arguments for arithmetic circuits, one can use the three-move arithmetic circuit argument (Section 8.4) or the five-move arithmetic circuit argument (Section 8.5), with the compiler based on Pedersen commitments (9.2). With the five-move argument, one can then apply the recursive argument for scalar products (Section 10.1) to obtain a protocol with logarithmic communication complexity.

To construct arguments for specialised languages, such as polynomial evaluation arguments, membership arguments, and range proofs, one can use the low-depth circuit argument (Section 8.3) with either compiler.

The combinations described above are summarised in Figure 1.1.

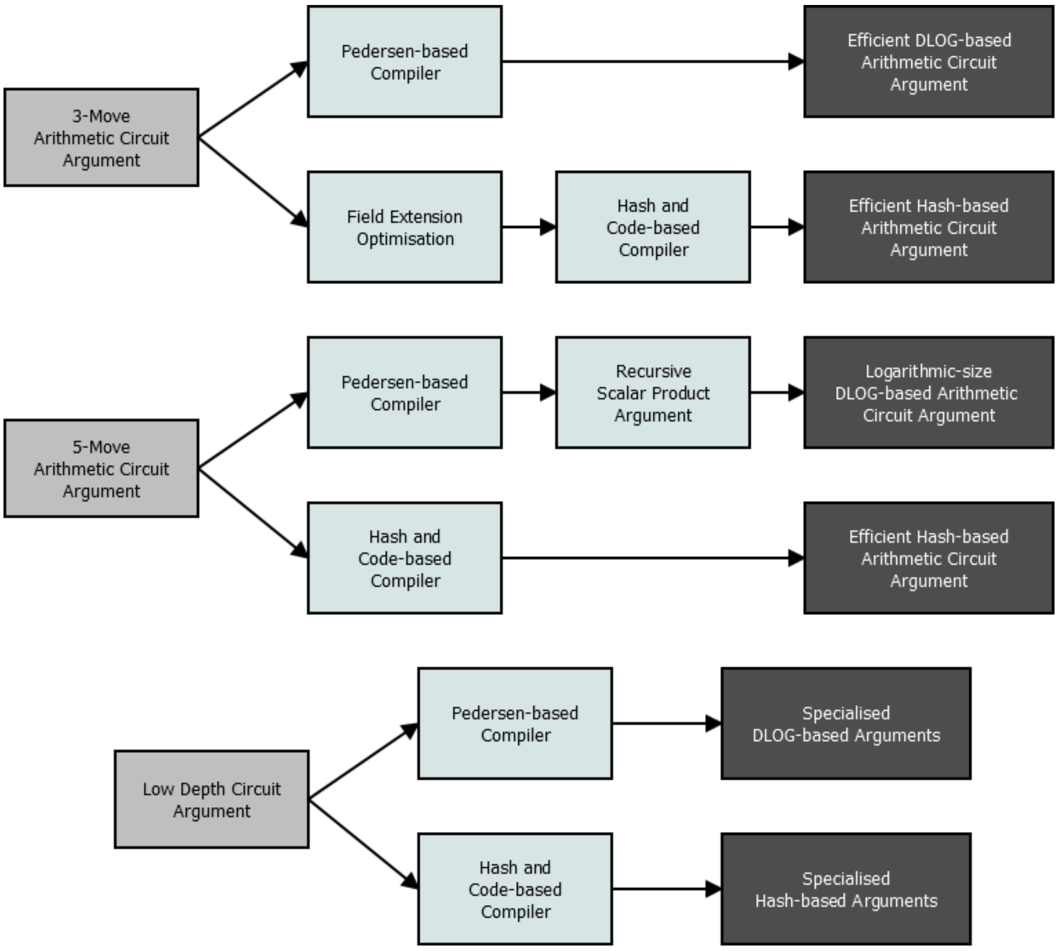

3-Move Arithmetic Circuit Argument
Pedersen-based Compiler
Efficient DLOG-based Arithmetic Circuit Argument
Field Extension Optimisation
Hash and Code-based Compiler
Efficient Hash-based Arithmetic Circuit Argument
5-Move Arithmetic Circuit Argument
Pedersen-based Compiler
Recursive Scalar Product Argument
Logarithmic-size DLOG-based Arithmetic Circuit Argument
Hash and Code-based Compiler
Efficient Hash-based Arithmetic Circuit Argument
Low Depth Circuit Argument
Pedersen-based Compiler
Specialised DLOG-based Arguments
Hash and Code-based Compiler
Specialised Hash-based Arguments

Figure 1.1: Different ways to combine the results in this thesis.

Chapter 2

Background and Related Work

Zero-knowledge proofs were invented by Goldwasser, Micali, and Rackoff [4]. In defining zero-knowledge proofs, the authors solved several important conceptual problems.

Firstly, the definition of Interactive Turing Machines put the concept of interactive protocols between two parties on a rigourous theoretical footing. This lead to the new computational complexity classes IP and ZK of languages which can be recognised by interactive proofs and zero-knowledge proofs, respectively. They gave a zero-knowledge proof for quadratic residuosity, the first zero-knowledge proof, showing that the class ZK was non-empty.

Secondly, they solved the problem of what it means for some party to know something. The knowledge of a party, or computing device, was captured by whatever it is possible for that party to compute, given the information available to it, and its own computational constraints.

Finally, the problem of what it means to gain no knowledge from an interaction was captured using a simulation-based definition. That is, if it is possible to efficiently simulate the contents of an interactive protocol without taking part in the protocol or knowing any secret information that the participants are privy to, then observing the interaction cannot confer any new knowledge. What can be computed from viewing the execution of the protocol is exactly the same as what can be computed without seeing it, and using a simulated execution instead.

Goldreich et al. [42] later showed that all languages in NP have zero-knowledge proofs, so that NP is contained inside ZK. Informally, this means that for any problem for which one can check the solution efficiently, one can also convince somebody else that you hold the solution, without giving away any information about the solution. In fact, more is true. By taking several results together, we know that there are zero-knowledge proofs for every language in IP [43, 44, 45], assuming the existence of one-way functions.

Feige et al. [46] introduced zero-knowledge proofs-of-knowledge. These are different to the original proposal of zero-knowledge proofs. Let us consider the difference for NP languages $\mathscr{L}$, with an instance u and a witness w. The original zero-knowledge proofs could be referred to as 'zero-knowledge proofs of membership' in this context. They prove that u lies in $\mathscr{L}$, without leaking any further information, such as w. So the verifier learns that some valid w exists, but this does not guarantee that the prover actually knows a witness. In a proof of knowledge, the verifier learns that the prover knows a valid w, and nothing more. These are useful for identification schemes, for example, where the prover might authenticate themselves by proving that the know the secret key corresponding to a particular public key.

We can classify zero-knowledge proofs according to the number of rounds of interaction that take place between the prover and the verifier. Non-interactive zero-knowledge proofs were introduced in [47]. In these proofs, the proof consists of a single message sent from the prover to the verifier, who then accepts it or rejects it. Non-interactive zero-knowledge proofs require a common reference string as input to the protocol, to be used by both the prover and the verifier. Without a common reference string, it is only possible to construct non-interactive zero-knowledge proofs for languages in the complexity class BPP, which are seen as trivial, as the verifier can efficiently decide whether an instance is in a BPP-language without receiving any help from the prover.

The soundness and zero-knowledge properties of zero-knowledge proofs usually come in three different flavours:

- Perfect; the property is always satisfied, even against computationally unbounded adversaries.

- Statistical; the property fails to be satisfied with at most negligible probability, even against computationally unbounded adversaries.

- Computational; the property fails to be satisfied with at most negligible probability, against computationally bounded adversaries.

So far, we have discussed zero-knowledge proofs, which have perfect or statistical soundness. However, these can only have computational zero-knowledge. Protocols with computational soundness and perfect or statistical zero-knowledge are called zero-knowledge *arguments*. Brassard et al. [48] showed that all languages in NP have zero-knowledge arguments with perfect zero-knowledge. Micali [49] introduced the related notion of CS (computationally-sound) proofs, where proofs for false statements exist, but are computationally difficult to find.

Gentry et al. [50] used fully homomorphic encryption to construct zero-knowledge proofs where the communication complexity corresponds to the size of the witness. For circuit satisfiability, for example, the scheme works by encrypting the witness using a symmetric key encryption scheme, and using fully homomorphic encryption to decrypt the witness and evaluate the circuit homomorphically in the witness while still in encrypted form. This result gives protocols with the lowest communication complexity that one might expect, as proofs cannot in general have communication that is smaller than the witness size unless surprising results about the complexity of solving SAT instances hold [51, 52].

Kilian [53] showed that in contrast to zero-knowledge proofs, zero-knowledge arguments can have very low communication complexity. His construction relied on the PCP theorem. Probabilistically checkable proofs, or PCPs, are proofs consisting of strings of many elements, whose correctness can be checked by examining only a small number of elements, sampled at random. Kilian's scheme has an excellent polylogarithmic communication complexity, but does not yield a practical scheme due to the large computational overhead required to convert statements into PCPs. In his scheme, the prover converts the statement to be proved into a PCP consisting of bits, commits to each of the bits using a single commitment, and then hashes all of the commitments in a Merkle tree. The verifier chooses a few bits of the PCP to verify. The prover reveals those commitments from the Merkle tree, and uses a simple, auxiliary zero-knowledge proof system to prove that the committed bits will pass the verifier's checks.

Ishai et al. [54] introduce commitment schemes with linear decommitment. After the committer commits to several values, they can open a linear combination of the commitments, in a verifiable manner. This is a weakening of the property of homomorphic commitments, where one can use the homomorphic property to compute the correct linear combination of the commitments, and then open them. These commitments are closely related to the ILC model. One could view parts of our compilation of ILC protocols into zero-knowledge protocols based on hash functions and error-correcting codes as a proof that one can construct a commitment scheme with linear decommitment from these ingredients; one with an interactive decommitment phase.

An interactive protocol is zero-knowledge if for any verifier, even a malicious one, there exists an efficient simulator for the protocol. Honest verifier zero-knowledge [55] is a weaker property, which only guarantees that there exists a simulator for the interaction between an honest prover and an honest verifier. On introducing the notion, [55] show that any protocol with statistical honest-verifier zero-knowledge can be converted into a fully statistical zero-knowledge protocol under the discrete logarithm assumption. [56] show the same result under general one-way permutations, and [57] improved the result to one-way functions.

We say a proof system is *public coin* if the verifier's messages are chosen randomly and correspond exactly to the verifier's randomness. Another result by Damgaard [58] showed that public-coin honest verifier zero-knowledge protocols with a constant number of rounds can be transformed into a fully zero-knowledge protocol without any complexity assumptions. [59] show that statistical honest-verifier zero-knowledge proofs can be converted into statistical fully zero-knowledge proofs without any complexity assumptions.

An interactive zero-knowledge protocol is concurrently secure if even concurrent protocol executions involving a single prover and one or more verifiers do not leak information about the witness. Assuming a common reference string and relying on trapdoor commitments, Damgård [60] gave a transformation yielding concurrently secure protocols for Σ-Protocols. The transformation can be optimized [61] using the idea that for each public-coin challenge x, the prover first commits to a value x', then the verifier sends a value x'', after which the prover opens the commitment and uses the challenge $x = x' + x''$. The coin-flipping can be interleaved with the rest of the proof, which means the transformation preserves the number of rounds and only incurs a very small efficiency cost to do the coin-flipping for the challenges.

If one does not wish to rely on a common reference string for security, one can use a private-coin transformation where the verifier does not reveal the random coins used to generate the challenges sent to the prover (hence the final protocol is no longer public coin). One example is the Micciancio and Petrank [62] transformation (yielding concurrently secure protocols) while incurring a small overhead with respect to the number of rounds of interaction as well as the computational and communication cost in each round. The transformation preserves the soundness and completeness errors of the original protocol; however, it does not preserve statistical zero-knowledge as the obtained protocol only has computational zero-knowledge.

There are other public-coin transformations to general zero-knowledge e.g. Goldreich et al. [59]. The transformation relies on a random-selection protocol between the prover and verifier to specify a set of messages and restricting the verifier to choose challenges from this set. This means to get negligible soundness error these transformations require $\omega(1)$ sequential repetitions so the round complexity goes up.

The Fiat-Shamir transformation [63] is a method of converting public-coin interactive zero-knowledge arguments into non-interactive zero-knowledge proofs. The new non-interactive protocol include a hash function in the common reference string. The prover replaces the verifier's messages with a hash of the protocol transcript up to that point. The resulting arguments are highly efficient in practice and are provably secure in the random oracle model [64]. In the random oracle model, even if the initial interactive proof only has honest verifier zero-knowledge, the resulting argument will have full zero-knowledge.

However, it has been shown [65, 66] that there are interactive protocols which are sound in the random oracle model, but which are insecure for any choice of hash function. Despite this theoretical problem, the Fiat-Shamir heuristic is still used to produce arguments for practical applications, where the hope is that it does give sound arguments for "natural" protocols.

Schnorr [5] and Guillou and Quisquater [67] gave early examples of practical zero-knowledge arguments for concrete number theoretic problems. Schnorr's protocol proves knowledge of a discrete logarithm, and Guillou-Quisquater's protocol proves knowledge of the message corresponding to an RSA encryption. Extending Schnorr's protocols, there have been many constructions of zero-knowledge arguments based on the discrete logarithm assumption. Cramer and Damgård [6] gave a zero-knowledge argument for arithmetic circuit satisfiability, which has linear communication complexity. The argument uses homomorphic commitments to all wire values in the circuit, using the homomorphic property to verify that that the addition gates in the circuit are satisfied, and giving a protocol to verify multiplications which is used for each multiplication gate in the circuit.

Before the logarithmic protocol presented in this thesis, the most efficient discrete logarithm based zero-knowledge arguments for arithmetic circuits were the protocols by Groth [7] and Seo [8], which are constant move arguments with a communication proportional to the square root of the circuit size. Both of these protocols fit into the Ideal Linear Commitment model. The square-root communication cost comes from the fact that all of the wire values in the arithmetic circuit are arranged into a matrix, and the prover sends the verifier a commitment to each row, and a linear combination of the rows. Balancing the number of rows and columns in the matrix gives a square-root communication cost in total. This thesis also gives two arguments for arithmetic circuit satisfiability based on the discrete logarithm assumption. One has fewer rounds of interaction than these previous works, and the other has lower concrete communication costs, and fewer verification equations.

Using pairing-based cryptography instead of just relying on the discrete logarithm assumption, Groth [68] extended these techniques to give a zero-knowledge argument with cube-root communication complexity. In this argument, the prover arranges the wire-values into a cuboid. Each slice of the cuboid is a matrix, and the prover commits to each row of each matrix using a Pedersen commitment, which collapses the cuboid of wire-values into a matrix of Pedersen commitments. Then, the prover uses a pairing-based commitment scheme to collapse the matrix of Pedersen commitments into a vector of pairing-based commitments. The cube-root communication complexity comes from the fact that the prover has to send values to the verifier for each dimension of the cuboid, and balancing the dimensions gives the cube-root. The argument requires only a constant number of moves. It is the ability to use multiple related commitment schemes that allows the compression. In fact, given a candidate cryptographic multi-linear map, one could continue committing to commitments at multiple different levels. For a multi-linear map with l levels, one could achieve $(l+2)$th-root communication complexity.

Our logarithmic-communication-complexity protocol for arithmetic circuit satisfiability employs a similar concept, but works in a slightly different way. Wire-values can be arranged in an d-dimensional hypercube. All elements are committed to using a Pedersen commitment. At each step in the argument, the prover receives a random challenge from the verifier, and takes a random linear combination of $(d-1)$dimensional hypercubes to reduce the dimension of the hypercube by one. This operation is compatible with the Pedersen commitment scheme, up to some correction factors, and results in a Pedersen commitment to fewer elements. It is the sequential interaction over many rounds, and the special properties of the Pedersen commitment scheme, that allows the compression. This led to the protocol given in [9], which shows that not only is the commitment scheme compatible with the compressing operation, but one can reduce a scalar-product check on the original elements to a check on the new compressed elements, leading to a highly efficient protocol for verifying the scalar product of committed vectors.

Bunz et al [1] observe that the original protocol uses separate Pedersen commitments for each input vector to each scalar product, and optimises the argument by giving a new argument where multiple values are contained in a single commitment.

Hyrax [69] uses the same scalar-product argument in a different way, combined with a multi-variate polynomial commitment scheme, to give efficient proofs for highly structured circuits, which achieve sub-linear proof size, linear prover time, and sub-linear verification time when giving proofs for a highly parallelisable circuit, or computing a batch proof for a large number of identical circuits. This approach performs best for circuits of low depth.

[39] adapts the square-root communication protocol of [9] to the post-quantum setting, using commitments based on the hardness of the shortest vector problem for lattices. This work can be seen as the compilation of a particular ILC protocol into the new lattice-based setting. However, the commitment scheme used does not seem to admit the same special properties needed to replicate the argument with logarithmic communication complexity which is possible in the discrete logarithm setting. Like the compilation of ILC protocols based on hash functions, the lattice-based protocol requires an check on all commitments, in the form of a proof-of-knowledge. The main difference between that protocol and our work is the adaptation of the same techniques to a new algebraic setting where the size of elements is important, and which is not a field, so that proof-techniques based on linear algebra become much more difficult to apply.

An exciting line of research [31, 32, 33, 34, 35, 13, 36, 37] has developed many proposals for succinct non-interactive arguments (SNARGs) yielding pairing-based constructions where the arguments consist of a constant number of group elements. The arguments have a constant size, and a constant verification time, which allowed for effective recursive composition [35] and exciting proof-carrying-data techniques. The disadvantage of these arguments is the super-linear computational complexity of the prover. The techniques have also been extended to give proofs with simulation-extractability [70, 71].

Like the ILC model's relationship with many discrete-logarithm-based protocols, the pairing-based protocols above can all be captured using the model of Linear Interactive Proofs and Linear PCPs [72]. In linear interactive proofs, the prover and verifier send vectors of field elements to one another. The verifier makes linear queries on a proof vector created by the prover. However, the prover can only send linear (or affine) transformations of the verifier's previously sent vectors, which distinguishes these systems from ILC protocols, in which the prover is allowed to perform more general computation. It is possible to convert ILC protocols into linear interactive proofs, and vice-versa, but the resulting protocols are usually inefficient, reflecting the fact that the models were tailored to different use-cases. Furthermore, linear interactive proofs usually involve a verifier of algebraic degree two. The ILC protocols presented in this work sometimes have a higher algebraic degree, so they could not be converted into linear interactive proofs and compiled using the same methods.

The idealised linear interactive proofs are compiled into real arguments in pairing groups in works like [37] by creating a common reference string with all of the verifier's queries embedded into the exponents of group elements. The resulting protocols are proved secure under strong assumptions implying that whenever one can find group elements satisfying particular equations, then the group elements must be linear combinations of the group elements in the CRS. However, due to the nature of the verifier's queries in the idealised proofs, the common reference strings are highly structured and must either be generated by a trusted third party or an expensive multiparty computation protocol. Other works such as [73] attempt to mitigate the problem. They give a protocol which updates common reference strings, so that the updated common reference strings are trustworthy, even if the old ones were not. [74, 75] present secure multi-party computation protocols used to generate the common reference strings of pairing-based SNARKs.

Furthermore, non-falsifiable knowledge extractor assumptions are used to guarantee security. In contrast, the arguments we develop here are based solely on the discrete logarithm assumption, or collision resistant hash functions, and use a small common reference string which is independent of the circuit. This is because we choose to compile our idealised protocols under these alternative assumptions.

Bootle et al [16] used error-correcting codes and linear-time collision-resistant hash functions to give the first zero-knowledge proof and argument systems for arithmetic circuit satisfiability with constant computational overhead. The prover uses a linear number of field multiplications, and verification is even more efficient, requiring only a linear number of additions. They proposed the ILC model, which forms the basis of this work. Their methodology was also similar, designing an ideal protocol with the correct security properties and compiling it into a real proof. The result hinges on choices of particularly efficient linear-time-computable hash functions and codes. This work includes similar compilations, modified to account for changes in the ILC model, some optimisations using Reed-Solomon codes, and the discrete logarithm setting.

STARKs [17] give an argument with logarithmic communication costs, and logarithmic verification costs. Computational complexity for the prover is quasilinear, but the large constants involved give this approach higher computational time for the prover than other cryptographic proof implementations such as [1, 69] for giving proofs about practical instances.

Another effective way to construct efficient zero-knowledge proofs is to follow the so-called MPC-in-the-head paradigm of [76]. This approach leads to very efficient constructions both in theory and in practice. When the prover wants to prove, for example, that a circuit is satisfiable, they simulate a secure multi-party computation protocol to evaluate the circuit on secret-shared inputs. They commit to the view of each party in the multi-party computation protocol. The verifier randomly selects some fraction of the views, and checks that they are consistent. ZKBoo [77] and subsequent optimisation ZKB++ [78] use hash functions to construct zero-knowledge arguments for the satisfiability of boolean circuits. Their communication complexity is linear in the circuit size, but the use of symmetric primitives gives good performance in practice.

Ligero [15] provides another implementation of the MPC-in-the-head paradigm and used techniques similar to [16] to construct sublinear arguments for arithmetic circuits. The approach used in Ligero is similar to the approach used in this work. One difference is that Ligero uses the multiplicative properties of Reed-Solomon codewords to help verify multiplications. This work does not require codewords to have any multiplicative properties.

Jawurek et al. [79] gave a different approach to zero-knowledge proofs derived from multiparty computation protocols, using garbled circuits.

All of the arguments mentioned above which rely on collision-resistant hash-functions for security only require a simple common reference string including a description of the hash-function. This makes them suitable for blockchain applications where a trusted-setup procedure is particularly undesirable.

Other works give a composite approach, combining different zero-knowledge proof systems for both algebraic and non-algebraic techniques. Chase et al [80] uses two approaches to interactive zero-knowledge. They use the garbled-circuit techniques of [79] to prove non-algebraic statements, and algebraic protocols based on the discrete logarithm assumption and RSA assumption to prove algebraic statements. They leverage both techniques at the same time in order to construct efficient privacy-preserving credentials. [81] uses similar ideas to give non-interactive zero-knowledge proofs, this time by combining pairing-based SNARKs with discrete-logarithm-based interactive zero-knowledge proofs, after using the Fiat-Shamir heuristic to make the interactive protocols non-interactive.

Another model for protocols is that of interactive oracle proofs, introduced in [82]. Interactive oracle proofs are interactive proofs between a prover, and a verifier, in which the verifier only has query access to the prover's messages. They simultaneously generalise interactive proofs and PCPs. Intuitively, one way to view interactive oracle proofs is as PCPs where the interaction between the prover and the verifier means that the prover only has to compute a small part of the PCP for the verifier to check. In fact, Ideal Linear Commitment protocols can also be seen as interactive oracle proofs with some extra restrictions.

Recently, the notion of linear PCPs was generalised to fully-linear PCPs in [83]. Just as PCPs were generalised by IOPs, the same paper, [83] goes on to generalise LPCPs further to (fully-)linear IOPs. In particular, linear and fully-linear IOPs are very similar to Ideal Linear Commitment protocols, allowing the verifier to make linear queries on a committed proof string, but ultimately measures the efficiency of a protocol using different metrics, as the authors aim to describe different types of interactive protocols.

As well as general proof proof systems, various works give protocols with low communication complexity for specific languages. For example, in a membership argument [84, 85], a prover demonstrates that a secret committed value λ is an element of a list $\mathscr{L} = \{\lambda_0, \dots, \lambda_{N-1}\}$, without revealing any other information about λ. In a polynomial evaluation argument [86, 85], a prover demonstrates that a secret committed value v is the evaluation of a public polynomial $h(U)$ at another secret committed value u. In a range proof [87, 88], a prover demonstrates that a secret committed value a is an element of the interval $[A;B]$.

The goals of membership arguments are related to those of zero-knowledge sets [89]. Membership arguments allow a prover to commit to a secret value and show that it lies in a public set, without leaking information on the value. On the other hand, zero-knowledge sets allow the prover to commit to a secret set, and handle membership and non-membership queries in a verifiable manner, without leaking information on the set.

Herranz constructs attribute-based signatures [90] using what is essentially a set membership argument for multiple values. The argument relies only on the discrete logarithm assumption, but the communication complexity is linear in the size of the set. Camenisch et al. [91] also provide set membership proofs with logarithmic communication complexity, and Fauzi et al. [92] construct constant size arguments for more complex relations between committed sets. The latter two works both rely on pairing-based assumptions.

Groth and Kohlweiss [2], and a follow-up work [10] show that one can prove that one out of N commitments contain 0 with logarithmic communication complexity, based on the discrete logarithm assumption. For homomorphic commitments, it is easy to reduce the task of a membership argument to the task of checking that one commitment contains a zero, by dividing every commitment in the public set by the prover's own commitment. Both arguments work by arranging the values in the list into a tree, and having the prover commit to a sequence of bits which describe a path from the root of the tree to the leaf which is equal to the prover's own commitment. [10] optimises the protocol given in [2] by generalising to an n-ary tree rather than a binary one.

Range arguments can be seen as a special case of membership arguments, where $\mathscr{L}$ is simply a list of consecutive integers. Many are based on the strong RSA assumption, and use Lagrange's Four-Square Theorem. Couteau et al. show that this assumption can be replaced by an RSA-variant which is much closer to the standard RSA assumption [93]. Their techniques can be applied to existing works such as [94, 88]. Chaabouni et al. [95] give an argument with sub-logarithmic communication complexity in the size of the list, which is comparable to the efficiency we achieve, and also relies on the hardness of the discrete logarithm problem, but uses pairings for verification.

Membership arguments also generalise arguments that a committed value lies in a linear subspace such as [96, 97, 98]. The protocols in these works all operate in bilinear-pairing groups. Peng [99] achieves a square-root complexity. Some existing protocols [3], [2] even achieve logarithmic communication complexity. Our single-value membership proof is an extension of the latter works where we reduce the number of commitments from logarithmic to constant.

Cryptographic accumulators [100, 101, 102, 103] can also be used to give membership proofs. The members of a set are absorbed into a constant-size accumulated value. Witnesses for set-membership can then be generated and verified using the accumulated value. Efficient instantiations of accumulators exist and often rely on the Strong RSA assumption or pairing-based assumptions. An RSA modulus has to be $\frac{\lambda^3}{\text{polylog}\lambda}$ bits to provide security against factorisation using the General Number Field Sieve. For a given security level, pairing-based schemes with constant embedding degree scale similarly due to sub-exponential algorithms for attacking the discrete logarithm problem in the target group. Furthermore, such schemes require a trusted setup. By contrast, when instantiating our proofs using Pedersen commitments or collision resistant hash functions, we only require commitments of size $O(\lambda)$ bits for security against discrete logarithm attacks in elliptic curve groups, or collision-finding attacks against the hash functions.

Some of the schemes can be adapted to give zero-knowledge arguments for non-membership, from a variety of settings. For example, [3, 99] also give non-membership arguments in the discrete logarithm setting. Accumulators that support non-membership arguments have been constructed, based on both pairing assumptions ([104]) and the strong RSA assumption ([105]).

Our polynomial commitment protocol is a key part of our zero-knowledge argument. Polynomial commitments were first introduced by Kate et al. [106], who give a construction using bilinear maps. The original construction has also been extended to the multivariate case [107, 108]. Libert et al. [109] also gave a construction relying on much simpler pairing-based assumptions. Our polynomial commitment protocol builds on the polynomial commitment protocol presented in [9], and gives a square-root communication complexity when instantiated with compact commitments. Later, Hyrax [69] gives a commitment scheme for multilinear polynomials using the scalar-product argument of [9, 1], with a logarithmic communication complexity. The same idea can be incorporated into our batch protocol for low-degree polynomials, but does not improve asymptotic performance, so for ease of exposition, we do not discuss this.

Some zero-knowledge proofs and arguments use the idea of embedding many statements into a single polynomial using Lagrange interpolation polynomials in a challenge x. The idea originates in the quadratic arithmetic programs of Gennaro et al. [34]. It was used in the context of interactive zero-knowledge arguments by Bayer [12]. The technique was originally applied to construct a Hadamard product argument and batched polynomial evaluation argument. Earlier work by Gennaro et al. [110] batches Schnorr proofs using simple powers of x.

Other batch arguments in the literature use methods from [111] and multiply different instances of the proof by small exponents before compressing the proofs together. This approach may be used to trade soundness for efficiency. The batch argument in this thesis proves and verifies the logical AND of many statements simultaneously. There are also batch proofs for OR statements [112], and k-out-of-N batch proofs [113]. Finally, Henry and Goldberg [113] define a notion of conciseness to characterise batch proofs.

Camenisch and Stadler [114] also propose a general framework of relations for zero-knowledge proofs based on the discrete logarithm assumption. Their notation is useful for describing large and complex statements. We take some inspiration from their notation, but use different notation since the ILC model describes general relations over fields, and values committed using a generic commitment scheme.

Chapter 3

Formal Definitions

3.1 Notation and Computational Model

This thesis is concerned with algorithms modelled as probabilistic polynomial-time Turing machines, and interactive probabilistic polynomial-time Turing machines. We use the abbreviations PPT and DPT for algorithms running in probabilistic polynomial time and deterministic polynomial time respectively, where the running time is polynomial in the size of the algorithm inputs. Unless otherwise stated, the sizes of inputs and outputs to the machines will be bounded above by a polynomial in a security parameter λ, usually provided to the algorithms in unary form as 1^{λ}.

For functions $f, g : \mathbb{N} \to [0,1]$, we write $f(\lambda) \approx g(\lambda)$ if $|f(\lambda) - g(\lambda)| = \frac{1}{\lambda^{\omega(1)}}$. We say a function f is *overwhelming* if $f(\lambda) \approx 1$ and f is *negligible* if $f(\lambda) \approx 0$.

Write $y = A(x;r)$ when the algorithm A outputs y on input x with randomness r. We write $y \leftarrow A(x)$ to mean selecting r at random and setting $y = A(x;r)$. We write $y \leftarrow S$ for sampling y uniformly at random from a set S. We define $[n]$ to be the set of integers $1, \ldots, n$.

We use $\mathbb{F}$ to denote a finite field. We use bold letters such as $\mathbf{v}$ for row vectors. For $\mathbf{v} \in \mathbb{F}^n$ and a set $J = \{j_1, \ldots, j_k\} \subset [n]$ with $j_1 < \cdots < j_k$ we define the vector $\mathbf{v}|_J$ to be $(\mathbf{v}_{j_1}, \ldots, \mathbf{v}_{j_k})$. Similarly, for a matrix $V \in \mathbb{F}^{m \times n}$ we let $V|_J \in \mathbb{F}^{m \times k}$ be the submatrix of V restricted to the columns indicated in J.

Let $z_1, \ldots, z_m$ be distinct points in a finite field $\mathbb{F}$. and let $l_1(X), \ldots, l_m(X)$ be their associated Lagrange polynomials. Explicitly,

$$l_i(X) = \prod_{j \neq i} \frac{X - z_j}{z_i - z_j}$$

Note that $l_i(z_j) = \delta_{i,j}$, so it is easy to see how to combine the Lagrange polynomials to produce a polynomial which interpolates a given function at $z_1, \ldots, z_m$. Let $l_0(X) = \prod_{i=1}^{m}(X - z_i)$.

3.2 Arithmetic Circuits

Arithmetic circuits are a model for algebraic computation over fields. An arithmetic circuit consists of addition and multiplication gates. Our satisfiability arguments consider arithmetic circuits described as a list of multiplication gates together with a set of linear consistency equations relating the inputs and outputs of the gates. In this section, we show how to reduce an arbitrary arithmetic circuit to this format.

Definition 1 *An arithmetic circuit over a field $\mathbb{F}$ and variables $(A_1, \ldots, A_m)$ is a directed acyclic graph whose vertices are called gates. Gates of in-degree 0 are inputs to the circuit and labelled with some A_i or a constant field element. All other gates are labelled $+$ or $\times$.*

Given field elements $a_i \in \mathbb{F}$, an arithmetic circuit is evaluated in several steps. First, label the inputs with the a_i. Then, take gates whose inputs are all labelled with field elements, apply the operation on the gate to the inputs and write the answer on the output wire. and repeating this process until all gates have been labelled with output field elements.

We may consider fan-in 2 circuits, in which case all of the $+$ and $\times$ gates have in-degree 2, or arbitrary fan-in circuits. We consider circuits with arbitrary fan-out, in which case all of the $+$ and $\times$ gates have unbounded out-degree.

Arithmetic circuits can be measured in various ways. The size of an arithmetic circuit is the number of gates in the circuit. This can be further split into the number of addition gates and the number of multiplication gates. The depth of a circuit is the length of the longest path beginning from any circuit input. It is easy to see that arithmetic circuits compute polynomial functions of their inputs, and the degree of the arithmetic circuit is the total degree of the polynomial that it computes.

Arithmetic circuits can be described alternatively as a list of multiplication gates with a collection of linear consistency equations relating the inputs and outputs of the gates. Our zero-knowledge protocols for circuit satisfiability use circuits in this form. Any circuit described as an acyclic graph can be efficiently converted into the alternative description.

At a high level, we transform an arithmetic circuit into two kinds of equations. Multiplication gates are directly represented as equations of the form $a \cdot b = c$, where a, b, c represent the left, right and output wires. We will arrange these values in matrix form producing a Hadamard matrix product. This process will lead to duplicate values, when a wire is the output of one multiplication gate and the input of another, or when it is used as input multiple times. We keep track of this by using a series of linear constraints. For example, suppose we have two multiplication gates with wire values a_1, b_1, c_1 and a_2, b_2, c_2. If the output of the first multiplication gate is the right input of the second, we would write $c_1 - b_2 = 0$.

We also add linear constraints representing the addition and multiplication by constant gates of the circuit. We then rewrite those equations so that the only wires that are referenced in the equations are those linked to (non-constant) multiplication gates. We now describe this process.

3.2.1 Preprocessing Arithmetic Circuits for Arguments

We show how to remove addition and multiplication-by-constant gates from an arithmetic circuit A, and replace them with bilinear consistency equations on the inputs and outputs of the remaining gates, such that satisfiability of the equations is equivalent to satisfiability in the original circuit.

Let B be the sub-circuit of A containing all wires and gates before a multiplication gate, with m input wires and n output wires. Label the m inputs of B with the unit vectors $\mathbf{e}_i = (0, \ldots, 1, \ldots, 0)$ of length m. For every addition gate with inputs labelled as $\mathbf{x}, \mathbf{y}$, label the output wire as $\mathbf{x} + \mathbf{y}$. For every multiplication-by-constant gate with inputs $\mathbf{x}$ and constant c label the output with $c\mathbf{x}$. By proceeding inductively, the n outputs of B are now labelled with vectors of length m representing them as linear combinations of the inputs.

This requires at most $m\,|B|$ arithmetic operations. Note however that all outputs of B are linear combinations of the inputs, and that B can be written with $n(2m-1)$ fan-in 2 gates in such a way that the consistency equations can be trivially read off from the circuit description. More specifically, a linear combination $\sum_{i=1}^{m} a_i x_i$ can be produced using m multiplication-by-constant gates and $m-1$ addition gates to add the answers together.

We can now remove the gates of B from A. We also remove any multiplication gates whose inputs are the inputs of the new circuit. Now we simply repeat the process of finding consistency equations until we have considered the whole of A. In Figure 3.1 there is an example of a circuit together and the corresponding consistency equations.

The first (input) and final (output) sub-circuits require additional processing. We show how to do this for the output sub-circuit. The input sub-circuit is very similarly handled.

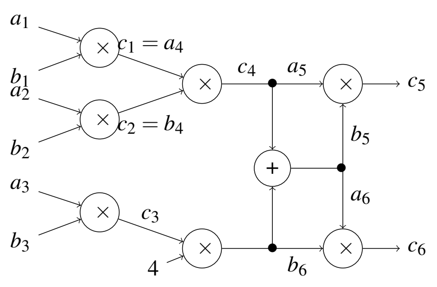
Let B be the output sub-circuit. Write $(a_1,\ldots,a_m) = \mathbf{a}$ for the input wires of B and $(b_1,\ldots,b_n) = \mathbf{b}$ for the output wires. Without loss of generality, we may ignore variable output wires. By construction of B, each output b_i is of the form $\sum_{i=1}^{n} q_{ij} a_j + p_i$, with consistency equations obtained as above. We write this in terms of an $m \times n$ matrix Q and a column vector $\mathbf{p}$ of size m, namely

$$\mathbf{b} = \mathsf{Q}\mathbf{a} + \mathbf{p}.$$

Let r be the rank of Q. We convert Q into reduced row echelon form R, writing

$$\mathbf{b}'' = \mathsf{R}\mathbf{a}.$$

By the properties of reduced row echelon form, after relabelling the a_i and permuting the columns of R to match, we have that $b''_i = a_i + \sum_{j=l+1}^{m} r_{ij} a_j$ for $1 \leq i \leq l$. Therefore, we may consider $a_{l+1},\ldots,a_m$ as free wires and express other



$$c_1 = a_4$$
$$c_2 = b_4$$
$$c_4 = a_5$$
$$4c_3 + c_4 = b_5$$
$$4c_3 + c_4 = a_6$$
$$4c_3 = b_6$$

Figure 3.1: A simple arithmetic circuit, and the corresponding consistency equations. The first sub-circuit contains the wires $a_1, b_1, c_1, a_2, b_2, c_2, a_3, a_3, c_3$. The second sub-circuit contains the wires $c_1, a_4, c_2, b_4, c_4, c_5, c_6$. The third sub-circuit B contains the wires c_3, c_4, a_5, b_5, a_6.

a_i as linear functions of these wires plus constants.

Note that if $b''_i \neq 0$ for some $i > l$, the circuit can never be satisfied anyway. However, assuming that our statement is a satisfiable circuit, with a witness consisting of satisfying wire values, this never occurs. Then the original circuit is satisfied if and only if the a_i values satisfy the consistency equations.

If Q is an $m \times n$ matrix then it can be converted into reduced row echelon form using $O(\max(m,n)mn)$ operations. It is trivial that $m \leq 2|B|$ and $n \leq |B|$. This gives an upper bound of $O(|B|^3)$ computation for the output sub-circuit. Note that this is often a large over-estimate; this upper bound occurs for circuits of depth 1 where inputs feed into distinct gates. For circuits of large depth, where the same input is fed into several gates, the upper bound will definitely not be reached.

The case of the input sub-circuit is very similar, except that we take the transpose of the matrix.

3.2.2 Reduction of Circuit Satisfiability Problem to a Hadamard Matrix Product and Linear Constraints.

Having preprocessed the arithmetic circuit as in the previous section, we may assume that the input and the output wires feed into and go out from multiplication gates only. We number the multiplication gates from 1 to N and we arrange the inputs and outputs of these gates into three $m \times n$ matrices A, B and C such that the (i, j) entries of the matrices correspond to the left input, right input and output of the same multiplication gate.

As shown in [9], an arithmetic circuit can be described as a system of equations in the entries of the above matrices. The multiplication gates define a set of N equations

$$A \circ B = C \tag{3.1}$$

where $\circ$ is the Hadamard (entry-wise) product. The circuit description also contains constraints on the wires between multiplication gates. Denoting the rows of the matrices A, B, C as

$$\mathbf{a}_i = (a_{i,1}, \dots, a_{i,n}) \quad \mathbf{b}_i = (b_{i,1}, \dots, b_{i,n}) \quad \mathbf{c}_i = (c_{i,1}, \dots, c_{i,n}) \quad \text{for } i \in \{1, \dots, m\}$$

these constraints can be expressed as $Q < 2N$ linear equations of inputs and outputs of multiplication gates of the form

$$\sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{w}_{q,a,i} + \sum_{i=1}^{m} \mathbf{b}_i \cdot \mathbf{w}_{q,b,i} + \sum_{i=1}^{m} \mathbf{c}_i \cdot \mathbf{w}_{q,c,i} = K_q \quad \text{for } q \in \{1, \dots, Q\} \tag{3.2}$$

for constant vectors $\mathbf{w}_{q,a,i}, \mathbf{w}_{q,b,i}, \mathbf{w}_{q,c,i}$ and scalars K_q.

For example, suppose that the circuit contains a single addition gate, with $a_{1,1}$ and $a_{1,2}$ as inputs, and $b_{1,1}$ as output. In this case, $Q = 1$ and we would set $\mathbf{w}_{1,a,1} = (1, 1, 0, \dots, 0)$, $\mathbf{w}_{1,b,1} = (-1, 0, \dots, 0)$, and all other $\mathbf{w}$ vectors would be set to $\mathbf{0}$. Then (3.2) would simply read

$$a_{1,1} + a_{1,2} - b_{1,1} = 0$$

to capture the constraint imposed by the addition gate.

In total, to capture all multiplications and linear constraints, we have $N + Q$ equations that the wires must satisfy in order for the circuit to be satisfiable.

3.3 Commitment Schemes

A non-interactive commitment scheme allows a sender to commit to a secret message and later reveal the message in a verifiable way. Here we are interested in commitment schemes that take as input an arbitrary length message so the message space is $\{0,1\}^*$. A commitment scheme is defined by a pair of PPT algorithms $(\mathsf{Setup}, \mathsf{Commit})$.

$\mathsf{Setup}(1^\lambda) \to ck$**:** Given a security parameter, this returns a commitment key ck.

$\mathsf{Commit}_{ck}(m) \to c$**:** Given a message m from a message space $\mathscr{M}_{ck}$, this picks randomness $r \leftarrow \mathscr{R}_{ck}$, from a randomness space, and computes a commitment $c = \mathsf{Commit}_{ck}(m;r) \in \mathscr{C}_{ck}$.

A commitment scheme must be *binding* and *hiding*. The binding property means that it is infeasible to open a commitment to two different messages, whereas the hiding property means that the commitment does not reveal anything about the committed message.

Definition 2 (Binding) *A commitment scheme is* computationally binding *if for all PPT adversaries* $\mathscr{A}$

$$\Pr\left[\begin{array}{c} ck \leftarrow \mathsf{Setup}(1^\lambda);\ (m_0, r_0, m_1, r_1) \leftarrow \mathscr{A}(ck) : \\ m_0 \neq m_1 \ \wedge\ \mathsf{Commit}_{ck}(m_0; r_0) = \mathsf{Commit}_{ck}(m_1; r_1) \end{array}\right] \approx 0.$$

If this holds also for unbounded adversaries, we say the commitment scheme is statistically binding.

Definition 3 (Hiding) *A commitment scheme is* computationally hiding *if for all PPT stateful adversaries* $\mathscr{A}$

$$\Pr\left[\begin{array}{c} ck \leftarrow \mathsf{Setup}(1^\lambda);\ (m_0, m_1) \leftarrow \mathscr{A}(ck);\ b \leftarrow \{0,1\}; \\ c \leftarrow \mathsf{Commit}_{ck}(m_b) :\ \mathscr{A}(c) = b \end{array}\right] \approx \frac{1}{2},$$

where $\mathscr{A}$ *outputs messages of equal length* $|m_0| = |m_1|$. *If the definition holds also for unbounded adversaries, we say the commitment scheme is* statistically hiding.

Suppose further that $(\mathscr{M}_{ck}, \cdot)$, $(\mathscr{R}_{ck}, \circ)$ and $(\mathscr{C}_{ck}, \oplus)$ are groups.

Definition 4 (Homomorphic Commitment Scheme) *We call the commitment scheme homomorphic if* $\mathsf{Commit} : \mathscr{M}_{ck} \times \mathscr{R}_{ck} \to \mathscr{C}_{ck}$ *is a group-homomorphism, i.e.*

$$\mathsf{Commit}(m \cdot m'; r \circ r') = \mathsf{Commit}(m; r) \oplus \mathsf{Commit}(m'; r')$$

3.4 Error-Correcting Codes

A *code* over an alphabet Σ is a subset $\mathscr{C} \subseteq \Sigma^n$. A code $\mathscr{C}$ is associated with an encoding function $E_{\mathscr{C}} : \Sigma^k \to \Sigma^n$ mapping messages of length k into *codewords* of length n. We assume there is a setup algorithm $\mathsf{Gen}_{\mathsf{E}_{\mathscr{C}}}$ which takes as input a finite field $\mathbb{F}$ and the parameter $k \in \mathbb{N}$, and outputs an encoding function $E_{\mathscr{C}}$.

We use error-correcting codes as part of our compilation from ILC protocols to proof systems based on collision-resistant hash-functions. We restrict our attention to $\mathbb{F}$-*linear codes* for which the alphabet is a finite field $\mathbb{F}$, the code $\mathscr{C}$ is a k-dimensional linear subspace of $\mathbb{F}^n$, and $E_{\mathscr{C}}$ is an $\mathbb{F}$-linear map. The *rate* of the code is defined to be $\frac{k}{n}$. The *Hamming distance* between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}^n$ is denoted by $\mathsf{hd}(\mathbf{x}, \mathbf{y})$ and corresponds to the number of coordinates in which $\mathbf{x}, \mathbf{y}$ differ. The *(minimum) distance* of a code is defined to be the minimum Hamming distance $\mathsf{hd}_{\min}$ between distinct codewords in $\mathscr{C}$. We denote by $[n, k, \mathsf{hd}_{\min}]_{\mathbb{F}}$ a linear code over $\mathbb{F}$ with length n, dimension k and minimum distance $\mathsf{hd}_{\min}$. The *Hamming weight* of a vector $\mathbf{x}$ is $\mathsf{wt}(\mathbf{x}) = |\{i \in [n] : \mathbf{x}_i \neq 0\}|$.

To get good results when compiling, we will use families of linear codes achieving asymptotically good parameters. More precisely, we require codes with linear length, $n = \Theta(k)$, and linear distance, $\mathsf{hd}_{\min} = \Theta(k)$, in the *dimension* k of the code. We recall that random linear codes achieve with high probability the best trade-off between distance and rate. However, in this work we are concerned with the computational efficiency of the encoding procedure and random codes are not known to be very efficient. The reader can keep Reed-Solomon codes in mind for concrete instantiations, as these as simple, satisfy all of the necessary requirements, and are practically efficient as well as asymptotically.

Chapter 4

Zero-Knowledge Proofs and the Ideal Linear Commitment Model

In this chapter, we begin by defining zero-knowledge proofs. Then we define what it means to be an Ideal Linear Commitment protocol, and compare that model with other similar models which give rise to zero-knowledge proof systems.

This chapter is important, as the core contribution of this thesis is to provide efficient Ideal Linear Commitment protocols and show that they can be converted into real proof systems. None of this would be possible without first precisely definining the model.

We choose to present zero-knowledge proofs and the definition of the Ideal Linear Commitment model together in the same chapter, because the two types of protocol will satisfy largely similar security definitions.

4.1 Zero-Knowledge Proofs

A *proof system* is defined by a triple of stateful PPT algorithms $(\mathscr{K}, \mathscr{P}, \mathscr{V})$, which we call the common-reference-string *generator*, the *prover* and *verifier*, respectively.

The setup generator $\mathscr{K}$ creates public parameters σ which provide the necessary setup information for $\mathscr{P}$and $\mathscr{V}$to run the protocol. On input 1^λ, the generator $\mathscr{G}$ produces a common reference string σ. We think of σ as being honestly generated and use the public parameter model purely for simplicity and efficiency in our proofs. However, in the proofs we construct, σ consists of parts that are either publicly verifiable or could be generated by the verifier, so we do not rely on the public parameter model for security in any way.

Different Types of Interactive Protocols. The prover and verifier interact with each other in some sort of *interaction environment*, or *communication channel* which we will denote by $\overset{\text{chan}}{\longleftrightarrow}$. In the usual interaction environment $\longleftrightarrow$, all messages are forwarded between prover and verifier. In fact, $\longleftrightarrow$ refers to exactly the sort of interaction defined by the complexity class IP. We also consider an *ideal linear commitment* environment, ILC, defined properly in Section 4.2. This thesis is concerned with producing ILC proof protocols which are defined by interactions between the prover and verifier in the ILC environment.

Why speak of interaction environments at all? We do this because the definitions of what it means to be a secure zero-knowledge proof protocol will really depend very little on whether the interaction takes place as an interactive proof protocol or an ILC protocol. This is to be expected, as ILC proof protocols were designed as an information theoretic abstraction of certain zero-knowledge proofs.

When $\mathscr{P}$ and $\mathscr{V}$ interact on inputs s and t via $\overset{\text{chan}}{\longleftrightarrow}$, then let $\mathsf{view}_{\mathscr{V}} \leftarrow \langle \mathscr{P}(s) \overset{\text{chan}}{\longleftrightarrow} \mathscr{V}(t) \rangle$ be the view of the verifier in the execution, which is made up of all of the verifier's inputs, including random coins, and let $\mathsf{trans}_{\mathscr{P}} \leftarrow \langle \mathscr{P}(s) \overset{\text{chan}}{\longleftrightarrow} \mathscr{V}(t) \rangle$ denote the transcript of the communication between prover and channel. This overloads the notation $\leftarrow \langle \mathscr{P}(s) \overset{\text{chan}}{\longleftrightarrow} \mathscr{V}(t) \rangle$ but it will always be clear from the variable name if we get the verifier's view or the prover's transcript. At the end of the interaction the verifier accepts or rejects. We write $\langle \mathscr{P}(s) \overset{\text{chan}}{\longleftrightarrow} \mathscr{V}(t) \rangle = b$ depending on whether the verifier rejects ($b = 0$) or accepts ($b = 1$).

We say a proof system is *public coin* if the verifier's messages to the communication channel are chosen uniformly at random and independently of the actions of the prover, i.e., the verifier's messages to the prover correspond to the verifier's randomness ρ. All of our protocols, whether they are ILC protocols, or real protocols compiled under cryptographic assumptions, will be public coin. Public coin protocols are very relevant as they are conceptually simple, and one can prove simpler variants of standard security properties, such as special-honest-verifier zero-knowledge, with the understanding that standard transformations and techniques will provide full security later.

We will model our zero-knowledge proofs using ternary relations $\mathscr{R}$ consisting of tuples (σ, u, w). The first item in the tuple is the *common reference string* σ containing the setup information required for the protocol. Typically, σ will specify the security parameter λ, perhaps implicitly through its length, and may also contain other parameters used for specifying the specific relation, e.g. a description of a field. Often, σ will also contain parameters that do not influence membership of $\mathscr{R}$ but may aid the prover and verifier, for instance, a description of an encoding function that they will use. The second item in the tuple, u is the *instance* and represents what the prover wants to prove. The final item, w, is the prover's secret *witness* that $(\sigma, u) \in \mathscr{L}_{\mathscr{R}}$ where the language $\mathscr{L}_{\mathscr{R}} \subset \{0,1\}^*$ as follows.

$$\mathscr{L}_{\mathscr{R}} = \{(\sigma, u) | \exists w : (\sigma, u, w) \in \mathscr{R}\}$$

Intuitively speaking, this is the collection of statements which are 'true', and for which the verifier should output 1 after running the protocol with an honest prover. The languages $\mathscr{L}_{\mathscr{R}}$ are decidable in polynomial time.

The protocol $(\mathscr{K}, \mathscr{P}, \mathscr{V})$ is called a *proof of knowledge* over communication channel $\overset{\text{chan}}{\longleftrightarrow}$ for relation $\mathscr{R}$ if it has perfect completeness and computational knowledge soundness as defined below.

Definition 5 (Perfect Completeness) *The proof is* perfectly complete *if for all PPT adversaries $\mathscr{A}$*

$$\Pr\left[\begin{array}{c} \sigma \leftarrow \mathscr{K}(1^\lambda); (u,w) \leftarrow \mathscr{A}(\sigma) : \\ (\sigma, u, w) \notin \mathscr{R} \vee \langle \mathscr{P}(\sigma, u, w) \overset{\text{chan}}{\longleftrightarrow} \mathscr{V}(\sigma, u) \rangle = 1 \end{array}\right] = 1.$$

Definition 6 (Knowledge soundness) *A public-coin proof system has* computational (strong black-box) knowledge soundness *if for all DPT $\mathscr{P}^*$ there exists* *an expected PPT extractor $\mathscr{E}$ such that for all PPT adversaries $\mathscr{A}$*

$$\Pr\left[\begin{array}{c}\sigma \leftarrow \mathscr{K}(1^\lambda); (u,s) \leftarrow \mathscr{A}(\sigma); w \leftarrow \mathscr{E}^{\langle \mathscr{P}^*(s) \overset{\text{chan}}{\longleftrightarrow} \mathscr{V}(\sigma,u)\rangle}(\sigma,u) : \\ b = 1 \wedge (\sigma,u,w) \notin \mathscr{R}\end{array}\right] \approx 0.$$

Here the oracle $\langle \mathscr{P}^*(s) \overset{\text{chan}}{\longleftrightarrow} \mathscr{V}(\sigma,u)\rangle$ runs a full protocol execution and if the proof is successful it returns a transcript of the prover's communication with the channel. The extractor $\mathscr{E}$ can ask the oracle to rewind the proof to any point in a previous transcript and execute the proof again from this point on with fresh public-coin challenges from the verifier. We define $b \in \{0,1\}$ to be the verifier's output in the first oracle execution, i.e., whether it accepts or not, and we think of s as the state of the prover. The definition can then be paraphrased as saying that if the prover in state s makes a convincing proof, then we can extract a witness.

If the definition holds also for unbounded $\mathscr{P}^*$ and $\mathscr{A}$ we say the proof has statistical knowledge soundness.

If the definition of knowledge soundness holds for a non-rewinding extractor, i.e., a single transcript of the prover's communication with the communication channel suffices, we say the proof system has knowledge soundness with straight-line extraction.

This definition gives a security guarantee against computationally bounded adversaries. Zero-knowledge protocols satisfying this definition are properly called zero-knowledge arguments of knowledge to distinguish them from zero-knowledge proofs of knowledge, for which knowledge soundness is guaranteed even against unbounded adversaries. However, the term 'proofs' is often used in both cases.

Remark. The definition of knowledge soundness uses a knowledge extractor running in expected polynomial time. One might question the validity of proofs of computational knowledge soundness based on some cryptographic assumption, which prove that either the knowledge extractor extracts a witness, or manages to break some computational assumption in expect polynomial time. This is because cryptographic assumptions are usually defined using probabilistic polynomial time adversaries rather than expected polynomial time adversaries. Fortunately, Markov's inequality allows us to relate the expected polynomial time extractor to a probabilistic polynomial time algorithm. Suppose that the expected running time of the extractor is given by T. Then, by Markov's inequality, the probability that the knowledge extractor runs for time longer than $2T$ is at most $1/2$. If we define a new algorithm that runs the knowledge extractor for time at most $2T$ and then aborts if the running time exceeds $2T$, then we have a probabilistic polynomial time algorithm capable of either finding a witness or breaking a computational assumption. This means that it is really possible to reduce the knowledge soundness of the scheme to a standard cryptographic assumption, at the cost of some looseness in the reduction coming from the probability that the knowledge extractor runs for too long and the algorithm aborts.

Another way to define a proof of knowledge follows Groth and Ishai [115] who borrowed the term witness-extended emulation from Lindell [116]. Informally, their definition says that given an adversary that produces an acceptable argument with some probability, there exists an emulator that produces a similar argument with the same probability together with a witness w. Note that the emulator is allowed to rewind the prover and verifier's interaction to any previous move.

Definition 7 (Witness-extended emulation) $(\mathscr{P},\mathscr{V})$ *has* statistical witness-extended emulation *if for all deterministic polynomial time $\mathscr{P}^*$ there exists an expected polynomial time emulator $\mathscr{E}$ such that for all interactive adversaries $\mathscr{A}$*

$$\Pr\left[(u,s)\leftarrow\mathscr{A}(1^\lambda); tr\leftarrow\langle\mathscr{P}^*(u,s)\overset{\text{chan}}{\longleftrightarrow}\mathscr{V}(u)\rangle : \mathscr{A}(tr)=1\right]$$
$$\approx \Pr\left[\begin{array}{c}(u,s)\leftarrow\mathscr{A}(1^\lambda);(tr,w)\leftarrow\mathscr{E}^{\langle\mathscr{P}^*(u,s)\overset{\text{chan}}{\longleftrightarrow}\mathscr{V}(u)\rangle}(u):\\ \mathscr{A}(tr)=1 \text{ and if } tr \text{ is accepting then } (u,w)\in R\end{array}\right]$$

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

Note that in this definition, the role of the adversary $\mathscr{A}$ is as a distinguisher, while it is the job of $\mathscr{P}^*$ to produce proofs. That is why the two are decoupled from one another.

We can interpret s as the state of $\mathscr{P}^*$, including the randomness. So, whenever $\mathscr{P}^*$ is able to make a convincing argument when in state s, $\mathscr{E}$ can extract a witness. Witness Extended Emulation implies Knowledge Soundness [61].

We will construct public-coin proofs that have special honest-verifier zero-knowledge. This means that if the verifier's challenges are known, or even adversarially chosen, then it is possible to simulate the verifier's view without the witness. In other words, the simulator works for verifiers who may use adversarial coins in choosing their challenges but they follow the specification of the protocol as an honest verifier would.

Definition 8 (Special Honest-Verifier Zero-Knowledge) *The proof of knowledge is* computationally special honest-verifier zero-knowledge (SHVZK) *if there exists a PPT simulator $\mathscr{S}$ such that for all stateful interactive PPT adversaries $\mathscr{A}$ that output (u, w) such that $(\sigma, u, w) \in R$ and randomness ρ for the verifier*

$$\Pr\left[\begin{array}{c}\sigma \leftarrow \mathscr{K}(1^\lambda); (u, w, \rho) \leftarrow \mathscr{A}(\sigma); \\ \mathsf{view}_{\mathscr{V}} \leftarrow \langle \mathscr{P}(\sigma, u, w) \overset{\text{chan}}{\longleftrightarrow} \mathscr{V}(\sigma, u; \rho)\rangle : \mathscr{A}(\mathsf{view}_{\mathscr{V}}) = 1\end{array}\right]$$
$$\approx \Pr\left[\sigma \leftarrow \mathscr{K}(1^\lambda); (u, w, \rho) \leftarrow \mathscr{A}(\sigma); \mathsf{view}_{\mathscr{V}} \leftarrow \mathscr{S}(\sigma, u, \rho) : \mathscr{A}(\mathsf{view}_{\mathscr{V}}) = 1\right].$$

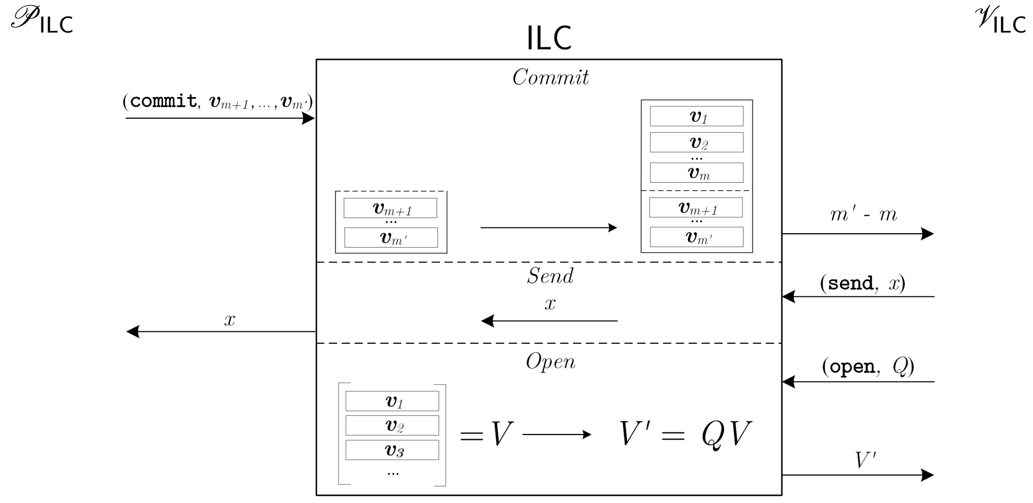
We say the proof is statistically SHVZK *if the definition holds also against unbounded adversaries, and we say the proof is* perfect SHVZK *if the probabilities are exactly equal.*

4.2 Introduction to the Ideal Linear Commitment Model

In this section, we will precisely define Ideal Linear Commitment protocols. We will then compare and contrast this interaction model with other models. Later, in Section 7, we give some simple examples of discrete-logarithm-based protocols and their equivalent protocols in the ILC model.

Ideal Linear Commitment protocols were first described in [16], but never completely formally defined. The definitions given in this section are new, and an important contribution of this thesis. Though the definitions are new, they draw inspiration from the definitions of similar models for information-theoretic proof systems, such as Interactive Oracle Proof protocols [82].

In contrast to standard interactions between a prover and a verifier in IP protocols, we will consider interactions in an *ideal linear commitment* interaction environment, ILC. Figure 4.1 provides some graphical insight into how parties in an ILC proof protocol interact. Our motivation for introducing this new interaction environment was to model zero-knowledge proof protocols using compressing and homomorphic cryptographic commitment schemes.



$\mathscr{P}_{\mathsf{ILC}}$
ILC
$\mathscr{V}_{\mathsf{ILC}}$
Commit
(commit, $v_{m+1}, \dots, v_{m'}$)
v_1
v_2
v_m
v_{m+1}
$v_{m'}$
m' - m
Send
x
(send, x)
Open
(open, Q)
$=V \longrightarrow V' = QV$
v_3
V'

Figure 4.1: Description of the ILC interaction environment. The $\mathbf{v}_i$ here are vectors over a field $\mathbb{F}$, x is a value from $\mathbb{F}$ and V, V' and Q are matrices over $\mathbb{F}$.

As such, when using the ILC channel, the prover can submit a `commit` command to commit to vectors of field elements of some fixed length k, specified in σ_{ILC}. The vectors remain secretly stored, and will not be forwarded to the verifier. Instead, the verifier only learns how many vectors the prover has committed to, and their lengths.

The ILC could be viewed in several possible different ways, as a commitment functionality, communication channel, a trusted third party, or an oracle for the verifier. When Ideal Linear Commitment protocols are compiled into real zero-knowledge protocols, the functionality previously guaranteed by the ILC will be enforced using various cryptographic tools.

The verifier can send single field elements to the prover. The verifier can also submit queries to an `open` oracle for obtaining the opening of any linear combinations of the vectors *of the same length* sent by the prover. We stress that the verifier can request several linear combinations within a single `open` query, as depicted in Figure 4.1.

In addition to the ILC commands and oracles used in the original model [16], we introduce a new `check` oracle. Inside the ILC model, this command behaves in a very similar way to the `open` oracle. However, the two commands will be treated slightly differently when ILC protocols are compiled into real protocols. The reason for making the distinction is that in some of our protocols, the verifier needs to check whether a large vector, that they have computed themselves, is the correct linear combination of vectors committed by the prover. This could be solved by having the verifier make an `send` query for the correct linear combination and checking whether the result is equal to the large vector, which incurs a communication cost for the large vector in the ILC protocol. However, in a real protocol, where the verifier's queries will actually be computed and sent by the prover, the verifier can re-commit to the vector that they have computed, and check this against the prover's commitments. In other words, since the verifier has already computed the vector for themself, there is no need for them to receive it again. Therefore, the `check` oracle is used to distinguish this case, which should not be counted as part of the communication costs of a proof. This issue will be discussed further in later chapters.

4.2.1 Definitions

We will now define general Ideal Linear Commitment protocols, and then specialise to the case of public-coin protocols.

Definition 9 (Ideal Linear Commitment Protocol) *In a μ-round Ideal Linear Commitment protocol, a prover $\mathscr{P}_{\mathsf{ILC}}$ and a verifier $\mathscr{V}_{\mathsf{ILC}}$ will interact with each other via the* ILC*, each sending μ messages. An Ideal Linear Commitment protocol is defined over a field $\mathbb{F}$ and will use a fixed vector length k. Let $(t_1,\dots,t_\mu) \in \mathbb{N}^\mu$ be a tuple of message lengths. There is a setup generator $\mathscr{K}_{\mathsf{ILC}}$ which outputs $\sigma_{\mathsf{ILC}} = (\mathbb{F}, k, (t_1,\dots,t_\mu), \mathsf{aux})$, where* aux *consists of extra elements of $\mathbb{F}$ which might* *be useful when running the protocol.*

The prover and the verifier will run the protocol on given inputs. They will use random coins $\rho_{\mathscr{P}}$ and $\rho_{\mathscr{V}}$, and they will maintain states $\mathsf{state}_i^{\mathscr{P}}$ and $\mathsf{state}_i^{\mathscr{V}}$ that will allow them to remember information over several different rounds of the protocol. The initial state $\mathsf{state}_0^{\mathscr{P}}$ of the prover will be equal to the prover's input. The initial state $\mathsf{state}_0^{\mathscr{V}}$ of the verifier will be equal to the prover's input. Initially, the ILC *will store V_0, which is either a matrix with k columns, or equal to $\perp$. The matrix V_0 corresponds to any commitments that have been made before the protocol begins, that might form part of the prover or verifier's input.*

In every round $i \in \{1, \dots, \mu\}$:

- *The prover takes as input the round number i, the setup information σ_{ILC}, the verifier's previous messages $m_1, \dots, m_{i-1}$, internal state $\mathsf{state}_{i-1}^{\mathscr{P}}$, and randomness $\rho_{\mathscr{P}}$, and outputs a matrix $V_i \in \mathbb{F}^{t_i \times k}$ and a new internal state $\mathsf{state}_i^{\mathscr{P}}$. It commits to V_i using the* `commit` *command.*

- *After round i, the* ILC *stores the vertical concatenation of the matrices $V_0, \dots, V_i$.*

- *The verifier takes as input the round number i, the setup information σ_{ILC}, internal state $\mathsf{state}_{i-1}^{\mathscr{V}}$, and randomness $\rho_{\mathscr{P}}$. It makes linear* `open` *and* `check` *queries on the contents of the* ILC. *Then it outputs a new message $m_i \in \mathbb{F}$ and a new internal state $\mathsf{state}_i^{\mathscr{V}}$.*

- *The verifier's final message m_μ is either* 1*, in which case we say the verifier has accepted, or* 0*, in which case we say that the verifier has rejected.*

We call a proof system over the ILC channel *non-adaptive* if the verifier makes one *open* query and then immediately makes one *check* query to the ILC channel, before terminating his interaction with the channel, and these are the only *open* and *check* queries that the verifier makes. Otherwise we call it *adaptive*. Although adaptive proof systems are allowed, we will only consider non-adaptive ILC proof systems to simplify the exposition. In non-adaptive ILC proof systems with vectors of length k, the verifier will produce two query matrices, Q for the `open` query and Q' for the `check` query.

We give pseudocode descriptions of an Ideal Linear Commitment protocol in Figure 4.2. Here, $\mathscr{V}_{\mathsf{ILC}}^{\mathtt{open}(\cdot),\mathtt{check}(\cdot,\cdot)}$ means that $\mathscr{V}_{\mathsf{ILC}}$ has access to `open` and `check` oracles.

```
An ILC Protocol on $\sigma_{\mathsf{ILC}}, \rho_{\mathscr{P}}, \rho_{\mathscr{V}}, \mathsf{state}_0^{\mathscr{P}}, \mathsf{state}_0^{\mathscr{V}}$

• Initialise protocol.:
    – Set $M = V_0$.
• For $i = 1$ to $\mu$:
•     Run prover algorithm:
    – $(V_i, \mathsf{state}_i^{\mathscr{P}}) \leftarrow \mathscr{P}_{\mathsf{ILC}}(i, \{m_j\}_{j<i}, \mathsf{state}_{i-1}^{\mathscr{P}}, \rho_{\mathscr{P}})$
    – commit$(V_i)$
•     Run verifier algorithm:
    – $(m_i, \mathsf{state}_i^{\mathscr{V}}) \leftarrow \mathscr{V}_{\mathsf{ILC}}^{\mathtt{open}(\cdot),\mathtt{check}(\cdot,\cdot)}(i, \mathsf{state}_{i-1}^{\mathscr{V}}, \rho_{\mathscr{V}})$
• Accept or Reject
```

```
commit$(V)$

• If $M = \perp$ then set $M = V$.
• If $V = \perp$ then do nothing.
• Otherwise, update $M$ by vertically concatenating it with $V$, with $V$ at the bottom.
```

```
open$(Q)$

• If $M = \perp$ then output $\perp$.
• Output $QM$.
```

```
check$(Q', V')$

• If $M = \perp$ then output $\perp$.
• If $Q'M = V'$ then output $\top$.
• Otherwise output $\perp$.
```

Figure 4.2: Description of how the parties in an ILC protocol interact.

The security properties of ILC proof protocols, including the completeness, soundness, and zero-knowledge properties, are defined in exactly the same way as for normal zero-knowledge protocols.

4.2.2 Complexity Measures

The first useful complexity measure for an ILC protocol is the vector length k. We have already mentioned μ, the number of rounds of an ILC protocol. This is another useful complexity measure.

Set $t = \sum_{i=1}^{\mu} t_i$. This is the *total number of vectors* that the prover commits to as part of the ILC protocol. Later one, we will see that this can impact the communication complexity and computational complexity of real protocols based on ILC protocols.

We define qc to be the `open` query complexity, i.e. the number of linear combinations that the verifier obtains using the `open` oracle. Every time the verifier makes an `open` query on a matrix $Q \in \mathbb{F}^{l} \times t$, we consider this to be l queries for the purpose of measuring the query complexity. Then, qc is the total number of rows from all matrices Q on which the verifier has called the `open` oracle. Similarly, we define qc$'$ to be the `check` query complexity, i.e. the number of times that the verifier queries certain linear combinations using the `check` oracle.

4.2.3 Public Coin Protocols

As with zero-knowledge protocols, we say that an Ideal Linear Commitment protocol is *public coin* if the following conditions hold.

1. *Except for the final message m_μ*, the verifier's messages to the communication channel are chosen uniformly at random from $\mathbb{F}$ and independently of the actions of the prover, i.e., the verifier's messages m_i correspond to the verifier's randomness $\rho_{\mathcal{V}}$.

2. The query matrices Q and Q' in the verifier's `open` and `check` queries are determined only by the values of $m_1, \ldots, m_i$ that have appeared in the protocol so far. The query matrices $\mathcal{V}'$ in the verifier's `check` queries are determined only by the values of $m_1, \ldots, m_i$ that have appeared in the protocol so far and by the values of `open` query responses that the verifier has seen for far. In particular, this means that the prover can actually antipicate the queries that will be made, because the verifier never uses information that the prover doesn't know when making queries.

3. The verifier's final message m_μ is determined by the messages $m_1, \ldots, m_{\mu-1}$ and the results of the queries made so far.

In this case, we are able to make several simplifications. Since for $i < \mu$ the verifier's messages m_i only depend on $\rho_{\mathcal{V}}$, all `open` and `check` queries can be deferred until the end of the protocol, since they will not affect the messages in any way. In fact, the verifier will only need to make one `open` query and then one `check` query.

Similarly, the state of the verifier will not affect any of the messages m_i for $i < \mu$, so the verifier need not update state their state either, apart from possibly producing a final state $\mathsf{state}_\mu^{\mathcal{V}}$ after making the `open` and `check` queries.

4.2.4 Remarks on the Interaction Model

Remark. We have presented an interaction environment using vectors of fixed length k. In fact, in our protocols, we will allow the prover to commit to vectors of *several different fixed lengths* $k_1, \ldots, k_r$, which will be specified at the beginning of the protocol. This is easily formalised by defining a new interaction environment, giving the prover and verifier access to r copies of different `commit`, `open` and `check` commands and oracles using vectors of lengths $k_1, \ldots, k_r$. Such a channel will be referred to as an ILC channel for vectors of several fixed lengths, and will be specified simply by including multiple vector lengths in σ_{ILC} rather than just one. The resulting interaction environment is shown in Figure 4.3.

Alternatively, we can easily incorporate vectors of different lengths into the model by padding all vectors with zeroes until they are the same length as the longest vector. This will have no impact on the asymptotic efficiency of our protocols. Some of our ILC protocols require single values to be committed as well as vectors.

In this case, the matrices $V_{0,j}$, which correspond to commitments to data made before the protocol begins, each have k_j columns, or maybe be equal to $\perp$.

Remark. It is of course easy to generalise to the case where the verifier's messages are not single field elements. We use the case of single field elements for notational simplicity and because it suffices for all of our protocols.

Remark. We will assume that all parties output messages which parse correctly. In other words, no party or functionality will ever deviate from the protocol using messages which are from the wrong domain, or of incorrect length. This includes the verifier making `open` and `check` queries Q such that matrices Q and M cannot be multipled together, or such that $Q'V$ and V' are of different dimensions. This will not be a problem for security. Indeed, provided that checking the format of a message can be done efficiently, it would be simple, but tedious, to add such checks at the beginning of every algorithm, and instruct all algorithms to abort if the checks are

An ILC Protocol on $\sigma_{\mathsf{ILC}}, \rho_{\mathscr{P}}, \rho_{\mathscr{V}}, \mathsf{state}_0^{\mathscr{P}}, \mathsf{state}_0^{\mathscr{V}}$

- **Initialise protocol.**:
  - Parse $\sigma_{\mathsf{ILC}} = (\mathbb{F}, k_1, \dots, k_r)$.
  - Set $M_j = V_{0,j}$ for each $j \in [r]$.
- **For** $i = 1$ **to** μ:
- **Run prover algorithm**:
  - $(V_{i,1}, \dots, V_{i,r}, \mathsf{state}_i^{\mathscr{P}}) \leftarrow \mathscr{P}_{\mathsf{ILC}}(i, \{m_j\}_{j<i}, \mathsf{state}_{i-1}^{\mathscr{P}}, \rho_{\mathscr{P}})$
  - $\texttt{commit}(k_j, V_{i,j})$ for each $j \in [r]$.
- **Run verifier algorithm**:
  - $(m_i, \mathsf{state}_i^{\mathscr{V}}) \leftarrow \mathscr{V}_{\mathsf{ILC}}^{\texttt{open}(\cdot,\cdot),\texttt{check}(\cdot,\cdot,\cdot)}(i, \mathsf{state}_{i-1}^{\mathscr{V}}, \rho_{\mathscr{V}})$
- **Accept or Reject**

$\texttt{commit}(k_i, V)$

- If $M_i = \perp$ then set $M_i = V$.
- If $V = \perp$ then do nothing.
- Otherwise, update M_i by vertically concatenating it with V, with V at the bottom.

$\texttt{open}(k_i, Q)$

- If $M_i = \perp$ then output $\perp$.
- Output QM_i.

$\texttt{check}(k_i, Q', V')$

- If $M_i = \perp$ then output $\perp$.
- If $Q'M_i = V'$ then output $\top$.
- Otherwise output $\perp$.

Figure 4.3: Description of how the parties interact in an ILC protocol with several vector lengths.

failed. We assume that this is the case in all of our compiled protocols.

Remark. If we had followed the original model of [16], we might have allowed sequences of `commit`, `send` and `open` and `check` queries to be performed in an arbitrary order. Our definition does not incur a loss in generality by having the prover and verifier alternate in the protocol, as valid protocols in the original model could be recovered by having rounds in which either the prover does nothing or the verifier does not use certain oracles. Rather, our definition corresponds to a sensible reordering of such protocols. For example, the verifier gains no power by being able to make an `open` query directly after sending a message to the prover, but before the prover has committed to any new vectors. The verifier might just as well compute their message without sending it, make their query, and then send the message.

Remark. The verifier also produces a state in their final round, in addition to their final message $m_\mu \in \{0,1\}$ which signifies whether they want to accept or reject. This serves a useful purpose. In Section 8.2, we define a polynomial commitment scheme in the ILC model. At the end of the protocol, the verifier is able to compute the evaluation of a committed polynomial and then store it in their final state. In this way, the polynomial commitment scheme can be used as a subprotocol in other ILC protocols, as the evaluation can be passed around as part of the final state.

Remark. We have formally defined how the prover and verifier interact as part of an ILC protocol, and how the prover receives the verifier's computed outputs m_i, for example. Later on, when designing protocols, we will be slightly less formal, and refer, for example, to the verifier sending m_i to the prover. We will also suppress the vector $(t_1, \ldots, t_\mu)$ from our descriptions of σ_{ILC} and protocols, and assume that both the prover and verifier know the message schedule of the protocol.

4.2.5 Comparison with Other Models and Types of Protocol

Interactive Oracle Proofs. Interactive Oracle Proofs (IOPs) were introduced in [82]. They were designed to simultaneously generalise the interactive protocols of the class IP and probabilistically-checkable proofs (PCPs). IOP protocols share a similar syntax and pattern of interaction to ILC protocols, but the most significant difference is in the verifier's query oracles. In IOP protocols, the prover sends functions to the verifier, and the verifier may make queries to learn the values of these functions at certain points. If one views a function as a long string which is the concatenation of all possible outputs of the function, then one can imagine the verifier making pointwise queries on the string. This naturally leads to the idea of instantiating IOP protocols using commitments based on Merkle trees [17] which can be efficiently opened to individual positions of a committed string.

Thus, the key difference between IOP and ILC protocols is that IOP protocols use *pointwise* queries rather than linear queries. This difference leads to another difference in the interaction environment. In IOP protocols, the prover commits to a different function or string in each round, and the verifier has separate oracle access to each string committed so far. On the other hand, the ILC stores a matrix M which is updated using the prover's `commit` commands, accumulating all committed vectors into a single matrix which the verifier can make queries on, rather than allowing the verifier to make queries on each matrix separately. This clearly leads to more powerful protocols when linear queries are involved; as the verifier can, for example, make a query and obtain a vector $\mathbf{a}x + \mathbf{b}$, where $\mathbf{a}$ and $\mathbf{b}$ are from different committed matrices, without learning the value of either $\mathbf{a}$ or $\mathbf{b}$.

Linear Interactive Proofs. Linear Interactive Proofs (LIPs) were introduced in [72]. In linear interactive proofs both the prover and verifier send vectors of field elements, but the prover can only send linear (or affine) transformations of the verifier's previously sent vectors. This is a good model for certain types of zero-knowledge protocol which use a trusted setup. In such cases, the trusted setup consists of various messages, encoded, for example, inside elements from a finite group or suitable homomorphic encryption scheme or commitment scheme. The prover can manipulate the messages in a linear fashion, via the group elements or ciphertexts, but cannot produce new encodings by any other method. An example of a zero-knowledge proof that can be modelled in this way is given in [34].

Both models use vectors of fixed lengths over fields. However, for LIPs, it is the prover who is restricted to certain types of linear computations, as the inspiration for this model came from proofs where the *prover* manipulates homomorphic commitments. For our constructions it is important that the prover can compute on field elements received by the verifier and for instance evaluate polynomials, while it is the *verifier* who is restricted to linear queries, as the inspiration for our model came from proofs where the verifier manipulates homomorphic commitments.

(Fully) Linear PCPs and (Fully) Linear IOPs Linear PCPs were introduced in [72] as a further abstraction of arguments lying behind some LIPs. They were generalised to fully-linear PCPs in [83]. Just as PCPs were generalised by IOPs, the same paper, [83] goes on to generalise LPCPs further to (fully-)linear IOPs.

All of these formal models are very closely related to the ILC model. In these models, the prover produces proof strings defined over some field, and the verifier is allowed to make linear queries on the proof strings. The prover produces one proof string in an LPCP protocol, whereas in a LIOP protocol, the prover produces several proof strings over several rounds of interaction with the verifier. In the fully-linear case, the verifier is further restricted so that in addition, they can only make linear queries on the instance u.

The differences between ILC protocols and LIOP protocols are largely motivated by different efficiency metrics. Intuitively, a LIOP protocol behaves like an ILC protocol where the fixed vector length k is equal to 1, and both types of protocol can be converted into the other, although the resulting protocols may suffer unusual parameter choices. For example, an ILC protocol with $k = 1$ is possible, but when converted into a real protocol using a highly-compressing homomorphic commitment scheme, it is easy to imagine that we create a more efficient protocol by committing to more than one element at once.

LPCP and LIOP protocols are suitable for modelling zero-knowledge proofs where the prover uses non-compressing methods to hide their messages, perfectly binding homomorphic commitment schemes, or homomorphic encryption schemes. The 'fully-linear' abstraction captures the case where the instance u also relates to encrypted or committed data, for example, and the prover wishes to give a proof about the data.

Chapter 5

Cryptographic Assumptions and Concrete Commitment Schemes

Zero-knowledge protocols are often built based on cryptographic assumptions stating that various mathematical problems cannot be solved efficiently. In this thesis, we show that efficient zero-knowledge proofs and arguments can be built easily based on two different cryptographic assumptions: the discrete logarithm assumption, and the existence of collision resistant hash functions. Here, we formally define both of these assumptions.

Note that our compilation proofs are not limited to the use of the commitment schemes in this chapter. The Pedersen commitment scheme could be replaced by any homomorphic commitment scheme over a field in our compilation in Section 9.2. Similarly, we would like to emphasise that we have specified a commitment scheme based on collision-resistant hash-functions in this chapter purely for concreteness, and the compilation procedure in Section 9.1 could be based on an arbitrary string commitment scheme.

We define cryptographic assumptions relative to instance generators $\mathscr{G}$.

5.1 The Discrete Logarithm Assumption

The discrete logarithm assumption is a computational hardness assumption relative to a given group $\mathbb{G}$. Given two group elements g and h in $\mathbb{G}$, the discrete logarithm problem is to compute an integer k such that $g^k = h$ in $\mathbb{G}$.

The discrete logarithm problem has been extensively studied. It forms the basis for cryptosystems and commitments, such as the ElGamal encryption scheme and the Pedersen commitment scheme. Researchers have also devised algorithms to solve the discrete logarithm problem, some of which work for any group, and some of which work for particular choices of group.

Formal Definition Let $\mathscr{G}$ be a probabilistic, polynomial time algorithm that on input 1^λ returns a description $\mathsf{gk} = (\mathbb{G}, p, g)$ of a group $\mathbb{G}$ of prime order p, with a generator g. Assume that the group has associated polynomial time algorithms for computing group operations and deciding membership.

We say that the discrete logarithm assumption holds relative to $\mathscr{G}$ if for all probabilistic polynomial time adversaries $\mathscr{A}$ and all $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathsf{gk} = (\mathbb{G}, p, g) \leftarrow \mathscr{G}(1^\lambda); x \leftarrow \mathbb{Z}_p; h := g^x : \mathscr{A}(\mathsf{gk}, h) = x\right] \approx 0$$

Commitment Scheme The Pedersen commitment scheme [117] is a well-known commitment scheme based on the discrete logarithm assumption. The original commitment scheme allows a committer to commit to a single element, but is easily generalised to a commitment scheme for multiple elements, shown in Figure 5.1. Here, the message space is $\mathbb{Z}_p^n$ and the randomness space is $\mathbb{Z}_p$. Commitments lie in $\mathbb{G}$. The binding property comes from the discrete logarithm assumption, and the hiding property from the fact that r is chosen uniformly at random, so that commitments are distributed as random group elements.

Perhaps suprisingly, a Pedersen commitment on a message of length n can be computed using $O(n/\log n)$ group operations rather than the $O(n)$ operations that one might expect, using Pippenger's algorithm. A special case of the algorithm- which is sufficient to obtain our results- is explained in Appendix C.

Pedersen commitments are an attractive choice because they are both succinct, and simple and easy to compute using exponentiation operations on group elements. Our ILC protocols use only arithmetic operations over fields. When instantiated using Pedersen commitments, although the protocols are more complicated, the types of operations required to compute the resulting zero-knowledge proofs are

Figure 5.1: Pedersen Commitments

```
1: KeyGen $(1^\lambda, n)$:
2: $\mathsf{gk} = (\mathbb{G}, p, g) \leftarrow \mathscr{G}(1^\lambda)$
3: Select $\mathbf{g} \leftarrow \mathbb{G}^{n+1}$
4: Return $(\mathsf{gk}, \mathbf{g})$
```

```
1: Commit_ck $(\mathbf{m}; r)$:
2: Parse $(m_1, \ldots, m_n) = \mathbf{m}$
3: Parse $((\mathbb{G}, p, g), (g_1, \ldots, g_n, h)) = \mathsf{ck}$
4: Return $h^r \prod_{i=1}^{n} g_i^{m_i}$
```

fundamentally the same as those required to compute DSA or ECDSA signatures, with the addition of well-known Fast Fourier transform algorithms to perform some polynomial operations. This should make it extremely easy to produce efficient implementations of our ILC protocols when they are compiled into the discrete logarithm setting.

Parameter Choices The discrete logarithm problem is believed to be difficult when $\mathbb{G}$ is chosen to be a subgroup of the multiplicative group of a prime field, or the group of points of an elliptic curve over a finite field, with large prime order. In order to achieve λ bits of security, different sizes of group are required in each case. Over prime fields, the best known algorithms are based on the general number field sieve [118], and in light of these algorithms, a prime of roughly $\mathscr{O}(\lambda^3)$ bits is required for λ bits of security. For most elliptic curves, except those from a few special families, the best algorithms are generic algorithms such as Pollard's rho algorithm, and the order of $\mathbb{G}$ should be approximately 2λ bits.

The problem of breaking the binding property of the Pedersen commitment scheme can be reduced to the discrete logarithm problem.

The discrete logarithm assumption is well-known, well-examined, and widely used in cryptography. Our protocols rely on the discrete logarithm assumption in groups with prime order p. The assumption is believed to hold in suitable subgroups of elliptic-curve groups. The best algorithms for finding discrete logarithms in such elliptic curve groups are still generic algorithms with complexity $\Omega(\sqrt{p})$. For these groups we therefore enjoy lower parameter sizes than protocols based on RSA groups that are subject to sub-exponential attacks.

The discrete logarithm assumption is also believed to hold in well-chosen multiplicative sub-groups of finite fields. Finite fields of prime order should have moduli of $\frac{\lambda^3}{\text{polylog}\lambda}$ bits in order to achieve λ bits of security against the best known attacks. This makes protocols communicating large numbers of group elements highly impractical in this setting. Some of our protocols can be tuned so that they only require a constant number of group elements, resulting in much better efficiency when instantiated in finite fields of prime order, since the $\frac{\lambda^3}{\text{polylog}\lambda}$ communication cost can then appear as a constant additive factor rather than a multiplicative one.

5.2 The Collision Problem for Hash Functions

Hash functions are functions which produce a short digest, or hash, of fixed length when applied to an input of arbitrary size. When using hash functions for cryptographic security guarantees, a common requirement is collision resistance. The computational problem here is to generate two different inputs which give the same hash value.

Like Pedersen commitments, hash-based commitments can be extremely simple and efficient to compute. But Pedersen commitments are based on the hardness of the discrete logarithm assumption, and so we expect that a quantum computer running Shor's algorithm would be able to break the binding property of Pedersen commitments, and thus the soundness property of our ILC protocols when compiled in the discrete logarithm setting. However, there is currently no efficient quantum algorithm for breaking the collision-resistance of hash-functions much faster than classical algorithms.

Recent works, most notably [119], argue that when made into non-interactive zero-knowledge proofs, certain zero-knowledge proofs are secure in strong threat models against quantum adversaries who are able to perform quantum interactions with the prover or the verifier. Making such arguments about our own protocols is beyond the scope of this thesis, but as they are based on a cryptographic assumption which still resists cryptanalytic attempts by quantum algorithms, it is possible that they could be proved secure against quantum adversaries in the future, or inspire other protocols with quantum security proofs.

Formal Definition. Collision resistance is properly defined for a family of functions. Let $m, l : \mathbb{N} \to \mathbb{N}$ be increasing functions such that $m(\lambda) > l(\lambda)$ for all $\lambda \in \mathbb{N}$. Let $\mathscr{G}$ be a probabilistic, polynomial time algorithm that on input 1^λ returns a key s. Then suppose that there exists a DPT algorithm h such that for every λ and every s, h defines a function $h : \{0,1\}^{m(\lambda)} \to \{0,1\}^{l(\lambda)}$. Then $(\mathscr{G}, h)$ is a fixed-length hash function.

We say that $(\mathscr{G}, h)$ is a collision resistant hash function if for all probabilistic polynomial time adversaries $\mathscr{A}$ and all $\lambda \in \mathbb{N}$,

$$\Pr\left[s \leftarrow \mathscr{G}(1^\lambda) : \mathscr{A}(s) = (x, x'),\ x, x' \in \{0,1\}^{m(\lambda)}, h^s(x) = h^s(x')\right] \approx 0$$

Commitment Scheme. Halevi and Micali [120] show that a collision-resistant hash function gives rise to an efficient and compact statistically hiding commitment scheme. In the following, we tacitly assume that $m(\lambda)$ is large enough to create a functional commitment scheme. If this is not the case, one can use the Merkle-Damgaard construction or alternatives [121, 122] used to extend the hash function to a larger input length.

Let $T_{m,n}$ be the space of $m \times n$ Toeplitz matrices over $\mathbb{Z}_2$. This is the collection of matrices such that for each top-left to bottom-right diagonal, every element in the diagonal is equal. Note that A can be described using $m+n-1$ bits.

Figure 5.2 gives a commitment scheme with message space $\{0,1\}^m$, randomness space $\{0,1\}^{O(l)}$ and commitment space $\{0,1\}^{O(l)}$.

Parameter Choices The best known generic attack on collision resistant hash functions is the birthday attack, which shows that a collision can be found in $O(2^{l/2})$ operations. This is a square-root of the cost of the simplest possible attack, which simply hashes every possible message until a collision is found, and uses $O(2^l)$ operations in the worst case.

Figure 5.2: Hash-Based Commitments

```
1: **KeyGen** $(1^\lambda, m)$:
2: Set $L = 6\lambda + 4$.
3: $s \leftarrow \mathscr{G}(1^\lambda)$.
4: **Return** $(h, s, L)$
```

```
1: **Commit**$_{\mathsf{ck}}$ $(\mathbf{m}; r)$:
2: Parse $\mathbf{m}$ as a bitstring.
3: Parse $\mathbf{r} \in \{0,1\}^L$.
4: Compute $\mathbf{m}' = h^s(\mathbf{m})$.
5: Select $A \leftarrow T_{l,L}$.
6: Compute $\mathbf{b} = \mathbf{m}' - A\mathbf{r}$.
7: Compute $y = h^s(\mathbf{r})$.
8: **Return** $(A, \mathbf{b}, y)$
```

Chapter 6

Lemmata for Security Proofs

In this chapter, we present a variety of useful lemmata which will be used to prove that the ILC protocols in Chapter 8 are secure, and to prove that those protocols have a few useful additional properties for the purpose of compiling them into discrete-logarithm-based protocols in Chapter 9.

These results are presented here together because they are technical in nature. They are useful for proving the security of the protocols but unnecessary for understanding the flow of any of the protocols. In particular, Lemma 3 and Lemma 4 prevent a good deal of repetition in the proofs of the properties given after each ILC protocol.

6.1 Variant of the Schwarz-Zippel Lemma

This lemma is a non-standard variant of the Schwarz-Zippel Lemma, and was proved for the first time by the author of this thesis in [16]. The role of this lemma is to help prove the soundness of the ILC protocol in Section 8.5, which uses two random challenges x and y. When $d' = 0$, we also recover the original Schwarz-Zippel lemma which is useful for proving the soundness of the other protocols in Chapter 8.

Intuitively, in protocols with two random challenges represented by X and Y, where Y was produced after X, the coefficients of $Q(X)$ correspond to values that the prover committed before having seen X or Y, and the functions $R_j(X)$ correspond to values that the prover committed to after having seen X, but before having seen Y. This is why the $R_j(X)$ are allowed to be arbitrary functions of X. As in the example in Section 7.2, the coefficients of $P(X,Y)$ will all be zero if the prover has behaved honestly, and the verifier checks an equality of polynomials evaluated at random points. If $P(X,Y)$ is not identically zero, then we would like to know the probability that the prover succeeds if $P(X,Y)$ nevertheless evaluates to zero. The lemma provides an upper bound on the probability that a cheating prover will fail to be caught out during this process.

A Laurent polynomial is a polynomial which also has terms with negative powers in some variable. For example, a Laurent polynomial in one formal variable X with coefficients in the field $\mathbb{F}$ is an expression of the form $\sum_{i \in I} a_i X^i$ where I is a finite set of integers (some of which may be negative) and a_i is an element of $\mathbb{F}$ for each $i \in I$.

Lemma 1 *Let $\mathbb{F}$ be a field. Let P be a function of the following form, where $Q(X)$ is a Laurent polynomial of total degree at most d and $R_j(X)$ are* arbitrary functions *and not necessarily polynomials.*

$$P(X,Y) = Q(X) + \sum_{j=-d',j\neq 0}^{d'} R_j(X)\, Y^j$$

Let S be a finite subset of $\mathbb{F}^{\times}$. Let x, and y be selected at random independently and uniformly from S. Let F be the event that either $Q(X)$ is not the zero polynomial or one of the values of $R_j(x)$ is not zero. Then

$$\Pr\left[\{P(x,y) = 0\} \wedge F\right] \leq \frac{(d + 2d' + 1)}{|S|}$$

Remark This lemma can be extensively generalised to functions with more variables, and used to prove the security of ILC protocols with a larger number of random challenges. See in particular [16].

Proof Assume that the result holds for $u-1$. We prove the lemma for u. Write

$$P(X,Y) = Q(X) + \sum_{j=-d',j\neq 0}^{d'} R_j(X)\, Y^j$$

For a fixed value x of X, this is a Laurent polynomial of total degree $2d'+1$ in Y. Let G be the event that P is the zero polynomial in Y. Let F' be the event that Q is not the zero polynomial.

$$\begin{aligned}\Pr\left[\{P(x,y)=0\}\wedge F\right] &= \Pr\left[\{P(x,y)=0\}\wedge F\wedge G\right]\\ &\quad+\Pr\left[\{P(x,y)=0\}\wedge F\wedge\neg G\right]\end{aligned}$$

If G holds, then P is the zero polynomial in Z_u, so since $Q(x)$ is the constant term, then $Q(x)$ is necessarily zero. On the other hand, if G and F hold simultaneously, then each value $R_j(x)$ must be zero, so Q must be a non-zero polynomial. Therefore, F' holds. We use these facts to bound the first probability. We bound the second probability by simply removing the event F.

$$\begin{aligned}\Pr\left[\{P(x,y)=0\}\wedge F\right] &\leq \Pr\left[\{Q(x)=0\}\wedge F'\right]\\ &\quad+\Pr\left[\{P(x,y)=0\}\wedge\neg G\right]\end{aligned}$$

The first probability is bounded by the fact that $Q(X)$ is a polynomial of degree d and has at most d roots. To bound the second probability, observe that for any value of x such that $\neg G$ holds, P is a non-zero Laurent polynomial of degree at most $2d'+1$ in Y, and has at most $2d'+1$ roots y. The result follows. ■

6.2 A general forking lemma.

In this section, we prove a general forking lemma which connects two security notions for interactive protocols. One type of notion includes properties such as *special soundness*, where one can use a collection of accepting protocol transcripts in the correct format to compute a witness for the protocol. The other includes notions such as knowledge soundness and witness-extended emulation, where a knowledge extractor with rewindable black-box access to the prover and verifier must compute a witness. This forking lemma connects the two by showing that the knowledge extractor can efficiently sample the collection of transcripts required to compute a witness. This lemma was first proved in [9].

Suppose that we have a $(2\mu+1)$-move public-coin argument with μ challenges, $x_1,\ldots,x_\mu$ in sequence. Let $n_i \geq 1$ for $1 \leq i \leq \mu$. Consider $\prod_{i=1}^{\mu} n_i$ accepting transcripts with challenges in the following tree format. The tree has depth μ and $\prod_{i=1}^{\mu} n_i$ leaves. The root of the tree is labelled with the statement. Each node of depth $i < \mu$ has exactly n_i children, each labelled with a distinct value for the ith challenge x_i. This can be referred to as an $(n_1,\ldots,n_\mu)$-tree of accepting transcripts.

Definition 10 ($(n_1,\ldots,n_\mu)$-tree special soundness) *Let $(\mathcal{G},\mathcal{P},\mathcal{V})$ be an interactive argument. Let $n_1(\lambda),\ldots,n_\mu(\lambda) \in \mathbb{N}$ such that $\prod_{i=1}^{\mu} n_i$ is bounded above by a polynomial in the security parameter λ. Suppose that there exists a DPT algorithm χ such that given $\mathsf{view}_{\mathcal{V}}$ for each transcript of an $(n_1,\ldots,n_\mu)$-tree of accepting transcripts as input, χ outputs a valid witness w for the statement at the root of the tree. Then we say that $(\mathcal{G},\mathcal{P},\mathcal{V})$ has $(n_1,\ldots,n_\mu)$-tree special soundness.*

All of our arguments allow a witness to be extracted efficiently from an appropriate tree of accepting transcripts. This lemma therefore shows that they all have knowledge soundness. This is a natural generalisation of special-soundness for Sigma-protocols, where $\mu = 1$ and $n = 2$. For simplicity in the following lemma, we assume that the challenges are chosen uniformly from a field $\mathbb{F}$ where $|\mathbb{F}| = 2^{\lambda}$, but any sufficiently large challenge space would suffice.

Lemma 2 (Forking Lemma) *Let $(\mathcal{P},\mathcal{V})$ be a $(2\mu+1)$-move, public coin interactive protocol with $(n_1,\ldots,n_\mu)$-tree soundness. Then $(\mathcal{P},\mathcal{V})$ has witness-extended emulation.*

For simplicity in the following proof, we assume challenges are chosen uniformly from $\mathbb{Z}_p$ where $|p| = \lambda$, but any sufficiently large challenge space would suffice.

We now provide a proof of the Forking Lemma we need for Theorems 7 and 11. For simplicity in the following lemma, we assume that the challenges are chosen uniformly from $\mathbb{Z}_p$ where $|p| = \lambda$, but any sufficiently large challenge space would suffice.

Proof Suppose that for deterministic polynomial time $\mathscr{P}^*$ there is a polynomial time interactive adversary $\mathscr{A}$ in the sense of witness-extended emulation, such that

$$\Pr\left[(u,s) \leftarrow \mathscr{A}(1^\lambda); tr \leftarrow \langle \mathscr{P}^*(u,s), \mathscr{V}(u)\rangle : \mathscr{A}(tr) = 1\right] = \varepsilon.$$

Note that if ε is negligible, then we do not need to extract a witness, since the emulator can simply fail every time and trivially achieve witness-extended emulation. Therefore, from now on, we assume that ε is not negligible.

We construct an expected polynomial time emulator $\mathscr{E}$, which has access to a rewindable transcript oracle $\langle \mathscr{P}^*, \mathscr{V}\rangle$ and produces a witness. This is done via recursive calls to tree-finders $\mathscr{T}$ that deal with the protocol after the first few challenges are already fixed. The *i*th tree-finder takes the previous challenges and partial transcript given to it as input, picks random values for x_{i+1}, runs the prover on these values and hands the result to the next tree-finder. Each tree-finder may fail on the first value of x_{i+1}, ensuring that the whole process runs in expected polynomial time. With overwhelming probability, the emulator obtains an $(n_1, \ldots, n_\mu)$-tree of transcripts and is then able to extract a witness, using the efficient algorithm χ that exists by assumption.

$\mathscr{E}^{\langle \mathscr{P}^*, \mathscr{V}\rangle}(u) \rightarrow (tr, w)$:

- Run $\mathscr{T}^{\langle \mathscr{P}^*, \mathscr{V}\rangle}(1) \rightarrow (tr, \mathsf{tree})$
- If $\mathsf{tree} = \perp$ then return $(tr, \perp)$.
- If tree is not a valid $(n_1, \ldots, n_\mu)$-tree of transcripts (i.e. there are collisions in certain challenges) then return $(tr, \perp)$.
- Else run $w \leftarrow \chi(u, \mathsf{tree})$.
- Return (tr, w)

For $1 \leq i \leq \mu + 1$:

$\mathscr{T}^{\langle \mathscr{P}^*, \mathscr{V}\rangle}(i) \rightarrow (tr, \mathsf{tree})$:

- If $i = \mu + 1$

```
•     Obtain a complete protocol transcript from $tr \leftarrow \langle \mathscr{P}^*, \mathscr{V} \rangle$
•     Run $\mathscr{V}(tr) \rightarrow b$
•     If $b = 0$ then return $(tr, \perp)$.
•     If $b = 1$ then set $\mathsf{tree} = \{tr\}$ and return $(tr, \mathsf{tree})$.
• Run $\langle \mathscr{P}^*, \mathscr{V} \rangle$ up to and including move $2i+1$.
• Run $\mathscr{T}^{\langle \mathscr{P}^*, \mathscr{V} \rangle}(i+1) \rightarrow (tr, \mathsf{tree})$
• If $\mathsf{tree} = \perp$ then return $(tr, \perp)$.
• Set $\mathsf{counter} = 1$
• While $\mathsf{counter} < n_i$:
•     Rewind $\langle \mathscr{P}^*, \mathscr{V} \rangle$ back until just before move $2i$.
•     Run $\mathscr{T}^{\langle \mathscr{P}^*, \mathscr{V} \rangle}(i+1) \rightarrow (tr', \mathsf{tree}')$
•     If $\mathsf{tree} \neq \perp$, then append the transcripts in $\mathsf{tree}'$ to $\mathsf{tree}$, and increment $\mathsf{counter}$.
• Return $(tr, \mathsf{tree})$
```

Fix $1 \leq i \leq \mu$, and fix $x_1, \ldots, x_{i-1}$. We say that $\mathscr{T}(i)$ has failed if it returns $(tr, \perp)$.

Let ε' be the probability that $\mathscr{T}(i)$ fails for this choice of challenges, and let $\varepsilon'(x_i)$ be the probability that $\mathscr{T}(i+1)$ fails for this choice of challenges continued with x_i. The ith tree-finder can fail only if the $(i+1)$th tree-finder fails the first time it is called. This implies that for uniformly random x_i, the probability that $\mathscr{T}(i+1)$ fails is $\varepsilon' = \sum_{x_i \in \mathbb{F}} \Pr[X = x_i]\varepsilon'(x_i)$.

Therefore, the expected number of times that $\mathscr{T}(i)$ runs $\mathscr{T}(i+1)$ is $1 + \varepsilon' \frac{(n_i - 1)}{\varepsilon'} = n_i$. The final tree-finder $\mathscr{T}(k+1)$ merely checks whether the transcript is accepting or not. Hence, the total expected running time for $\mathscr{T}(1)$ to be $\prod_{i=1}^{\mu} n_i$ multiplied by the time taken to check whether a transcript is accepting. We conclude that the emulator $\mathscr{E}$ runs in expected polynomial time.

The first tree-finder $\mathscr{T}(1)$ only outputs $(tr, \perp)$ if the very first set of challenges generated by all of the emulators fails to produce an accepting transcript. This is exactly the probability that $\mathscr{P}^*$ successfully produces an accepting transcript in one run.

Given that we receive $\prod_{i=1}^{\mu} n_i$ accepting transcripts in tree, we now consider the probability that they do not form an $(n_1,\dots,n_\mu)$-tree. This occurs only when the n_i values of challenge x_i used by $\langle \mathscr{P}^*, \mathscr{V} \rangle$ while in the loop controlled by counter are not distinct, or in other words, there is a collision between these values, for some i.

By Markov's inequality, an algorithm whose expected running time is t will only run for longer than time $T > t$ with probability $\frac{t}{T}$. Let t be the expected running time of $\mathscr{E}$, which is bounded above by a polynomial in the security parameter. For easier analysis, we limit the actual running time of $\mathscr{E}$ to T, whose value will be chosen later.

When $\mathscr{E}$ runs in time at most T, then at most T uniformly random public coin challenges were selected by $\mathscr{V}$ in $\langle \mathscr{P}^*, \mathscr{V} \rangle$. If there are no collisions between *any* of the public coins chosen, then there are certainly no collisions of the type which would prevent tree from being a $(n_1,\dots,n_\mu)$-tree of transcripts. The probability that there is a collision between T values sampled uniformly from $\mathbb{F}$ is at most $\frac{T^2}{p}$.

Now, we choose $T = \sqrt[3]{p}$. The probability that tree fails to be an $(n_1,\dots,n_\mu)$-tree is at most $\frac{t}{T} + \frac{T^2}{p}$ which is now equal to $\frac{t}{\sqrt[3]{p}} + \frac{1}{\sqrt[3]{p}}$. This is negligible. Therefore, there is negligible probability of the tree-finding algorithms succeeding, yet $\mathscr{E}$ failing to extract a witness. This proves the argument has statistical witness-extended emulation. ■

6.3 Extractable Algebraic Queries

The following lemma shows that matrices from a certain family are always invertible. This fact is used to prove a further lemma which will be useful for arguing about the security of ILC protocols that have been compiled into discrete logarithm protocols.

Lemma 3 *Let $\mathbf{p} = (p_1(X),\dots,p_k(X))$ be linearly independent (Laurent) polynomials over $\mathbb{F}[X]$ of degree at most $k-1$. Let $\mathbf{x} = (x_1,\dots,x_k)$ be distinct points in $\mathbb{F}^*$.*

Then the following matrix is invertible.

$$M(\mathbf{p},\mathbf{x}) = \begin{pmatrix} p_1(x_1) & p_1(x_2) & \cdots & p_1(x_k) \\ p_2(x_1) & p_2(x_2) & \cdots & p_2(x_k) \\ \vdots & \vdots & \ddots & \vdots \\ p_k(x_1) & p_k(x_2) & \cdots & p_k(x_k) \end{pmatrix}$$

Proof The matrix $M(\mathbf{p},\mathbf{x})$ is square. In order to prove that the matrix is invertible, it suffices to prove that the rows are linearly independent. Suppose for contradiction that there was a linear relation between the rows of $M(\mathbf{p},\mathbf{x})$. Since all of the polynomials have degree at most $k-1$, the coefficients of each polynomial are completely determined by their evaluations at the points of $\mathbf{x}$, by applying a linear map. Therefore, any linear relation over the rows implies a linear relation over the polynomials, contradicting linear independence. ■

Using the invertibility of matrices from the previous lemma, the next lemma shows that for certain types polynomials in multiple variables, where one has a lot of polynomial evaluations at a structured set of points, one can solve a linear system to recover the coefficients of the polynomials.

Intuitively, in real protocols in the discrete logarithm setting, the verifier often receives an opening of a linear combination of many commitments. The coefficients of the linear combination come from the polynomials given below. Given a large number of openings at different points x and y, one can solve the linear system and recover openings to all of the commitments. This lemma is new but not general, with each type of polynomial corresponding to one of the ILC protocols in Chapter Chapter 8.

Lemma 4 *Consider an* ILC *protocol with random verifier challenges x and y. Consider* ILC *queries of the following polynomial forms given in the list below, where $\mathbf{p}_i$ are constant vectors, and $\mathbf{q}_j(y)$ are vectors which were committed after the prover saw y. We prove the result for each type of polynomial in the list. Suppose that we have an (n_1,n_2)-tree of query results, written as a vector $\mathbf{q}$. There exist n_1 and n_2 such that given the result of the query for n_1 different values of x, and for each* *value of x, for n_2 different values of y, there exists a linear map L such that $L\mathbf{q}$ is the concatenation of all of the $\mathbf{p}_i$ values, and the values of $\mathbf{q}_j(y)$ for each query y. In other words,* ILC *protocols that use queries of this form have (n_1, n_2)-tree special soundness for some n_1 and n_2.*

1. $\mathbf{v} = \sum_{i=-m}^{m} \mathbf{h}_i x^i$.

2. $\mathbf{v} = \sum_{i=0}^{m} \mathbf{h}_i l_i(x)$, *where $l_1(X), \dots, l_m(X)$ is the set of Lagrange interpolation polynomials with distinct interpolation points $z_1, \dots, z_m$ and $l_0(X) = \prod_{i=1}(X - z_i)$.*

3. $\mathbf{v} = \sum_{i=1}^{m} \mathbf{h}_i l_i(x) + l_0(x) \sum_{i=0}^{m-2} \mathbf{d}_i x^i$.

4. $\mathbf{v} = \mathbf{v}_0 y + \sum_{i=0}^{m} \mathbf{h}_i l_i(x)$.

5. $\mathbf{v} = \mathbf{f}_2 y^2 + \mathbf{f}_1 y + \sum_{i=1}^{m} \mathbf{h}_i l_i(x) + l_0(x) \sum_{i=0}^{m-2} \mathbf{d}_i x^i$.

6. $\mathbf{v} = \mathbf{e} y + \sum_{i=-m}^{m} \mathbf{h}_i x^i$.

7. $\mathbf{v} = \sum_{i=1}^{m} \mathbf{a}_i x^i y^i + \sum_{i=1}^{m} \mathbf{b}_i x^{-i} + x^m \sum_{i=1}^{m} \mathbf{c}_i x^i + \mathbf{d} x^{2m+1}$.

8. $\mathbf{v} = \sum_{i=1}^{m} \mathbf{a}_i y^i + \sum_{k=-m,k\neq 0}^{m} \mathbf{b}_k x^k$.

Proof

1. Given queries for $2m+1$ different values of the challenge x, this follows from Lemma 3 with $\mathbf{p}$ a vector of powers of X, which are clearly linearly independent, and $\mathbf{x}$ a vectors of the different values of the challenge.

2. Given queries for m different values of the challenge x, this follows from Lemma 3 with $\mathbf{p}$ a vector containing all of the polynomials $l_i(X)$. The Lagrange interpolation polynomials on points $z_1, \dots, z_m$ are clearly linearly independent, and $l_0(X)$ is independent from all of these Lagrange polynomials as it has degree m while they have degree $m-1$.

3. Given queries for $2m-1$ different values of the challenge x, this follows from Lemma 3 in a similar way to the previous two items.

4. Fix x. Given query answers for two different values of y, we have a linear system in $\mathbf{v}_0$ and $\sum_{i=0}^{m} \mathbf{h}_i l_i(x)$. Applying Lemma 3 with $\mathbf{p} = (Y, 1)$ and $\mathbf{Y}$ a vector containing the two different values of the challenge y, we can invert the linear system to recover $\mathbf{v}_0$ and $\sum_{i=0}^{m} \mathbf{h}_i l_i(x)$. Now, recovering the $\mathbf{h}_i$ values from m values of $\sum_{i=0}^{m} \mathbf{h}_i l_i(x)$ for m different values of x reduces to an earlier case.

5. Fixing a single value of x, given three query answers for three different values of y, we can eliminate y in a similar way to the previous case. Then, recovering the $\mathbf{h}_i$ and $\mathbf{d}_i$ from many different values of x is exactly the third case already covered.

6. Fixing a single value of x, given two query answers for two different values of y, we can eliminate y in a similar way to the previous case. Then, recovering the $\mathbf{h}_i$ from many different values of x is exactly the first case already covered.

7. For $4m+3$ different values of x, fix the value of y. Now ignore the powers of y multiplying the $\mathbf{a}_i$ and apply the first case to recover $\mathbf{a}_i y^i$, $\mathbf{b}_i$, $\mathbf{c}_i$ and $\mathbf{d}$. If we assume that we have the query results for at least two distinct values of y for each x, then at least one of the y values from the pair must be non-zero. Then we can simply divide by y to recover the $\mathbf{a}_i$.

8. Fix y. Suppose that we have query answers for $2m+1$ distinct of x. Then apply the result of the first case in order to recover all $\mathbf{b}_k$ and the value of $\sum_{i=1}^{m} \mathbf{a}_i y^i$. Now, apply the first case again for many different values $\sum_{i=1}^{m} \mathbf{a}_i y^i$ for distinct values of y to recover $\mathbf{a}_i$ for each i.

■

Chapter 7

Simple Protocols through the ILC Lens

In this chapter, to motivate the ILC model and to help the reader warm up before they read about more complicated matters in later chapters, we present some very simple zero-knowledge protocols in the discrete logarithm setting, and rewrite them as ILC protocols.

In the process, we explain the rationale and design process for protocols in the Ideal Linear Commitment model with reference to two simple, concrete examples of a zero-knowledge protocol based on the discrete logarithm assumption. We also explain why in later chapters we prove properties about the rank and dimension of the query matrices.

7.1 Basic Proof-of-Knowledge

In this section, we present one of the simplest possible useful examples of a discrete-logarithm-based protocol, and show how it can be rewritten as an ILC protocol.

Let $\mathbb{G}$ be a group of prime order p in which the discrete logarithm problem is hard to solve. Let g and h be elements of $\mathbb{G}$. We will use the Pedersen commitment scheme. A commitment A to message m with randomness α shall be computed as $A = g^m \cdot h^\alpha$.

The relation corresponding to the proof is

$$\mathscr{R} = \left\{ \ \sigma = (\mathbb{G}, p, g, h), u = C, w = \{a, r\} \ : \ C = g^a \cdot h^r \ \right\}$$

Now we present the protocol.

Common input: Setup information $\sigma = (\mathbb{G}, p, g, h)$.

Instance: Group element $u = C \in \mathbb{G}$.

Prover's witness: A field element. $w = (a, r) \in \mathbb{Z}_p^2$ such that $C = g^a \cdot h^r$.

Protocol:

P: The prover randomly selects $b, s \leftarrow \mathbb{Z}_p$. The prover computes $B = g^b \cdot h^s$ and sends B to the verifier.

V: The verifier randomly selects $x \leftarrow \mathbb{Z}_p$, and sends x to the prover.

P: The prover computes $f = ax + b$, and $u = rx + t$, and sends f and u to the verifier.

V: The verifier checks the equation $g^f \cdot h^u \stackrel{?}{=} C^x \cdot B$.

Theorem 5 *The Schnorr protocol 7.1 has perfect completeness, knowledge-soundness and special honest verifier zero-knowledge.*

Proof Perfect completeness follows by careful inspection.

For SHVZK, we describe a simulator. Given a challenge x, the simulator selects f and u uniformly at random from $\mathbb{Z}_p$ so that they are distributed exactly as in a real argument. Now, B is uniquely determined by the verification equation, and can be efficiently computed by rearranging the equation to see that $B = g^{-f} \cdot h^{-u} \cdot C^{-x}$. All values are distributed exactly as they would be in a real argument. Therefore the simulated argument is indistinguishable from a real argument and we have SHVZK.

Finally, we prove knowledge-soundness. To do this, we show that the protocol has *special soundness*. That is, given two proofs with the same first message and distinct challenges x and x', we show that it is possible to extract a valid witness w. This is the same as 2-tree special soundness from Definition 10.

Given $B \in \mathbb{G}$, distinct $x, x' \in \mathbb{Z}_p$ and $f, f', u, u' \in \mathbb{Z}_p$ such that $g^f \cdot h^u = C^x \cdot B$, and $g^{f'} \cdot h^{u'} = C^{x'} \cdot B$, we can divide the two copies of the verification equation to eliminate B. We obtain $g^{f-f'} \cdot h^{u-u'} = C^{x-x'}$. Since x and x' are distinct, $x - x'$ is invertible in $\mathbb{Z}_p$. Setting $a = \frac{f-f'}{x-x'}$ and $r = \frac{u-u'}{x-x'}$, we see that $g^a \cdot h^r = C$.

To see that the protocol has knowledge soundness, apply Lemma 2, which shows that the protocol has witness extended emulation. Then, as mentioned previously, [61] shows that witness extended emulation implies knowledge soundness. ■

Next, we give the ILC version of this protocol. The idea behind the basic proof-of-knowledge protocol is that the prover gives the verifier a linear combination of their secret value a and a random blinding value b used to hide a. Thus, we can reimagine Schnorr's protocol as an ILC protocol.

Notice that the verifier always accepts! This is not a mistake. ILC protocols model zero-knowledge protocols where the verifier wants to check relations between different linear combinations of committed values. In the real proof-of-knowledge protocol, the prover sends the values $B = g^b \cdot h^s$, f, and u to the verifier. We know that $C = g^a \cdot h^r$. Then, the verification equation checks that $z = sx + t$ in the exponent of the group element g. If the check is satisfied, then we know that $z = sx + t$.

In the ILC version of the protocol, the verifier is already sure that f is of the correct form, having used the `open` oracle to get it. The point of the ILC abstraction is to remove the need to think about a concrete commitment scheme, assume that all linear combinations that were previously checked using the homomorphic commitments are of the correct form, and focus on the how the prover and verifier compute on and check relations between the values in the finite field. We also remove the randomness used in the commitment scheme. This means that in the ILC version of the basic proof-of-knowledge protocol, the verifier simply needs to see the value of f, and nothing more.

Of course, in a real protocol it is important that this condition is actually checked. To see how the ILC version of the basic proof-of-knowledge protocol is converted into the actual basic proof-of-knowledge protocol, first, replace the prover's uses of the `commit` command with real commitments, and send the commitments to the verifier as they are made. Next, notice that the verifier's `open` query is split into two parts in the real protocol. The prover sends f and the randomness u corresponding to it, and the verifier checks the verification equation.

Common Reference String: $\sigma = (\mathbb{Z}_p, 1, \mathsf{aux} = \perp)$.

Instance: The committed field element $[a]$, which is already committed or stored in the ILC.

Prover's witness: The field element a.

P: The prover randomly selects $b \leftarrow \mathbb{Z}_p$. The prover commits to b using `commit`.

V: The verifier samples the challenge $x \leftarrow \mathbb{Z}_p$ uniformly at random.

The verifier makes an `open` query to get $f = ax + b$.

The verifier outputs 1 to accept.

Lemma 6 *The protocol has perfect completeness, knowledge-soundness, and perfect special honest verifier zero-knowledge.*

Proof Perfect completeness of the protocol follows by careful inspection.

For perfect special honest verifier zero knowledge, we provide an efficient simulator for the protocol. The simulator simulates the view of the verifier by picking f uniformly at random from $\mathbb{Z}_p$.

Finally, we prove knowledge-soundness. This is trivial as the ILC knowledge extractor already has access to all messages that the prover has committed to using `commit`. ■

7.2 Proof-of-Knowledge of a Committed Bit

Next, we give an example of a more complicated zero-knowledge protocol based on the discrete logarithm assumption. In this protocol, the prover proves to the verifier that they know how to open a given commitment to either 0 or 1.

This protocol will illustrate, for this basic case, some of the techniques used to design ILC protocols for particular statements.

The relation corresponding to the proof is

$$\mathscr{R} = \left\{ \ \sigma = (\mathbb{G}, p, g, h), u = C, w = \{a \in \{0,1\}, r \in \mathbb{Z}_p\} \ : \ B = g^a \cdot h^r \ \right\}$$

In the basic proof-of-knowledge protocol, the prover sent a linear combination $f = ax + b$ to the verifier, and this linear combination contained the secret, but looked random to the verifier. In this protocol, we want to check that a is a bit. One way to do this would be to find a way to use f to compute a useful function of a. Since a is a bit, it satisfies $a(1-a)$. So we can design a protocol around this fact.

We can use f to compute $a(1-a)$, although there will be some extra terms too, since $f = ax + b$. Observe that $f(x - f) = a(1-a)x^2 + b(1-2a)x - a^2$. The leading coefficient is $a(1-a)$. This means that if the prover is honest and a is really a bit, then $f(x-f)$ will be a polynomial of degree 1, instead of a polynomial of degree 2 as we might expect. So if the verifier can check this condition using commitments, then they will be satisfied that a really is a bit.

To check this, in the protocol, the prover will act as in the previous example, but will compute extra commitments K_1 and K_2 to $k_1 := b(1-2a)$ and $k_2 := -a^2$. Then the verifier will be able to commit to $f(x-f)$ later on and check that it is a linear polynomial. Note that the verifier doesn't need to worry about what the values inside K_1 and k_2 are. They simply serve to prove that $f(x-f)$ has the correct degree.

This example demonstrates a useful strategy for designing protocols. The prover commits to their witness, and extra random blinding values which will be used to hide the witness. They send the prover a masked version of the witness, and the verifier uses this to compute a polynomial which has a useful expression involving the witness in one of the coefficients. If the prover really knows a witness, that coefficient of the polynomial will be equal to zero, or some other known value. The prover commits to the other coefficients of the polynomial in advance to convince the verifier that the polynomial really is of this special form.

Now we present the protocol.

Common input: Setup information $\sigma = (\mathbb{G}, p, g, h)$.

Instance: Group element $u = C \in \mathbb{G}$.

Prover's witness: $w = \{a \in \{0,1\}, r \in \mathbb{Z}_p\}$ such that $C = g^a \cdot h^r$

Protocol:

P: The prover randomly selects $b, s, t_1, t_2 \leftarrow \mathbb{Z}_p$.

The prover computes $B = g^b \cdot h^s$, $K_1 = g^{b(1-2a)} \cdot h^{t_1}$ and $K_2 = g^{-a^2} \cdot h^{t_2}$.

The prover sends B, K_1 and K_2 to the verifier.

V: The verifier randomly selects $x \leftarrow \mathbb{Z}_p$, and sends x to the prover.

P: The prover computes $f = ax + b$, $u = rx + s$, and $v = t_1 x + t_2$, and sends f, u and v to the verifier.

V: The verifier checks the equations $g^f \cdot h^u \overset{?}{=} C^x \cdot B$ and $g^{f(x-f)} \cdot h^v \overset{?}{=} K_1^x \cdot K_2$.

Theorem 7 *The Schnorr protocol 7.2 has perfect completeness, knowledge-soundness and special honest verifier zero-knowledge.*

Proof Perfect completeness follows by careful inspection.

For SHVZK, we describe a simulator. Given a challenge x, the simulator selects f, u and v uniformly at random from $\mathbb{Z}_p$ so that they distributed exactly as in a real argument. Choose K_2 uniformly at random from $\mathbb{G}$. Now, B and K_1 are uniquely determined by the verification equation, and can be efficiently computed by rearranging the equations. All values are distributed exactly as they would be in a real argument. Therefore the simulated argument is indistinguishable from a real argument and we have SHVZK.

Finally, we prove knowledge-soundness. To do this, as before, we show that the protocol has special soundness. That is, given two proofs with the same first message and distinct challenges x and x', we show that it is possible to extract a valid witness a. This is the same as 2-tree special soundness from Definition 10.

Suppose that we are given $B, K_1, K_2 \in \mathbb{G}$, distinct $x, x' \in \mathbb{Z}_p$ and f, f', u, u', v, $v' \in \mathbb{Z}_p$ such that $g^f \cdot h^u = C^x \cdot B$, $g^{f(x-f)} \cdot h^v \overset{?}{=} K_1^x \cdot K_2$ and similarly equalities hold for the other copies of the verification equations in x'.

We can divide the two copies of the first verification equation to eliminate B. We obtain $g^{f-f'} \cdot h^{u-u'} = C^{x-x'}$. Since x and x' are distinct, $x - x'$ is invertible in $\mathbb{Z}_p$. Setting $a = \frac{f-f'}{x-x'}$ and $r = \frac{u-u'}{x-x'}$, we see that $g^a \cdot h^r = C$. By substituting this value for C into a copy of the first verification equation, we can also compute b and s such that $B = g^b \cdot h^s$. In a similar way, we can use the two copies of the second verification equation to find openings k_1, κ_1 of K_1 and k_2, κ_2 of K_2.

Now, by the binding property of the commitment scheme, we know that $f = ax + b$ and that $f(x - f) = k_1 x + k_2$. Substituting the first expression into the second, we obtain the quadratic polynomial $a(1-a)x^2 + (b(1-2a) - k_1)x - a^2 - k_2$. If the leading coefficient, $a(1-a)$, is non-zero, then this is a linear or quadratic equation, and there are at most two values of x which are roots. So the probability that the prover is able to produce a proof that the verifier will accept is at most $2/p$. Therefore, we conclude that $a(1-a) = 0$, and that a is a bit.

To see that the protocol has knowledge soundness, apply Lemma 2, which shows that the protocol has witness extended emulation. Then, as mentioned previously, [61] shows that witness extended emulation implies knowledge soundness. $\blacksquare$

Now we present the ILC version of the protocol. In this case, the verifier is already sure that f is of the correct form $ax + b$, having used the `open` oracle to get it. The prover commits to k_1 and k_2 in advance. Since the verifier has $f = ax + b$, they can compute $f(x - f)$ for themselves and then use an `check` command to check that it is equal to $k_1 x + k_2$. Remember that it does not matter exactly what the values of k_1 and k_2 are, so long as they exist and $f(x - f)$ is a linear and not a quadratic polynomial.

This example in particular shows how the ILC model can be a useful abstraction. The details of the commitment scheme used, and the randomness in the commitments, are removed. One can focus on the polynomial arithmetic that actually makes the protocol work.

Common Reference String: $\sigma = (\mathbb{Z}_p, 1, \mathsf{aux} = \perp)$.

Instance: The committed element $[a]$, which is already committed or stored in the ILC.

Prover's witness: The bit $a \in \{0,1\}$.

P: The prover randomly selects $b \leftarrow \mathbb{Z}_p$. The prover computes $k_1 = b(1-2a)$ and $k_2 = -a^2$. The prover commits to b, k_1 and k_2 using `commit`.

V: The verifier samples the challenge $x \leftarrow \mathbb{Z}_p$ uniformly at random.

The verifier makes an `open` query to get $f = ax + b$.

The verifier computes $f(x-f)$, and makes a `check` query to check whether it is equal to $k_1 x + k_2$.

The verifier outputs 1 if the `check` query passed, and 0 otherwise.

Lemma 8 *The protocol has perfect completeness, knowledge-soundness, and perfect special honest verifier zero-knowledge.*

Proof Perfect completeness of the protocol follows by careful inspection.

For perfect special honest verifier zero knowledge, we provide an efficient simulator for the protocol. The simulator simulates the view of the verifier by picking f uniformly at random from $\mathbb{Z}_p$, and setting the output of the `check` query to $\top$.

Finally, we prove knowledge-soundness. Firstly, note that the ILC knowledge extractor already has access to all messages that the prover has committed to using `commit`. As in the previous protocol, we know that $f = ax + b$ and that $f(x-f) = k_1 x + k_2$. Substituting the first expression into the second, we obtain the quadratic polynomial $a(1-a)x^2 + (b(1-2a) - k_1)x - a^2 - k_2$. If $a(1-a)$ is non-zero, then this is a quadratic equation, and there are at most two values of x which are roots. So the probability that the prover is able to produce a proof that the verifier will accept is at most $2/p$. Therefore, we conclude that $a(1-a) = 0$, and that a is a bit. ■

7.2.1 Query Matrices

After many protocols in Chapter 8 of this thesis, and in the security proof for the compilation procedure in Section 9.2, we are concerned with the condition that the ILC query matrices have full rank, and fewer rows than columns. With the aid of the protocols from Sections 7.2 and 7.2, we explain why this is the case.

Let us look at the query matrix for the protocol of Section 7.2. The matrix is

$$\begin{pmatrix} x & 1 & 0 & 0 \\ 0 & 0 & x & 1 \end{pmatrix}.$$

The columns correspond to A, B, K_1 and K_2 from 7.2, and the rows correspond to the first and second verification equations. We can use this matrix to connect the randomness values sent by an honest prover to the randomness values used in the prover's commitments.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x & 1 & 0 & 0 \\ 0 & 0 & x & 1 \end{pmatrix} \begin{pmatrix} r \\ s \\ t_1 \\ t_2 \end{pmatrix}.$$

This is a system of linear equations.

Consider the rank-nullity theorem. Over a field $\mathbb{F}$, the rank-nullity theorem says when the matrix is considered to be a linear map, the dimension of the image of the map, plus the dimension of the kernel of the map, are equal to the number of columns, or the length of the vectors to which the matrix is applied. In an ILC protocol, this is equal to the number of commitments that the prover has made.

Imagine that this system of linear equations had a trivial kernel. In that case, given the randomness openings u and v from an honest prover, the verifier would be able to solve the system to recover the randomness used in the prover's commitments. In general, this seems as though it will not lead to protocols with the zero-knowledge property. This leads us to a first condition on the query matrix. If the kernel is non-trivial, this implies that *the rank of the query matrix should be strictly less than the number of columns in the matrix.*

Returning to the protocols for a second, we can also see that the protocol from Section 7.2 is very easy to simulate. One picks a commitment opening f at random. One also has to pick some randomness values v and u for the commitments in the verification equations, and these are also picked uniformly at random. Then, it is always possible to select some of the commitment group elements uniformly at random, and determine the others by rearranging the verification equations. In the ILC version of this protocol in Section 7.2, the commitments and commitment randomness have been abstracted out of the protocol, and we only need to simulate f.

When considering more complicated ILC protocols, and how they might be converted into real protocols in the discrete logarithm setting, we could hope to use the same strategy to prove zero knowledge. Namely, we could produce a simulator for the discrete-logarithm based protocol by using the simulator for the underlying ILC protocol to simulate all of the commitment openings, and then if we needed to simulate any values for commitment randomness, we could select all of those values uniformly at random. Then, one could select some of the commitment group elements uniformly at random and rearrange the verification equations to determine what the rest of the commitments should be in the simulated transcript. One could refer to this as the 'simple' simulator.

Now, suppose that the rank of the ILC query matrix is strictly less than the number of rows in the matrix. This means that the simulated openings for the randomness values could not be chosen to be independently and uniformly at random; as they would then come from a space of higher dimension than the space of an honest prover's openings for the randomness. In such a case, we would not be able to use the simple simulator. So we have a second condition on the query matrix, namely that *the rank of the query matrix should be greater than or equal to the number of rows in the matrix.*

If we take the condition that the rank of the ILC query matrix is at least the number of rows in the matrix, but strictly less than the number of columns in the matrix, then we must have that the number of rows is less than the number of columns. Then, the matrix must achieve maximum possible rank, which is the number of rows in the matrix. So, motivated by some simple situations which seem like they might be problematic for the zero-knowledge property and the use of the simple simulator, we have placed some limitations on the ILC query matrix.

In Section 9.2, we give a proof that these conditions on the query matrix are sufficient for honest-verifier zero-knowledge; that is, we show that the simple simulator suffices. These conditions on the query matrix apply to all of our protocols. However, we do not claim that the conditions are *necessary* for zero-knowledge.

Chapter 8

Generic ILC Protocols

In this section, we present efficient ILC protocols for a variety of different statements.

8.1 Remarks

Before stating our ILC protocols, we briefly discuss how ILC proofs will be compiled into real proofs. This will motivate the efficiency discussions after each protocol. In the ILC protocols, the prover will commit to vectors by sending them to ILC. After interacting with the prover, and simply receiving vectors from the `open` and `check` oracles, the verifier will make several ILC queries in order to obtain certain linear combinations of the committed vectors. In real, compiled protocols, the prover will commit to vectors using the commitment scheme, and will personally compute and send the linear combinations requested by the verifier. Therefore, when discussing the communication costs of ILC protocols, we will refer to the number and length of committed vectors, and the number and length of vectors which form the verifier's queries to the ILC. When discussing the computational costs of the protocols, we will refer to the number and length of committed vectors, and the cost in field multiplications for the prover, verifier, and ILC.

As discussed earlier, when the verifier makes a `check` query, the result will not be counted as part of the communication complexity of the protocol. This is because in the original compilation from ILC protocols to standard zero-knowledge protocols given in [16], all `open` queries made by the verifier end up being computed and sent by the prover in the compiled protocol. Then, the verifier checks all query answers against commitments, and checks that the query answers satisfy certain equations. For example, the verifier might query ILC and obtain vectors $\bar{\mathbf{a}}$, $\bar{\mathbf{b}}$ and $\bar{\mathbf{c}}$. Then, in the real proof, the verifier would check $\bar{\mathbf{a}}$, $\bar{\mathbf{b}}$ and $\bar{\mathbf{c}}$ against committed values, and check some equations, such as $\bar{\mathbf{a}} \circ \bar{\mathbf{b}} = \bar{\mathbf{c}}$, for example. However, rather than actually make a query for $\bar{\mathbf{c}}$, the verifier could simply check $\bar{\mathbf{a}} \circ \bar{\mathbf{b}}$ against the commitments for $\bar{\mathbf{c}}$. Since the verifier can compute for themselves what $\bar{\mathbf{c}}$ ought to be from the verification equations, there is no need for the verifier to receive this values. The `check` command is used to distinguish this case, which should not be counted as part of the communication costs of a proof. Furthermore, linear combinations queried using `check` queries do not need to be generated by the simulator, as they can already be computed using the answers to other `open` queries.

Throughout this chapter, we will present ILC protocols over a field $|\mathbb{F}|$ and prove that they have soundness error $poly(\lambda)/|\mathbb{F}|$. We will assume that $|\mathbb{F}|$ is sufficiently large so that the protocols have negligible soundness error. However, if this is not the case, then in Section 10.2, we provide a technique to boost soundness with minimal overhead, which applies to the three-move arithmetic circuit protocol in Section 8.4.

For each protocol, we will prove that with high probability, the ILC query matrix for the protocol has full rank, and fewer rows than columns. This will be useful in Chapter 9 when arguing that compiled protocols satisfy the zero-knowledge property.

8.2 Polynomial Commitment Sub-Protocol

The basis for the polynomial commitment protocol in this section was originally published in joint work [11] with Jens Groth, as a discrete-logarithm based protocol.

We present an ILC protocol that allows the prover to commit to a polynomial, so that the prover can later reveal the evaluation of the polynomial at a specific point x chosen by the verifier and prove that the evaluation is correct. This protocol is the same as the polynomial commitment protocol in [11], and similar to the polynomial commitment protocol of [9], but is rewritten slightly as an ILC protocol.

Motivation. It may seem unnecessary to produce a special protocol for committing to a polynomial and revealing the evaluation in the ILC model. After all, the prover could simply commit to each coefficient of the polynomial separately, and using a single query, the verifier could obtain an evaluation of the polynomial by requesting a single linear combination of the coefficients. However, the protocol in this subsection allows a trade-off between the number of commitments made by the prover, and the length of the vector which is given to the verifier as a response to the verifier's query. In most cases, this will lead to an improvement in communication complexity over the naive method of producing a separate commitment to each coefficient.

Looking ahead to the compilation of our ILC protocols into real zero-knowledge protocols, the flexibility that the polynomial commitment protocol affords will be extremely useful. Our compilation based on hash-functions results in one commitment for each element of the longest committed vector. Our compilation based on the discrete-logarithm assumption results in one commitment for each committed vector. The polynomial commitment protocol allows a trade-off between these two dimensions, so that the same protocol can be tuned for each compilation method.

8.2.1 Definitions

A polynomial commitment scheme enables a prover to commit to a secret vector of polynomials $\mathbf{h}(X) \in \mathbb{F}^l[X]$. By this, we mean that $\mathbf{h}(X)$ is a vector with l entries, and that each entry is a polynomial with coefficients in $\mathbb{F}$, in the formal variable X. Later on the verifier can learn the evaluation of the committed polynomial at a given point.

We add an integer argument *id* to both the prover and verifier algorithms. Some protocols will use the polynomial commitment scheme more than once to commit to different polynomials, and the *id* argument will allow the verifier to request the correct polynomial evaluation as part of PolyVerify.

Let us discuss the algorithms in more detail. Let us suppose that we wish to commit to a polynomial of degree d, meaning that the maximum of the degrees of the l entries of $\mathbf{h}(X)$ is d. Set $d = mn$, for some positive integers m and n. In the ILC model, we could commit to the l coefficients of X^i as a vector, for each X^i, and use an ILC protocol with vector length l. However, it will be more efficient if we combine more data from different powers of X into the different entries of the same vector, so we use a longer vector length $N = nl$, and the prover who commits to a polynomial will make roughly m commitments.

Let $\mathscr{K}$, $\mathscr{P}$ and $\mathscr{V}$ define an ILC protocol with μ rounds. The public information available to both the prover and the verifier is $\sigma = (\mathbb{F}, N)$ and $u = (m, l)$. Only the prover has access to a secret vector of polynomials $\mathbf{h}(X) \in \mathbb{F}^l[X]$ of degree mn. The verifier receives an evaluation point $x \in \mathbb{F}$ as input, and queries the ILC to obtain some useful values. The verifier then saves the evaluation $\mathbf{h}(x)$ as part of their final state $\mathsf{state}^{\mathscr{V}}_{\mu}$.

The polynomial commitment scheme is not considered, alone, to be a zero-knowledge protocol. However, it will be a useful subroutine in other zero-knowledge protocols. As such, we need to describe some security properties of the scheme so that we can use them to prove the security properties of other protocols where polynomial commitment is used as a subroutine. For the definitions, we draw inspiration from the Oblivious Polynomial Evaluation scheme of [123]. We give three security properties. These are correctness, soundness, and prover privacy. Correctness and prover privacy directly correspond to the definitions of correctness and sender security given in [123], except for small adjustments, since the scheme will be running as an ILC protocol. Since we do not require the verifier's evaluation point to be hidden from the prover, we refer to the functionality of polynomial evaluation rather than oblivious polynomial evaluation. This corresponds to abandoning the notion of receiver privacy from the original scheme.

First, we define the functionality of polynomial evaluation for the prover and verifier. Suppose that the prover and verifier are running as part of an ILC protocol over field $\mathbb{F}$ with vector length k.

- **Prover Input:** $\mathbf{h}(X) \in \mathbb{F}^l[X]$, with $\deg(\mathbf{h}) \leq mn$.
- **Verifier Input:** $x \in \mathbb{F}$.
- **Prover Output:** Nothing.
- **Verifier Output:** $\mathbf{h}(x)$.

The definition of correctness simply guarantees that if the protocol is carried out honestly, then the verifier will return the correct polynomial evaluation $\mathbf{h}(x)$.

Definition 11 (Correctness) *The polynomial commitment scheme is* correct *if for all* $\mathbf{h}(X) \in \mathbb{F}^l[X]$ *with* $deg(\mathbf{h}) \leq mn$ *and* $x \in \mathbb{F}$*, the verifier receives* $\mathbf{h}(x)$*, the output of the polynomial evaluation functionality.*

Definition 12 (Prover Privacy) *For every probabilistic polynomial time* $\mathscr{V}^*$ *playing the role of the verifier, there exists a probabilistic polynomial time simulator* $\mathscr{S}$ *that produces output identically distributed to the view of* $\mathscr{V}^*$.

The original definitions given in [123] do not bind the prover to a fixed polynomial of particular degree. We saw in Section 7.2 that it could be important to check that `open` values were evaluations of a polynomial of a particular degree. Therefore, we provide an extra definition that ensures that the verifier's outputs must be consistent with evaluations of the same polynomial.

Definition 13 (Knowledge soundness) *A polynomial commitment scheme has* knowledge soundness *if for all DPT* $\mathscr{P}^*$ *there exists an expected PPT extractor* $\mathscr{E}$ *such that for all PPT adversaries* $\mathscr{A}$

$$\Pr\left[\begin{array}{c} \sigma \leftarrow \mathscr{K}(1^\lambda); ((m,l),s,x,x') \leftarrow \mathscr{A}(\sigma); \\ \mathbf{h}(X) \leftarrow \mathscr{E}^{\langle \mathscr{P}^*(s) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathscr{V}(\sigma,(m,l))\rangle}(\sigma,(m,l),x); \\ \langle \mathscr{P}^*(s) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathscr{V}(\sigma,(m,l))\rangle(\sigma,(m,l),x') : \\ \mathsf{state}_\mu^{\mathscr{V}} \neq \mathbf{h}(x') \ \wedge \ \mathbf{h}(X) \in \mathbb{F}^l[x] \ \wedge \ deg(\mathbf{h}) = mn \end{array}\right] \approx 0.$$

Here the oracle $\langle \mathscr{P}^*(s) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathscr{V}(\sigma,(m,l))\rangle$ *runs a full protocol execution and if the proof is successful it returns a transcript of the prover's communication with the channel. The extractor* $\mathscr{E}$ *can ask the oracle to rewind the proof to any point in a previous transcript and execute the proof again from this point on with fresh public-coin challenges from the verifier. We think of s as the state of the prover.*

8.2.2 Idea

In the following, we will build a polynomial commitment scheme as in ILC sub-protocol using vectors in $\mathbb{F}^{nl}$. Let us first give some intuition about how the construction will work.

Let $\mathbf{h}(X) = \sum_{i=0}^{N} \mathbf{h}_i X^i$ be a polynomial of degree $N = (n+1)m-1$ with coefficients that are row-vectors in $\mathbb{F}^l$. Define an $m \times (n+1)l$ matrix

$$\begin{pmatrix} \mathbf{h}_{0,0} & \mathbf{h}_{0,1} & \cdots & \mathbf{h}_{0,n} \\ \mathbf{h}_{1,0} & \mathbf{h}_{1,1} & \cdots & \mathbf{h}_{1,n} \\ \vdots & \vdots & \ddots & \\ \mathbf{h}_{m-1,0} & \mathbf{h}_{m-1,1} & \cdots & \mathbf{h}_{m-1,n} \end{pmatrix} = \begin{pmatrix} \mathbf{h}_0 & \mathbf{h}_m & \cdots & \mathbf{h}_{nm} \\ \mathbf{h}_1 & \mathbf{h}_{m+1} & \cdots & \mathbf{h}_{nm+1} \\ \vdots & \vdots & \ddots & \\ \mathbf{h}_{m-1} & \mathbf{h}_{2m-1} & \cdots & \mathbf{h}_N \end{pmatrix}$$

With this matrix we have $\mathbf{h}(X) = \sum_{j=0}^{n} (\sum_{i=0}^{m-1} \mathbf{h}_{i,j} X^i) X^{mj}$. In the polynomial commitment scheme, the prover commits to each row of the matrix. Then, the verifier can make a single ILC query to obtain $(\bar{\mathbf{h}}_0, \ldots, \bar{\mathbf{h}}_n) = \sum_{i=0}^{m} (\mathbf{h}_{i,0}, \ldots, \mathbf{h}_{i,n}) x^i$. The verifier can use this value to compute $\mathbf{h}(x) = \bar{\mathbf{h}}_0 + \sum_{j=1}^{n} \bar{\mathbf{h}}_j x^{(j-1)m+d}$.

While the main idea we have sketched above gives the verifier assurance that the committed polynomial has been correctly evaluated, the prover may not be happy. The problem is that the solution gives away information about the coefficients of $\mathbf{h}(X)$. We will therefore introduce some random blinding vectors to ensure no information is leaked about the committed coefficients except the evaluation of the polynomial. We will also adjust the protocol to handle an arbitrary polynomial degree $N = mn + d$ for $0 \leq d < m$ by shifting the first column of the matrix.

We pick random blinders $\mathbf{b}_1, \ldots, \mathbf{b}_n \leftarrow \mathbb{F}^l$ and define an $(m+1) \times (n+1)l$ matrix $\{\mathbf{h}_{i,j}\}_{i=0,j=0}^{m,n}$ as follows:

$$\begin{pmatrix} \mathbf{h}_0 & \mathbf{b}_1 & \cdots & \mathbf{b}_{n-1} & \mathbf{b}_n \\ \mathbf{h}_1 & \mathbf{h}_{d+1} & \cdots & \mathbf{h}_{(n-2)m+d+1} & \mathbf{h}_{(n-1)m+d+1} \\ & & & & \vdots \\ \mathbf{h}_d - \mathbf{b}_1 & \vdots & \ddots & & \mathbf{h}_{nm} \\ 0 & & & & \mathbf{h}_{nm+1} \\ \vdots & & & & \vdots \\ 0 & \mathbf{h}_{m+d-1} & \cdots & \mathbf{h}_{(n-2)m+d-1} & \mathbf{h}_{N-1} \\ 0 & \mathbf{h}_{m+d} - \mathbf{b}_2 & \cdots & \mathbf{h}_{(n-2)m+d} - \mathbf{b}_n & \mathbf{h}_N \end{pmatrix}$$

We can therefore rewrite the polynomial as

$$\mathbf{h}(X) = \sum_{i=0}^{m} \mathbf{h}_{i,0} X^i + \sum_{j=1}^{n} \left(\sum_{i=0}^{m} \mathbf{h}_{i,j} X^i \right) X^{(j-1)m+d}.$$

Intuitively, the prover will be committing to the rows of the matrix, and the verifier will be using the `open` oracle to reveal a linear combination of the rows, and then computation a polynomial evaluation using a method similar to that of the previous example. We describe the scheme below.

8.2.3 Protocol

In this protocol, we suppress the inputs σ, m, l and $\mathbf{h}(X)$ for notational simplicity.

$\mathbf{P}_{\mathrm{PC}}(id)$**:** The prover randomly selects $\mathbf{b}_1, \ldots, \mathbf{b}_n \leftarrow \mathbb{F}^l$ and arranges them into a matrix with entries $\{\mathbf{h}_{i,j}\}_{i=0,j=0}^{m,n}$ as follows:

$$\begin{pmatrix}
\mathbf{h}_0 & \mathbf{b}_1 & \cdots & \mathbf{b}_{n-1} & \mathbf{b}_n \\
\mathbf{h}_1 & \mathbf{h}_{d+1} & \cdots & \mathbf{h}_{(n-2)m+d+1} & \mathbf{h}_{(n-1)m+d+1} \\
 & & & & \vdots \\
\mathbf{h}_d - \mathbf{b}_1 & \vdots & \ddots & & \mathbf{b}_n \\
0 & & & & \mathbf{h}_{nm+1} \\
\vdots & & & & \vdots \\
0 & \mathbf{h}_{m+d-1} & \cdots & \mathbf{h}_{(n-2)m+d-1} & \mathbf{h}_{N-1} \\
0 & \mathbf{h}_{m+d} - \mathbf{b}_2 & \cdots & \mathbf{h}_{(n-2)m+d} - \mathbf{b}_n & \mathbf{h}_N
\end{pmatrix}$$

The prover commits to each row of the matrix above using `commit` (i.e. for $1 \le i \le m$, the prover commits to the row vector $(\mathbf{h}_{i,0}||\ldots||\mathbf{h}_{i,n})$).

$\mathbf{V}_{\mathrm{PC}}(x, id)$**:** The verifier makes an `open` query to the ILC to obtain $(\bar{\mathbf{h}}_0, \ldots, \bar{\mathbf{h}}_n) = \sum_{i=0}^{m} (\mathbf{h}_{i,0}, \ldots, \mathbf{h}_{i,n}) x^i$.

The verifier computes and returns $\mathbf{h}(x) = \bar{\mathbf{h}}_0 + \sum_{j=1}^{n} \bar{\mathbf{h}}_j x^{(j-1)m+d}$.

8.2.4 Security Proof

Lemma 9 *The polynomial commitment protocol given above has correctness, knowledge-soundness, and prover privacy.*

Proof By inspection, it follows that when the prover is honest, the verifier always recovers $\bar{\mathbf{h}} = \mathbf{h}(x)$.

We describe an efficient simulator to prove that the protocol has prover privacy. The simulator is given $\mathbf{h}(x)$, and also given ρ, which determines the value of x. The simulator first picks random $\bar{\mathbf{h}}_1, \ldots, \bar{\mathbf{h}}_n \leftarrow \mathbb{F}^l$ and then computes $\bar{\mathbf{h}}_0 = \mathbf{h}(x) - \sum_{j=1}^{n} \bar{\mathbf{h}}_j x^{(j-1)m+d}$. In other words, the $\mathbf{h}_j$ are chosen uniformly at random, conditional on giving the correct evaluation $\mathbf{h}(x)$.

This is a perfect simulation. The values $\bar{\mathbf{h}}_1, \ldots, \bar{\mathbf{h}}_n$ are chosen independently and uniformly at random in real proofs due to the choices of $\mathbf{b}_1, \ldots, \mathbf{b}_n$ just as in the simulated proofs. Finally, given these random values both real and simulated proofs, the matching $\bar{\mathbf{h}}_0$ are uniquely determined. This means we have identical distributions of real and simulated proofs which are consistent with the evaluation $\mathbf{h}(x)$.

Finally, we prove knowledge-soundness. The knowledge extractor already has access to the vectors committed as $(\mathbf{h}_{i,0}|| \ldots ||\mathbf{h}_{i,n})$ for $0 \leq i \leq m$, having seen all messages sent between the prover and the ILC. Therefore, we define the polynomial to be

$$\mathbf{h}(X) = \sum_{i=0}^{m} \mathbf{h}_{i,0} X^i + \sum_{j=1}^{n} \left(\sum_{i=0}^{m} \mathbf{h}_{i,j} X^i \right) X^{(j-1)m+d}$$

In the protocol, the verifier queries ILC to obtain $(\bar{\mathbf{h}}_0, \ldots, \bar{\mathbf{h}}_n) = \sum_{i=0}^{m} (\mathbf{h}_{i,0}, \ldots, \mathbf{h}_{i,n}) x^i$ and then computes and returns $\bar{\mathbf{h}} = \bar{\mathbf{h}}_0 + \sum_{j=1}^{n} \bar{\mathbf{h}}_j x^{(j-1)m+d}$. Substituting the query values into the expression for $\bar{\mathbf{h}}$ shows that $\mathbf{h}(x) = \bar{\mathbf{h}}$ and completes the proof of soundness. ■

Lemma 10 *The protocol has tree-special soundness, and the extraction algorithm χ is a linear map.*

Proof The verifier's view takes the form $(\bar{\mathbf{h}}_0, \ldots, \bar{\mathbf{h}}_n) = \sum_{i=0}^{m} (\mathbf{h}_{i,0}, \ldots, \mathbf{h}_{i,n}) x^i$. This corresponds to case 1 of Lemma 4. Therefore, given the verifier's view for sufficiently many distinct values of x, one can recover the committed polynomial using a linear map.

Lemma 11 *With high probability, the* ILC *query matrix for the protocol has full rank, and fewer rows than columns.*

Proof The rows of the query matrix correspond to the number of ILC queries made by the verifier. The columns correspond to vectors which the prover has committed to. In this protocol, there is only one query, so there are clearly fewer queries than commitments whenever m is greater than 1. The query matrix consists of only a single vector, and is therefore full rank whenever the query matrix is non-zero. With high probability over the choice of x, the query is non-trivial.

8.2.5 Efficiency

Communication. The prover must send $m+1$ vectors in $\mathbb{F}^{nl}$ to ILC. The verifier receives a single vector of length $N = nl$ from ILC.

Computation. The prover does not need to do any algebraic computation. The verifier's computation is dominated by the ml multiplications over $\mathbb{F}$ required to compute $\mathbf{h}(x)$. It costs the ILC mnl multiplications to compute the verifier's query.

8.3 3-Move Low Depth Circuit Protocol

The basis for the protocol in this section was originally published in joint work [11] with Jens Groth, as a discrete-logarithm based protocol.

We present an ILC protocol that allows the prover to prove knowledge of values which satisfy a polynomial relation of low-degree. This type of relation is useful when giving zero-knowledge membership and non-membership proofs, such as the membership proof in [2], its subsequent optimisation in [10], and the polynomial evaluation proof of [3]. This protocol is the same as the main protocol given in [11], but is rewritten slightly as an ILC protocol.

A special case of this protocol, a zero-knowledge proof of membership described in 8.3.4.1, has been prototyped in Python.

In the following we will refer to the parameters $\ell_a, \ell_b, \ell_P, d_P, \ell_Q, d_Q$ such that

$\mathbf{a}_{i,j} \in \mathbb{F}^{\ell_a}$, $\mathbf{b}_{i,j} \in \mathbb{F}^{\ell_b}$, $\mathbf{P}$ is a vector of length ℓ_P with entries $(\ell_a + \ell_b)$-variate polynomials of total degree d_P, and $\mathbf{Q}$ is a vector of length ℓ_Q containing $(\ell_a + \ell_b)$-variate polynomials of total degree d_Q.

The idea is to prove that the prover knows solutions to polynomial relations encoding various statements, but allow the statements to be tweaked through an extra input to the polynomial. Let $\{\mathbf{b}_{i,j}\}_{i\in[m],j\in[n]}$ be some public vectors, representing the extra inputs or 'tweaks'. Let $\mathbf{P}$ and $\mathbf{Q}$ be vectors of polynomials which take two inputs, $\{\mathbf{a}_{i,j}, \mathbf{b}_{i,j}\}_{i\in[m],j\in[n]}$. We assume that the prover has already committed to C which we will write with square brackets in the instance $[C]$. We give a proof of knowledge of values $\{\mathbf{a}_{i,j}\}_{i\in[m],j\in[n]}$, such that $\mathbf{P}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}) = \mathbf{0}$ for $i \in [m], j \in [n]$, and such that

$$C = \begin{pmatrix} \mathbf{Q}(\mathbf{a}_{1,1}, \mathbf{b}_{1,1}), & \mathbf{Q}(\mathbf{a}_{1,2}, \mathbf{b}_{1,2}), & \ldots, \mathbf{Q}(\mathbf{a}_{1,n}, \mathbf{b}_{1,n}) \\ \mathbf{Q}(\mathbf{a}_{2,1}, \mathbf{b}_{2,1}), & \mathbf{Q}(\mathbf{a}_{2,2}, \mathbf{b}_{2,2}), & \ldots, \mathbf{Q}(\mathbf{a}_{2,n}, \mathbf{b}_{2,n}) \\ \vdots & & \\ \mathbf{Q}(\mathbf{a}_{m,1}, \mathbf{b}_{m,1}), & \mathbf{Q}(\mathbf{a}_{m,2}, \mathbf{b}_{m,2}), & \ldots, \mathbf{Q}(\mathbf{a}_{m,n}, \mathbf{b}_{m,n}) \end{pmatrix}$$

The protocol we design will be more efficient than repeating $t = mn$ instances of the basic protocol in parallel, as the communication complexity will depend on $\sqrt{t}$ rather than t.

Let us explain what all of the different letters above represent, and give an example of how to instantiate the parameters to get a simple range proof, where the prover wishes to prove that a secret committed integer lies in a given interval. The range proof example is stated again more precisely later on in Section 8.3.4.3.

- Firstly, we have $\mathbf{a}$. This is a secret vector, and part of the prover's witness. For our simple range proof, $\mathbf{a}$ will be a vector containing the the k-bit binary expansion of the prover's secret integer.

- Secondly, we have $\mathbf{b}$. This is a public vector of extra inputs to the protocol. For our simple range proof, we can imagine setting $\mathbf{b} = (1, 2, 4, \ldots, 2^{k-1})$. This will give a range proof for the interval $[0, 2^k - 1]$, but by tweaking the last entry of $\mathbf{b}$ to make it smaller, we could get range proofs for other intervals.

- Next, we have the polynomial $\mathbf{P}$. This represents a polynomial that the witness $\mathbf{a}$ should satisfy, possibly using the extra tweaks $\mathbf{b}$. For a range proof, $\mathbf{a}$ is made up of bits, so $\mathbf{P} = \mathbf{a} \circ (\mathbf{1} - \mathbf{a})$, and $\mathbf{P}(\mathbf{a}) = 0$ if and only if $\mathbf{a}$ consists of bits.

- Finally, we have the polynomial $\mathbf{Q}$ and the commitment $[C]$. Polynomial $\mathbf{Q}$ connects the witness with the contents of the commitment $[C]$. For example, for the simple range proof, we set $\mathbf{Q}(\mathbf{a}, \mathbf{b}) = \mathbf{a} \cdot \mathbf{b}$, so that $\mathbf{Q}$ computes the number represented by the bits of $\mathbf{a}$. The commitment $[C]$ should contain the value of $\mathbf{Q}(\mathbf{a}, \mathbf{b})$, which is the prover's secret integer in this case.

The relation corresponding to the proof is as follows, where "LD" is short for "low-depth".

$$\mathscr{R}_{\mathrm{LD}} = \left\{ \begin{array}{c} (\sigma, u, w) = \big((\mathbb{F}, n, z_1, \ldots, z_m) \ , \ ([C], \mathbf{P}, \mathbf{Q}, \{\mathbf{b}_{i,j}\}_{i \in [n], j \in [m]}), \{\mathbf{a}_{i,j}\}_{i \in [n], j \in [m]} \big) : \\ \forall i \in [m], \forall j \in [n], \mathbf{P}(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}) = \mathbf{0} \\ C = \begin{pmatrix} \mathbf{Q}(\mathbf{a}_{1,1}, \mathbf{b}_{1,1}), & \mathbf{Q}(\mathbf{a}_{1,2}, \mathbf{b}_{1,2}), & \ldots, \mathbf{Q}(\mathbf{a}_{1,n}, \mathbf{b}_{1,n}) \\ \mathbf{Q}(\mathbf{a}_{2,1}, \mathbf{b}_{2,1}), & \mathbf{Q}(\mathbf{a}_{2,2}, \mathbf{b}_{2,2}), & \ldots, \mathbf{Q}(\mathbf{a}_{2,n}, \mathbf{b}_{2,n}) \\ \vdots & & \\ \mathbf{Q}(\mathbf{a}_{m,1}, \mathbf{b}_{m,1}), & \mathbf{Q}(\mathbf{a}_{m,2}, \mathbf{b}_{m,2}), & \ldots, \mathbf{Q}(\mathbf{a}_{m,n}, \mathbf{b}_{m,n}) \end{pmatrix} \end{array} \right\}.$$

8.3.1 Intuition behind Protocol

The protocol embeds multiple instances of the same polynomial equality into a single polynomial by using Lagrange interpolation polynomials, inspired by [34, 12]. To recover a single instance, simply evaluate the polynomial in one of the interpolation points.

More concretely, the prover commits to the following vectors.

$$\begin{array}{llll} (\mathbf{a}_{0,1} || & \mathbf{a}_{0,2} || & \ldots || & \mathbf{a}_{0,n}) \\ (\mathbf{a}_{1,1} || & \mathbf{a}_{1,2} || & \ldots || & \mathbf{a}_{1,n}) \\ (\mathbf{a}_{2,1} || & \mathbf{a}_{2,2} || & \ldots || & \mathbf{a}_{2,n}) \\ \vdots & & & \\ (\mathbf{a}_{m,1} || & \mathbf{a}_{m,2} || & \ldots || & \mathbf{a}_{m,n}) \end{array}$$

Here, the values $\mathbf{a}_{0,1},\dots,\mathbf{a}_{0,n}\in\mathbb{F}^{l_a}$, where the value of the first index is 0, are blinding values chosen uniformly at random. These are completely unrelated to the values of the witness, which are $\mathbf{a}_{1,1},\dots,\mathbf{a}_{m,n}$, where the first index has a value strictly greater than 0. The verifier chooses a random challenge x and queries the ILC to obtain $(\bar{\mathbf{a}}_1,\dots,\bar{\mathbf{a}}_n)=\sum_{i=0}^m(\mathbf{a}_{i,1},\dots,\mathbf{a}_{i,n})l_i(x)$.

We must demonstrate that $\mathbf{a}_{i,j},\mathbf{b}_{i,j}$ satisfy the polynomial relations in the statement. Let $\bar{\mathbf{b}}_j=\sum_{i=1}^m\mathbf{b}_{ij}l_i(x)$. The verifier evaluates $\mathbf{P}$, $\mathbf{Q}$ using $\bar{\mathbf{a}}_j$ and $\bar{\mathbf{b}}_j$ for each j. By definition of $\bar{\mathbf{a}}_j$ and $\bar{\mathbf{b}}_j$, when evaluating at an interpolation point z_i, we obtain the single evaluation of the original polynomial, $\mathbf{P}(\mathbf{a}_{i,j},\mathbf{b}_{i,j})$. This implies, for example, that $\mathbf{P}(\bar{\mathbf{a}}_j,\bar{\mathbf{b}}_j)\equiv\mathbf{0}\mod l_0(x)$, or in other words, that $\mathbf{P}(\bar{\mathbf{a}}_j,\bar{\mathbf{b}}_j)$ is a multiple of $l_0(X)$ for each j. The prover must commit to the coefficients of $\mathbf{P}(\bar{\mathbf{a}}_j,\bar{\mathbf{b}}_j)/l_0(x)$ in advance (as a polynomial in x), and uses the polynomial commitment scheme to achieve this for every j simultaneously.

Finally, the prover needs to convince the verifier that the committed matrix C contains the values of $\mathbf{Q}(\mathbf{a}_{i,j},\mathbf{b}_{i,j})$. This is done in a similar way to the $\mathbf{P}$ polynomial, except here we build up polynomial equalities over committed values. The full protocol can be found below.

Common Reference String: $\sigma=(\mathbb{F},n,k_1,t_1,z_1,\dots,z_m)$ where $z_1,\dots,z_m$ are distinct points in $\mathbb{F}$ defining Lagrange polynomials $l_1(X),\dots,l_m(X)$ such that $l_i(z_j)=\delta_{i,j}$ and defining $l_0(X)=\prod_{j=1}^m(X-z_j)$.

Instance: $\{\mathbf{b}_{i,j}\}_{i\in[m],j\in[n]},\mathbf{P},\mathbf{Q}$ polynomials, and the $[C]$ that are already stored in the ILC.

P: Pick $\mathbf{a}_{0,1},\dots,\mathbf{a}_{0,n}\leftarrow\mathbb{F}^{\ell_a}$ and $\mathbf{c}_{0,1},\dots,\mathbf{c}_{0,n}\leftarrow\mathbb{F}^{\ell_Q}$. Commit to $(\mathbf{c}_{0,1}||\dots||\mathbf{c}_{0,n})$ and $(\mathbf{a}_{i,1}||\dots||\mathbf{a}_{i,n})$ for $i\in\{0\}\cup[m]$ using `commit`.

Define

$$\bar{\mathbf{a}}_j(X)=\textstyle\sum_{i=0}^m\mathbf{a}_{i,j}l_i(X)\qquad\bar{\mathbf{b}}_j(X)=\sum_{i=1}^m\mathbf{b}_{i,j}l_i(X)$$

$$\mathbf{P}^*_j(X)=\frac{\mathbf{P}(\bar{\mathbf{a}}_j(X),\bar{\mathbf{b}}_j(X))}{l_0(X)}\qquad\mathbf{Q}^*_j(X)=\mathbf{c}_{0,j}+\frac{\sum_{i=1}^m\mathbf{Q}(\mathbf{a}_{i,j},\mathbf{b}_{i,j})l_i(X)-\mathbf{Q}(\bar{\mathbf{a}}_j(X),\bar{\mathbf{b}}_j(X))}{l_0(X)}$$

Run $\mathbf{P}_{\mathrm{PC}}(\sigma = (\mathbb{F}, k_1), id = 1)$ and commit to $(\mathbf{P}_1^*(X)||\ldots||\mathbf{P}_n^*(X))$.

Run $\mathbf{P}_{\mathrm{PC}}(\sigma = (\mathbb{F}, t_1), id = 2)$ and commit to $(\mathbf{Q}_1^*(X)||\ldots||\mathbf{Q}_n^*(X))$.

V: The verifier samples the challenge $x \leftarrow \mathbb{F} \setminus \{z_1, \ldots, z_m\}$ randomly. The verifier runs

$$\mathbf{V}_{\mathrm{PC}}(x, id = 1) \rightarrow \bar{\mathbf{p}} = (\bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_n) \quad \mathbf{V}_{\mathrm{PC}}(x, id = 2) \rightarrow \bar{\mathbf{q}} = (\bar{\mathbf{q}}_1, \ldots, \bar{\mathbf{q}}_n)$$

Let $\mathbf{c}_i = (\mathbf{c}_{i,1}||\ldots||\mathbf{c}_{i,n})$ denote the rows of C, where $\mathbf{c}_{i,j} \in \mathbb{F}^{\ell_Q}$. The verifier makes an `open` query for the following value.

$$(\bar{\mathbf{a}}_1||\ldots||\bar{\mathbf{a}}_n) = \sum_{i=0}^{m} (\mathbf{a}_{i,1}||\ldots||\mathbf{a}_{i,n}) l_i(x)$$

The verifier computes $\bar{\mathbf{b}}_j = \bar{\mathbf{b}}_j(x)$. The verifier performs the following checks.

$$(\mathbf{P}(\bar{\mathbf{a}}_j, \bar{\mathbf{b}}_j))_{j \in [n]} \stackrel{?}{=} \bar{\mathbf{p}} l_0(x)$$
$$(\bar{\mathbf{q}}_j l_0(x) + \mathbf{Q}(\bar{\mathbf{a}}_j, \bar{\mathbf{b}}_j))_{j \in [n]} \stackrel{?}{=} \sum_{i=0}^{m} (\mathbf{c}_{i,1}||\ldots||\mathbf{c}_{i,n}) l_i(x)$$

The first check is made directly. The second check uses one `check` query. If all checks are satisfied, then the verifier outputs 1, and otherwise 0.

Lemma 12 *The protocol has perfect completeness, knowledge-soundness, and perfect special honest verifier zero-knowledge.*

Proof Perfect completeness of the protocol follows by correctness of the PolyCommit sub-protocol, which was used to commit to $(\mathbf{P}_1^*(X)||\ldots||\mathbf{P}_n^*(X))$ and $(\mathbf{Q}_1^*(X)||\ldots||\mathbf{Q}_n^*(X))$, and by careful inspection.

For perfect special honest verifier zero knowledge, we provide an efficient simulator for the protocol. The simulator selects $\bar{\mathbf{a}}_j \leftarrow \mathbb{F}^{\ell_a}$ and $\bar{\mathbf{q}}_j \leftarrow \mathbb{F}^{\ell_Q}$ for each j. All these values are distributed exactly as in a real protocol, where they are also uniformly random. She then simulates the view of the verifier when running $\mathcal{V}_{\mathrm{PC}}(x, id = 1)$ and $\mathcal{V}_{\mathrm{PC}}(x, id = 2)$ using the evaluation point x and evaluations $\bar{\mathbf{p}}$ and $\bar{\mathbf{q}}$, which are determined by the values already simulated, and using the simulator which comes from the prover privacy property of the PolyCommit sub-protocol. By the prover privacy of the polynomial commitment scheme, the simulated values have identical distribution to the real proofs. Finally, the simulator sets the output of all `check` queries to $\top$.

Finally, we prove knowledge-soundness. The ILC knowledge extractor already has access to all messages sent between the prover and ILC. It remains to show that if the committed values are not a valid witness, then there is a negligible probability that the verifier will accept. Recall the verification checks. The verifier runs

$$\mathscr{V}_{\mathrm{PC}}(x, id = 1) \rightarrow \bar{\mathbf{p}} = (\bar{\mathbf{p}}_1, \dots, \bar{\mathbf{p}}_n) \qquad \mathscr{V}_{\mathrm{PC}}(x, id = 2) \rightarrow \bar{\mathbf{q}} = (\bar{\mathbf{q}}_1, \dots, \bar{\mathbf{q}}_n)$$

The verifier queries ILC to obtain the following values.

$$(\bar{\mathbf{a}}_1 || \dots || \bar{\mathbf{a}}_n) = \sum_{i=0}^{m} (\mathbf{a}_{i,1} || \dots || \mathbf{a}_{i,n}) l_i(x)$$
$$(\bar{\mathbf{c}}_1 || \dots || \bar{\mathbf{c}}_n) = \sum_{i=0}^{m} (\mathbf{c}_{i,1} || \dots || \mathbf{c}_{i,n}) l_i(x)$$

The verifier computes $\bar{\mathbf{b}}_j = \bar{\mathbf{b}}_j(x)$. The verifier performs the following checks, for all $j \in [n]$.

$$\mathbf{P}(\bar{\mathbf{a}}_j, \bar{\mathbf{b}}_j) \stackrel{?}{=} \bar{\mathbf{p}}_j l_0(x) \qquad \bar{\mathbf{q}}_j l_0(x) + \mathbf{Q}(\bar{\mathbf{a}}_j, \bar{\mathbf{b}}_j) \stackrel{?}{=} \bar{\mathbf{c}}_j$$

Now, substitute the expressions for $\bar{\mathbf{a}}_j$, $\bar{\mathbf{b}}_j$ and $\bar{\mathbf{c}}_j$ into the left-hand side of the verification equations. By the soundness of the polynomial commitment protocol, we know that $\mathbf{p}_j$ is a polynomial of degree $m(d_P - 1)$ in x, and that $\mathbf{q}_j$ is a polynomial of degree $m(d_Q - 1)$ in x. The verifier only accepts if the equations hold. By assumption $\mathscr{P}^*_{\mathrm{prod}}$ is deterministic, and we know when it made its commitments. Hence, $\mathbf{a}_{i,j}$, $\mathbf{b}_{i,j}$ and $\mathbf{c}_{i,j}$ are constants. By the soundness of the polynomial commitment protocol, and the fact that the polynomials were committed by the prover before seeing x, we know that $\mathbf{p}_j$ is a polynomial of degree $m(d_P - 1)$ in x, and that $\mathbf{q}_j$ is a polynomial of degree $m(d_Q - 1)$ in x.

We can now apply Lemma 1, which implies the Schwarz-Zippel Lemma. Suppose that there exist i and j such that

$$\mathbf{P}(\mathbf{a}_{i,j},\mathbf{b}_{i,j}) \neq 0 \quad \text{or} \quad \mathbf{Q}(\mathbf{a}_{i,j},\mathbf{b}_{i,j}) \neq \mathbf{c}_{i,j}$$

This implies that

$$\mathbf{P}(\bar{\mathbf{a}}_j(X),\bar{\mathbf{b}}_j(X)) \neq \bar{\mathbf{p}}_j(X)l_0(X) \quad \text{or} \quad \bar{\mathbf{q}}_j(X)l_0(X)+\mathbf{Q}(\bar{\mathbf{a}}_j(X),\bar{\mathbf{b}}_j(X)) \neq \bar{\mathbf{c}}_j(X)$$

By the Schwarz-Zippel Lemma, if there is not an equality of polynomials, then the probability that that verification checks are satisfied is $\max(d_P,d_Q)m/|\mathbb{F}|$. The verifier can only accept if we have equality, so this shows that the probability that the committed values does not satisfy the product relation but that the verifier accepts is negligible, which proves knowledge soundness. ■

Lemma 13 *The protocol has tree-special soundness, and the extraction algorithm χ is a linear map.*

Proof The verifier sees the following queries as part of the protocol.

$$(\bar{\mathbf{a}}_1||\dots||\bar{\mathbf{a}}_n) = \sum_{i=0}^{m}(\mathbf{a}_{i,1}||\dots||\mathbf{a}_{i,n})l_i(x)$$
$$(\bar{\mathbf{c}}_1||\dots||\bar{\mathbf{c}}_n) = \sum_{i=0}^{m}(\mathbf{c}_{i,1}||\dots||\mathbf{c}_{i,n})l_i(x)$$

These both correspond to case 2 of Lemma 4. Therefore, given the verifier's view for sufficiently many distinct values of x, one can recover the committed polynomial using a linear map. ■

Lemma 14 *With high probability, the* ILC *query matrix for the protocol has full rank, and fewer rows than columns.*

Proof The rows of the query matrix correspond to the number of ILC queries made by the verifier. The columns correspond to vectors which the prover has committed to. In this protocol, there are a constant number of queries, including the queries from the polynomial commitment sub-protocol, so there are clearly fewer queries than commitments whenever m is large enough. The query matrix has full rank. This is because the two submatrices corresponding to the polynomial commitment sub-protocols have full rank. The submatrix corresponding to the main protocol also has full rank, since the two queries are non-trivial and operate on different commitments. Since each sub-matrix operates on different commitments to each other, the entire query matrix has full rank. With high probability over the choice of x, all queries are non-trivial. ■

8.3.2 Communication

Let k_1, k_2 be the dimensions of the matrix used in the PolyCommit subprotocol when committing to $\mathbf{P}^*$, and similarly, let t_1, t_2 be the dimensions of the matrix in the subprotocol for committing to $\mathbf{Q}^*$. In total, the prover commits to $m + k_1 + t_1 + 4$ vectors using ILC (k_1 vectors of length k_2, t_1 vectors of length t_2, and m vectors of length n). The verifier's queries amount to a total of $\ell_a n + \ell_P n(k_2 + 1) + \ell_Q n(t_2 + 1) + 4$ field elements. The verifier makes 3 queries to ILC, with queries on vectors of lengths k_2, t_2 and n respectively.

Single Proof Case When $t = mn = 1$ and the prover is proving a single relation, we may choose parameters so that the protocol only sends a constant number of vectors to ILC. This will be useful for the compiled proofs, where commitments often use extremely large group elements. Set $k_1 = t_1 = 1$, $k_2 = d_P - 1$, $t_2 = d_Q - 1$. Then the prover need only send 7 vectors to ILC. The verifier's queries amount to a total of $\ell_a + \ell_P d_P + \ell_Q d_Q + 4$ field elements. This will minimise communication in the case where the protocol is compiled and instantiated over a multiplicative subgroup of a finite field, where group elements are much bigger than field elements.

In the case where the protocol is later instantiated using an elliptic curve group, commitments and field elements have roughly the same size. Then, we can minimise the total communication costs by choosing $k_2 = \left\lceil \sqrt{\frac{d_P}{\ell_P}} \right\rceil$, $k_1 \approx \frac{d_P}{k_2}$. Set $t_2 = \left\lceil \sqrt{\frac{d_Q}{\ell_Q}} \right\rceil$, $t_1 \approx \frac{d_Q}{t_2}$. Then the prover must send $\sqrt{\ell_P d_P} + \sqrt{\ell_Q d_Q} + 5$ vectors to ILC. The verifier's queries amount to a total of $\ell_a + \sqrt{\ell_P d_P} + \sqrt{\ell_Q d_Q} + 4$ field elements.

Batch Proof Case When t is large, we choose parameters so that the communication costs in compiled proofs are proportional to $\sqrt{t}$ rather than t. Set $k_2 = \left\lceil \sqrt{\frac{d_P m}{\ell_P n}} \right\rceil$,

$k_1 \approx \frac{d_P m}{k_2}$. Set $t_2 = \left\lceil \sqrt{\frac{d_Q m}{\ell_Q n}} \right\rceil$, $t_1 \approx \frac{d_Q m}{t_2}$. Finally, set $m \approx \sqrt{\ell_a t}, n \approx \frac{t}{m}$. Then the protocol requires the prover to send roughly $\sqrt{\ell_a t} + \sqrt{d_P \ell_P t} + \sqrt{d_Q \ell_Q t}$ vectors to ILC. The verifier's queries amount to a total of $\sqrt{\ell_a t} + \sqrt{d_P \ell_P t} + \sqrt{d_Q \ell_Q t}$ field elements.

8.3.3 Computation

Over $\mathbb{F}$, the prover must perform

$$O((\ell_a + \ell_b + \ell_P) t d_P \log m d_P + (\ell_a + \ell_b + \ell_Q) t d_Q \log m d_Q + t d_P \mathsf{Eval}_P + t d_Q \mathsf{Eval}_Q)$$

multiplications. Here, Eval_P is the cost of evaluating P once, and similarly for Q. The vectors of polynomials $\mathbf{P}^*(X), \mathbf{Q}^*(X)$ are computed using FFT techniques. Over $\mathbb{F}$, the verifier must perform $O((\ell_P + \ell_Q)n + n\mathsf{Eval}_P + n\mathsf{Eval}_Q)$ multiplications. To compute the verifier's queries, the ILC uses $\ell_P d_P mn + \ell_Q d_Q mn$ multiplications to compute the verifier's polynomial commitment queries, and $mn(\ell_a + \ell_b)$ multiplications to compute the verifier's other ILC queries.

8.3.4 Applications

In this section, we specify concrete choices of relations for $\mathbf{P}, \mathbf{Q}$, which give rise to zero-knowledge arguments for several useful applications.

8.3.4.1 Membership Argument with Public List

In membership arguments [84, 85], the prover wishes to convince the verifier that a commitment contains one of the values in a given list $\mathscr{L} = (\lambda_0, \ldots, \lambda_{N-1})$. Groth and Kohlweiss [2] give an efficient membership argument, which with minor tweaks fits into our framework. For simplicity, we will in the following assume N is a power of 2.

Statement: $([c], \lambda_0, \ldots, \lambda_{N-1})$

Witness: ℓ such that $c = \lambda_\ell$

Polynomial Encoding: Let $m = \log_2 N$ and let $(l_0, \ldots, l_{m-1})$ be the binary expansion of l, satisfying $l_j(1 - l_j) = 0$ for $0 \leq j \leq m-1$. Define $l_{j,1} := l_j$ and

$l_{j,0} = 1 - l_j$. We have that

$$\sum_{i=0}^{N-1} \lambda_i \prod_{j=0}^{m-1} l_{j,i_j} = \lambda_l$$

where we write the binary expansion of i as $(i_0, \dots, i_{m-1})$.

Parameter Choice: Writing $\circ$ for the entry-wise product of two vectors

- $\ell_a = \log_2 N, \ell_b = N, \ell_P = \log_2 N, d_P = 2, \ell_Q = 1, d_Q = \log_2 N$
- $\mathbf{a} = (l_0, \dots, l_{m-1})$
- $\mathbf{b} = (\lambda_0, \dots, \lambda_{N-1})$
- $\mathbf{P}(\mathbf{a}, \mathbf{b}) = \mathbf{a} \circ (\mathbf{1} - \mathbf{a})$
- $Q(\mathbf{a}, \mathbf{b}) = \sum_{i=0}^{N-1} \lambda_i \prod_{j=0}^{m-1} l_{j,i_j}$

An alternative construction was given in [10] that optimises the membership argument by using an n-ary representation of l. This alternative construction is captured by our framework as follows, this time assuming for simplicity that N is a power of n, using different polynomials $\mathbf{P}$ and $\mathbf{Q}$.

Polynomial Encoding: Let $m = \log_n N$ and let $(l_0, \dots, l_{m-1})$ be the n-ary expansion of l. Let δ_{rs} be the Kronecker delta symbol, which is equal to 1 if $r = s$ and 0 otherwise. Consider the bit-string $(\delta_{l_0,0}, \delta_{l_0,1}, \dots, \delta_{l_{m-1},n-1})$, each element satisfying $\delta_{i,j}(1 - \delta_{i,j}) = 0$, and with $\sum_{i=0}^{n-1} \delta_{l_j,i} = 1$ for each j. As described in [10], we have that

$$\sum_{i=0}^{N-1} \lambda_i \prod_{j=0}^{m-1} \delta_{j,i_j} = \lambda_l$$

where i_j the jth n-ary digit of i.

Parameter Choice:

- $\ell_a = n \log_n N$, $\ell_b = N$, $\ell_P = n \log_n N$, $d_P = 2$, $\ell_Q = 1$, $d_Q = \log_n N$
- $\mathbf{a} = (\delta_{l_0,1}, \dots, \delta_{l_{m-1},n-1})$, not including $\delta_{j,0}$ for any j.
- $\mathbf{b} = (\lambda_0, \dots, \lambda_{N-1})$.

- $\delta_{l_j,0} = 1 - \sum_{i=1}^{n-1} \delta_{l_j,i}$ for each j.
- $\mathbf{v} = \left(\delta_{l_0,0}, \dots, \delta_{l_{m-1},n-1}\right)$, with the $\delta_{j,0}$ included.
- $P(\mathbf{a},\mathbf{b}) = \mathbf{v} \circ (\mathbf{1} - \mathbf{v})$
- $Q(\mathbf{a},\mathbf{b}) = \sum_{i=0}^{N-1} \lambda_i \prod_{j=0}^{m-1} \delta_{j,i_j} = \lambda_l$

When $t = 1$ and we are aiming for a constant number of commitments, the simple binary version of the argument gives the lowest communication costs. Otherwise, in the cases where t is large, or where $t = 1$ and we aim to minimise the total number of elements communicated in the compiled proof, setting $n = 3$ gives the lowest communication costs. The protocol efficiency is reported in Table 1.2.

8.3.4.2 Polynomial Evaluation Argument

In a polynomial evaluation argument [86, 85], we have a polynomial of degree N and commitments to a point and its purported evaluation in that point. The prover wants to convince the verifier that the committed evaluation of the polynomial is correct.

The most efficient discrete logarithm based polynomial evaluation argument was given by Bayer and Groth [3]. We will now use our framework of polynomial relations to capture their protocol.

Statement: $([c_u],[c_v],h(X))$, where $h(X)$ is a polynomial of degree N.

Witness: u, v such that $c_u = u, c_v = v$, and $h(u) = v$.

Polynomial Encoding: Set $u_i = u^{2^i}$ for $0 \le i \le \log_2 N - 1$, so that $u_i = u_{i-1}^2$ for each i. If $h(X) = \sum_{i=0}^{N-1} h_i X^i$, then we can write $h(u) = \sum_{i=0}^{N-1} h_i \prod_{j=0}^{\log_2 N-1} u_j^{i_j}$.

Parameter Choice:

- $\ell_a = \log_2 N$, $\ell_b = N$, $\ell_P = \log_2 N - 1$, $d_P = 2$, $\ell_Q = 1$, $d_Q = \log_2 N$
- $\mathbf{a} = (u_0, \dots, u_{\log N-1})$
- $\mathbf{b} = (h_0, \dots, h_{N-1})$
- $\mathbf{P}(\mathbf{a},\mathbf{b}) = (u_1 - u_0^2, \dots, u_{\log N-1} - u_{\log N-2}^2)$
- $\mathbf{Q}(\mathbf{a},\mathbf{b}) = \sum_{i=0}^{N-1} h_i \prod_{j=0}^{\log N-1} u_j^{i_j}$

With alternative choices of the matrices $\mathbf{P},\mathbf{Q}$, we can improve the communication costs of their argument by switching to an n-ary encoding of the powers in the polynomial.

Polynomial Encoding: Set $u_i = u^{n^i}$ for $0 \leq i \leq \log_n N - 1$, so that $u_i = u_{i-1}^n$ for each i. If $h(X) = \sum_{i=0}^{N-1} h_i X^i$, then we can write $h(u) = \sum_{i=0}^{N-1} h_i \prod_{j=0}^{\log_n N-1} u_j^{i_j}$, where this time, i_j is the jth digit of the $n$ary representation of i. This gives rise to the efficiencies listed in Table **??**.

Parameter Choice:

- $\ell_a = \log_n N$, $\ell_b = N$, $\ell_P = \log_n N$, $d_P = n$, $\ell_Q = 1$, $d_Q = \log_n N$
- $\mathbf{a} = (u_0, \ldots, u_{\log_n N-1})$
- $\mathbf{b} = (h_0, \ldots, h_{N-1})$
- $\mathbf{P}(\mathbf{a},\mathbf{b}) = (u_1 - u_0^n, \ldots, u_{\log_n N-1} - u_{\log_n N-2}^n)$
- $\mathbf{Q}(\mathbf{a},\mathbf{b}) = \sum_{i=0}^{N-1} h_i \prod_{j=0}^{\log_n N-1} u_j^{i_j}$

When $t = 1$ and we are aiming for a constant number of commitments, setting $n = 4$ gives the lowest communication costs. When $t = 1$ and we aim to minimise the total number of elements communicated in the compiled proof, we set $n = \frac{\log_2 N}{\log_2 \log_2 N}$. Otherwise, in the cases where t is large, setting $n = 6$ gives the lowest communication costs. The protocol efficiency is reported in Table 1.3.

We would like to highlight the results when $t = 1$ and $n = \frac{\log_2 N}{\log_2 \log_2 N}$. In this case, the total communication complexity of the polynomial evaluation protocol is $O\left(\frac{\log N}{\log \log N}\right)$ commitments and field elements in the discrete logarithm settting. In particular, this protocol has the best asymptotic efficiency for polynomial evaluation in zero-knowledge based on the discrete-logarithm assumption. If we use the polynomial to specify a set through its roots, then argument gives a membership argument where the communication costs are asymptotically less than $\log N$, the number of bits required to specify an element of the set. The best known arguments for general arithmetic circuits [9, 1] achieve a logarithmic communication complexity, as the number of multiplication gates required to compute a polynomial of degree N is N.

This protocol also uses commitments to vectors of length $O(\log N)$ rather than $O(N)$ which translates to dramatically fewer cryptographic operations.

We note that [12] gives a batch argument for polynomial evaluation based on similar ideas. However, ours is more communication efficient.

Remark. The relations above arise from choices of a small set of powers of u which generate all powers from u to u^{N-1}. This is the same as choosing an additive basis for $[N-1]$. For certain parameter choices, we have found modest benefits to using more complex bases, such as generalised Zeckendorf bases, but these give only slight improvements, so are omitted for simplicity.

8.3.4.3 Range Proof

In range proofs [87, 88], we have a commitment and a range $[A;B]$. The prover wants to convince the verifier that the committed value inside the commitment falls in the given range. A common strategy for constructing a range proof is to write the committed value in binary, prove all the bits are indeed 0 or 1, and that their weighted sum yields a number within the range. We now describe this type of range proof in our framework of polynomial relations, where we for simplicity focus on intervals $[0,N]$ with $N = 2^m - 1$.

Statement: $(N, [c])$

Witness: a, r such that $c = a, a \in [0,N]$.

Polynomial Encoding: Let $a_0, \ldots, a_{m-1}$ be the binary representation of a, so that $a_i(1-a_i) = 0$ for $0 \le i \le m-1$. Then $a = \sum_{i=0}^{m-1} a_i 2^i$.

Parameter Choice:

- $\ell_a = m, \ell_b = m, \ell_P = m, d_P = 2, \ell_Q = 1, d_Q = m+1$
- $\mathbf{a} = (a_0, \ldots, a_{m-1})$
- $\mathbf{b} = (2^0, 2^1, \ldots, 2^{m-1})$
- $\mathbf{P}(\mathbf{a}, \mathbf{b}) = \mathbf{a} \circ (\mathbf{1} - \mathbf{a})$
- $\mathbf{Q}(\mathbf{a}, \mathbf{b}) = \sum_{i=0}^{m-1} a_i 2^i$

With an alternative choice of $\mathbf{P}, \mathbf{Q}$, following [95], it is possible to improve the communication costs of the argument by using an n-ary base. This gives rise to the efficiencies listed in Table 1.3.

Polynomial Encoding: Let $N = n^m - 1$. Let $a_0, \ldots, a_{m-1}$ be the n-ary representation of a, so that $\prod_{k=0}^{n-1}(a_i - k) = 0$ for $0 \leq i \leq m-1$. Then $a = \sum_{i=0}^{m-1} a_i n^i$.

Parameter Choice:

- $\ell_a = m$, $\ell_b = m$, $\ell_P = m$, $d_P = n$, $\ell_Q = 1$, $d_Q = 1$
- $\mathbf{a} = (a_0, \ldots, a_{m-1})$
- $\mathbf{b} = (1, n, \ldots, n^{m-1})$
- $P(\mathbf{a}, \mathbf{b}) = \mathbf{a} \circ (\mathbf{a} - \mathbf{1}) \circ \ldots (\mathbf{a} - n + 1)$
- $Q(\mathbf{a}, \mathbf{b}) = \sum_{i=0}^{m-1} a_i n^i$

When $t = 1$ and we are aiming for a constant number of commitments, setting $n = 4$ gives the lowest communication costs. When $t = 1$ and we aim to minimise the total number of elements communicated in the compiled proof, we set $n = \frac{\log_2 N}{\log_2 \log_2 N}$. Otherwise, in the cases where t is large, setting $n = 6$ gives the lowest communication costs. The protocol efficiency is reported in Table 1.3.

8.4 3-Move Square-Root Protocol

In this section, we present an ILC protocol for arithmetic circuit satisfiability which consists of only 3 moves.

One might ask why it is necessary to include both a 3-move protocol and a 5-move protocol in this thesis. The 5-move protocol is more efficient in almost every way except for the number of moves. Furthermore, many users are more interested in non-interactive protocols anyway. After making the 5-move argument non-interactive using the Fiat-Shamir heuristic, does the existence of the 3-move protocol matter?

There are several reasons why it was important to include the 3-move argument. Firstly, it yields, for example, using the later compilation with discrete logarithm comitments, the first known zero-knowledge argument of knowledge for arithmetic circuit satisfiability that has square-root communication complexity, and only 3 moves, based on the discrete logarithm assumption. The previous best arguments [8, 7] consisted of 5 moves.

Secondly, it was this argument that drew attention to some corner cases within the ILC model, and led to the creation of the `check` command alongside the `open` command. Finally, when both proofs have been compiled into real protocols using homomorphic commitments and the compilation procedure in Section 9.2, the 3-move protocol requires much less rewinding of the prover and verifier in order to prove knowledge soundness. This is a potential advantage either when one is thinking about the concrete security of each protocol and the tightness of security reductions, or when one is trying to give security proofs in a quantum security model[1] where rewinding techniques are still only applicable in quite limited scenarios.

We will construct SHVZK proofs of knowledge for the relation $\mathscr{R}_{\mathsf{AC}}$, where the instances are arithmetic circuits over a field $\mathbb{F}$ specified by σ. An instance consists of many fan-in 2 addition and multiplication gates over $\mathbb{F}$, a description of how wires in the circuit connect to the gates, and values assigned to some of the input wires. Witnesses w are the remaining inputs such that the output of the circuit is 0. For an exact definition of how we represent an arithmetic circuit, see Section 3.2.

In this argument, we use the earlier reduction of arithmetic circuit satisfiability give in 3.2, and check that some committed vectors have the correct Hadamard product and satisfy some linear consistency equations. An approach similar to that of [13] works well for doing many operations in parallel, so this will be used for the Hadamard product. Embedding field elements as the coefficients of polynomials as in [7] works well for scalar products, so will be used for the consistency equations.

The Hadamard product component of this argument is not new. It is the Hadamard Product argument of section 5.4, [12].

One disadvantage of this approach is that the prover sends the blinded wire

[1]The compilation procedure from Section 9.2 was given with discrete-logarithm-based commitments in mind, but applies to all homomorphic commitments, including those which might be secure against attacks by quantum adversaries.

values to the verifier more than once. Nevertheless, the compiled protocol achieves a square-root complexity in 3 rounds.

The relation corresponding to the proof is

$$\mathscr{R}_{\text{AC1}} = \left\{ \begin{array}{c} (\sigma_{\mathsf{ILC}}, u, w);\ \sigma_{\mathsf{ILC}} = (\mathbb{F}, n, Q, (z_1, \ldots, z_m)), \\ u = \left(\{\mathbf{w}_{q,a,i}, \mathbf{w}_{q,b,i}, \mathbf{w}_{q,c,i}\}_{q \in [Q], i \in [m]}, \{K_q\}_{q \in [Q]}\right), \\ w = \{\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i \in m} : \\ \forall q \in [Q], \forall i \in [m], \quad \mathbf{w}_{q,a,i}, \mathbf{w}_{q,b,i}, \mathbf{w}_{q,c,i} \in \mathbb{F}^n \\ \forall i \in [m], \quad \mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i \in \mathbb{F}^n, \quad \mathbf{a}_i \circ \mathbf{b}_i = \mathbf{c}_i \\ \wedge \quad \forall q \in [Q], \quad \sum_{q=1}^{Q} \mathbf{w}_{q,a,i} \cdot \mathbf{a}_i + \sum_{q=1}^{Q} \mathbf{w}_{q,b,i} \cdot \mathbf{b}_i + \sum_{q=1}^{Q} \mathbf{w}_{q,c,i} \cdot \mathbf{c}_i = K_q \end{array} \right\}$$

8.4.1 Idea

The prover will begin by commiting to each $\mathbf{a}_i$, $\mathbf{b}_i$ and $\mathbf{c}_i$ for each i.

Given distinct points $z_1, \ldots, z_m$, let $l_i(X)$ be the Lagrange polynomials using distinct interpolation points $z_1, \ldots, z_m \in \mathbb{F}$. Explicitly,

$$l_i(X) = \prod_{j \neq i} \frac{X - z_j}{z_i - z_j}$$

Recall that $l_i(z_j) = \delta_{i,j}$. Set $l_0(X) = \prod_{i=1}^{m}(X - z_i)$.

The Hadamard product holds if and only if there exists a polynomial $\mathbf{d}(X)$ of degree $m-2$ such that the following polynomial equation holds:

$$\left(\sum_{i=1}^{m} \mathbf{a}_i l_i(X)\right) \circ \left(\sum_{i=1}^{m} \mathbf{b}_i l_i(X)\right) - \left(\sum_{i=1}^{m} \mathbf{c}_i l_i(X)\right) = \mathbf{d}(X) l_0(X)$$

The prover can commit to the coefficients $\mathbf{d}_i$ of $\mathbf{d}(X)$ beforehand. Then, for a randomly chosen challenge x, the verifier can make the following ILC queries.

$$\bar{\mathbf{a}} = \sum_{i=1}^{m} \mathbf{a}_i l_i(x) \qquad \bar{\mathbf{c}} = \sum_{i=1}^{m} \mathbf{c}_i l_i(x) + l_0(x) \sum_{i=0}^{m-2} \mathbf{d}_i x^i$$

$$\bar{\mathbf{b}} = \sum_{i=1}^{m} \mathbf{b}_i l_i(x)$$

with suitable random blinding values used in the real protocol to hide the wire values. The verifier can then verify that the Hadamard product holds by checking whether $\bar{\mathbf{a}} \circ \bar{\mathbf{b}} \overset{?}{=} \bar{\mathbf{c}}$.

Define the following polynomials.

$$\bar{\mathbf{e}}(X) = \sum_{i=1}^{m} \mathbf{a}_i X^i + X^m \sum_{i=1}^{m} \mathbf{b}_i X^i + X^{2m} \sum_{i=1}^{m} \mathbf{c}_i X^i$$

$$\mathbf{w}_q(X) = \sum_{i=1}^{m} \mathbf{w}_{q,a,i} X^{-i} + X^{-m} \sum_{i=1}^{m} \mathbf{w}_{q,b,i} X^{-i} + X^{-2m} \sum_{i=1}^{m} \mathbf{w}_{q,c,i} X^{-i}$$

The linear consistency equations hold if and only if there exist vectors $\mathbf{h}_j \in \mathbb{F}^Q$ such that the following polynomial equation holds:

$$(\bar{\mathbf{e}}(X) \cdot \mathbf{w}_1(X), \ldots, \bar{\mathbf{e}}(X) \cdot \mathbf{w}_Q(X)) = (K_1, \ldots, K_Q) + \sum_{j \neq 0, j=-3m}^{3m} \mathbf{h}_j X^j$$

The prover can commit to the vectors $\mathbf{h}_j$ in before receiving x. Then verifier can make ILC queries to obtain $\mathbf{e}(X)$ and also

$$\mathbf{h} = \sum_{j \neq 0, j=-3m}^{3m} \mathbf{h}_j x^j$$

Since all of the coefficients are public, the verifier can compute $\mathbf{w}_q(x)$ by themselves for each $q \in [Q]$. Finally, the verifier can verify that the linear consistency equations hold by checking whether

$$(\bar{\mathbf{e}} \cdot \mathbf{w}_1(x), \ldots, \bar{\mathbf{e}} \cdot \mathbf{w}_Q(x)) \overset{?}{=} (K_1, \ldots, K_Q) + \sum_{j \neq 0, j=-3m}^{3m} \mathbf{h}_j x^j$$

This protocol was the main motivation for introducing the `check` command to the ILC model. If this command had not been introduced, then the verifier might have made a `open` query for $(K_1, \ldots, K_Q) + \sum_{j \neq 0, j=-3m}^{3m} \mathbf{h}_j x^j$. Following the compilation in [16], this would have resulted in the prover sending a vector of length Q to the verifier. In general, Q could be as large as N, the number of multiplication gates in the arithmetic circuit. However, it is unnecessary for the prover to send this value to the verifier, as in a real proof, the verifier can compute $(\bar{\mathbf{e}} \cdot \mathbf{w}_1(x), \ldots, \bar{\mathbf{e}} \cdot \mathbf{w}_Q(x))$ and check this value against succinct commitments to the $\mathbf{h}_j$.

Common input: Setup information $\sigma_{\mathsf{ILC}} = (\mathbb{F}, n, Q, (z_1, \ldots, z_m))$. The description of an arithmetic circuit $u = \left(\{\mathbf{w}_{q,a,i}, \mathbf{w}_{q,b,i}, \mathbf{w}_{q,c,i}\}_{q \in [Q], i \in [m]}, \{K_q\}_{q \in [Q]}\right)$.

Prover's witness: Satisfying assignments $w = \{\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i \in m}$ to the wires of the circuit.

Protocol:

P: The prover randomly selects $\mathbf{a}_0, \mathbf{b}_0, \mathbf{c}_0 \leftarrow \mathbb{F}^n$, and commits to them using `commit`. For $i \in [m]$, the prover commits to $\mathbf{a}_i$, $\mathbf{b}_i$ and $\mathbf{c}_i \in \mathbb{F}^n$ using `commit`. The prover computes $\mathbf{d}_0, \ldots, \mathbf{d}_m \in \mathbb{F}^n$ such that

$$\left(\sum_{i=0}^{m} \mathbf{a}_i l_i(X)\right) \circ \left(\sum_{i=0}^{m} \mathbf{b}_i l_i(X)\right) - \left(\sum_{i=0}^{m} \mathbf{c}_i l_i(X)\right) = l_0(X) \sum_{i=0}^{m} \mathbf{d}_i X^i$$

Note that the sums on the left begin from $i = 0$ to incorporate the blinders. For $0 \leq i \leq m$, the prover commits to $\mathbf{d}_i$ using `commit`.

The prover randomly selects $\mathbf{e} \leftarrow \mathbb{F}^n$ and commits to it using `commit`. The prover computes $\mathbf{h}_j \in \mathbb{F}^Q$ such that

$$(\bar{\mathbf{e}}(X) \cdot \mathbf{w}_1(X), \ldots, \bar{\mathbf{e}}(X) \cdot \mathbf{w}_Q(X)) = (c_1, \ldots, c_Q) + \sum_{j \neq 0, j = -3m}^{3m} \mathbf{h}_j X^j$$

where

$$\bar{\mathbf{e}}(X) = \mathbf{e} + \sum_{i=1}^{m} \mathbf{a}_i X^i + X^m \sum_{i=1}^{m} \mathbf{b}_i X^i + X^{2m} \sum_{i=1}^{m} \mathbf{c}_i X^i$$
$$\mathbf{w}_q(X) = \sum_{i=1}^{m} \mathbf{w}_{q,a,i} X^{-i} + X^{-m} \sum_{i=1}^{m} \mathbf{w}_{q,b,i} X^{-i} + X^{-2m} \sum_{i=1}^{m} \mathbf{w}_{q,c,i} X^{-i}$$

For $j \neq 0, -3m \leq j \leq 3m$, the prover commits to $\mathbf{h}_j$ using `commit`.

V: The verifier randomly selects $x \leftarrow \mathbb{F}^*$, and makes the following `open` queries to

ILC.

$$\bar{\mathbf{e}} = \mathbf{e} + \sum_{i=1}^{m} \mathbf{a}_i x^i + x^m \sum_{i=1}^{m} \mathbf{b}_i x^i + x^{2m} \sum_{i=1}^{m} \mathbf{c}_i x^i$$

$$\bar{\mathbf{a}} = \sum_{i=0}^{m} \mathbf{a}_i l_i(x) \qquad \bar{\mathbf{b}} = \sum_{i=0}^{m} \mathbf{b}_i l_i(x)$$

For each $q \in [Q]$, the verifier computes

$$\mathbf{w}_q = \sum_{i=1}^{m} \mathbf{w}_{q,a,i} x^{-i} + x^{-m} \sum_{i=1}^{m} \mathbf{w}_{q,b,i} x^{-i} + x^{-2m} \sum_{i=1}^{m} \mathbf{w}_{q,c,i} x^{-i}$$

The verifier performs the following checks, using one `check` query for each.

$$\bar{\mathbf{a}} \circ \bar{\mathbf{b}} \overset{?}{=} \sum_{i=0}^{m} \mathbf{c}_i l_i(x) + l_0(x) \sum_{i=0}^{m} \mathbf{d} x^i$$

$$(\bar{\mathbf{a}} \cdot \mathbf{w}_1, \ldots, \bar{\mathbf{a}} \cdot \mathbf{w}_Q) \overset{?}{=} (c_1, \ldots, c_Q) \quad + \sum_{j \neq 0, j=-3m}^{3m} \mathbf{h}_j x^j$$

Theorem 15 *The argument of subsection 8.4.1 for satisfiability of an arithmetic circuit has perfect completeness, knowledge-soundness and special honest verifier zero-knowledge.*

Proof Perfect completeness follows by careful inspection.

For SHVZK, we describe a simulator. Given a challenge x such that $l_0(x) \neq 0$, the simulator randomly selects $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ uniformly at random from $\mathbb{F}^n$ and $\bar{\mathbf{e}}$ uniformly at random from $\mathbb{F}^Q$. Finally, the simulator sets the output of all `check` queries to $\top$. These are distributed exactly as in a real argument. Therefore the simulated argument is indistinguishable from a real argument and we have SHVZK.

Finally, we prove knowledge-soundness. The ILC knowledge extractor already has access to all messages sent between the prover and ILC. It remains to show that if the committed values are not a valid witness, then there is a negligible probability that the verifier will accept. Recall that the verifier makes `open` queries in order to obtain the following values.

$$\bar{\mathbf{a}} = \sum_{i=0}^{m} \mathbf{a}_i l_i(x) \qquad \bar{\mathbf{b}} = \sum_{i=0}^{m} \mathbf{b}_i l_i(x)$$

Now, substitute these expressions into the expression containing $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ that arises from the `check` query. This yields the following polynomial equation evaluated at x.

$$\left(\sum_{i=0}^{m} \mathbf{a}_i l_i(x)\right) \circ \left(\sum_{i=0}^{m} \mathbf{b}_i l_i(x)\right) - \sum_{i=0}^{m} \mathbf{c}_i l_i(x) - \sum_{i=0}^{m} \mathbf{d}_i l_0(x) = 0 \tag{8.1}$$

By assumption $\mathscr{P}^*_{\text{prod}}$ is deterministic, and we know when it made its commitments. Hence, all of the vector coefficients in Equation 8.1 are constant with respect to x. If there exist i and j such that $\mathbf{a}_{i,j} \circ \mathbf{b}_{i,j} \neq \mathbf{c}_{i,j}$, then Equation 8.1 is a non-zero polynomial of degree $2m$ which is zero at x. Since a polynomial of degree $2m$ can have at most $2m$ roots, the probability that this equality holds is at most $\frac{2m}{|\mathbb{F}|-m}$. The verifier can only accept if we have equality, so this shows that the probability that the committed values does not satisfy the product relation but verifier accepts is negligible, which proves knowledge soundness. ■

Lemma 16 *The protocol has tree-special soundness, and the extraction algorithm χ is a linear map.*

Proof The verifier sees the following queries as part of the protocol.

$$\bar{\mathbf{e}} = \mathbf{e} + \sum_{i=1}^{m} \mathbf{a}_i x^i + x^m \sum_{i=1}^{m} \mathbf{b}_i x^i + x^{2m} \sum_{i=1}^{m} \mathbf{c}_i x^i$$

$$\bar{\mathbf{a}} = \sum_{i=0}^{m} \mathbf{a}_i l_i(x) \qquad \bar{\mathbf{b}} = \sum_{i=0}^{m} \mathbf{b}_i l_i(x)$$

$$\bar{\mathbf{a}} \circ \bar{\mathbf{b}} \overset{?}{=} \sum_{i=0}^{m} \mathbf{c}_i l_i(x) + l_0(x) \sum_{i=0}^{m} \mathbf{d} x^i$$

$$(\bar{\mathbf{a}} \cdot \mathbf{w}_1, \ldots, \bar{\mathbf{a}} \cdot \mathbf{w}_Q) \overset{?}{=} (c_1, \ldots, c_Q) \quad + \sum_{j \neq 0, j=-3m}^{3m} \mathbf{h}_j x^j$$

These all correspond to cases of Lemma 4. Therefore, given the verifier's view for sufficiently many distinct values of x, one can recover the committed polynomial using a linear map. ■

Lemma 17 *With high probability, the* ILC *query matrix for the protocol has full rank, and fewer rows than columns.*

Proof The rows of the query matrix correspond to the number of ILC queries made by the verifier. The columns correspond to vectors which the prover has committed to. In this protocol, there are a constant number of queries, including the queries from the polynomial commitment sub-protocol, so there are clearly fewer queries than commitments whenever m is large enough. The query matrix has full rank. This is because the rows corresponding to each query operate on different commitments. Since each row operates on different commitments to the others, the entire query matrix has full rank. With high probability over the choice of x, all queries are non-trivial. ■

8.4.2 Efficiency

Communication The prover sends $4m+4$ vectors of length n and $6m$ vectors of length Q to ILC. Choosing $n \approx m \approx \sqrt{N}$ minimises the total cost in the compiled proof.

Computation The dominant cost for the prover in field multiplications is $O(Qmn \log m)$ from multiplying vectors of polynomials using FFT techniques. For the verifier, the number of field multiplications is dominated by $3Qmn$.

8.5 5-Move Square-Root Protocol

The basis for the protocol in this section was originally published in joint work [9] with Andrea Cerulli, Pyrros Chaidos, Jens Groth and Christophe Petit, as a discrete-logarithm based protocol.

We present and ILC protocol for arithmetic circuit satisfiability which consists of 5 moves. This protocol is the similar to the protocol in [11], but is rewritten slightly as an ILC protocol.

The relation corresponding to the proof is

$$\mathscr{R}_{\mathrm{AC2}} = \left\{ \begin{array}{c} (\sigma_{\mathsf{ILC}}\ , u, w);\ \sigma_{\mathsf{ILC}} = (\mathbb{F}, n), \\ u = \left(\{\mathbf{w}_{q,a,i}, \mathbf{w}_{q,b,i}, \mathbf{w}_{q,c,i}\}_{q\in[Q],i\in[m]}, \{K_q\}_{q\in[Q]}\right), \\ w = \{\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i\in m} : \\ \forall q \in [Q], \forall i \in [m], \quad \mathbf{w}_{q,a,i}, \mathbf{w}_{q,b,i}, \mathbf{w}_{q,c,i} \in \mathbb{F}^n \\ \forall i \in [m], \quad \mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i \in \mathbb{F}^n, \quad \mathbf{a}_i \circ \mathbf{b}_i = \mathbf{c}_i \\ \wedge \quad \forall q \in [Q], \quad \sum_{q=1}^{Q} \mathbf{w}_{q,a,i} \cdot \mathbf{a}_i + \sum_{q=1}^{Q} \mathbf{w}_{q,b,i} \cdot \mathbf{b}_i + \sum_{q=1}^{Q} \mathbf{w}_{q,c,i} \cdot \mathbf{c}_i = K_q \end{array} \right\}$$

This relation has been modified slightly in comparison with the previous relation $\mathscr{R}_{\mathrm{AC1}}$ used for the previous two arithmetic circuit protocols. The only changes are in the setup information. The previous protocols used Lagrange polynomials, so interpolation points $(z_1, \ldots, z_m)$ were included in the setup material. The previous protocols also required the prover to commit to vectors of lengths n and Q, whereas this protocol only requires commitments to vectors of length n.

Motivation Although 8.4 already presents a protocol for arithmetic circuit satisfiability, it does not seem to admit further optimisation techniques or improvements. The protocol in this section reduces checking arithmetic circuit satisfiability to a scalar product check on committed values. If one considers the scalar-product argument of Chapter 10 as a special opening protocol for scalar products on Pedersen commitments, one can combine the two protocols to produce a protocol for arithmetic circuit satisfiability with logarithmic communication complexity, as in [9], and later in [1].

Overview The argument works by folding both the Hadamard matrix product and the linear constraints of Section 3.2 into a single polynomial equation, where a Laurent polynomial has 0 as its constant term, and use an ILC protocol to prove that this is the case. We can optionally integrate the inner product argument of Section 10.1 into the compiled proof to reduce communication.

Our technique improves on the efficiency of arguments such as [7] by making two main changes, each resulting in efficiency improvements.

- We do not need commitments to the input and output wires of addition gates. We handle addition gates with linear consistency equations thus yielding a significant performance improvement proportional to the number of addition gates. This parallels [34] who also manage to eliminate addition gates when constructing Quadratic Arithmetic Programs from circuits.

- We avoid black-box reductions to zero-knowledge arguments for generic linear algebra statements and instead design an argument directly for arithmetic circuit satisfiability. As a result, our square-root argument has only 5 moves, while the argument from [7] requires 7 moves. We note that [8] reduced the complexity of [7] to 5 moves as well, but at a significant computational overhead whereas we also reduce the computational cost.

These improvements give us a square root communication complexity with respect to the number of multiplication gates in the circuit. This is because for a circuit with $N = mn$ multiplication gates, the prover makes $3m$ commitments to wire values in his first move, and later provides an opening consisting of n field elements to a homomorphic combination of these commitments. Optimising the parameters by choosing $m \approx n \approx \sqrt{N}$ leads to square root complexity.

Below we give a first informal exposition of our arguments, and follow with a formal description.

8.5.1 Idea

Let us consider the relation which encodes arithmetic circuit satisfiability.

$$\mathscr{R}_{\text{AC2}} = \left\{ \begin{array}{c} (\sigma_{\mathsf{ILC}}, u, w);\ \sigma_{\mathsf{ILC}} = (\mathbb{F}, n), \\ u = \left(\{\mathbf{w}_{q,a,i}, \mathbf{w}_{q,b,i}, \mathbf{w}_{q,c,i}\}_{q\in[Q], i\in[m]}, \{K_q\}_{q\in[Q]}\right), \\ w = \{\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i\in m} : \\ \forall q \in [Q], \forall i \in [m], \quad \mathbf{w}_{q,a,i}, \mathbf{w}_{q,b,i}, \mathbf{w}_{q,c,i} \in \mathbb{F}^n \\ \forall i \in [m], \quad \mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i \in \mathbb{F}^n, \quad \mathbf{a}_i \circ \mathbf{b}_i = \mathbf{c}_i \\ \wedge \quad \forall q \in [Q], \quad \sum_{q=1}^{Q} \mathbf{w}_{q,a,i} \cdot \mathbf{a}_i + \sum_{q=1}^{Q} \mathbf{w}_{q,b,i} \cdot \mathbf{b}_i + \sum_{q=1}^{Q} \mathbf{w}_{q,c,i} \cdot \mathbf{c}_i = K_q \end{array} \right\}$$

This contains $N = mn$ multiplication constraints, and Q linear consistency constraints.

Let Y be a formal indeterminate. We will reduce the $N+Q$ equations above to a single polynomial equation in Y by embedding each equation into a distinct power of Y. In our argument we will then require the prover to prove that this single equation holds when replacing Y by a random challenge received from the verifier.

Let $\mathbf{Y}'$ denote the vector $(Y^m, \dots, Y^{mn})$ and $\mathbf{Y}$ denote $(Y, Y^2, \dots, Y^m)$. Then, we can multiply (3.1) by $\mathbf{Y}$ from the left and $\mathbf{Y}'^T$ on the right to obtain $\mathbf{Y}(A \circ B)\mathbf{Y}'^T = \mathbf{Y}C\mathbf{Y}'^T$, or equivalently

$$\sum_{i=1}^{m} Y^i (\mathbf{a}_i \circ \mathbf{b}_i) \cdot \mathbf{Y}' = \sum_{i=1}^{m} Y^i (\mathbf{c}_i \cdot \mathbf{Y}')$$

Since $(\mathbf{a} \circ \mathbf{b}) \cdot \mathbf{Y}' = \mathbf{a} \cdot (\mathbf{b} \circ \mathbf{Y}')$, we obtain the following expression

$$\sum_{i=1}^{m} \mathbf{a}_i \cdot (\mathbf{b}_i \circ \mathbf{Y}') Y^i = \left(\sum_{i=1}^{m} \mathbf{c}_i Y^i \cdot \mathbf{Y}' \right)$$

This is easily seen to be equivalent to (3.1), because $(\mathbf{a}_i)_j (\mathbf{b}_i)_j = (\mathbf{c}_i)_j$ appears in the coefficients of Y^{i+jm}, and $i + jm$ takes every value from $m+1$ to $M = N + m$ exactly once.

Moreover, the Q linear constraints on the wires are satisfied if and only if

$$\sum_{q=1}^{Q} \left(\sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{w}_{q,a,i} + \sum_{i=1}^{m} \mathbf{b}_i \cdot \mathbf{w}_{q,b,i} + \sum_{i=1}^{m} \mathbf{c}_i \cdot \mathbf{w}_{q,c,i} \right) Y^q = \sum_{q=1}^{Q} K_q Y^q$$

since the qth constraint arises from comparing the coefficients of Y^q. Combining the two polynomial equations by adding them after multiplying the latter by Y^M, and swapping summations, we see that the circuit is satisfied if and only if

$$\left(\sum_{i=1}^{m} \mathbf{a}_i \cdot (\mathbf{b}_i \circ \mathbf{Y}')Y^i\right) + \sum_{i=1}^{m} \mathbf{a}_i \cdot \left(\sum_{q=1}^{Q} \mathbf{w}_{q,a,i} Y^{M+q}\right) + \sum_{i=1}^{m} \mathbf{b}_i \cdot \left(\sum_{q=1}^{Q} \mathbf{w}_{q,b,i} Y^{M+q}\right)$$
$$+ \sum_{i=1}^{m} \mathbf{c}_i \cdot \left(-Y^i \mathbf{Y}' + \sum_{q=1}^{Q} \mathbf{w}_{q,c,i} Y^{M+q}\right) \quad = \quad \left(\sum_{q=1}^{Q} K_q Y^{M+q}\right)$$

Let us define

$$\mathbf{w}_{a,i}(Y) = \sum_{q=1}^{Q} \mathbf{w}_{q,a,i} Y^{M+q} \qquad \mathbf{w}_{b,i}(Y) = \sum_{q=1}^{Q} \mathbf{w}_{q,b,i} Y^{M+q}$$
$$\mathbf{w}_{c,i}(Y) = -Y^i \mathbf{Y}' + \sum_{q=1}^{Q} \mathbf{w}_{q,c,i} Y^{M+q} \qquad K(Y) = \sum_{q=1}^{Q} K_q Y^{M+q}$$

Then the circuit is satisfied if and only if

$$\sum_{i=1}^{m} \mathbf{a}_i \cdot (\mathbf{b}_i \circ \mathbf{Y}')Y^i + \sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{w}_{a,i}(Y) + \sum_{i=1}^{m} \mathbf{b}_i \cdot \mathbf{w}_{b,i}(Y) + \sum_{i=1}^{m} \mathbf{c}_i \cdot \mathbf{w}_{c,i}(Y) - K(Y) = 0 \tag{8.2}$$

In the argument, the prover will commit to $\mathbf{a}_i, \mathbf{b}_i$ and $\mathbf{c}_i$. The verifier will then issue a random challenge $y \leftarrow \mathbb{F}^*$ and the prover will convince the verifier that the committed values satisfy Eq. 8.2, evaluated on y. If the committed values do not satisfy the polynomial equation, the probability the equality holds for a random y is negligible, so the prover is unlikely to be able to convince the verifier.

In order to show that (8.2) is satisfied, we craft a special Laurent polynomial $t(X)$ in a second formal indeterminate X, whose constant coefficient is exactly twice the left-hand side of (8.2). Therefore, this polynomial will have zero constant term if and only if (8.2) is satisfied. We define

$$\begin{aligned}
\mathbf{r}(X) &:= \sum_{i=1}^{m} \mathbf{a}_i y^i X^i + \sum_{i=1}^{m} \mathbf{b}_i X^{-i} + X^m \sum_{i=1}^{m} \mathbf{c}_i X^i + \mathbf{d} X^{2m+1} \\
\mathbf{s}(X) &:= \sum_{i=1}^{m} \mathbf{w}_{a,i}(y) y^{-i} X^{-i} + \sum_{i=1}^{m} \mathbf{w}_{b,i}(y) X^i + X^{-m} \sum_{i=1}^{m} \mathbf{w}_{c,i}(y) X^{-i} \\
\mathbf{r}'(X) &:= \mathbf{r}(X) \circ \mathbf{y}' + 2\mathbf{s}(X) \\
t(X) &:= \mathbf{r}(X) \cdot \mathbf{r}'(X) - 2K(y)
\end{aligned}$$

Here $\mathbf{y}'$ is the vector $\mathbf{Y}'$ evaluated at y, and $\mathbf{d}$ is a blinding vector consisting of random scalars that the prover commits to in the first round. In the protocol, the prover will reveal $\mathbf{r}(x)$ for a randomly chosen challenge $x \in \mathbb{F}^*$, and the blinding vector $\mathbf{d}$ ensures that we can reveal $\mathbf{r}(x)$ without leaking information about $\mathbf{a}_i, \mathbf{b}_i$ and $\mathbf{c}_i$. We also observe that $\mathbf{s}(x)$ is efficiently computable from public information about the circuit and the challenges.

We have designed these polynomials such that the constant term of $\mathbf{r} \cdot (\mathbf{r} \circ \mathbf{y}')$ is equal to $2\sum_{i=1}^{m} \mathbf{a}_i \cdot (\mathbf{b}_i \circ \mathbf{y}') y^i$ and the constant term of $\mathbf{r} \cdot \mathbf{s}$ is equal to $\sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{w}_{a,i}(y) + \sum_{i=1}^{m} \mathbf{b}_i \cdot \mathbf{w}_{b,i}(y) + \sum_{i=1}^{m} \mathbf{c}_i \cdot \mathbf{w}_{c,i}(y)$. We conclude that the constant term of $t(X)$ is exactly twice the left-hand side of (8.2), and is therefore zero if and only if the circuit is satisfied.

We are now in a position to describe a protocol with square root communication complexity.

The prover first commits to vectors $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$ and $\mathbf{d}$ and the verifier replies with a challenge $y \leftarrow \mathbb{F}^*$. The prover computes $t(X)$ and commits to it by committing to each coefficient using the ILC functionality. Then, the verifier selects a random challenge $x \leftarrow \mathbb{F}^*$ and uses ILC queries to get $\mathbf{r}(x)$ and an evaluation $v = t(x)$.

The verifier computes $\mathbf{s}(x), \mathbf{r}'(x)$ and K, then checks if $v = \mathbf{r}(x) \cdot \mathbf{r}'(x) - 2K$. The verifier accepts the argument if both checks are satisfied.

8.5.2 Protocol Description

Common input: Setup information $\sigma_{\mathsf{ILC}} = (\mathbb{F}, n)$. The description of an arithmetic circuit $u = \left(\{\mathbf{w}_{q,a,i}, \mathbf{w}_{q,b,i}, \mathbf{w}_{q,c,i}\}_{q \in [Q], i \in [m]}, \{K_q\}_{q \in [Q]}\right)$.

Prover's witness: Satisfying assignments $w = \{\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i \in m}$ to the wires of the circuit.

Protocol:

P: The prover chooses $\mathbf{d} \leftarrow \mathbb{F}^n$ uniformly at random, and commits to it using `commit`. For $i \in [m]$, commit to $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$ to `commit`.

V: The verifier randomly selects $y \leftarrow \mathbb{F}^*$ and sends it to the prover.

As argued before, the circuit determines vectors of polynomials $\mathbf{w}_{a,i}(Y)$, $\mathbf{w}_{b,i}(Y)$, $\mathbf{w}_{c,i}(Y)$ and $K(Y)$ such that C is satisfiable if and only if

$$\sum_{i=1}^{m} \mathbf{a}_i \cdot (\mathbf{b}_i^T \circ \mathbf{Y}')Y^i + \sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{w}_{a,i}(Y) + \sum_{i=1}^{m} \mathbf{b}_i \cdot \mathbf{w}_{b,i}(Y) + \sum_{i=1}^{m} \mathbf{c}_i \cdot \mathbf{w}_{c,i}(Y) = K(Y)$$

where $\mathbf{Y}' = (Y^m, \ldots, Y^{mn})$. Given y, both the prover and verifier can compute $K = K(y)$, $\mathbf{w}_{a,i} = \mathbf{w}_{a,i}(y)$, $\mathbf{w}_{b,i} = \mathbf{w}_{b,i}(y)$ and $\mathbf{w}_{c,i} = \mathbf{w}_{c,i}(y)$.

P: The prover computes Laurent polynomials $\mathbf{r}, \mathbf{s}, \mathbf{r}'$, which have vector coefficients, and Laurent polynomial t, in the indeterminate X.

$$\begin{aligned}
\mathbf{r}(X) &= \sum_{i=1}^{m} \mathbf{a}_i y^i X^i + \sum_{i=1}^{m} \mathbf{b}_i X^{-i} + X^m \sum_{i=1}^{m} \mathbf{c}_i X^i + \mathbf{d} X^{2m+1} \\
\mathbf{s}(X) &= \sum_{i=1}^{m} \mathbf{w}_{a,i} y^{-i} X^{-i} + \sum_{i=1}^{m} \mathbf{w}_{b,i} X^i + X^{-m} \sum_{i=1}^{m} \mathbf{w}_{c,i} X^{-i} \\
\mathbf{r}'(X) &= \mathbf{r}(X) \circ \mathbf{y}' + 2\mathbf{s}(X) \\
t(X) &= \mathbf{r}(X) \cdot \mathbf{r}'(X) - 2K = \sum_{k=-3m, k \neq 0}^{4m+2} t_k X^k
\end{aligned}$$

When the wires $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$ correspond to a satisfying assignment, the Laurent polynomial $t(X)$ will have constant term $t_0 = 0$.

The prover commits to $t(X)$ by committing to each coefficient t_k using a separate `commit` command.

V: The verifier randomly selects $x \leftarrow \mathbb{F}^*$ and makes an `open` queries to obtain

$$\mathbf{r} = \sum_{i=1}^{m} \mathbf{a}_i x^i y^i + \sum_{i=1}^{m} \mathbf{b}_i x^{-i} + x^m \sum_{i=1}^{m} \mathbf{c}_i x^i + \mathbf{d} x^{2m+1}$$

$$v = \sum_{k=-3m,k\neq 0}^{4m+2} t_k x^k$$

The verifier computes $\mathbf{r}' = \mathbf{r} \circ \mathbf{y}' + 2\mathbf{s}(x)$. The verifier accepts only if the following `check` is satisfied.

$$\mathbf{r} \cdot \mathbf{r}' - 2K \stackrel{?}{=} v$$

8.5.3 Security Analysis.

Theorem 18 *The argument of subsection 8.5.2 for satisfiability of an arithmetic circuit has perfect completeness, perfect special honest verifier zero-knowledge and statistical witness-extended emulation for extracting either a breach of the binding property of the commitment scheme or a witness for the satisfiability of the circuit.*

Proof Perfect completeness follows by inspection and using the fact that the polynomial commitment protocol and inner product argument also have perfect completeness.

For perfect special honest verifier zero-knowledge we are given $y, x \in \mathbb{F}^*$, which allows us to compute $\mathbf{w}_{a,i}, \mathbf{w}_{b,i}, \mathbf{w}_{c,i}$ and K from the circuit. The simulator picks $\mathbf{r} \leftarrow \mathbb{F}^n$. To see that the simulated components have the same distribution as a real argument observe $\mathbf{r}$ is uniformly random. Finally, the simulator sets the output of all `check` queries to $\top$.

It remains to show that we have knowledge soundness. The ILC knowledge extractor already has access to all messages sent between the prover and ILC. It remains to show that if the committed values are not a valid witness, then there is a negligible probability that the verifier will accept. Recall that the verifier makes an `open` query in order to obtain the following value

$$\mathbf{r} = \sum_{i=1}^{m} \mathbf{a}_i x^i y^i + \sum_{i=1}^{m} \mathbf{b}_i x^{-i} + x^m \sum_{i=1}^{m} \mathbf{c}_i x^i + \mathbf{d} x^{2m+1}$$

then computes $\mathbf{r}' = \mathbf{r} \circ \mathbf{y}' + 2\mathbf{s}(x)$. Substitute these values into the verifier's `check` query $\mathbf{r} \cdot \mathbf{r}' - 2K \overset{?}{=} \sum_{k=-3m,k\neq 0}^{4m+2} t_k x^k$. Following the explanation before the description of the protocol, we obtain a polynomial equation in x and y of the following form:

$$\left(2\sum_{i=1}^{m} \mathbf{a}_i \cdot (\mathbf{b}_i \circ \mathbf{y}')y^i + \sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{w}_{a,i}(y) + \sum_{i=1}^{m} \mathbf{b}_i \cdot \mathbf{w}_{b,i}(y) + \sum_{i=1}^{m} \mathbf{c}_i \cdot \mathbf{w}_{c,i}(y) \right) + \sum_{k=-3m,k\neq 0}^{4m+2} t'_k x^k = 0$$

The verifier only accepts if the equation holds. By assumption $\mathscr{P}^*_{\text{prod}}$ is deterministic, and we know when it made it's commitments. Hence, all of the terms in $\mathbf{a}_i$, $\mathbf{b}_i$ and $\mathbf{c}_i$ are constants, and the t'_k terms are functions of y. We can now apply Lemma 1. If the circuit is not satisfied, then the coefficient of some power of y in the equation above will be non-zero, which means that we have F. The verifier can only accept if we have equality, so this shows that the probability that the committed values do not satisfy the circuit but that the verifier still accepts is negligible, which proves knowledge soundness. ■

Lemma 19 *The protocol has tree-special soundness, and the extraction algorithm χ is a linear map.*

Proof The verifier sees the following queries as part of the protocol.

$$\mathbf{r} = \sum_{i=1}^{m} \mathbf{a}_i x^i y^i + \sum_{i=1}^{m} \mathbf{b}_i x^{-i} + x^m \sum_{i=1}^{m} \mathbf{c}_i x^i + \mathbf{d} x^{2m+1}$$

$$\mathbf{r} \cdot \mathbf{r}' - 2K \overset{?}{=} \sum_{k=-3m,k\neq 0}^{4m+2} t_k x^k$$

After relabelling the variables, these all correspond to cases of Lemma 4. Therefore, given the verifier's view for sufficiently many distinct values of x and y, one can recover the committed polynomial using a linear map. ■

Lemma 20 *With high probability, the* ILC *query matrix for the protocol has full rank, and fewer rows than columns.*

Proof The rows of the query matrix correspond to the number of ILC queries made by the verifier. The columns correspond to vectors which the prover has committed to. In this protocol, there are a constant number of queries, including the queries from the polynomial commitment sub-protocol, so there are clearly fewer queries than commitments whenever m is large enough. The query matrix has full rank. This is because the rows corresponding to each query operate on different commitments. Since each row operates on different commitments to the others, the entire query matrix has full rank. With high probability over the choices of x and z, all queries are non-trivial. ■

8.5.4 Efficiency

Communication The argument above has the prover send $O(m)$ elements to ILC. Setting $m \approx \sqrt{N}$, $n \approx \sqrt{N}$ will minimise the communication complexity in the compiled proof.

Computation The main computational cost for the prover is computing $t(X)$ using FFT-based techniques, which costs $O(mn \log m)$ multiplications in $\mathbb{F}$. The main cost in the verification is computing $\mathbf{s}(X)$ given the description of the circuit which requires in the worst case Qn multiplications in $\mathbb{F}$, considering arbitrary fan-in addition gates. In case of $O(N)$-size circuits with fan-in 2 gates, computing $\mathbf{s}(X)$ requires $O(N)$ multiplications. Evaluating $\mathbf{s}(x)$ requires $3N$ multiplications.

Chapter 9

Compiling Ideal Linear Commitment Protocols into Standard Zero-Knowledge Proofs

Remark When considering ILC protocols with multiple fixed vector lengths, as explained earlier, one can either define a new ILC channel consisting of many copies of ILC channels of single lengths to handle the issue, or simply pad all vectors to the length of the maximum vector. In the first case, we obtain compilation proofs for the new channel simply by running the compilation on the ILC protocol for each single-length ILC channel that makes up the multiple length channel.

9.1 Compiling into Hash-Based Proofs

In this section, we show how to compile a proof of knowledge with straight-line extraction for relation $\mathscr{R}$ over the communication channel ILC into a proof of knowledge for the same relation over the standard channel, based on the existence of collision-resistant hash functions.

The compilation proof given in this section comes from joint work in [16] with Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi and Sune K. Jakobsen. In particular, Sune K Jakobsen is responsible for the technical lemmas in the compilation proof, but the proof and compilation have been adapted for use with our modified ILC model and optimisations to the compilation method.

As well as the fact that we can instantiate protocols designed for homomorphic commitment schemes without any homomorphic commitments, and showing that the ILC model really is very useful and general, the resulting protocols have the potential to be extremely efficient in practice. Ligero [15], a similar protocol which is also based on hashes and error-correcting codes with an additional special property, has been implemented in C++ and provides evidence for the deployability of interactive protocols based on hash functions and error-correcting codes.

The idea behind the compilation of an ILC proof is that instead of committing to vectors $\mathbf{v}_\tau$ using `commit`, the prover encodes each vector $\mathbf{v}_\tau$ as $\mathsf{E}_{\mathscr{C}}(\mathbf{v}_\tau)$ using a linear error-correcting code $\mathsf{E}_{\mathscr{C}}$. In any given round, we can think of the codewords as rows $\mathsf{E}_{\mathscr{C}}(\mathbf{v}_\tau)$ in a matrix $\mathsf{E}_{\mathscr{C}}(V)$. However, instead of committing to the rows of the matrix, the prover commits to the columns of the matrix. When the verifier wants to make an `open` query for a linear combination of the original vectors, he sends the coefficients $\mathbf{q} = (q_1, \ldots, q_t)$ of the linear combination to the prover, and the prover responds with the linear combination $\mathbf{v}_{(\mathbf{q})} \leftarrow \mathbf{q}V$.

We also add a check, where the verifier sends an extra random linear combination $\gamma \in \mathbb{F}^t$ to ensure that if a malicious prover commits to values of $\mathbf{e}_\tau$ that are far from being codewords, the verifier will most likely reject. The reason the challenges $\mathbf{q}$ and $\mathbf{q}'$ from the ILC proof are not enough to ensure this is that they are not chosen uniformly at random. One could, for instance, imagine that there was a vector $\mathbf{v}_\tau$ that was never queried in a non-trivial way, and hence the prover could choose it to be far from a codeword. To make sure this extra challenge γ does not reveal information to the verifier, the prover picks a random blinding vector $\mathbf{v}_0$, which is added as the first row of V and will be added to the linear combination of the challenge γ.

Notice that we will use the notation $\mathbf{v}_{(\mathbf{q})}$, and later on $\mathbf{v}_{(\gamma)}$, to denote vectors that depend on $\mathbf{q}$ and γ: the $\mathbf{q}$ and γ are not indices. Now, to spot check that the prover is not giving a wrong $\mathbf{v}_{(\mathbf{q})}$, the verifier may request the jth element of each committed codeword $\mathbf{e}_\tau$. This corresponds to revealing the jth column of error-corrected matrix $\mathsf{E}_{\mathscr{C}}(V)$. Since the code $\mathsf{E}_{\mathscr{C}}$ is linear, the revealed elements should satisfy $\mathsf{E}_{\mathscr{C}}(\mathbf{v}_{(\mathbf{q})})_j = \sum_{\tau=1}^{t} q_\tau \mathsf{E}_{\mathscr{C}}(\mathbf{v}_\tau)_j = \mathbf{q}(\mathsf{E}_{\mathscr{C}}(V)|_j)$. The verifier will spot check on multiple columns, so that if the code has sufficiently large minimum distance and the prover gives a wrong $\mathbf{v}_{(\mathbf{q})}$, then with overwhelming probability, the verifier will open at least one column j where the above equality does not hold.

Revealing entries in a codeword may leak information about the encoded vector. To get SHVZK, instead of using $\mathsf{E}_{\mathscr{C}}$, we use a randomized encoding $\tilde{\mathsf{E}}_{\mathscr{C}}$ defined by $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v};\mathbf{r}) = (\mathsf{E}_{\mathscr{C}}(\mathbf{v}) + \mathbf{r}, \mathbf{r})$. This doubles the code-length to $2n$ but ensures that when you reveal entry j, but not entry $j+n$, then the verifier only learns a random field element. The spot checking technique using $\tilde{\mathsf{E}}_{\mathscr{C}}$ is illustrated in Fig. 9.1. In the following, we use the notation $\mathbf{e}_\tau = (\mathsf{E}_{\mathscr{C}}(\mathbf{v}_\tau) + \mathbf{r}_\tau, \mathbf{r}_\tau)$ and $E = (\mathsf{E}_{\mathscr{C}}(V) + R, R)$.

The original compilation given in [16] handled an earlier version of the ILC model without `check` queries. We must discuss how to treat these queries. The verifier makes `check` queries $\mathbf{q}' = (q'_1, \ldots, q'_t)$ of linear combinations of committed vectors. Now, `check` queries handle the case where the verifier can compute what $\mathbf{v}'_{(\mathbf{q})} \leftarrow \mathbf{q}'V$ should be by themselves, using the results of other `open` queries. Therefore, the prover need not send $\mathbf{v}'_{(\mathbf{q})}$ to the verifier. The prover, given $\mathbf{q}'$, must still reveal $\mathbf{r}'_{(\mathbf{q})} = \mathbf{q}'R$. The verifier now asks for openings of 2λ columns $J = \{j_1, \ldots, j_{2\lambda}\}$ in E and verifies for these columns that $\mathbf{q}'E|_J = \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}'_{(\mathbf{q})}; \mathbf{r}'_{(\mathbf{q})})|_J$, using the value of $\mathbf{v}'_{(\mathbf{q})}$ that they have computed using other openings. To avoid revealing any information about $\mathsf{E}_{\mathscr{C}}(V)$, we must ensure that $\forall j \in [n] : j \in J \Rightarrow j+n \notin J$. If the spot checks pass, the verifier believes that $\mathbf{v}'_{(\mathbf{q})} = \mathbf{q}'V$.

$$\begin{array}{ccc}
\begin{pmatrix} \mathbf{v}_0 \\ \vdots \\ \mathbf{v}_t \end{pmatrix} & \xrightarrow{\tilde{\mathsf{E}}_{\mathscr{C}}} & \left(\begin{array}{c|c} \mathsf{E}_{\mathscr{C}}(\mathbf{v}_0)+\mathbf{r}_0 & \mathbf{r}_0 \\ \vdots & \vdots \\ \mathsf{E}_{\mathscr{C}}(\mathbf{v}_t)+\mathbf{r}_t & \mathbf{r}_t \end{array}\right) \\
\mathbf{q}\downarrow & & \begin{array}{cc} \mathbf{q}\downarrow_{j_1} \cdots \mathbf{q}\downarrow_{j_\lambda} & \mathbf{q}\downarrow_{j_{\lambda+1}} \cdots \mathbf{q}\downarrow_{j_{2\lambda}} \end{array} \\
\begin{pmatrix} \mathbf{v}_{(\mathbf{q})} \end{pmatrix} & \xrightarrow{\tilde{\mathsf{E}}_{\mathscr{C}}} & \left(\begin{array}{c|c} \mathsf{E}_{\mathscr{C}}(\mathbf{v}_{(\mathbf{q})})+\mathbf{r}_{(\mathbf{q})} & \mathbf{r}_{(\mathbf{q})} \end{array}\right)
\end{array}$$

Figure 9.1: Vectors $\mathbf{v}_\tau$ organized in matrix V are encoded row-wise as matrix $E = \tilde{\mathsf{E}}_{\mathscr{C}}(V;R)$. The vertical line in the right matrix and vector denotes concatenation of matrices respectively vectors. The prover commits to each column of E. When the prover, given $\mathbf{q}$ and $\mathbf{q}'$, wants to reveal the linear combination $\mathbf{v}_{(\mathbf{q})} = \mathbf{q}V$ she also reveals $\mathbf{r}_{(\mathbf{q})} = \mathbf{q}R$ and $\mathbf{r}'_{(\mathbf{q})} = \mathbf{q}'R$. The verifier now asks for openings of 2λ columns $J = \{j_1, \ldots, j_{2\lambda}\}$ in E and verifies for these columns that $\mathbf{q}E|_J = \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\mathbf{q})};\mathbf{r}_{(\mathbf{q})})|_J$ and $\mathbf{q}'E|_J = \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}'_{(\mathbf{q})};\mathbf{r}'_{(\mathbf{q})})|_J$, where the verifier computed $\mathbf{v}'_{(\mathbf{q})}$ by themselves. To avoid revealing any information about $\mathsf{E}_{\mathscr{C}}(V)$, we must ensure that $\forall j \in [n] : j \in J \Rightarrow j+n \notin J$. If the spot checks pass, the verifier believes that $\mathbf{v}_{(\mathbf{q})} = \mathbf{q}V$ and $\mathbf{v}'_{(\mathbf{q})} = \mathbf{q}'V$.

9.1.1 Reed-Solomon Codes

In order to show that we can instantiate our construction efficiently, we give a short description of Reed-Solomon codes. Reed Solomon codes are a simple example of linear codes with a constant relative minimum distance and an efficient (quasilinear) encoding algorithm. As such, they are a good choice for instantiating the error-correcting codes in our proofs.

A Reed-Solomon code over a field $\mathbb{F}$ is defined as follows. Fix distinct points $a_1, \ldots, a_n \in \mathbb{F}$.

$$\{(p(a_1), \ldots, p(a_n)) : p(X) \text{ is a polynomial of degree at most } k \text{ over } \mathbb{F}\}$$

We can encode a vector of length k by embedding the coefficients into the coefficients of a polynomial p, and then using the codeword corresponding to p. Reed-Solomon codes have minimum distance $n-k+1$. Choosing $a_1, \ldots, a_n$ to be suitable roots of unity in $\mathbb{F}$, the codeword can be computed in $O(n \log n)$ field operations. To achieve constant relative minimum distance, we can choose $n = 2k-1$, for example.

Note that we will not use the multiplicative properties of Reed-Solomon codes. In our compilation, the verifier only does linear calculations on elements of codewords. The one place where the verifier may do non-linear calculations on values sent by the prover is as part of `check` queries, but here, the operations are done prior to encoding. This is important to note. Otherwise, the algebraic degree of the ILC verifier might lead to restrictions on our choices of parameters, since for the product of Reed-Solomon codewords to be a valid codeword, the degrees of the polynomials associated with the codewords must not be too high.

9.1.2 Construction

Let $(\mathscr{K}_{\mathsf{ILC}}, \mathscr{P}_{\mathsf{ILC}}, \mathscr{V}_{\mathsf{ILC}})$ be a *non-adaptive* μ-round SHVZK proof of knowledge with straight-line extraction over ILC for a relation $\mathscr{R}$. Here, non-adaptive means that the verifier waits until the last round before querying linear combinations of vectors and they are queried all at once instead of the queries depending on each other.[1] Let $\mathrm{Gen}_{\mathsf{E}_{\mathscr{C}}}$ be a generator that given field $\mathbb{F}$ and length parameter k outputs a constant rate linear code $\mathsf{E}_{\mathscr{C}}$ that is efficiently computable given its description, and has linear minumum distance. Define the $\tilde{\mathsf{E}}_{\mathscr{C}}$ with code length $2n$ as above: $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v};\mathbf{r}) = (\mathsf{E}_{\mathscr{C}}(\mathbf{v})+\mathbf{r},\mathbf{r})$. Finally, let $(\mathsf{Setup},\mathsf{Commit})$ be a non-interactive commitment scheme.

We now define a proof of knowledge $(\mathscr{K},\mathscr{P},\mathscr{V})$ in Fig. 9.2, where we use the following notation: given matrices $V_1,\ldots,V_\mu$, $R_1,\ldots,R_\mu$ and $E_1,\ldots,E_\mu$ we define

$$V = \begin{pmatrix} V_1 \\ \vdots \\ V_\mu \end{pmatrix} \qquad R = \begin{pmatrix} R_1 \\ \vdots \\ R_\mu \end{pmatrix} \qquad E = \begin{pmatrix} E_1 \\ \vdots \\ E_\mu \end{pmatrix}.$$

The matrices $V_1,\ldots,V_\mu$ are formed by the row vectors $\mathscr{P}_{\mathsf{ILC}}$ commits to, and we let $t_1,\ldots,t_\mu$ be the numbers of vectors in each round, i.e., for all i we have $V_i \in \mathbb{F}^{t_i \times k}$.

We say that a set $J \subset [2n]$ is *allowed* if $|J \cap [n]| = \lambda$ and $|J \setminus [n]| = \lambda$ and there is no $j \in J$ such that $j+n \in J$. In the following we will always assume codewords have

[1]The construction can be easily modified to an adaptive ILC proof. For each round of queries in the ILC proof, there will one extra round in the compiled proof.

length $n \geq 2\lambda$. We use $\tilde{\mathsf{E}}_{\mathscr{C}}(V;R)$ to denote the function that applies $\tilde{\mathsf{E}}_{\mathscr{C}}$ *row-wise* to V and R. In the protocol for $\mathscr{V}$, we are using that $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v};\mathbf{r})|_J$ can be computed from just $\mathbf{v}$ and $\mathbf{r}|_{\{j\in[n]:j\in J\vee j+n\in J\}}$. We use $\mathsf{Commit}(E;\mathbf{s})$ to denote the function that applies Commit *column-wise* on E and returns a vector $\mathbf{c}$ of $2n$ commitments. We group all $\mathscr{V}_{\mathsf{ILC}}$'s queries in two matrices $Q \in \mathbb{F}^{\mathrm{qc}\times t}$ and $Q' \in \mathbb{F}^{\mathrm{qc}'\times t}$, where t is the total number of vectors committed to by $\mathscr{P}$, qc is the `open` query complexity of $\mathscr{V}_{\mathsf{ILC}}$, i.e., the total number of linear combinations $\mathbf{q}$ that $\mathscr{V}_{\mathsf{ILC}}$ requests to be opened, and qc$'$ is the `check`query complexity of $\mathscr{V}_{\mathsf{ILC}}$, i.e., the total number of linear combinations $\mathbf{q}'$ that $\mathscr{V}_{\mathsf{ILC}}$ requests to be checked.

Remark For the protocol to be well-defined, we need $n \geq 2\lambda$ so we have sufficiently many columns. Otherwise no allowed J exists. If the output of $\mathsf{E}_{\mathscr{C}}$ is shorter than 2λ, $\mathscr{K}_{\mathsf{IOP}}$ will extend $\mathsf{E}_{\mathscr{C}}$ by a factor $\left\lceil \frac{2\lambda}{n} \right\rceil$ to ensure that output length n satisfies $n \geq 2\lambda$. Here the extension of $\mathsf{E}_{\mathscr{C}}$ is given by $\{(e,\ldots,e) : e \in \mathsf{E}_{\mathscr{C}}\}$, where the number of e's is the factor of the extension. In particular, extending by a factor 1 leaves everything unchanged. Extending by a larger factor keeps the input length constant while extending the output length. Notice that we are interested in efficiency when ℓ is large compared to λ, in which case no extension is needed.

9.1.3 Security Analysis

Theorem 1 (Completeness) *If* $(\mathscr{K}_{\mathsf{ILC}}, \mathscr{P}_{\mathsf{ILC}}, \mathscr{V}_{\mathsf{ILC}})$ *is complete for relation* $\mathscr{R}$ *over* ILC*, then* $(\mathscr{K}, \mathscr{P}, \mathscr{V})$ *in Fig. 9.2 is complete for relation* $\mathscr{R}$.

Proof All the commitment openings are correct, so they will be accepted by the verifier. In the execution of $\langle \mathscr{P}(\sigma,u,w) \longleftrightarrow \mathscr{V}(\sigma,u)\rangle$, the fact that $\mathsf{E}_{\mathscr{C}}$ is linear implies $\tilde{\mathsf{E}}_{\mathscr{C}}$ is linear and hence all the linear checks will be true. If $(\sigma,u,w) \in \mathscr{R}$ then $(\sigma_{\mathsf{ILC}},u,w) \in \mathscr{R}$ and being complete $\langle \mathscr{P}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},u,w) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathscr{V}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},stm)\rangle = 1$ so $\mathscr{V}$'s internal copy of $\mathscr{V}_{\mathsf{ILC}}$ will accept. Thus, in this case, $\langle \mathscr{P}(\sigma,u,w) \longleftrightarrow \mathscr{V}(\sigma,u)\rangle = 1$, which proves completeness. ∎

Theorem 2 (Knowledge Soundness) *If* $(\mathscr{K}_{\mathsf{ILC}}, \mathscr{P}_{\mathsf{ILC}}, \mathscr{V}_{\mathsf{ILC}})$ *is statistically knowledge sound with a straight-line extractor for relation* $\mathscr{R}$ *over* ILC *and* $(\mathsf{Setup}, \mathsf{Commit})$ *is computationally (statistically) binding, then* $(\mathscr{K}, \mathscr{P}, \mathscr{V})$ *as* *constructed above is computationally (statistically) knowledge sound for relation $\mathscr{R}$.*

Proof We prove the computational case. The statistical case is similar.

In order to argue that $(\mathscr{K},\mathscr{P},\mathscr{V})$ is computationally knowledge sound, we will first show that for every DPT $\mathscr{P}^*$ there exists a deterministic (but not necessarily efficient) $\mathscr{P}^*_{\mathsf{ILC}}$ such that for all PPT $\mathscr{A}$ we have

$$\Pr\left[\begin{array}{c}\sigma \leftarrow \mathscr{K}(1^\lambda);(\sigma_{\mathsf{ILC}},\cdot)=\sigma;(u,s)\leftarrow\mathscr{A}(\sigma):\\ \langle\mathscr{P}^*(s)\longleftrightarrow\mathscr{V}(\sigma,u;(\rho_{\mathsf{ILC}},\rho))\rangle=1\\ \wedge\ \langle\mathscr{P}^*_{\mathsf{ILC}}(s,\sigma,u)\overset{\mathsf{ILC}}{\longleftrightarrow}\mathscr{V}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},u;\rho_{\mathsf{ILC}})\rangle=0\end{array}\right]\approx 0. \tag{9.1}$$

Note that the randomness ρ_{ILC} in $\mathscr{V}$ which comes from the internal $\mathscr{V}_{\mathsf{ILC}}$ in line two is the same as the randomness used by $\mathscr{V}_{\mathsf{ILC}}$ in line three.

Our constructed $\mathscr{P}^*_{\mathsf{ILC}}$ will run an internal copy of $\mathscr{P}^*$. When the internal $\mathscr{P}^*$ in round i sends a message $(\mathbf{c}_i,t_i)$, $\mathscr{P}^*_{\mathsf{ILC}}$ will simulate $\mathscr{P}^*$ on every possible continuation of the transcript, and for each $j=1,\dots,2n$ find the most frequently occurring correct opening $((E_i)_j,(\mathbf{s}_i)_j)$ of $(\mathbf{c}_i)_j$. $\mathscr{P}^*_{\mathsf{ILC}}$ will then use this to get matrices E_i^*. For each row $\mathbf{e}^*_\tau$ of these matrices, $\mathscr{P}^*_{\mathsf{ILC}}$ finds a vector $\mathbf{v}_\tau$ and randomness $\mathbf{r}_\tau$ such that $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_\tau,\mathbf{r}_\tau),\mathbf{e}^*_\tau)<\frac{\mathsf{hd}_{\min}}{3}$ if such a vector exists. If for some τ no such vector $\mathbf{v}_\tau$ exists, then $\mathscr{P}^*_{\mathsf{ILC}}$ aborts. Otherwise we let V_i and R_i denote the matrices formed by the row vectors $\mathbf{v}_\tau$ and $\mathbf{r}_\tau$ in round i and $\mathscr{P}^*_{\mathsf{ILC}}$ sends V_i to the ILC. Notice that since the minimum distance of $\tilde{\mathsf{E}}_{\mathscr{C}}$ is at least $\mathsf{hd}_{\min}$, there is at most one such vector $\mathbf{v}_\tau$ for each $\mathbf{e}^*_\tau$.

The internal copy of $\mathscr{P}^*$ will expect to get two extra rounds, where in the first it should receive γ, Q and Q', and should respond with $\mathbf{v}^*_{(\gamma)},\mathbf{r}^*_{(\gamma)},V_{(Q)}$, $R_{(Q)}$ and $R'_{(Q)}$, and in the second it should receive J and send $E_{01}|_J,\mathbf{s}_1|_J,\dots,E_\mu,\mathbf{s}_\mu|_J$. Since $\mathscr{P}^*_{\mathsf{ILC}}$ does not send and receive corresponding messages, $\mathscr{P}^*_{\mathsf{ILC}}$ does not have to run this part of $\mathscr{P}^*$. Of course, for each commitment sent by $\mathscr{P}^*$, these rounds are internally simulated many times to get the most frequent opening. Notice that a $\mathscr{V}_{\mathsf{ILC}}$ communicating over ILC with our constructed $\mathscr{P}^*_{\mathsf{ILC}}$ will, on query Q receive $V_{(Q)}=QV$ from the ILC.

The verifier $\mathscr{V}$ accepts only if its internal copy of $\mathscr{V}_{\mathsf{ILC}}$ accepts. Hence, the only three ways $\langle \mathscr{P}^*(s) \longleftrightarrow \mathscr{V}(\sigma, u; (\rho_{\mathsf{ILC}}, \rho))\rangle$ can accept without $\langle \mathscr{P}^*_{\mathsf{ILC}}(s, \sigma, u) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathscr{V}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, u; \rho_{\mathsf{ILC}})\rangle$ being accepting are

1. if $\mathscr{P}^*$ makes an opening of a commitment that is not its most frequent opening of that commitment, or

2. if $\mathscr{P}^*_{\mathsf{ILC}}$ has an error because for some τ no $\mathbf{v}_\tau, \mathbf{r}_\tau$ with $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_\tau, \mathbf{r}_\tau), \mathbf{e}^*_\tau) < \frac{\mathsf{hd}_{\min}}{3}$ exists, or

3. if $\mathscr{P}^*$ sends some $V^*_{(Q)} \neq V_{(Q)}$.

We will now argue that for each of these three cases, the probability that they happen and $\mathscr{V}$ accepts is negligible.

First Case Since $\mathscr{P}^*$ runs in polynomial time and the commitment scheme is computationally binding, there is only negligible probability that $\mathscr{P}^*$ sends a valid opening that is not the most frequent. Since $\mathscr{V}$ will reject any opening that is not valid, the probability of $\mathscr{V}$ accepting in case 1 is negligible.

Second Case To do so, define the event Err that E^* is such that for some $\gamma^* \in \mathbb{F}^t$ we have $\mathsf{hd}(\tilde{\mathscr{C}}, \gamma^* E^*) \geq \frac{\mathsf{hd}_{\min}}{3}$. Here $\tilde{\mathscr{C}}$ denotes the image of $\tilde{\mathsf{E}}_{\mathscr{C}}$, i.e. $\tilde{\mathscr{C}} = \{(c+r, r) : c \in \mathscr{C}, r \in \mathbb{F}^n\}$. Clearly, if $\mathscr{P}^*_{\mathsf{ILC}}$ returns an error because no $\mathbf{v}_i, \mathbf{r}_i$ with $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_i, \mathbf{r}_i), \mathbf{e}^*_i) < \frac{\mathsf{hd}_{\min}}{3}$ exist then we have Err.

Lemma 21 *Let $\mathbf{e}^*_0, \ldots, \mathbf{e}^*_t \in \mathbb{F}^{2n}$. If Err occurs, then for uniformly chosen $\gamma \in \mathbb{F}^t$, there is probability at most $\frac{1}{|\mathbb{F}|}$ that $\mathsf{hd}(\tilde{\mathscr{C}}, \mathbf{e}^*_0 + \gamma E^*) < \frac{\mathsf{hd}_{\min}}{6}$.*

Proof Assume Err, that is, there exist $\gamma^* \in \mathbb{F}^t$ with $\mathsf{hd}\left(\tilde{\mathscr{C}}, \gamma^* E^*\right) \geq \frac{\mathsf{hd}_{\min}}{3}$. We will show that for any $r \in \mathbb{F}^\times$ we have

$$\mathsf{hd}\left(\tilde{\mathscr{C}}, \mathbf{e}^*_0 + \gamma E^*\right) + \mathsf{hd}\left(\tilde{\mathscr{C}}, e^*_0 + (\gamma + r\gamma^*)E^*\right) \geq \mathsf{hd}\left(\tilde{\mathscr{C}}, \gamma^* E^*\right) \geq \frac{\mathsf{hd}_{\min}}{3}. \qquad (9.2)$$

This implies that at most one of $\mathbf{e}^*_0 + \gamma E^*$ and $\mathbf{e}^*_0 + (\gamma + r\gamma^*)E^*$ can have distance less than $\frac{\mathsf{hd}_{\min}}{6}$ to $\tilde{\mathscr{C}}$. That is, for at most one $\gamma \in \mathbb{F}^t$ in each equivalence class in $\mathbb{F}^t/\gamma^*\mathbb{F}$ can $\mathbf{e}^*_0 + \gamma E^*$ have distance less than $\frac{\mathsf{hd}_{\min}}{6}$ to $\tilde{\mathscr{C}}$. Since each such equivalence class contains $|\mathbb{F}|$ elements, there is probability at most $\frac{1}{|\mathbb{F}|}$ that a random $\gamma \in \mathbb{F}^t$ satisfies $\mathsf{hd}\left(\tilde{\mathscr{C}}, \mathbf{e}_0^* + \gamma E^*\right) < \frac{\mathsf{hd}_{\min}}{6}$.

To finish the proof, we need to prove (9.2). Write $\mathbf{e}_0^* + \gamma E^* = c_1 + v_1$ and $\mathbf{e}_0^* + (\gamma + r\gamma^*)E^* = c_2 + v_2$ with $c_1, c_2 \in \tilde{\mathscr{C}}$ and $\mathsf{wt}(v_1) = \mathsf{hd}\left(\tilde{\mathscr{C}}, \mathbf{e}_0^* + \gamma E^*\right), \mathsf{wt}(v_2) = \mathsf{hd}\left(\tilde{\mathscr{C}}, \mathbf{e}_0^* + (\gamma + r\gamma^*)E^*\right)$. Now

$$\begin{aligned}\gamma^* E^* =& (\mathbf{e}_0^* + (\gamma + r\gamma^*)E^* - (\mathbf{e}_0^* + \gamma E^*))r^{-1} \\ =& (c_2 + v_2 - c_1 - v_1)r^{-1} \\ =& (c_2 - c_1)r^{-1} + (v_2 - v_1)r^{-1}\end{aligned}$$

Here $(c_2 - c_1)r^{-1} \in \tilde{\mathscr{C}}$ and $(v_2 - v_1)r^{-1}$ has at most

$$\mathsf{wt}(v_1) + \mathsf{wt}(v_2) = \mathsf{hd}\left(\tilde{\mathscr{C}}, \mathbf{e}_0^* + \gamma E^*\right) + \mathsf{hd}\left(\tilde{\mathscr{C}}, \mathbf{e}_0^* + (\gamma + r\gamma^*)E^*\right)$$

non-zero elements. This proves inequality (9.2), and hence the lemma. ■

Thus, if Err then with probability at least $1 - \frac{1}{|\mathbb{F}|}$ the vector γ is going to be such that $\mathsf{hd}(\tilde{\mathscr{C}}, \mathbf{e}_0^* + \gamma E^*) \geq \frac{\mathsf{hd}_{\min}}{6}$. If this happens, then for the vectors $(\mathbf{v}_{(\gamma)}^*, \mathbf{r}_{(\gamma)}^*)$ sent by $\mathscr{P}^*$, we must have $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}^*, \mathbf{r}_{(\gamma)}^*), \mathbf{e}_0^* + \gamma E^*) \geq \frac{\mathsf{hd}_{\min}}{6}$. This means that either in the first half of the codeword $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}^*, \mathbf{r}_{(\gamma)}^*)$ or in the second half, there will be at least $\frac{\mathsf{hd}_{\min}}{12}$ values of j where it differs from $\mathbf{e}_0^* + \gamma E^*$. It is easy to see that the λ values of j in one half of $[2n]$ are chosen uniformly and independently at random conditioned on being different.

For each of these j, there is a probability at most $1 - \frac{\mathsf{hd}_{\min}}{12n}$ that $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}, \mathbf{r}_{(\gamma)})_j = \mathbf{e}_{0,j}^* + \gamma E^*|_j$, and since the j's are chosen uniformly under the condition that they are distinct, given that this holds for the first i values, the probability is even smaller for the $i+1$'th. Hence, the probability that it holds for all j in this half is negligible. This shows that the probability that Err happens and $\mathscr{V}$ accepts is negligible.

Third Case Now we turn to the third case, where Err does not happen but $\mathscr{P}^*$ sends a $V_{(Q)}^* \neq V_{(Q)}$. In this case, for all $\gamma^* \in \mathbb{F}^t$, we have $\mathsf{hd}(\tilde{\mathscr{C}}, \sum_{\tau=1}^{t} \gamma_\tau^* \mathbf{e}_\tau^*) < \frac{\mathsf{hd}_{\min}}{3}$. In particular, this holds for the vector γ given by $\gamma_\tau = 1$ and $\gamma_{\tau'} = 0$ for $\tau' \neq \tau$, so the $\mathbf{v}_\tau$'s are well-defined.

For two matrices A and B of the same dimensions, we define their Hamming distance $\mathsf{hd}_2(A,B)$ to be the number of j's such that the jth column of A and jth column of B are different. This agrees with the standard definition of Hamming distance, if we consider each matrix to be a string of column vectors.

Lemma 22 *Assume $\neg Err$ and let V and R be defined as above. Then for any $\mathbf{q} \in \mathbb{F}^t$ there exists an $\mathbf{r}_{(\mathbf{q})}$ with $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{q}V, \mathbf{r}_{(\mathbf{q})}), \mathbf{q}E^*) < \frac{\mathsf{hd}_{\min}}{3}$.*

In particular, for any $V^*_{(Q)} \neq QV$, and any $R^*_{(Q)}$ we have

$$\mathsf{hd}_2\left(\tilde{\mathsf{E}}_{\mathscr{C}}\left(V^*_{(Q)}, R^*_{(Q)}\right), QE^*\right) \geq 2\frac{\mathsf{hd}_{\min}}{3}.$$

Of course, the same statement holds for $\mathbf{q}'$, $\mathbf{r}'_{(\mathbf{q})}$, $V'^*_{(Q)} \neq Q'V$ and $R'^*_{(Q)}$.

Proof Assume that $\neg Err$, that is for all $\mathbf{q} \in \mathbb{F}^t$ we have $\mathsf{hd}(\tilde{\mathscr{C}}, \mathbf{q}E^*) < \frac{\mathsf{hd}_{\min}}{3}$. Informally, we need to strengthen this by showing that the elements in $\tilde{\mathscr{C}}$ that are close to each $\mathbf{q}E^*$, are themselves linear in $\mathbf{q}$.

We have chosen $\mathbf{v}_\tau$'s and $\mathbf{r}_\tau$'s such that $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_\tau, \mathbf{r}_\tau), \mathbf{e}^*_\tau) < \frac{\mathsf{hd}_{\min}}{3}$, and V is the matrix where the τth row is v_τ. We will show by induction on number of non-zero elements $\mathsf{wt}(\mathbf{q})$ in $\mathbf{q}$ that there exists $\mathbf{r}_{(\mathbf{q})}$ with $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{q}V, \mathbf{r}_{(\mathbf{q})}), \mathbf{q}E^*) < \frac{\mathsf{hd}_{\min}}{3}$.

This is trivially true for $\mathsf{wt}(\mathbf{q}) = 0$. For $\mathsf{wt}(\mathbf{q}) = 1$ it follows from our choice of $\mathbf{v}_\tau$. Assume for induction that it is true for all $\mathbf{q}$ with $\mathsf{wt}(\mathbf{q}) \leq \kappa$ and consider a $\mathbf{q}$ with $\mathsf{wt}(\mathbf{q}) \leq 2\kappa$. We can now write $\mathbf{q} = \mathbf{q}_1 + \mathbf{q}_2$ where $\mathsf{wt}(\mathbf{q}_1), \mathsf{wt}(\mathbf{q}_2) \leq \kappa$. By the induction hypothesis, there exists $\mathbf{r}_{(\mathbf{q}_1)}$ such that $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{q}_1V, \mathbf{r}_{(\mathbf{q}_1)}), \mathbf{q}_1E^*) < \frac{\mathsf{hd}_{\min}}{3}$ and similar for $\mathbf{q}_2$. Since $\mathbf{q} = \mathbf{q}_1 + \mathbf{q}_2$ this implies

$$\begin{aligned}
&\mathsf{hd}\left(\tilde{\mathsf{E}}_{\mathscr{C}}\left(\mathbf{q}V, \mathbf{r}_{(\mathbf{q}_1)} + \mathbf{r}_{(\mathbf{q}_2)}\right), \mathbf{q}E^*\right) \\
&\quad = \mathsf{hd}\left(\tilde{\mathsf{E}}_{\mathscr{C}}\left((\mathbf{q}_1 + \mathbf{q}_2)V, \mathbf{r}_{(\mathbf{q}_1)} + \mathbf{r}_{(\mathbf{q}_2)}\right), (\mathbf{q}_1 + \mathbf{q}_2)E^*\right) \\
&\quad \leq \mathsf{hd}\left(\tilde{\mathsf{E}}_{\mathscr{C}}\left(\mathbf{q}_1V, \mathbf{r}_{(\mathbf{q}_1)}\right), \mathbf{q}_1E^*\right) + \mathsf{hd}\left(\tilde{\mathsf{E}}_{\mathscr{C}}\left(\mathbf{q}_2V, \mathbf{r}_{(\mathbf{q}_2)}\right), \mathbf{q}_2E^*\right) \\
&\quad < 2\frac{\mathsf{hd}_{\min}}{3}.
\end{aligned}$$

Since we assume $\neg Err$, we know that there exist *some* $\mathbf{v}_{(\mathbf{q})}$ and $\mathbf{r}_{(\mathbf{q})}$ such that $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\mathbf{q})}, \mathbf{r}_{(\mathbf{q})}), \mathbf{q}E^*) < \frac{\mathsf{hd}_{\min}}{3}$. Now, by the triangle inequality for Hamming distance, this implies

$$\begin{aligned}
&\mathsf{hd}\left(\tilde{\mathsf{E}}_{\mathscr{C}}\left(\mathbf{v}_{(\mathbf{q})},\mathbf{r}_{(\mathbf{q})}\right),\tilde{\mathsf{E}}_{\mathscr{C}}\left(\mathbf{q}V,\mathbf{r}_{(\mathbf{q}_1)}+\mathbf{r}_{(\mathbf{q}_2)}\right)\right)\\
&\quad\le \mathsf{hd}\left(\tilde{\mathsf{E}}_{\mathscr{C}}\left(\mathbf{v}_{(\mathbf{q})},\mathbf{r}_{(\mathbf{q})}\right),\mathbf{q}E^*\right)+\mathsf{hd}\left(\mathbf{q}E^*,\tilde{\mathsf{E}}_{\mathscr{C}}\left(\mathbf{q}V,\mathbf{r}_{(\mathbf{q}_1)}+\mathbf{r}_{(\mathbf{q}_2)}\right)\right)\\
&\quad< \frac{\mathsf{hd}_{\min}}{3}+2\frac{\mathsf{hd}_{\min}}{3}=\mathsf{hd}_{\min}
\end{aligned}$$

Since $\mathsf{hd}_{\min}$ is the minimum distance of $\tilde{\mathsf{E}}_{\mathscr{C}}$, we must have $\mathbf{v}_{(\mathbf{q})}=\mathbf{q}V$, and hence $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{q}V,\mathbf{r}_{(\mathbf{q})}),\mathbf{q}E^*)<\frac{\mathsf{hd}_{\min}}{3}$. This finishes the induction argument.

The triangle inequality for Hamming distance shows that for any $(\mathbf{v}^*_{(\mathbf{q})},\mathbf{r}^*_{(\mathbf{q})})$ with $\mathbf{v}^*_{(\mathbf{q})}\neq\mathbf{q}V$ we have $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}^*_{(\mathbf{q})},\mathbf{r}^*_{(\mathbf{q})}),\mathbf{q}E^*)\ge 2\frac{\mathsf{hd}_{\min}}{3}$. Now for any $V^*_{(Q)}\neq QV$ there is a row τ where the two matrices differ. Let $\mathbf{q}$ be the τth row of Q. Then $\mathsf{hd}(\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}^*_{(\mathbf{q})},\mathbf{r}^*_{(\mathbf{q})}),\mathbf{q}E^*)\ge 2\frac{\mathsf{hd}_{\min}}{3}$ tells us that the τth row of $\tilde{\mathsf{E}}_{\mathscr{C}}(V^*_{(Q)},R^*_{(Q)})$ and τth row of QE^* differs in at least $2\frac{\mathsf{hd}_{\min}}{3}$ positions. In particular, $\mathsf{hd}_2\left(\tilde{\mathsf{E}}_{\mathscr{C}}\left(V^*_{(Q)},R^*_{(Q)}\right),QE^*\right)\ge 2\frac{\mathsf{hd}_{\min}}{3}$. ■

This means that if $\neg Err$ occurs and $\mathscr{P}^*$ attempts to open a $V^*_{(Q)}\neq V_{(Q)}=QV$ then

$$\mathsf{hd}_2\left(\tilde{\mathsf{E}}_{\mathscr{C}}\left(V^*_{(Q)},R^*_{(Q)}\right),QE^*\right)\ge 2\frac{\mathsf{hd}_{\min}}{3}.$$

As argued above, if the distance between two strings of length $2n$ is at least $\frac{\mathsf{hd}_{\min}}{3}$, the probability that J will not contain a j such that the two strings differ in position j is negligible. Hence, the probability that $\tilde{\mathsf{E}}_{\mathscr{C}}\left(V^*_{(Q)},R^*_{(Q)}\right)|_J=QE^*|_J$ is negligible. Thus, the probability that $\neg Err$ and $\mathscr{V}$ accepts while $\mathscr{V}_{\mathsf{ILC}}$ does not is negligible. This proves (9.1).

Next, we want to define a *transcript extractor* $\mathscr{T}$ that given rewindable access to $\langle\mathscr{P}^*(s)\longleftrightarrow\mathscr{V}(\sigma,u)\rangle$ outputs $\widetilde{\mathsf{trans}}_{\mathscr{P}_{\mathsf{ILC}}}$, which we would like to correspond to all messages committed by $\mathscr{P}^*_{\mathsf{ILC}}$ using `commit` in $\langle\mathscr{P}^*_{\mathsf{ILC}}(s,\sigma,u)\overset{\mathsf{ILC}}{\longleftrightarrow}\mathscr{V}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},u;\rho_{\mathsf{ILC}})\rangle$. Here ρ_{ILC} is the randomness used by the $\mathscr{V}_{\mathsf{ILC}}$ inside $\mathscr{V}$ in the first execution of $\mathscr{T}$'s oracle $\langle\mathscr{P}^*(s)\longleftrightarrow\mathscr{V}(\sigma,u)\rangle$. However, we allow $\mathscr{T}$ to fail if $\mathscr{V}$ does not accept in this first transcript and further to fail with negligible probability. Formally, we want $\mathscr{T}$ to run in expected PPT such that for all PPT $\mathscr{A}$:

$$\Pr\left[\begin{array}{c}\sigma \leftarrow \mathcal{K}(1^\lambda); (\sigma_{\mathsf{ILC}}, \cdot) = \sigma; (u,s) \leftarrow \mathcal{A}(\sigma);\\ \widetilde{\mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}}} \leftarrow \mathcal{T}^{\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(\sigma,u)\rangle}(\sigma,u);\\ \mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}} \leftarrow \langle \mathcal{P}^*_{\mathsf{ILC}}(s,\sigma,u) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, u; \rho_{\mathsf{ILC}})\rangle :\\ b = 1 \ \wedge \ \mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}} \neq \widetilde{\mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}}}\end{array}\right] \approx 0. \qquad (9.3)$$

Here b is the value output by $\mathcal{V}$ the first time $\mathcal{T}$'s oracle runs $\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(\sigma,u)\rangle$, and the randomness ρ_{ILC} used by $\mathcal{V}_{\mathsf{ILC}}$ in the third line is identical to the random value used by the $\mathcal{V}_{\mathsf{ILC}}$ inside $\mathcal{V}$ in the first transcript. On input (σ,u), the transcript extractor $\mathcal{T}$ will first use its oracle to get a transcript of $\langle \mathcal{P}^*(s) \longleftrightarrow \mathcal{V}(\sigma,u;(\rho_{\mathsf{ILC}},\rho))\rangle$. If $\mathcal{V}$ rejects, $\mathcal{T}$ will just abort. If $\mathcal{V}$ accepts, $\mathcal{T}$ will rewind the last message of $\mathcal{P}^*$ to get a transcript for a new random challenge J. $\mathcal{T}$ continues this way, until it has an accepting transcript for $2n$ independently chosen sets J. Notice that if there is only one choice of J that results in $\mathcal{V}$ accepting, $\mathcal{P}^*$ will likely have received each allowed challenge around $2n$ times and $\mathcal{T}$ will get the exact same transcript $2n$ times before it is done rewinding. Still, $\mathcal{T}$ runs in expected polynomial time: if a fraction p of all allowed sets J results in accept, the expected number of rewindings *given* that the first transcript accepts is $\frac{2n-1}{p}$. However, the probability that the first run accepts is p, and if it does not accept, $\mathcal{T}$ does not do any rewindings. In total, that gives $\frac{(2n-1)p}{p} = 2n-1$ rewindings in expectation.

We let $J_1,\ldots,J_{2n}$ denote the set of challenges J in the accepting transcripts obtained by $\mathcal{T}$. If $\bigcup_{i=1}^{2n} J_i$ has less than $2n - \frac{\mathsf{hd}_{\min}}{3}$ elements, $\mathcal{T}$ terminates. Otherwise, $\mathcal{T}$ is defined similarly to $\mathcal{P}^*_{\mathsf{ILC}}$: it uses the values of the openings to get at least $2n - \frac{\mathsf{hd}_{\min}}{3}$ columns of each E_i. For each of the row vectors, $\mathbf{e}_\tau$, it computes $\mathbf{v}_\tau$ and $\mathbf{r}_\tau$ such that $\tilde{\mathsf{E}}_{\mathcal{C}}(\mathbf{v}_\tau, \mathbf{r}_\tau)$ agrees with $\mathbf{e}_\tau$ in all entries $(\mathbf{e}_\tau)_j$ for which the j'th column have been revealed, if such $\mathbf{v}$ exists. Since $\mathcal{T}$ will not correct any errors, finding such $\mathbf{v}_\tau$ and $\mathbf{r}_\tau$ corresponds to solving a linear set of equations. Notice that since the minimum distance is more than $2\frac{\mathsf{hd}_{\min}}{3}$ there is at most one such $\mathbf{v}_\tau$ for each $\tau \in [t]$. If for some τ there is no such $\mathbf{v}_\tau$, then $\mathcal{T}$ aborts, otherwise $\mathcal{T}$ use the resulting vectors $\mathbf{v}_\tau$ as the prover messages to define $\widetilde{\mathsf{trans}_{\mathcal{P}_{\mathsf{ILC}}}}$.

If $|\bigcup_{i=1}^{\kappa} J_i| < 2n - \frac{\mathsf{hd}_{\min}}{3}$, there are at least $\frac{\mathsf{hd}_{\min}}{6}$ numbers in $[n] \setminus \bigcup_{i=1}^{\kappa} J_i$ or in $\{n+1,\dots,2n\} \setminus \bigcup_{i=1}^{\kappa} J_i$. In either case, a random allowed J has negligible probability of being contained in $\bigcup_{i=1}^{\kappa} J_i$. Since $\mathscr{T}$ runs in expected polynomial time, this implies by induction that there is only negligible probability that $|\bigcup_{i=1}^{\kappa} J_i| < \min(\kappa, 2n - \frac{\mathsf{hd}_{\min}}{3})$ and therefore $|\bigcup_{i=1}^{2n} J_i| < 2n - \frac{\mathsf{hd}_{\min}}{3}$.

Finally, we need to show

Lemma 23 *The probability that for some τ there are no $\mathbf{v}_\tau$ and $\mathbf{r}_\tau$ such that $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_\tau, \mathbf{r}_\tau)$ agrees with $\mathbf{e}_\tau$ on the opened $j \in \bigcup_{i=1}^{2n} J_i$ and $b = 1$ is negligible.*

In particular, the probability that $b = 1$ but $\mathscr{T}$ does not extract the transcript of $\mathscr{P}^*_{\mathsf{ILC}}$ is negligible.

Proof Since we can ignore events that happen with negligible probability, and the expected number of rewindings is polynomial, we can assume that in all the rewindings, $\mathscr{P}^*$ only makes openings to the most common openings. We showed that the probability that $b = 1$ but $\mathscr{P}^*$ sends a $V^*_{(Q)} \neq QV$, or $\mathscr{V}$ computes a $V'^*_{(Q)} \neq Q'V$ is negligible and by the same argument the probability that $b = 1$ but $\mathscr{P}^*$ sends $\mathbf{v}^*_{(\gamma)} \neq \mathbf{v}_{(\gamma)}$ is negligible. Therefore, in the following, we will assume $\mathbf{v}^*_{(\gamma)} = \mathbf{v}_{(\gamma)}$.

Now suppose that there is some $\mathbf{e}_\tau$ such that the opened values are inconsistent with being $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_\tau, \mathbf{r}_\tau)$ for any $\mathbf{r}_\tau$. That is, there is some j such that $j, n+j \in \bigcup_{i=1}^{2n} J_i$ and $(\mathbf{e}_\tau)_j - (\mathbf{e}_\tau)_{n+j} \neq \mathsf{E}_{\mathscr{C}}(\mathbf{v})_j$. For uniformly chosen $\gamma_\tau \in \mathbb{F}$, we get that $\gamma_\tau((\mathbf{e}_\tau)_j - (\mathbf{e}_\tau)_{n+j} - \mathsf{E}_{\mathscr{C}}(\mathbf{v})_j)$ is uniformly distributed in $\mathbb{F}$. Hence for a random $\gamma \in \mathbb{F}^t$, we have that $\gamma \cdot ((\mathbf{e})_j - (\mathbf{e})_{n+j} - \mathsf{E}_{\mathscr{C}}(\mathbf{v})_j)$ is uniformly distributed. When $\mathscr{V}$ sends γ, $\mathscr{P}^*$ will respond with $\mathbf{v}^*_{(\gamma)} = \mathbf{v}_{(\gamma)}$ and some $\mathbf{r}^*_{(\gamma)}$. $\mathscr{V}$ will only accept on a challenge J if for all $j \in J$ we have $(\mathbf{e}_0 + \gamma\mathbf{e})_j = \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}, \mathbf{r}^*_{(\gamma)})_j$. Since $j, n+j \in \bigcup_{i=1}^{2n} J_i$ we have $(\mathbf{e}_0 + \gamma\mathbf{e})_j = \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}, \mathbf{r}^*_{(\gamma)})_j$ and $(\mathbf{e}_0 + \gamma\mathbf{e})_{n+j} = \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}, \mathbf{r}^*_{(\gamma)})_{n+j}$ so

$$\begin{aligned}
(\mathbf{e}_0)_j - (\mathbf{e}_0)_{n+j} + \gamma\mathbf{e}_j - \gamma\mathbf{e}_{n+j} =& \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}, \mathbf{r}^*_{(\gamma)})_j - \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}, \mathbf{r}^*_{(\gamma)})_{n+j} \\
=& \mathsf{E}_{\mathscr{C}}(\mathbf{v}_{(\gamma)})_j \\
=& (\mathsf{E}_{\mathscr{C}}(\mathbf{v}_0) + \gamma \mathsf{E}_{\mathscr{C}}(\mathbf{v}))_j
\end{aligned}$$

that is,

$$\gamma \mathbf{e}_j - \gamma \mathbf{e}_{n+j} - \gamma \mathsf{E}_{\mathscr{C}}(\mathbf{v})_j = \mathsf{E}_{\mathscr{C}}(\mathbf{v}_0)_j - (\mathbf{e}_0)_j + (\mathbf{e}_0)_{n+j}$$

For random γ the left-hand side is uniform and the right-hand side is fixed, hence equality only happens with negligible probability. That proves the lemma. ■

Since $\mathscr{E}_{\mathsf{ILC}}^{\langle \mathscr{P}^*_{\mathsf{ILC}}(s,\sigma,u) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathscr{V}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},u)\rangle}(\sigma,u)$ is a straight-line extractor, we can simply assume that it gets the transcript as an input, and can be written as $\mathscr{E}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},u,\mathsf{trans}_{\mathscr{P}_{\mathsf{ILC}}})$. For any PPT $\mathscr{A}$ consider the following experiment.

$$\left[\begin{array}{c} \sigma \leftarrow \mathscr{K}(1^\lambda); (\sigma_{\mathsf{ILC}},\cdot) = \sigma; (u,s) \leftarrow \mathscr{A}(\sigma); \\ \widetilde{\mathsf{trans}}_{\mathscr{P}_{\mathsf{ILC}}} \leftarrow \mathscr{T}^{\langle \mathscr{P}^*(s) \longleftrightarrow \mathscr{V}(\sigma,u)\rangle}(\sigma,u); \\ \mathsf{trans}_{\mathscr{P}_{\mathsf{ILC}}} \leftarrow \langle \mathscr{P}^*_{\mathsf{ILC}}(s,\sigma,u) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathscr{V}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},u;\rho_{\mathsf{ILC}})\rangle = b_{\mathsf{ILC}}; \\ w \leftarrow \mathscr{E}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},u,\mathsf{trans}_{\mathscr{P}_{\mathsf{ILC}}}); \\ \widetilde{w} \leftarrow \mathscr{E}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},u,\widetilde{\mathsf{trans}}_{\mathscr{P}_{\mathsf{ILC}}}); \end{array}\right] \tag{9.4}$$

We have shown that when doing this experiment, the probability that $b = 1 \wedge b_{\mathsf{ILC}} = 0$ and the probability that $b = 1 \wedge \mathsf{trans}_{\mathscr{P}_{\mathsf{ILC}}} \neq \widetilde{\mathsf{trans}}_{\mathscr{P}_{\mathsf{ILC}}}$ are both negligible. By knowledge soundness of $(\mathscr{K}_{\mathsf{ILC}},\mathscr{P}_{\mathsf{ILC}},\mathscr{V}_{\mathsf{ILC}})$, the probability that $b_{\mathsf{ILC}} = 1 \wedge (\sigma,u,w) \notin \mathscr{R}$ is also negligible. Finally, if $\mathsf{trans}_{\mathscr{P}_{\mathsf{ILC}}} = \widetilde{\mathsf{trans}}_{\mathscr{P}_{\mathsf{ILC}}}$ then clearly $w = \widetilde{w}$. Taken together this implies that the probability of $b = 1 \wedge (\sigma,u,\widetilde{w}) \notin R$ is negligible. We now define $\mathscr{E}^{\langle \mathscr{P}^*(s) \longleftrightarrow \mathscr{V}(\sigma,u)\rangle}(\sigma,u)$ to compute $\mathscr{E}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},u,\mathscr{T}^{\langle \mathscr{P}^*(s) \longleftrightarrow \mathscr{V}(\sigma,u)\rangle}(\sigma,u))$. The above experiment shows that $(\mathscr{K},\mathscr{P},\mathscr{V})$ is knowledge sound with $\mathscr{E}$ as extractor. ■

Theorem 3 (SHVZK) *If* $(\mathscr{K}_{\mathsf{ILC}},\mathscr{P}_{\mathsf{ILC}},\mathscr{V}_{\mathsf{ILC}})$ *is perfect SHVZK and* $(\mathsf{Setup},\mathsf{Commit})$ *is computationally (statistically) hiding then* $(\mathscr{K},\mathscr{P},\mathscr{V})$ *is computationally (statistically) SHVZK.*

Proof To prove we have SHVZK we describe how the simulator $\mathscr{S}(\sigma,u,\rho)$ should simulate the view of $\mathscr{V}$. Along the way, we will argue why, the variables output by $\mathscr{S}$ have the correct joint distribution. To keep the proof readable, instead of saying that "the joint distribution of [random variable] and all previously defined random variables is identical to the distribution in the real view of $\mathscr{V}$ in $\langle \mathscr{P}(\sigma,u,w) \longleftrightarrow$

$\mathscr{V}(\sigma,u)\rangle$" we will simply say that "[random variable] has the correct distribution".

Using the randomness ρ the simulator learns the queries $\rho_{\mathsf{ILC}} = (x_1,\dots,x_{\mu-1},Q)$ the internal $\mathscr{V}_{\mathsf{ILC}}$ run by the honest $\mathscr{V}$ will send. $\mathscr{S}$ can therefore run $\mathscr{S}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},u,\rho_{\mathsf{ILC}})$ to simulate the view of the internal $\mathscr{V}_{\mathsf{ILC}}$. This gives it $(t_1,\dots,t_\mu,V_{(Q)})$. By the SHVZK property of $(\mathscr{K}_{\mathsf{ILC}},\mathscr{P}_{\mathsf{ILC}},\mathscr{V}_{\mathsf{ILC}})$ these random variables will all have the correct joint distribution.

Then $\mathscr{S}$ reads the rest of ρ to learn also the challenges γ and J that $\mathscr{V}$ will send. The simulator picks uniformly at random $\mathbf{v}_{(\gamma)} \leftarrow \mathbb{F}^k$. Since in a real proof $\mathbf{v}_0$ is chosen at random, we see that the simulated $\mathbf{v}_{(\gamma)}$ has the correct distribution. Now $\mathscr{S}$ picks $E_{01}|_J,\dots,E_\mu|_J$ uniformly at random. Recall that we defined $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v};\mathbf{r}) = (\mathsf{E}_{\mathscr{C}}(\mathbf{v})+\mathbf{r},\mathbf{r})$ and by definition of J being allowed, we have for all $j \in J$ that $j+n \notin J$. This means for any choice of $\mathbf{v}_0 \in \mathbb{F}^k$ and $V \in \mathbb{F}^{t\times k}$ that when we choose random $\mathbf{r}_0 \leftarrow \mathbb{F}^n$ and $R \leftarrow \mathbb{F}^{t\times n}$ we get uniformly random $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_0;\mathbf{r}_0)|_J$ and $\tilde{\mathsf{E}}_{\mathscr{C}}(V;R)$. Consequently, $E_{01}|_J,\dots,E_\mu|_J$ have the correct distribution.

Next, the simulator picks $\mathbf{r}_{(\gamma)} \in \mathbb{F}^n$ and $R_{(Q)} \in \mathbb{F}^{t\times n}$ one entry and column at a time. For all j such that $j \notin J$ and $j+n \notin J$ the simulator picks random $(\mathbf{r}_{(\gamma)})_j \leftarrow \mathbb{F}$ and random $R_j \leftarrow \mathbb{F}^t$. For all j such that $j \in J$ or $j+n \in J$, the simulator then computes the unique $(\mathbf{r}_{(\gamma)})_j \in \mathbb{F}$ and $R_j \in \mathbb{F}^t$ such that we get $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)};\mathbf{r}_{(\gamma)}) = \mathbf{e}_0|_J + \gamma E|_J$ and $\tilde{\mathsf{E}}_{\mathscr{C}}(V_{(Q)};R_{(Q)}) = QE|_J$ and $\tilde{\mathsf{E}}_{\mathscr{C}}(V'_{(Q)};R'_{(Q)}) = Q'E|_J$.

Finally, $\mathscr{S}$ defines $E_{01}|_{\bar{J}},\dots,E_\mu|_{\bar{J}}$ to be 0 matrices. It then picks $\mathbf{s}_1,\dots,\mathbf{s}_\mu$ at random and makes the commitments $\mathbf{c}_1,\dots,\mathbf{c}_\mu$ as in the protocol. For $j \in J$ we see that all the $\mathbf{c}_i|_j$ commitments are computed as in the real execution from values that have the same distribution as in a real proof. Hence, they will have the correct distribution. The $\mathbf{c}_i|_j$s for $j \notin J$ are commitments to different values than in a real proof. However, by the computational (statistical) hiding property of the commitment scheme, they have a distribution that is computationally (statistically) indistinguishable from the correct distribution. ■

9.1.4 Efficiency

We will now estimate the efficiency of a compiled proof of knowledge $(\mathscr{K},\mathscr{P},\mathscr{V})$ for $(\sigma,u,w) \in \mathscr{R}$. Let μ be the number of rounds, $t = \sum_{i=1}^{\mu} t_i$, k,n given in $\mathsf{E}_{\mathscr{C}}$, qc the `open` query complexity, i.e., $Q \in \mathbb{F}^{\mathrm{qc} \times t}$, and qc' the `check` query complexity, i.e., $Q' \in \mathbb{F}^{\mathrm{qc}' \times t}$. Let $T_{\mathscr{P}_{\mathsf{ILC}}}$ be the running time of $\mathscr{P}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, u, w)$, $T_{\tilde{\mathsf{E}}_{\mathscr{C}}}(k)$ be the encoding time for a vector in $\mathbb{F}^k$, $T_{\mathsf{Commit}}(t_i)$ be the time to commit to t_i field elements, $T_{\mathrm{Mmul}}(\mathrm{qc}, t, b)$ be the time it takes to multiply matrices in $\mathbb{F}^{\mathrm{qc} \times t}$ and $\mathbb{F}^{t \times b}$ and similarly for qc'. Let $T_{\mathscr{V}_{\mathsf{ILC}}}$ be the running time of $\mathscr{V}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, u)$, and let C_{ILC} be the communication from the verifier to the prover in $\langle \mathscr{P}_{\mathsf{ILC}} \overset{\mathsf{ILC}}{\longleftrightarrow} \mathscr{V}_{\mathsf{ILC}} \rangle$, $C_{\mathsf{Commit}}(t_i)$ be the combined size of commitment and randomness for a message consisting of t_i field elements.

We give the dominant factors of efficiency of the compiled proof in Fig. 9.3. The estimates presume $T_{\mathsf{Commit}}(t_1 + 1)$ is not too far from $T_{\mathsf{Commit}}(t_1)$. Of course, in many realistic cases, the cost of computing queries will be less than $T_{\mathrm{Mmul}}(\mathrm{qc}, t, b)$ if the query matrix contains large zero-submatrices, or has some other special form.

9.1.5 Optimisations

Motivated by the argument in Section 8.4, we discuss two possible optimisations to the compilation procedure. Consider the following ILC check query.

$$(\bar{\mathbf{a}} \cdot \mathbf{w}_1, \ldots, \bar{\mathbf{a}} \cdot \mathbf{w}_Q) \overset{?}{=} (c_1, \ldots, c_Q) \quad + \sum_{j \neq 0, j = -3m}^{3m} \mathbf{h}_j x^j$$

Since this is a check query, the prover will not need to send the value of $(c_1, \ldots, c_Q) \quad + \sum_{j \neq 0, j = -3m}^{3m} \mathbf{h}_j x^j$ to the verifier, because the verifier can compute what this value out to be for themself. This was the reason for introducing the `check` queries in the first place, because otherwise, if this was handled as a `send` query, the prover would have had to send a vector of length Q to the verifier in the compiled proof. Since Q may be as large as $O(N)$, this would have prevented the proof from having sub-linear communication costs.

However, the compiled proof introduces a further causes for concern. An obvious issue is that the prover commits to matrices column-wise, so the prover will produce Q commitments. This is easily solved however, by hashing all of these commitments into a Merkle tree, and opening the commitments required for the verification checks. This changes the communication costs associated with committing from $O(N)$ hash values to $O(1)$ hash values, and the cost of opening from $O(\lambda)$ openings to $O(\lambda \log N)$ openings, and adds some computational overhead. This option was not considered in [16], but was unnecessary for the asymptotic results in that protocol.

In the compiled proof, we use the randomised encoding $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v};\mathbf{r}) = (\mathsf{E}_{\mathscr{C}}(\mathbf{v}) + \mathbf{r}, \mathbf{r})$, and the prover commits to the randomised encodings. This means that the prover still needs to send some randomness to the verifier, computed from a linear combination of the randomness used to commit to the vectors $\mathbf{h}_i$, which are vectors of length Q. Then the vectors of randomness will be of length Q too, resulting in a vector of length Q being sent to the verifier.

We can remove the dependency of the length of the randomness vector on the length of the vector, k. Fix distinct points $a_1, \ldots, a_{2\lambda} \in \mathbb{F}$ at the beginning of the protocol. Instead of sampling a truly random vector $\mathbf{r} \in \mathbb{F}^k$, the prover selects a vector $\mathbf{r}' \leftarrow \mathbb{F}^{2\lambda}$. This specifies the coefficients of a polynomial of degree $2\lambda - 1$. The prover computes the Reed-Solomon encoding of $\mathbf{r}'$ with respect to points $a_1, \ldots, a_{2\lambda}$ and gets a vector $\mathbf{r} \in \mathbb{F}^k$.

Now, if the prover uses this $\mathbf{r}$ as randomness for the randomised encoding instead of choosing one uniformly at random, then they need only send a vector of length λ to the verifier. Changes to the security proofs are given in Appendix A.

Note that we should consider the ILC protocol compilation both with and without these optimisations, as these optimisations will add unecessary computational overhead into the compilation when applied to short vectors. Therefore, we can use the optimisations when compiling ILC protocols with vectors of multiple lengths, and only apply them to the long vectors.

We give the dominant factors of efficiency of the compiled proof in Fig. 9.6. The estimates presume $T_{\mathsf{Commit}}(t_1 + 1)$ is not too far from $T_{\mathsf{Commit}}(t_1)$. Notation is as before, with l the output length of the hash function used for committing, and $T_{RS}(2\lambda)$ the time taken to compute a Reed-Solomon codeword of length n from an input word of length 2λ.

9.2 Compiling into Discrete-Logarithm-Based Proofs

In this section, we show how to compile a proof of knowledge with straight-line extraction for relation $\mathscr{R}$ over the communication channel ILC into a proof of knowledge for the same relation over the standard channel based on the discrete logarithm assumption, provided that the ILC protocol satisfies some simple additional conditions. In fact, this compilation will work for any homomorphic commitment scheme over the field $\mathbb{F}$ in which the ILC protocol takes place, but for concreteness and efficiency, we use Pedersen commitments. These will give rise to efficient zero-knowledge arguments as Pedersen commitments are extremely succinct; only one group element is required to commit to a large number of field elements.

The idea behind this compilation of an ILC proof is that instead of committing to vectors $\mathbf{v}_\tau$ using the channel ILC, the prover commits to each vector using a real homomorphic commitment scheme. When the verifier wants to open a linear combination of the original vectors using an `open` command, he sends the coefficients $\mathbf{q} = (q_1, \ldots, q_t)$ of the linear combination to the prover, and the prover responds with the linear combination $\mathbf{v}_{(\mathbf{q})} \leftarrow \mathbf{q}V$. As before, we will use the notation $\mathbf{v}_{(\mathbf{q})}$ to denote vectors that depend on $\mathbf{q}$.

Now, to check that the prover is not giving a wrong $\mathbf{v}_{(\mathbf{q})}$, the verifier will compute a commitment to $\mathbf{v}_{(\mathbf{q})}$ using the real homomorphic commitment scheme, and check that this commitment is the correct linear combination of the commitments to $\mathbf{v}_\tau$. This works because the commitment scheme is homomorphic.

Of course, to prevent real commitments from leaking information about committed vectors, randomness is necessary. Each vector $\mathbf{v}_\tau$ will have the randomness $\mathbf{s}_\tau$ used to commit to it concatenated to it, and then the prover will respond with the linear combination $\mathbf{v}_{(\mathbf{q})} \leftarrow \mathbf{q}V$ as before, plus $\mathbf{s}_{(\mathbf{q})} = \mathbf{q}\mathbf{s}$. Then SHVZK for the compiled proof is inherited from the underlying ILC proof. Unlike the previous compilation technique which reveals some entries of committed codewords, almost all of the verifier's calculations are done on hiding commitments, along with a few openings that do not reveal any information due to the SHVZK property of the underlying ILC proof. Therefore, it is not necessary to use any additional linear combinations in the proof.

As before, we must discuss how to treat `check` queries. The verifier makes `check` queries $\mathbf{q}' = (q'_1, \dots, q'_t)$ of linear combinations of committed vectors. Now, `check` queries handle the case where the verifier can compute what $\mathbf{v}'_{(\mathbf{q})} \leftarrow \mathbf{q}'V$ should be by themselves, using the results of other `open` queries. Therefore, the prover need not send $\mathbf{v}'_{(\mathbf{q})}$ to the verifier. The prover, given $\mathbf{q}'$, must still reveal $\mathbf{s}'_{(\mathbf{q})} = \mathbf{q}'\mathbf{s}$. To check that the prover is not giving a wrong $\mathbf{v}'_{(\mathbf{q})}$, the verifier will compute a commitment to $\mathbf{v}'_{(\mathbf{q})}$ using the real homomorphic commitment scheme with randomness $\mathbf{s}'_{(\mathbf{q})}$, and check that this commitment is the correct linear combination of the commitments to $\mathbf{v}_\tau$.

9.2.1 Construction

Let $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ be a *non-adaptive* μ-round SHVZK proof of knowledge with tree extraction over ILC for a relation $\mathcal{R}$. We now define a proof of knowledge $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ in Fig. 9.5, where we use the same notation as in the previous section. That is, given matrices $V_1, \dots, V_\mu$ and vectors $\mathbf{s}_1, \dots, \mathbf{s}_\mu$, $\mathbf{c}_1, \dots, \mathbf{c}_\mu$, we define

$$V = \begin{pmatrix} V_1 \\ \vdots \\ V_\mu \end{pmatrix} \qquad \mathbf{s} = \begin{pmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_\mu \end{pmatrix} \qquad \mathbf{c} = \begin{pmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_\mu \end{pmatrix}.$$

The matrices $V_1, \dots, V_\mu$ are formed by the row vectors $\mathcal{P}_{\mathsf{ILC}}$ commits to, and we let $t_1, \dots, t_\mu$ be the numbers of vectors in each round, i.e., for all i we have $V_i \in \mathbb{F}^{t_i \times k}$.

This time, we use $\mathsf{Commit}(E; \mathbf{s})$ to denote the function that applies Commit *row-wise* on E and returns a vector $\mathbf{c}$ of $t = \sum_{i=1}^{\mu} t_i$ commitments.

9.2.2 Security Analysis

Theorem 4 (Completeness) *If $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ is complete for relation $\mathcal{R}$ over* ILC, *then $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ in Fig. 9.2 is complete for relation $\mathcal{R}$.*

Proof All the commitment openings are correct, so they will be accepted by the verifier. In the execution of $\langle \mathcal{P}(\sigma, u, w) \longleftrightarrow \mathcal{V}(\sigma, u) \rangle$, the fact that the commitment scheme is homomorphic implies that all the linear checks will be true.

If $(\sigma, u, w) \in \mathcal{R}$ then $(\sigma_{\mathsf{ILC}}, u, w) \in \mathcal{R}$ and being complete $\langle \mathcal{P}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, u, w) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, stm)\rangle = 1$ so $\mathcal{V}$'s internal copy of $\mathcal{V}_{\mathsf{ILC}}$ will accept. Thus, in this case, $\langle \mathcal{P}(\sigma, u, w) \longleftrightarrow \mathcal{V}(\sigma, u)\rangle = 1$, which proves completeness. ■

Theorem 5 (Knowledge Soundness) *If* $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ *is statistically knowledge sound for relation* $\mathcal{R}$ *over* ILC*, has tree-special soundness with a linear extraction algorithm, and* $(\mathsf{Setup}, \mathsf{Commit})$ *is computationally (statistically) binding, then* $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ *as constructed above is computationally (statistically) knowledge sound for relation* $\mathcal{R}$*.*

Proof Given that $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ has tree-special-soundness, we know that it is possible to extract a witness to the protocol from sufficiently many structured views of $\mathcal{V}_{\mathsf{ILC}}$. In $(\mathcal{K}, \mathcal{P}, \mathcal{V})$, the view of $\mathcal{V}_{\mathsf{ILC}}$ is given by the final message that $\mathcal{P}$ sends to $\mathcal{V}$. We first show that $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ satisfies the same tree-special-soundness as $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$, with the same values of n_i.

Suppose that we have a tree of accepting transcripts for $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ with the same values of n_i as in the tree-special-soundness of $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$. Now, $\mathcal{V}$ checks the verification equations $\mathsf{Commit}(V_{(Q)}; \mathbf{s}_{(Q)}) = Q\mathbf{c}$ and $\mathsf{Commit}(V'_{(Q)}; \mathbf{s}'_{(Q)}) = Q'\mathbf{c}$. In $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$, we would know, by the properties of the ILC model, that $V_{(Q)} = QV$ and $V'_{(Q)} = Q'V$, where V is the matrix consisting of vectors committed by $\mathcal{P}_{\mathsf{ILC}}$. Consider the linear system created by vertically stacking every copy of $V_{(Q)} = QV$ and $V'_{(Q)} = Q'V$ for every transcript in the tree. By the fact that the ILC protocol has a linear extraction algorithm, there is a linear map M which, when applied to the left and right hand side of this linear system, recovers each row of V. To extract a witness for $(\mathcal{K}, \mathcal{P}, \mathcal{V})$, consider a similar linear system created by vertically stacking every copy of the verification equations $\mathsf{Commit}(V_{(Q)}; \mathbf{s}_{(Q)}) = Q\mathbf{c}$ and $\mathsf{Commit}(V'_{(Q)}; \mathbf{s}'_{(Q)}) = Q'\mathbf{c}$. If we apply the linear map M to this system, which solves for the rows of the linear system, we obtain openings to each commitment in $\mathbf{c}$, given by the rows of a matrix V^*. We know when each commitment was made, so for each opening, we know that it cannot be a function of the challenges which the prover saw before the commitment was made. Furthermore, we know that by the binding property of the commitment scheme, $V_{(Q)} = QV^*$, and similarly, $V'_{(Q)} = Q'V^*$, because the compiled protocol $(\mathscr{K},\mathscr{P},\mathscr{V})$ checks the same equations as the ILC protocol, but in committed format. By the soundness of $(\mathscr{K}_{\mathsf{ILC}},\mathscr{P}_{\mathsf{ILC}},\mathscr{V}_{\mathsf{ILC}})$, the verifier $\mathscr{V}$ would not accept unless V^* provided a valid witness for the ILC protocol.

Finally, by Lemma 2, given rewindable black-box access to $(\mathscr{K},\mathscr{P},\mathscr{V})$, one can sample a tree of accepting transcripts with the correct values of n_i in expected polynomial time. ■

Theorem 6 (SHVZK) *If $(\mathscr{K}_{\mathsf{ILC}},\mathscr{P}_{\mathsf{ILC}},\mathscr{V}_{\mathsf{ILC}})$ is perfect SHVZK, with an ILC query matrix $Q^* = \begin{pmatrix} Q \\ Q' \end{pmatrix}$ which has full rank, and fewer rows than columns, and $(\mathsf{Setup},\mathsf{Commit})$ is computationally (statistically) hiding then $(\mathscr{K},\mathscr{P},\mathscr{V})$ is computationally (statistically) SHVZK.*

Proof To prove we have SHVZK we describe how the simulator $\mathscr{S}(\sigma,u,\rho)$ should simulate the view of $\mathscr{V}$. Along the way, we will argue why, the variables output by $\mathscr{S}$ have the correct joint distribution.

Using the randomness ρ the simulator learns the queries $\rho_{\mathsf{ILC}} = (x_1,\ldots,x_{\mu-1},Q)$ the internal $\mathscr{V}_{\mathsf{ILC}}$ run by the honest $\mathscr{V}$ will send. $\mathscr{S}$ can therefore run $\mathscr{S}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}},u,\rho_{\mathsf{ILC}})$ to simulate the view of the internal $\mathscr{V}_{\mathsf{ILC}}$. This gives it $(t_1,\ldots,t_\mu,V_{(Q)},V')(Q))$. By the SHVZK property of $(\mathscr{K}_{\mathsf{ILC}},\mathscr{P}_{\mathsf{ILC}},\mathscr{V}_{\mathsf{ILC}})$ these random variables will all have the correct joint distribution.

Since the query matrix has full rank, in an honest proof, where $\mathbf{s}_i$ are chosen uniformly at random, $\mathbf{s}_{(Q)}$ and $\mathbf{s}'_{(Q)}$ will also be uniformly random vectors. Therefore, the simulator selects each element of $\mathbf{s}_{(Q)}$ and $\mathbf{s}'_{(Q)}$ uniformly at random from $\mathbb{F}$. Finally, $\mathscr{S}$ picks $\mathbf{c}_1,\ldots,\mathbf{c}_\mu$ uniformly at random, conditioned on $\mathsf{Commit}(V_{(Q)},\mathbf{s}_{(Q)}) = Q\mathbf{c}$ and $\mathsf{Commit}(V'_{(Q)},\mathbf{s}'_{(Q)}) = Q'\mathbf{c}$. Since Q^* has fewer rows than columns, this can be done by writing Q^* in reduced row echelon form. Each column of Q^* corresponds to a commitment or committed value, and some entry of a $\mathbf{c}_i$. Then, all entries of $\mathbf{c}_1,\ldots,\mathbf{c}_\mu$ which do not correspond to a column of Q with a leading order 1 can be chosen uniformly at random. The other entries which do correspond to columns with a leading order 1 are now fully determined by the values of the entries which have already been chosen, and the fact that $\mathsf{Commit}(V_{(Q)},\mathbf{s}_{(Q)}) = Q\mathbf{c}$ and

$\mathsf{Commit}(V'_{(Q)}, \mathbf{s}'_{(Q)}) = Q'\mathbf{c}$.

Now, just as in the real protocol, $\mathbf{s}_{(Q)}$ and $\mathbf{s}'_{(Q)}$ are distributed uniformly at random, and the commitments are all uniformly random conditioned on satisfying $\mathsf{Commit}(V_{(Q)}, \mathbf{s}_{(Q)}) = Q\mathbf{c}$ and $\mathsf{Commit}(V'_{(Q)}, \mathbf{s}'_{(Q)}) = Q'\mathbf{c}$. ■

9.2.3 Efficiency

We will now estimate the efficiency of a compiled proof of knowledge $(\mathscr{K}, \mathscr{P}, \mathscr{V})$ for $(\sigma, u, w) \in \mathscr{R}$. Let μ be the number of rounds, $t = \sum_{i=1}^{\mu} t_i$, k, n given in $\mathsf{E}_{\mathscr{C}}$, qc the `open` query complexity, i.e., $Q \in \mathbb{F}^{\mathrm{qc} \times t}$, and qc$'$ the `check` query complexity, i.e., $Q' \in \mathbb{F}^{\mathrm{qc} \times t}$. Let $T_{\mathscr{P}_{\mathsf{ILC}}}$ be the running time of $\mathscr{P}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, u, w)$, $T_{\mathsf{Commit}}(t_i)$ be the time to commit to t_i field elements, $T_{\mathrm{Mmul}}(\mathrm{qc}, t, b)$ be the time it takes to multiply matrices in $\mathbb{F}^{\mathrm{qc} \times t}$ and $\mathbb{F}^{t \times b}$, and $T_{\mathscr{V}_{\mathsf{ILC}}}$ is the running time of $\mathscr{V}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, u)$. Let furthermore C_{ILC} be the communication from the verifier to the prover in $\langle \mathscr{P}_{\mathsf{ILC}} \overset{\mathsf{ILC}}{\longleftrightarrow} \mathscr{V}_{\mathsf{ILC}} \rangle$, $C_{\mathsf{Commit}}(t_i)$ be the combined size of commitment and randomness for a message consisting of t_i field elements.

We give the dominant factors of efficiency of the compiled proof in Fig. 9.6. As before, estimates presume $T_{\mathsf{Commit}}(t_1 + 1)$ is not too far from $T_{\mathsf{Commit}}(t_1)$.

```
$\mathscr{P}(\sigma, u, w)$

• Parse input:
    – Parse $\sigma = (\sigma_{\mathsf{ILC}}, \mathsf{E}_{\mathscr{C}}, ck)$
    – Parse $\sigma_{\mathsf{ILC}} = (\mathbb{F}, k)$
    – Get $n$ from $\mathsf{E}_{\mathscr{C}}$
• Round 1:
    – $\mathbf{v}_0 \leftarrow \mathbb{F}^k$
    – $\mathbf{e}_0 \leftarrow \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_0; \mathbf{r}_0)$
    – $(\mathtt{commit}, V_1) \leftarrow \mathscr{P}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, u, w)$
    – $E_1 \leftarrow \tilde{\mathsf{E}}_{\mathscr{C}}(V_1; R_1)$
    – Let $E_{01} = \begin{pmatrix} \mathbf{e}_0 \\ E_1 \end{pmatrix}$
    – $\mathbf{c}_1 = \mathsf{Commit}(E_{01}; \mathbf{s}_1)$
    – Send $(\mathbf{c}_1, t_1)$ to $\mathscr{V}$
• Rounds $2 \leq i \leq \mu$:
    – Get challenge $x_{i-1}$ from $\mathscr{V}$
    – $(\mathtt{commit}, V_i) \leftarrow \mathscr{P}_{\mathsf{ILC}}(x_{i-1})$
    – $E_i \leftarrow \tilde{\mathsf{E}}_{\mathscr{C}}(V_i; R_i)$
    – $\mathbf{c}_i = \mathsf{Commit}(E_i; \mathbf{s}_i)$
    – Send $(\mathbf{c}_i, t_i)$ to $\mathscr{V}$
• Round $\mu + 1$:
    – Get $(\gamma, Q, Q')$ from $\mathscr{V}$
    – $\mathbf{v}_{(\gamma)} \leftarrow \mathbf{v}_0 + \gamma V$
    – $\mathbf{r}_{(\gamma)} \leftarrow \mathbf{r}_0 + \gamma R$
    – $V_{(Q)} \leftarrow QV$
    – $R_{(Q)} \leftarrow QR$
    – $R_{(Q')} \leftarrow Q'R$
    – Send $(\mathbf{v}_{(\gamma)}, \mathbf{r}_{(\gamma)}, V_{(Q)}, R_{(Q)}, R'_{(Q)})$ to $\mathscr{V}$
• Round $\mu + 2$:
    – Get $J \subset [2n]$ from $\mathscr{V}$
    – Send $(E_{01}|_J, \mathbf{s}_1|_J, \ldots, E_\mu, \mathbf{s}_\mu|_J)$ to $\mathscr{V}$
```

```
$\mathscr{K}(1^\lambda)$

• $\sigma_{\mathsf{ILC}} \leftarrow \mathscr{K}_{\mathsf{ILC}}(1^\lambda)$
• Parse $\sigma_{\mathsf{ILC}} = (\mathbb{F}, k)$
• $\mathsf{E}_{\mathscr{C}} \leftarrow \mathsf{Gen}_{\mathsf{E}_{\mathscr{C}}}(\mathbb{F}, k)$
• $ck \leftarrow \mathsf{Setup}(1^\lambda)$
• Return $\sigma = (\sigma_{\mathsf{ILC}}, \mathsf{E}_{\mathscr{C}}, ck)$
```

```
$\mathscr{V}(\sigma, u)$

• Parse input
    – Parse $\sigma = (\sigma_{\mathsf{ILC}}, \mathsf{E}_{\mathscr{C}}, ck)$
    – Parse $\sigma_{\mathsf{ILC}} = (\mathbb{F}, k)$
    – Get $n$ from $\mathsf{E}_{\mathscr{C}}$
    – Give input $(\sigma_{\mathsf{ILC}}, u)$ to $\mathscr{V}_{\mathsf{ILC}}$
• Rounds $1 \leq i < \mu$:
    – Receive $(\mathbf{c}_i, t_i)$
    – $(\mathtt{send}, x_i) \leftarrow \mathscr{V}_{\mathsf{ILC}}(t_i)$
    – Send $x_i$ to $\mathscr{P}$
• Round $\mu$:
    – Receive $(\mathbf{c}_\mu, t_\mu)$
    – $\gamma \leftarrow \mathbb{F}^{\sum_{i=1}^{\mu} t_i}$
    – $(\mathtt{open}, Q) \leftarrow \mathscr{V}_{\mathsf{ILC}}(t_\mu)$
    – $(\mathtt{check}, Q') \leftarrow \mathscr{V}_{\mathsf{ILC}}(t_\mu)$
    – Send $(\gamma, Q, Q')$ to $\mathscr{P}$
• Round $\mu + 1$:
    – Receive $(\mathbf{v}_{(\gamma)}, \mathbf{r}_{(\gamma)}, V_{(Q)}, R_{(Q)})$
    – $\mathscr{V}_{\mathsf{ILC}}$ computes the answers $V'_{(Q)}$ to its check queries
    – Choose random allowed $J \subset [2n]$
    – Send $J$ to $\mathscr{P}$
• Round $\mu + 2$:
    – Receive $(E_{01}|_J, \mathbf{s}_1|_J, \ldots, E_\mu, \mathbf{s}_\mu|_J)$
    – Check $\mathbf{c}_1|_J = \mathsf{Commit}(E_{01}|_J; \mathbf{s}_1|_J), \ldots, \mathbf{c}_\mu|_J = \mathsf{Commit}(E_\mu|_J; \mathbf{s}_\mu|_J)$
    – Check $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}, \mathbf{r}_{(\gamma)})|_J = \mathbf{e}_0|_J + \gamma E|_J$
    – Check $\tilde{\mathsf{E}}_{\mathscr{C}}(V_{(Q)}, R_{(Q)})|_J = QE|_J$
    – Check $\tilde{\mathsf{E}}_{\mathscr{C}}(V'_{(Q)}, R'_{(Q)})|_J = Q'E|_J$
    – If all checks pass, return decision of $\mathscr{V}_{\mathsf{ILC}}(V_{(Q)})$, else return 0
```

Figure 9.2: Construction of $(\mathscr{K}, \mathscr{P}, \mathscr{V})$ from $(\mathscr{K}_{\mathsf{ILC}}, \mathscr{P}_{\mathsf{ILC}}, \mathscr{V}_{\mathsf{ILC}})$, commitment scheme $(\mathsf{Setup}, \mathsf{Commit})$ and error-correcting code $\mathscr{C}$.

Measure	**Cost**
Prover Computation	$T_{\mathscr{P}_{\mathsf{ILC}}} + t \cdot T_{\tilde{\mathsf{E}}_{\mathscr{C}}}(k) + 2n \cdot \sum_{i=1}^{\mu} T_{\mathsf{Commit}}(t_i)$ $+ T_{\mathrm{Mmul}}(\mathrm{qc} + \mathrm{qc}' + 1, t, k+n)$
Verifier Computation	$T_{\mathscr{V}_{\mathsf{ILC}}} + (\mathrm{qc} + \mathrm{qc}' + 1) \cdot T_{\tilde{\mathsf{E}}_{\mathscr{C}}}(k) + 2\lambda \cdot \sum_{i=1}^{\mu} T_{\mathsf{Commit}}(t_i)$ $+ T_{\mathrm{Mmul}}(\mathrm{qc} + \mathrm{qc}' + 1, t, 2\lambda)$
Communication	$C_{\mathsf{ILC}} + 2n \cdot \sum_{i=1}^{\mu} C_{\mathsf{Commit}}(t_i) + (\mathrm{qc}+1) \cdot (k+n) + \mathrm{qc}'n$ $+ (\mathrm{qc} + \mathrm{qc}' + 1) \cdot t + 2\lambda \cdot t$
Round Complexity	$\mu + 2$

Figure 9.3: Efficiency of a compiled proof of knowledge $(\mathscr{K}, \mathscr{P}, \mathscr{V})$ for $(\sigma, u, w) \in \mathscr{R}$. Communication is measured in field elements and computation in field operations.

Measure	**Cost**
Prover Computation	$T_{\mathscr{P}_{\mathsf{ILC}}} + t \cdot (T_{\mathsf{E}_{\mathscr{C}}}(k) + T_{RS}(2\lambda)) + 2n \cdot \sum_{i=1}^{\mu} T_{\mathsf{Commit}}(t_i)$ $+ 2\mu n \cdot T_{\mathrm{hash}}(2l) + T_{\mathrm{Mmul}}(\mathrm{qc} + \mathrm{qc}' + 1, t, k + 2\lambda)$
Verifier Computation	$T_{\mathscr{V}_{\mathsf{ILC}}} + (\mathrm{qc} + \mathrm{qc}' + 1) \cdot (T_{\mathsf{E}_{\mathscr{C}}}(k) + T_{RS}(2\lambda)) + 2\lambda \cdot \sum_{i=1}^{\mu} T_{\mathsf{Commit}}(t_i)$ $+ \lambda \log n \cdot T_{\mathrm{hash}}(2l) + T_{\mathrm{Mmul}}(\mathrm{qc} + \mathrm{qc}' + 1, t, 2\lambda)$
Communication	$C_{\mathsf{ILC}} + (\mu + \lambda(2\log n + 1)) \cdot C_{\mathsf{Commit}}(2l) + (\mathrm{qc}+1) \cdot (k + 2\lambda)$ $+ 2\lambda \mathrm{qc}' + (\mathrm{qc} + \mathrm{qc}' + 1) \cdot t + 2\lambda \cdot t$
Round Complexity	$\mu + 2$

Figure 9.4: Efficiency of a compiled proof of knowledge $(\mathscr{K}, \mathscr{P}, \mathscr{V})$ for $(\sigma, u, w) \in \mathscr{R}$ with optimisations made to the compiler. Communication is measured in field elements and computation in field operations.

```
$\mathscr{P}(\sigma, u, w)$
  • Parse input:
      – Parse $\sigma = (\sigma_{\mathsf{ILC}}, ck)$
      – Parse $\sigma_{\mathsf{ILC}} = (\mathbb{F}, k)$
      – Get $n$ from $\mathsf{E}_{\mathscr{C}}$
  • Round 1:
      – $(\texttt{commit}, V_1) \leftarrow \mathscr{P}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, u, w)$
      – $\mathbf{c}_1 = \mathsf{Commit}(V_1; \mathbf{s}_1)$
      – Send $(\mathbf{c}_1, t_1)$ to $\mathscr{V}$
  • Rounds $2 \le i \le \mu$:
      – Get challenge $x_{i-1}$ from $\mathscr{V}$
      – $(\texttt{commit}, V_i) \leftarrow \mathscr{P}_{\mathsf{ILC}}(x_{i-1})$
      – $\mathbf{c}_i = \mathsf{Commit}(V_i; \mathbf{s}_i)$
      – Send $(\mathbf{c}_i, t_i)$ to $\mathscr{V}$
  • Round $\mu + 1$:
      – Get $(Q, Q')$ from $\mathscr{V}$
      – $V_{(Q)} \leftarrow QV$
      – $\mathbf{s}_{(Q)} \leftarrow Q\mathbf{s}$
      – $\mathbf{s}'_{(Q)} \leftarrow Q'\mathbf{s}$
      – Send $(V_{(Q)}, \mathbf{s}_{(Q)}, \mathbf{s}'_{(Q)})$ to $\mathscr{V}$
```

```
$\mathscr{K}(1^\lambda)$
  • $\sigma_{\mathsf{ILC}} \leftarrow \mathscr{K}_{\mathsf{ILC}}(1^\lambda)$
  • Parse $\sigma_{\mathsf{ILC}} = (\mathbb{F}, k)$
  • $ck \leftarrow \mathsf{Setup}(1^\lambda)$
  • Return $\sigma = (\sigma_{\mathsf{ILC}}, ck)$

$\mathscr{V}(\sigma, u)$
  • Parse input
      – Parse $\sigma = (\sigma_{\mathsf{ILC}}, ck)$
      – Parse $\sigma_{\mathsf{ILC}} = (\mathbb{F}, k)$
      – Give input $(\sigma_{\mathsf{ILC}}, u)$ to $\mathscr{V}_{\mathsf{ILC}}$
  • Rounds $1 \le i < \mu$:
      – Receive $(\mathbf{c}_i, t_i)$
      – $(\texttt{send}, x_i) \leftarrow \mathscr{V}_{\mathsf{ILC}}(t_i)$
      – Send $x_i$ to $\mathscr{P}$
  • Round $\mu$:
      – Receive $(\mathbf{c}_\mu, t_\mu)$
      – $(\texttt{open}, Q) \leftarrow \mathscr{V}_{\mathsf{ILC}}(t_\mu)$
      – $(\texttt{check}, Q') \leftarrow \mathscr{V}_{\mathsf{ILC}}(t_\mu)$
      – Send $(Q, Q')$ to $\mathscr{P}$
  • Round $\mu + 1$:
      – Receive $(V_{(Q)}, \mathbf{s}_{(Q)}, \mathbf{s}'_{(Q)})$
      – $\mathscr{V}_{\mathsf{ILC}}$ computes the answers $V'_{(Q)}$ to its check queries
      – Check $\mathsf{Commit}(V_{(Q)}; \mathbf{s}_{(Q)}) = Q\mathbf{c}$
      – Check $\mathsf{Commit}(V'_{(Q)}; \mathbf{s}'_{(Q)}) = Q'\mathbf{c}$
      – If all checks pass, return decision of $\mathscr{V}_{\mathsf{ILC}}(V_{(Q)})$, else return 0
```

Figure 9.5: Construction of $(\mathscr{K}, \mathscr{P}, \mathscr{V})$ from $(\mathscr{K}_{\mathsf{ILC}}, \mathscr{P}_{\mathsf{ILC}}, \mathscr{V}_{\mathsf{ILC}})$, and homomorphic commitment scheme $(\mathsf{Setup}, \mathsf{Commit})$.

Measure	**Cost**
Prover Computation	$T_{\mathscr{P}_{\mathsf{ILC}}} + t \cdot \sum_{i=1}^{\mu} T_{\mathsf{Commit}}(k) + T_{\mathrm{Mmul}}(\mathrm{qc}+\mathrm{qc}'+1, t, k+1)$
Verifier Computation	$T_{\mathscr{V}_{\mathsf{ILC}}} + (\mathrm{qc}+\mathrm{qc}') \cdot T_{\mathsf{Commit}}(k+t+1)$
Communication	$C_{\mathsf{ILC}} + t \cdot C_{\mathsf{Commit}} + (k+1)\mathrm{qc} + \mathrm{qc}' + (\mathrm{qc}+\mathrm{qc}')t$
Round Complexity	$\mu+1$

Figure 9.6: Efficiency of a compiled proof of knowledge $(\mathscr{K}, \mathscr{P}, \mathscr{V})$ for $(\sigma, u, w) \in \mathscr{R}$ based on homomorphic commitments. Communication is measured in field elements and computation in field operations. Here κ is the cost of computing a group exponentiation in the commitment space.

Chapter 10

Optimisations outside the ILC Model

10.1 Recursive Argument for Inner Product Evaluation

Using Pedersen commitments as our commitment scheme, we will now give an inner product argument of knowledge of two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$ such that $A = \mathsf{Commit}_{ck}(\mathbf{a})$, $B = \mathsf{Commit}_{ck'}(\mathbf{b})$ and $\mathbf{a} \cdot \mathbf{b} = z$. Here, we assume that $z \in \mathbb{Z}_p$, commitments A, B and commitment keys ck, ck' are known to both the prover and the verifier. The argument will be used as a subroutine where zero-knowledge is not required, so the prover could in principle just reveal the witness $\mathbf{a}, \mathbf{b}$ to the verifier. In the following we show how to use interaction to reduce the communication from linear to logarithmic in n, the length of the vectors.

The basic step in our inner product argument is a 2-move reduction to a smaller statement using techniques similar to [18]. It will suffice for the prover to reveal the witness for the smaller statement in order to convince the verifier about the validity of the original statement. In the full argument, the prover and the verifier recursively run the reduction to obtain increasingly smaller statements. The argument is then concluded with the prover revealing a witness for a very small statement. The outcome of this is a $O(\log n)$-move argument with an overall communication of $O(\log n)$ group and field elements. The inner product argument will be used in the next section to build a logarithmic size argument for circuit satisfiability.

Due to the obvious relationship with Pedersen commitments, we will think of multi-exponentiations $\mathbf{g}^{\mathbf{a}}$ and $\mathbf{h}^{\mathbf{b}}$ as commitments with randomness set to zero, and to $\mathbf{a}, \mathbf{b}$ as openings with respect to commitment keys $\mathbf{g}, \mathbf{h}$.

10.1.1 Main Idea

We now describe the basic step in our argument. Consider the common input for both prover and verifier to be of the form $(\mathbb{G}, \mathbb{Z}_p, ck, A, ck', B, z, m)$ where m divides n, the length of the vectors. For arbitrary n one can always reduce to the case where $m|n$ by appending at most $m-1$ extra commitment keys for single elements to ck and ck'.

We split the bases for the multi-exponentiations into m sets $\mathbf{g} = (\mathbf{g}_1, \ldots, \mathbf{g}_m)$ and $\mathbf{h} = (\mathbf{h}_1, \ldots, \mathbf{h}_m)$, where each set has size $\frac{n}{m}$. We want to prove knowledge of vectors $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$ and $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_m)$ such that

$$A = \mathbf{g}^{\mathbf{a}} = \prod_{i=1}^{m} \mathbf{g}_i^{\mathbf{a}_i} \qquad B = \mathbf{h}^{\mathbf{b}} = \prod_{i=1}^{m} \mathbf{h}_i^{\mathbf{b}_i} \qquad \mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{b}_i = z$$

The key idea is for the prover to replace A with A', a commitment to a shorter vector $\mathbf{a}' = \sum_{i=1}^{m} \mathbf{a}_i x^i$, given a random challenge $x \leftarrow \mathbb{Z}_p^*$ provided by the verifier. In the argument, the prover first computes and sends

$$A_k = \prod_{i=\max(1,1-k)}^{\min(m,m-k)} \mathbf{g}_i^{\mathbf{a}_{i+k}} \quad \text{for } k = 1-m, \ldots, m-1$$

corresponding to the products over the diagonals of the following matrix

$$\begin{array}{cccccc}
 & \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m & \\
\mathbf{g}_1 & \mathbf{g}_1^{\mathbf{a}_1} & \mathbf{g}_1^{\mathbf{a}_2} & \cdots & \mathbf{g}_1^{\mathbf{a}_m} & \\
\vdots & \ddots & \mathbf{g}_2^{\mathbf{a}_2} & \ddots & \vdots & \\
\mathbf{g}_{m-1} & \mathbf{g}_{m-1}^{\mathbf{a}_1} & \ddots & \ddots & \mathbf{g}_{m-1}^{\mathbf{a}_m} & A_{m-1} \\
\mathbf{g}_m & \mathbf{g}_m^{\mathbf{a}_1} & \mathbf{g}_m^{\mathbf{a}_2} & \cdots & \mathbf{g}_m^{\mathbf{a}_m} & \vdots \\
 & & & & & A_{m-2} \\
 & A_{1-m} & A_{2-m} & \cdots & & A_0 = A
\end{array}$$

Notice that $A_0 = A$ is already known to the verifier since it is part of the statement. The verifier now sends a random challenge $x \leftarrow \mathbb{Z}_p^*$.

At this point, both the prover and the verifier can compute $\mathbf{g}' := \prod_{i=1}^{m} \mathbf{g}_i^{x^{-i}}$ and $A' := \prod_{k=1-m}^{m-1} A_k^{x^k}$. If the prover is honest then we have $A' = (\mathbf{g}')^{\mathbf{a}'}$, namely A' is a commitment to $\mathbf{a}'$ under the key $\mathbf{g}'$. Furthermore, even if the prover is dishonest, we can show that if the prover can open A' with respect to the key $\mathbf{g}'$ for $2m-1$ different challenges x, then we can extract openings $(\mathbf{a}_1, \ldots, \mathbf{a}_m)$ corresponding to $A = \prod_{i=1}^{m} \mathbf{g}_i^{\mathbf{a}_i}$.

The same type of argument can be applied in parallel to B with the inverse challenge x^{-1} giving us a sum of the form $\mathbf{b}' = \sum_{i=1}^{m} \mathbf{b}_i x^{-i}$ and a new base $\mathbf{h}' = \prod_{i=1}^{m} \mathbf{h}_i^{x^i}$.

All that remains is to demonstrate that z is the constant term in the product $\mathbf{a}' \cdot \mathbf{b}' = \sum_{i=1}^{m} \mathbf{a}_i x^i \cdot \sum_{j=1}^{m} \mathbf{b}_j x^{-j}$. Similarly to A and B, the prover sends values

$$z_k = \sum_{i=\max(1,1-k)}^{\min(m,m-k)} \mathbf{a}_i \cdot \mathbf{b}_{i+k} \qquad \text{for } k = 1-m, \ldots, m-1$$

where $z_0 = z = \sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{b}_i$, and shows that $z' := \mathbf{a}' \cdot \mathbf{b}' = \sum_{k=1-m}^{m-1} z_k x^{-k}$.

To summarise, after the challenge x has been sent, both parties compute $\mathbf{g}', A', \mathbf{h}', B', z'$ and then run an argument for the knowledge of $\mathbf{a}', \mathbf{b}'$ of length $\frac{n}{m}$. Given $n = m_\mu m_{\mu-1} \cdots m_1$, we recursively apply this reduction over the factors of n to obtain, after $\mu - 1$ iterations, vectors of length m_1. The prover concludes the argument by revealing a short witness associated with the last statement.

10.1.2 Formal description

We now give a formal description of the argument of knowledge introduced above.

Common input: $(\mathbb{G}, p, \mathbf{g}, A, \mathbf{h}, B, z, m_\mu = m, m_{\mu-1} = m', \ldots, m_1)$ such that $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$, $A, B \in \mathbb{G}$ and $n = \prod_{i=1}^{\mu} m_i$.

Prover's witness: $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathbf{b}_1, \ldots, \mathbf{b}_m)$ satisfying

$$A = \prod_{i=1}^{m} \mathbf{g}_i^{\mathbf{a}_i} \qquad B = \prod_{i=1}^{m} \mathbf{h}_i^{\mathbf{b}_i} \qquad \sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{b}_i = z$$

Argument if $\mu = 1$:

P: Send $(a_1, \dots, a_m, b_1, , \dots, b_m)$.

V: Accept if and only if

$$A = \prod_{i=1}^{m} g_i^{a_i} \qquad B = \prod_{i=1}^{m} h_i^{b_i} \qquad \sum_{i=1}^{m} a_i b_i = z$$

Reduction if $\mu \neq 1$:

P: Send $A_{1-m}, B_{1-m}, z_{1-m}, \dots, A_{m-1}, B_{m-1}, z_{m-1}$ where

$$A_k = \prod_{i=\max(1,1-k)}^{\min(m,m-k)} \mathbf{g}_i^{\mathbf{a}_{i+k}} \qquad B_k = \prod_{i=\max(1,1-k)}^{\min(m,m-k)} \mathbf{h}_i^{\mathbf{b}_{i+k}} \qquad z_k = \sum_{i=\max(1,1-k)}^{\min(m,m-k)} \mathbf{a}_i \cdot \mathbf{b}_{i+k}$$

Observe $A_0 = A, B_0 = B, z_0 = z$ so they can be omitted from the message.

V: $x \leftarrow \mathbb{Z}_p^*$.

Both prover and verifier compute a reduced statement of the form

$$(\mathbb{G}, p, \mathbf{g}', A', \mathbf{h}', B', z', m_{\mu-1}, \dots, m_1)$$

where

$$\mathbf{g}' = (\mathbf{g}'_1, \dots, \mathbf{g}'_{m'}) = \prod_{i=1}^{m} \mathbf{g}_i^{x^{-i}} \qquad A' = \prod_{k=1-m}^{m-1} A_k^{x^k}$$

$$\mathbf{h}' = (\mathbf{h}'_1, \dots, \mathbf{h}'_{m'}) = \prod_{i=1}^{m} \mathbf{h}_i^{x^i} \qquad B' = \prod_{k=1-m}^{m-1} B_k^{x^{-k}} \qquad z' = \sum_{k=1-m}^{m-1} z_k x^{-k}$$

The prover computes a new witness as $(\mathbf{a}'_1, \dots, \mathbf{a}'_{m'}) = \sum_{i=1}^{m} \mathbf{a}_i x^i$ and $(\mathbf{b}'_1, \dots, \mathbf{b}'_{m'}) = \sum_{i=1}^{m} \mathbf{b}_i x^{-i}$.

10.1.2.1 Security Analysis.

Theorem 7 *The argument has perfect completeness and computational witness extended emulation for either extracting a non-trivial discrete logarithm relation or a valid witness.*

Proof Perfect completeness can be verified directly. To prove witness-extended emulation we start by giving an extractor that either extracts a witness for the original statement or a non-trivial discrete logarithm relation.

For $\mu = 1$ we have (perfect) witness-extended emulation since the prover reveals a witness and the verifier checks it.

Before discussing extraction in the recursive step, note that if we get a non-trivial discrete logarithm relation for $\mathbf{g}'_1, \dots, \mathbf{g}'_{m'}$ then we also get a non-trivial discrete logarithm relation for $\mathbf{g}_1, \dots, \mathbf{g}_m$, since $x \neq 0$. A similar argument applies to $\mathbf{h}'_1, \dots, \mathbf{h}'_{m'}$ and $\mathbf{h}_1, \dots, \mathbf{h}_m$.

Now, assume we get witness $\mathbf{a}', \mathbf{b}'$ such that

$$A' = \prod_{k=1-m}^{m-1} A_k^{x^k} = \left(\prod_{i=1}^{m} \mathbf{g}_i^{x^{-i}}\right)^{\mathbf{a}'} \qquad B' = \prod_{k=1-m}^{m-1} B_k^{x^{-k}} = \left(\prod_{i=1}^{m} \mathbf{h}_i^{x^{i}}\right)^{\mathbf{b}'} \qquad \mathbf{a}' \cdot \mathbf{b}' = \sum_{k=1-m}^{m-1} z_k x^{-k}$$

for $2m-1$ different challenges $x \in \mathbb{Z}_p^*$. We will show that they yield either a witness for the original statement, or a non-trivial discrete logarithm relation for either $\mathbf{g}_1, \dots, \mathbf{g}_m$ or $\mathbf{h}_1, \dots, \mathbf{h}_m$.

Take $2m-1$ different challenges $x \in \mathbb{Z}_p^*$. They form a shifted Vandermonde matrix with rows $(x^{1-m}, x^{2-m}, \dots, x^{m-1})$. By taking appropriate linear combinations of the vectors we can obtain any unit vector $(0, \dots, 0, 1, 0, \dots, 0)$. Taking the same linear combinations of the $2m-1$ equations

$$\prod_{k=1-m}^{m-1} A_k^{x^k} = \left(\prod_{i=1}^{m} \mathbf{g}_i^{x^{-i}}\right)^{\mathbf{a}'} \quad \text{we get vectors } \mathbf{a}_{k,i} \text{ such that} \quad A_k = \prod_{i=1}^{m} \mathbf{g}_i^{\mathbf{a}_{k,i}}$$

For each of the $2m-1$ challenges, we now have $\prod_{k=1-m}^{m-1} A_k^{x^k} = \left(\prod_{i=1}^{m} \mathbf{g}_i^{x^{-i}}\right)^{\mathbf{a}'}$, which means that *for all i* we have

$$x^{-i}\mathbf{a}' = \sum_{k=1-m}^{m-1} \mathbf{a}_{k,i} x^k$$

unless we encounter a non-trivial discrete logarithm relation for $\mathbf{g}_1, \dots, \mathbf{g}_m$. This means that $\mathbf{a}' = \sum_{k=1-m}^{m-1} \mathbf{a}_{k,i} x^{k+i}$ for all i, and in particular $\sum_{k=1-m}^{m-1} \mathbf{a}_{k,i} x^{k+i} =$

$\sum_{k=1-m}^{m-1} \mathbf{a}_{k,1} x^{k+1} = \sum_{k=1-m}^{m-1} \mathbf{a}_{k,m} x^{k+m}$. Matching terms of degree outside $\{1,\ldots,m\}$ reveals $\mathbf{a}_{k,i} = 0$ for $k+i \notin \{1,\ldots,m\}$. Defining $\mathbf{a}_i = \mathbf{a}_{0,i}$, and matching terms of similar degree we get

$$\mathbf{a}_{k,i} = \begin{cases} \mathbf{a}_{k+i} & \text{if } k+i \in \{1,\ldots,m\} \\ 0 & \text{otherwise} \end{cases}$$

This means

$$\mathbf{a}' = \sum_{k=1-m}^{m-1} \mathbf{a}_{k,1} x^{k+1} = \sum_{k=0}^{m-1} \mathbf{a}_{k+1} x^{k+1} = \sum_{i=1}^{m} \mathbf{a}_i x^i$$

A similar analysis of $B_{1-m},\ldots,B_{m-1}$ and openings $\mathbf{b}'$ for $2m-1$ different challenges $x^{-1} \in \mathbb{Z}_p^*$ gives us either a non-trivial discrete logarithm relation for $\mathbf{h}_1,\ldots,\mathbf{h}_m$ or vectors $\mathbf{b}_i$ such that $\mathbf{b}' = \sum_{i=1}^{m} \mathbf{b}_i x^{-i}$ and $B = \prod_{i=1}^{m} \mathbf{h}_i^{\mathbf{b}_i}$.

Finally, with $\sum_{i=1}^{m} \mathbf{a}_i x^i \cdot \sum_{j=1}^{m} \mathbf{b}_j x^{-j} = \sum_{k=1-m}^{m-1} z_k x^{-k}$ for $2m-1$ different challenges we get $z = z_0 = \sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{b}_i$.

We can now apply the forking lemma to a tree of size $(2m_\mu - 1)(2m_{\mu-1} - 1)\cdots(2m_2 - 1) \leq n^2$, which is polynomial in λ, to conclude that the argument has witness-extended emulation. ■

10.1.3 Efficiency.

The recursive argument uses $2\mu - 1$ moves. The communication cost of all steps sums up to $4\sum_{i=2}^{\mu}(m_i - 1)$ group elements and $2\sum_{i=2}^{\mu}(m_i - 1) + 2m_1$ field elements.

At each iteration, the main cost for the prover is computing the A_k and B_k values, using less than $\frac{4(m_\mu^2 m_{\mu-1} \ldots m_1)}{\log(m_\mu \ldots m_1)}$ group exponentiations via multi-exponentiation techniques, and the z_k values using $m_\mu^2 m_{\mu-1} \cdots m_1$ field multiplications. The cost of computing the reduced statements is dominated by $\frac{2(m_\mu m_{\mu-1} \ldots m_1)}{\log m_\mu}$ group exponentiations for both the prover and the verifier. In the case where $m_\mu = \ldots = m_1 = m$, the verifier complexity is bounded above by $\frac{2m^\mu}{\log m} \frac{m}{m-1}$ group exponentiations. The prover complexity is bounded above by $\frac{6m^{\mu+1}}{\log m} \frac{m}{m-1}$ group exponentiations and $m^{\mu+1} \frac{m}{m-1}$ field multiplications.

10.1.4 How to use this Argument

At the end of the argument of Section 8.5, once compiled using Pedersen commitments, the verification equations verify exactly the statement of the recursive argument above. This means that rather than the prover sending the vector **r** to the verifier in that argument, along with some commitment randomness which appears in the compiled argument, the prover and verifier can conduct an inner product argument like the one above to prove that the prover knows values which satisfy the verification equations. To get a logarithmic communication complexity, one can use the original 5-move arithmetic circuit argument with extremely long vectors, of length $O(N)$, where N is the number of multiplication gates in the arithmetic circuit. Then the prover and verifier run the recursive argument with $m_i = 2$ for all i.

The argument is presented in stand-alone form in Appendix B. This argument has been implemented in Python. It is closely related to the Bulletproofs protocol given in [1], which has been implemented in various different languages including Java, C and Rust, and deployed as part of the Monero cryptocurrency.

10.1.5 Generalisations

Our original recursive argument applied to Pedersen commitments, which have the following form:

$ck = (g_1, \dots, g_n, h)$

$\mathsf{Commit}(a_1, \dots, a_n; r) = h^r \prod_{i=1}^n g_i^{a_i}$

We applied the argument to prove that committed values gave a particular scalar product:

$z = \mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^N a_i b_i$

Taking a step back, consider g^a and ab. Both of these functions are very simple bilinear maps. Scalar products and Pedersen Commitments (or multi-exponentiations) come from combining multiple instances of these simple bilinear maps.

Thus, under the right conditions, we can replace these with other similar bilinear maps and recover a recursive argument. For example, we have the following commitment scheme for group elements:

$ck = (g_1, \ldots, g_n, h)$

$\mathsf{Commit}(a_1, \ldots, a_n; r) = e(h, r) \prod_{i=1}^{n} e(g_i, a_i)$

We may wish to prove for example that a committed vector of group elements and a committed vector of field elements have a particular multi-exponentiation, or that two committed vectors of group elements have a particular pairing-product.

Bilinear Commitment Scheme Suppose that $f : G \times X \to \mathscr{A}$ is a non-degenerate bilinear map between abelian groups. For any N, we can create a homomorphic commitment scheme with key space G^N, message space X^N and commitments in $\mathscr{A}$ as follows:

$ck = (g_1, \ldots, g_N, g_R) \leftarrow G^{N+1}$

$F(g_1, \ldots, g_N, g_R; x_1, \ldots, x_N, x_R) = \mathsf{Commit}(x_1, \ldots, x_N; x_R) = f(g_R, x_R) \prod_{i=1}^{N} f(g_i, x_i)$

Since f is bilinear, the commitment scheme is homomorphic in both messages and keys. Since f is non-degenerate, if x_R is chosen uniformly at random, then the commitment will be uniformly random too.

Relevant instantiations do have the perfectly hiding property, but we are more interested in the binding property at present.

Lemma 24 (Linear Independence) *Assume that given $(g_1, \ldots, g_N, g_R) \leftarrow G^{N+1}$, it is difficult to efficiently find $x_1, \ldots, x_N, x_R$ such that $f(g_R, x_R) \prod_{i=1}^{N} f(g_i, x_i) = 1$. Then the commitment scheme is computationally binding.*

This includes the homomorphic commitment scheme for group elements used in [68], where $G = \mathbb{G}_1, X = \mathbb{G}_2, \mathscr{A} = \mathbb{G}_T$ and our bilinear map is the pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, under the Double Pairing Assumption.

It also includes the Pedersen commitment scheme where $G = \mathbb{G}, X = \mathbb{Z}_p, \mathscr{A} = \mathbb{G}$, and $f(g, x) = g^x$, under the Discrete Logarithm Assumption.

When all of the groups involved are $\mathbb{F}$-modules, one can generalise the recursive scalar product argument for two Pedersen-committed values to a recursive bilinear product argument for two values committed using bilinear commitment schemes, with an extremely similar protocol and security proof. The argument also works with two different bilinear commitment schemes. It would be interesting to see whether an argument of this form using the pairing-based commitment scheme of [68] could be used in conjunction with the logarithmic communication protocol resulting from the scalar product argument above, to get an interactive argument with sub-logarithmic communication complexity.

10.2 Field Extension Techniques

The basis for the techniques in this section was originally published in joint work [9] with Carsten Baum, Andrea Cerulli, Rafael del Pino, Jens Groth and Vadim Lyubashevsky as part of a post-quantum zero-knowledge protocol based on lattice cryptography. We do not explain the details of the protocol, here, as they are unnecessary and lattice cryptography is beyond the scope of this thesis. The inspiration for these techniques came from [41] and [40].

Suppose that we have an ILC protocol which works over a small finite field $\mathbb{F}$. Since soundness of these protocols is proved using the Schwarz-Zippel lemma, the soundness error in such protocols is limited to $O(1/|\mathbb{F}|)$. If the proof relates to an algebraic statement which is naturally phrased over a small field, one approach which yields protocols with low soundness error is to simulate the behaviour of the smaller field in the larger one, and produce a proof for an equivalent relation over a larger field. However, this usually leads to significant overhead costs. Following a naïve approach, one can imagine, for example, embedding a boolean circuit into a large prime field. Then, running an ILC protocol on large field elements is wasteful, when some large field elements actually only contain boolean values. Instead, we show how to embed many elements from a finite field into one single extension field element, and also to perform useful operations on base field elements using operations on extension field elements.

This is a viable approach when working with the hash-based commitment scheme from chapter 5.

Based on [39], we now show how to embed witness elements into field extensions. This will allow us to use the generic 3-move protocol from Chapter 8 for arithmetic circuit satisfiability with $O(1/|\mathbb{F}|^{2k})$ soundness error, which gives protocols with negligible soundness error in a single run.

For simplicity, we explain how our techniques work over $\mathbb{Z}_p$. However, they will also work over other finite fields.

Let $GF(p^{2k}) \simeq \mathbb{Z}_p[\phi]/\langle f(\phi)\rangle$, where f is a polynomial of degree $2k$ that is irreducible over $\mathbb{Z}_p$. Our goal is to embed k elements of $\mathbb{Z}_p$ into the extension field in a way so that we can multiply two $GF(p^{2k})$ elements in a way that does not interfere with the products of the original $\mathbb{Z}_p$ elements. Let $e_1,\ldots,e_k$ be distinct interpolation points in $\mathbb{Z}_p$ (note that in particular, this forces $p > k$). Let $l_1(X),\ldots,l_k(X)$ be the Lagrange polynomials associated with the points e_i, which have degree $k-1$. Let $l_0(X) = \prod_{j=1}^{k}(X-e_i)$, which has degree k.

Now, suppose that we have $a_1,\ldots,a_k$, $b_1,\ldots,b_k$ and $c_1,\ldots,c_k$ in $\mathbb{Z}_p$ such that $a_j \cdot b_j = c_j \mod p$ for each j. By evaluating the expression at each interpolation point, we see that the following statement about polynomials holds over $\mathbb{Z}_p$: $\left(\sum_{j=1}^{k} a_j l_j(X)\right)\cdot\left(\sum_{j=1}^{k} b_j l_j(X)\right) \equiv \left(\sum_{j=1}^{k} c_j l_j(X)\right) \mod l_0(X)$.

Therefore, there are $c'_0,\ldots,c'_{k-2} \in \mathbb{Z}_p$ such that $\left(\sum_{j=1}^{k} a_j l_j(X)\right)\cdot\left(\sum_{j=1}^{k} b_j l_j(X)\right) = \left(\sum_{j=1}^{k} c_j l_j(X)\right) + l_0(X)\sum_{j=0}^{k-2} c'_j X^j$.

The degree of f is $2k$, so if we choose the basis

$$\mathscr{B} = \{l_1(\phi),\ldots,l_k(\phi),l_0(\phi),\phi l_0(\phi),\ldots,\phi^{k-1}l_0(\phi)$$

for $GF(p^{2k})\}$ we can perform multiplications of extension field elements without any overflow modulo f interfering with the individual product relations $a_ib_i = c_i$ in $\mathbb{Z}_p$. We can therefore incorporate the above equality into $GF(p^{2k})$ as the equality $\left(\sum_{j=1}^{k} a_j l_j(\phi)\right)\cdot\left(\sum_{j=1}^{k} b_j l_j(\phi)\right) = \left(\sum_{j=1}^{k} c_j l_j(\phi)\right) + l_0(\phi)\sum_{j=0}^{k-2} c'_j \phi^j$.

This allows one multiplication of committed values to be performed without any overflow modulo f. As we shall see, this is sufficient for verifying multiplication triples for arithmetic circuit satisfiability.

We also need to be able to view single commitments to elements of $\mathbb{Z}_p$ as elements of the extension field in a way that helps to verify linear consistency relations between the elements.

Now, suppose that we have $a_1,\ldots,a_k$, $b_1,\ldots,b_k$ and $c_1,\ldots,c_k$ in $\mathbb{Z}_p$, and coefficients $w_{a,1},\ldots,w_{a,k}$, $w_{b,1},\ldots,w_{b,k}$ and $w_{c,1},\ldots,w_{c,k}$ in $\mathbb{Z}_p$ such that

$\sum_{j=1}^{k} a_j w_{a,j} + \sum_{j=1}^{k} b_j w_{b,j} + \sum_{j=1}^{k} c_j w_{c,j} = K \mod p$. By comparing coefficients, we see that the following statement about polynomials holds over $\mathbb{Z}_p$: $\left(\sum_{j=1}^{k} a_j X^{j-1}\right) \cdot \left(\sum_{j=1}^{k} w_{a,j} X^{k-j}\right) + \left(\sum_{j=1}^{k} b_j X^{j-1}\right) \cdot \left(\sum_{j=1}^{k} w_{b,j} X^{k-j}\right) + \left(\sum_{j=1}^{k} c_j X^{j-1}\right) \cdot \left(\sum_{j=1}^{k} w_{c,j} X^{k-j}\right) = KX^{k-1} + \sum_{j=0, j\neq k-1}^{2k-2} K_j X^j$, where the K_j are extra terms determined from the a, b, c and w values.

If we choose the basis $\mathscr{B}' = 1, \phi, \phi^2, \ldots, \phi^{2k-1}$ for $GF(p^{2k})$, we can perform multiplications of extension field elements in a way that always yields a useful linear relation in the ϕ^{k-1} term without any overflow modulo f.

By viewing multiplication in $GF(p^{2k})$ as a linear map over $\mathbb{Z}_p^{2k}$, we can simulate arithmetic in the extension field using arithmetic in $\mathbb{Z}_p^{2k}$.

Let $A_1, \ldots, A_{2k} \in \mathscr{C}^{2k}$ be homomorphic commitments to single elements, $a_1, \ldots, a_k \in \mathbb{Z}_p$. We can consider the tuple $\mathbf{A} = (A_1, \ldots, A_k)$ to be a commitment to an element $\mathbf{a} = (a_1, \ldots, a_{2k})$ of $GF(p^{2k})$. Now, if we consider $\mathbf{x} \in \mathbb{Z}_p^{2k}$ as an element of $GF(p^{2k})$, then there is a matrix $M_{\mathbf{x}}$ which simulates multiplication by $\mathbf{x}$ in $\mathbb{Z}_p^{2k}$ when we multiply on the left by $M_{\mathbf{x}}$. Since the A_i are homomorphic commitments, we can obtain a commitment to $\mathbf{x} * \mathbf{a}$ by computing $M_{\mathbf{x}}\mathbf{A}$, where $*$ represents multiplication in $GF(p^{2k})$.

10.2.1 Reduction of Circuit Satisfiability to a Hadamard Matrix Product and Linear Constraints over $GF(p^{2k})$.

Let $N = mnk$ be the number of multiplication gates in the arithmetic circuit. To reduce circuit satisfiability to constraints over $GF(p^{2k})$, we can consider the same polynomial equations as before, written over $GF(p^{2k})$ rather than $\mathbb{Z}_p$. We consider the rows of matrices A, B, and C as before, but this time, we label the row vectors of the matrices $\mathbf{a}_{i,j}, \mathbf{b}_{i,j}$ and $\mathbf{c}_{i,j} \in \mathbb{Z}_p^n$, for $1 \leq i \leq m$ and $1 \leq j \leq k$. Now, we consider the row vectors $\mathbf{a}_{i,1}, \ldots, \mathbf{a}_{i,k}$, which are elements of $\mathbb{Z}_p^n$, as an element in $GF(p^{2k})^n$.

Let $a_i = \left(\mathbf{a}_{i,1}, \mathbf{a}_{i,2}, \ldots, \mathbf{a}_{i,k}, \mathbf{0}, \ldots, \mathbf{0}\right)^T$ represent this element in $GF(p^{2k})^n$. Each column of the matrix represents a separate element of $GF(p^{2k})$.

Satisfiability conditions over $\mathbb{Z}_p$ were embedded using scalar products, denoted by $\cdot$, and element-wise products, denoted by $\circ$. If a and b in $\mathbb{Z}_p^{2k\times n}$ represent elements of $GF(p^{2k})^n$, then each column represents an element of $GF(p^{2k})$, and the scalar products and element-wise products of a and b are computed using the columns. We denote the element-wise product by $a \bigcirc b$ and the scalar product by $a \bigodot b$ to avoid confusion with any other matrix products on a and b.

$$a = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{pmatrix} \quad b = \begin{pmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_n \end{pmatrix}$$

$$a \bigcirc b = \begin{pmatrix} M_{\mathbf{v}_1}\mathbf{w}_1 & M_{\mathbf{v}_2}\mathbf{w}_2 & \dots & M_{\mathbf{v}_n}\mathbf{w}_n \end{pmatrix} \quad a \bigodot b = M_{\mathbf{v}_1}\mathbf{w}_1 + M_{\mathbf{v}_2}\mathbf{w}_2 + \dots + M_{\mathbf{v}_n}\mathbf{w}_n$$

Note that in the verification equations, although the verifier computes high powers of random challenges $\mathbf{x}$ and $\mathbf{y}$, the verifier only computes quadratic polynomials of values such as a and b which have been sent by the prover. This is important, because when we expand a and b in terms of their coefficients a_i and b_i, we see that the verifier only computes expressions which have degree 2 in the prover's secret committed wire values, embedded as elements of $GF(p^{2k})$. Therefore, considering a field extension of degree $2k$ with the basis $\mathscr{B}$ is sufficient for our purposes: we only need to ensure that a single multiplication in $GF(p^{2k})$ preserves the individual product relations embedded in the $GF(p)$ elements.

When embedding satisfiability conditions into a polynomial over $\mathbb{Z}_p$, using random challenges $x, y \in \mathbb{Z}_p$, the prover could send linear combinations of vectors $\mathbf{a}_i \in \mathbb{Z}_p^n$ such as $\mathbf{a}(x) = \mathbf{a}_0 + \sum_{i=1}^m \mathbf{a}_i y^i x^i$ to the verifier.

However, when embedding satisfiability conditions into a polynomial over $GF(p^{2k})$, using random challenges $\mathbf{x}, \mathbf{y} \in GF(p^{2k})$, the prover sends linear combinations of vectors $a_i \in GF(p^{2k})^n$ such as $a(x) = a_0 + \sum_{i=1}^m (M_\mathbf{y})^i (M_\mathbf{x})^i a_i$.

10.2.2 How to Apply This Idea

Suppose that we wish to give an argument for arithmetic circuit satisfiability over a small field $\mathbb{F}$. The way to use this technique to apply the 3-move square-root protocol over an algebraic extension of $\mathbb{F}$, and use the above technique to embed the conditions for circuit satisfiability into extension field operations. This yields an arithmetic circuit satisfiability argument over $\mathbb{F}$ with negligible soundness error in a single run, and the argument over the extension field will have the same asymptotic communication costs as one over the base field, but with negligible soundness error.

This idea was used by the author for the first time in [39], but drew inspiration from works such as [41] and [40] which used related techniques to prove much simpler statements. The advantage of using this idea over simply repeating the protocol many times in parallel is that the communication overhead is reduced. For example, if one uses a basic protocol over $\mathbb{Z}_q$ to prove a single statement u with soundness error approximately $1/q$, then the protocol may need to be repeated $O(t)$ times, for some t, to provide a negligible soundness error. Each repetition is only proving the same statement. By contrast, the techniques above allow one to embed $O(t)$ different statements, or $O(t)$ different portions of a single large statement, into a proof of the same size as the repeated proof. When the desired security level is λ bits, and a proof is given over $\mathbb{F}$, these techniques reduce communication overhead by a factor of $\lambda/\log|\mathbb{F}|$ compared with naively performing parallel repetitions.

Chapter 11

Conclusion

In this thesis, we introduced the ILC model, modified it to bring it closer to real protocols, and gave compilations from ILC protocols to real zero-knowledge protocols based on hash functions and error-correcting codes. The compilations separate the cryptographic and non-cryptographic parts of the design process and simplify the protocol design process. In particular, designing our ILC protocols and proving them secure was a matter of applying linear algebra and simple lemmata about polynomial identity testing. Proving that the ILC protocols could be securely compiled into real arguments was more complicated, but was done once and for all, and the compilations can be reused for many ILC protocols in the future. We presented protocols with state-of-the-art communication complexity and round complexity, and showed that the ILC model is powerful enough to reason about both general NP-Complete statements like arithmetic circuit satisfiability. We also gave ILC protocols for simpler statements such as polynomial evaluation or range proofs, in a manner that leads to highly efficient protocols. This included the framing of general relations to capture a class of zero-knowledge protocols characterised by low-degree polynomials, formalising the techniques used in such protocols, and providing a generic protocol for reasoning about such relations, which can be used to give batch-proofs for many statements at the same time.

We found techniques used in interactive zero-knowledge protocols in the discrete logarithm setting, and rewrote many of those protocols in the ILC model. Thus, this work shows that a great many discrete logarithm arguments follow the same basic design paradigms. Surprisingly, the same style of protocol and design techniques extend beyond the discrete logarithm setting to another commitment scheme which is not homomorphic! This shows that the ILC model addresses our goals of providing a good abstraction for discrete-logarithm-based protocols, which is also useful outside its original setting.

Using only this methodology, we were able to present some protocols with state-of-the-art communication complexity. Examples include a discrete-logarithm based polynomial evaluation argument, with a better asymptotic communication complexity than observed prevously, and a discrete-logarithm based membership argument, whose asymptotic communication complexity has improved constants over previous work, and which has highly tuneable parameters. These are of practical significance as they can be used as part of membership and non-membership arguments both in the designs of other primitives, like group and ring signatures, and in applications such as preventing double-spending in cryptocurrencies. Since ILC protocols can also be compiled based on hash functions and error-correcting codes, we also obtain some completely new arguments for polynomial evaluation and membership based on the existence of collision resistant hash functions. This shows that our goals of designing efficient protocols for a wide variety of different applications was also addressed.

We also presented some extra techniques which fall outside the ILC model, namely, a recursive argument to show that committed values have a particular scalar product, and a field extension technique which boosts the soundness of ILC protocols over small fields. This is at once a strength and a weakness of using idealised communication models. Protocols inside such models are highly constrained, which makes them easier to design and reason about, but may also limit their performance and utility. The fact that the most efficient protocol in this thesis, the logarithmic-communication argument for arithmetic circuit satisfiability, does not lie within the main model of communication, is a limitation. However, once a suitable model has been identified, one can also try to design useful protocols by attempting to create protocols outside the model.

There are other zero-knowledge protocols [39], some based on lattices, and some based on the Strong RSA assumption, which seem to work on the same basis as ILC protocols. That is, the prover commits to certain vectors, and the verifier picks a random challenge, and uses structured linear combinations of the committed vectors in a number of verification equations. Unlike in the ILC model, in which all elements belong to a field and the notion of size is not important, these settings require careful consideration of the size of committed elements to ensure zero-knowledge, and often for soundness too. The model falls short of capturing these protocols. Improving the model to take this into account, in particular for lattice-based protocols which may enjoy post-quantum security guarantees, is an attractive target for future research.

Another avenue that was not investigated is restricting the verifier's ILC queries. In all of the ILC protocols presented in this thesis, the coefficients of the verifier's linear queries are given by a linearly-independent set of polynomials evaluated at uniformly random challenges chosen by the verifier. The queries have a carefully chosen algebraic structure. For every protocol that we give, the query matrix appears to be a form of strongly universal hash function. The compilation from ILC protocols to discrete-logarithm based protocols requires restrictions on the rank and dimension of the matrix, and that a related system of linear equations can be solved. These conditions are treated in an ad-hoc manner outside of the proofs that the protocols are secure in the idealised model. There is still a gap between the model and the compiled protocols, and the communication model can be refined further. One could hope that such strong algebraic restrictions lead to interesting results, such as lower bounds on the communication complexity of ILC protocols, as linear algebra is an old discipline with many results that one could hope to apply to the structure of the query matrices.

Appendix A

Security Proofs for Optimised Hash and Error-Correcting-Code Compilation

Theorem 8 (Completeness) *If* $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ *is complete for relation* $\mathcal{R}$ *over* ILC*, then the compiled* $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ *with optimisations from Subsection 9.1.5 is complete for relation* $\mathcal{R}$*.*

Proof All the commitment openings are correct, so they will be accepted by the verifier. In the execution of $\langle \mathcal{P}(\sigma, u, w) \longleftrightarrow \mathcal{V}(\sigma, u) \rangle$, the fact that $\mathsf{E}_{\mathcal{C}}$ is linear and Reed-Solomon codes are linear implies that $\tilde{\mathsf{E}}_{\mathcal{C}}$ is linear and hence all the linear checks will be true. If $(\sigma, u, w) \in \mathcal{R}$ then $(\sigma_{\mathsf{ILC}}, u, w) \in \mathcal{R}$ and being complete $\langle \mathcal{P}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, u, w) \overset{\mathsf{ILC}}{\longleftrightarrow} \mathcal{V}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, stm) \rangle = 1$ so $\mathcal{V}$'s internal copy of $\mathcal{V}_{\mathsf{ILC}}$ will accept. Thus, in this case, $\langle \mathcal{P}(\sigma, u, w) \longleftrightarrow \mathcal{V}(\sigma, u) \rangle = 1$, which proves completeness. ■

Theorem 9 (Knowledge Soundness) *If* $(\mathcal{K}_{\mathsf{ILC}}, \mathcal{P}_{\mathsf{ILC}}, \mathcal{V}_{\mathsf{ILC}})$ *is statistically knowledge sound with a straight-line extractor for relation* $\mathcal{R}$ *over* ILC *and* $(\mathsf{Setup}, \mathsf{Commit})$ *is computationally (statistically) binding, then* $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ *with the optimisations from Subsection 9.1.5 is computationally (statistically) knowledge sound for relation* $\mathcal{R}$*.*

Proof

Throughout the proof of Theorem 2, we used the fact that $\tilde{\mathsf{E}}_{\mathcal{C}}$ is linear. This is still the case, due to the linearity of Reed-Solomon codes. The only part of the proof of Theorem 2 that used the form of $\tilde{\mathsf{E}}_{\mathscr{C}}$ as part of the proof was Lemma 23. We reprove this lemma.

Lemma 25 *The probability that for some τ there are no $\mathbf{v}_\tau$ and $\mathbf{r}_\tau$ such that $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_\tau, \mathbf{r}_\tau)$ agrees with $\mathbf{e}_\tau$ on the opened $j \in \bigcup_{i=1}^{2n} J_i$ and $b = 1$ is negligible.*

In particular, the probability that $b = 1$ but $\mathscr{T}$ does not extract the transcript of $\mathscr{P}^*_{\mathsf{ILC}}$ is negligible.

Proof Since we can ignore events that happen with negligible probability, and the expected number of rewindings is polynomial, we can assume that in all the rewindings, $\mathscr{P}^*$ only makes openings to the most common openings. We showed that the probability that $b = 1$ but $\mathscr{P}^*$ sends a $V^*_{(Q)} \neq QV$, or $\mathscr{V}$ computes a $V'^*_{(Q)} \neq Q'V$ is negligible and by the same argument the probability that $b = 1$ but $\mathscr{P}^*$ sends $\mathbf{v}^*_{(\gamma)} \neq \mathbf{v}_{(\gamma)}$ is negligible. Therefore, in the following, we will assume $\mathbf{v}^*_{(\gamma)} = \mathbf{v}_{(\gamma)}$.

Now suppose that there is some $\mathbf{e}_\tau$ such that the opened values are inconsistent with being $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_\tau, \mathbf{r}_\tau)$ for any $\mathbf{r}_\tau$. That is, either there is some j such that $j, n+j \in \bigcup_{i=1}^{2n} J_i$ and $(\mathbf{e}_\tau)_j - (\mathbf{e}_\tau)_{n+j} \neq \mathsf{E}_{\mathscr{C}}(\mathbf{v})_j$, or there exists some set $J' \subset [n+1, 2n]$ of size at least $2\lambda + 1$ such that $(\mathbf{e}_\tau)_j$ are not consistent with a Reed-Solomon codeword. Note that if $n = 2\lambda$, then a suitable Reed-Solomon codeword always exists by interpolation, so we do not need to consider this value of n. The first case with j and $n + j$ has already been covered by Lemma 23. Therefore, we prove the second case.

If we are in the second case, then there exist $\mu_j \in \mathbb{F}$ which give a parity check on J' such that $\sum_{j \in J'} \mu_j (\mathbf{e}_\tau)_j \neq 0$. This can be seen as follows. If the values are not consistent with a Reed-Solomon codeword, then that means that there is no polynomial whose evaluations would be consistent with polynomial evaluations $(\mathbf{e}_\tau)_j$ for $j \in J'$. Therefore, there exist $2\lambda + 1$ values $j \in J'$, and $j^* \in J'$ such that if we interpolate to compute the polynomial which evaluates to $(\mathbf{e}_\tau)_j$ at points a_j, and then evaluate at a_{j^*}, the result is not equal to $(\mathbf{e}_\tau)_{j^*}$. Interpolation from evaluations at the points a_j and then evaluation at a_{j^*} are both linear operations, and composing both linear maps yields the values μ_j.

Then,for uniformly chosen $\gamma_\tau \in \mathbb{F}$, we get that $\gamma_\tau (\sum_{j \in J'} \mu_j (\mathbf{e}_\tau)_j)$ is uniformly distributed in $\mathbb{F}$. Hence for a random $\gamma \in \mathbb{F}^t$, we have that $\gamma(\sum_{j\in J'} \mu_j(\mathbf{e})_j)$ is uniformly distributed. When $\mathscr{V}$ sends γ, $\mathscr{P}^*$ will respond with $\mathbf{v}^*_{(\gamma)} = \mathbf{v}_{(\gamma)}$ and some $\mathbf{r}^*_{(\gamma)}$. $\mathscr{V}$ will only accept on a challenge J if for all $j \in J$ we have $(\mathbf{e}_0 + \gamma\mathbf{e})_j = \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}, \mathbf{r}^*_{(\gamma)})_j$. Since $J' \subset \bigcup_{i=1}^{2n} J_i$ we have $(\mathbf{e}_0 + \gamma\mathbf{e})_j = \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}, \mathbf{r}^*_{(\gamma)})_j$ for each $j \in J'$ so

$$\left(\sum_{j\in J'} \mu_j(\mathbf{e}_0)_j\right) + \gamma\left(\sum_{j\in J'} \mu_j(\mathbf{e})_j\right) = \sum_{j\in J'} \mu_j \tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)}, \mathbf{r}^*_{(\gamma)})_j = 0$$

that is,

$$\left(\sum_{j\in J'} \mu_j(\mathbf{e}_0)_j\right) = -\gamma\left(\sum_{j\in J'} \mu_j(\mathbf{e})_j\right)$$

For random γ the right-hand side is uniform and the left-hand side is fixed, hence equality only happens with negligible probability. That proves the lemma. ■

Theorem 10 (SHVZK) *If* $(\mathscr{K}_{\mathsf{ILC}}, \mathscr{P}_{\mathsf{ILC}}, \mathscr{V}_{\mathsf{ILC}})$ *is perfect SHVZK and* $(\mathsf{Setup}, \mathsf{Commit})$ *is computationally (statistically) hiding then* $(\mathscr{K}, \mathscr{P}, \mathscr{V})$ *with the optimisations from Subsection 9.1.5 is computationally (statistically) SHVZK.*

Proof To prove we have SHVZK we describe how the simulator $\mathscr{S}(\sigma, u, \rho)$ should simulate the view of $\mathscr{V}$. Along the way, we will argue why, the variables output by $\mathscr{S}$ have the correct joint distribution. To keep the proof readable, instead of saying that "the joint distribution of [random variable] and all previously defined random variables is identical to the distribution in the real view of $\mathscr{V}$ in $\langle \mathscr{P}(\sigma, u, w) \longleftrightarrow \mathscr{V}(\sigma, u)\rangle$" we will simply say that "[random variable] has the correct distribution".

Using the randomness ρ the simulator learns the queries $\rho_{\mathsf{ILC}} = (x_1, \ldots, x_{\mu-1}, Q)$ the internal $\mathscr{V}_{\mathsf{ILC}}$ run by the honest $\mathscr{V}$ will send. $\mathscr{S}$ can therefore run $\mathscr{S}_{\mathsf{ILC}}(\sigma_{\mathsf{ILC}}, u, \rho_{\mathsf{ILC}})$ to simulate the view of the internal $\mathscr{V}_{\mathsf{ILC}}$. This gives it $(t_1, \ldots, t_\mu, V_{(Q)})$. By the SHVZK property of $(\mathscr{K}_{\mathsf{ILC}}, \mathscr{P}_{\mathsf{ILC}}, \mathscr{V}_{\mathsf{ILC}})$ these random variables will all have the correct joint distribution.

Then $\mathscr{S}$ reads the rest of ρ to learn also the challenges γ and J that $\mathscr{V}$ will send. The simulator picks uniformly at random $\mathbf{v}_{(\gamma)} \leftarrow \mathbb{F}^k$. Since in a real proof $\mathbf{v}_0$ is chosen at random, we see that the simulated $\mathbf{v}_{(\gamma)}$ has the correct distribution. Now $\mathscr{S}$ picks $E_{01}|_J, \ldots, E_\mu|_J$ uniformly at random. Recall that this time $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v};\mathbf{r}) = (\mathsf{E}_{\mathscr{C}}(\mathbf{v}) + \mathbf{r}, \mathbf{r})$

where $\mathbf{r}$ is the Reed-Solomon encoding of $\mathbf{r}' \in \mathbb{F}^{2\lambda}$. By definition of J being allowed, we have for all $j \in J$ that $j+n \notin J$. This means for any choice of $\mathbf{v}_0 \in \mathbb{F}^k$ and $V \in \mathbb{F}^{t\times k}$ that when we choose random $\mathbf{r}'_0 \leftarrow \mathbb{F}^{2\lambda}$ and $R' \leftarrow \mathbb{F}^{t\times 2\lambda}$ we get uniformly random $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_0;\mathbf{r}'_0)|_J$, and $\tilde{\mathsf{E}}_{\mathscr{C}}(V;R')$ is uniformly random subject to the second half of each vector being a Reed-Solomon codeword. Consequently, $E_{01}|_J,\ldots,E_\mu|_J$ have the correct distribution.

Next, the simulator picks $\mathbf{r}'_{(\gamma)} \in \mathbb{F}^{2\lambda}$ and $R'_{(Q)} \in \mathbb{F}^{t\times 2\lambda}$. To do this, for all j such that $j \in J$ or $j+n \in J$, the simulator computes the unique $(\mathbf{r}_{(\gamma)})_j \in \mathbb{F}$, $R_j \in \mathbb{F}^t$ and $R'_j \in \mathbb{F}^t$ such that we get $\tilde{\mathsf{E}}_{\mathscr{C}}(\mathbf{v}_{(\gamma)};\mathbf{r}_{(\gamma)}) = \mathbf{e}_0|_J + \gamma E|_J$ and $\tilde{\mathsf{E}}_{\mathscr{C}}(V_{(Q)};R_{(Q)}) = QE|_J$ and $\tilde{\mathsf{E}}_{\mathscr{C}}(V'_{(Q)};R'_{(Q)}) = Q'E|_J$. For all j such that $j \notin J$ and $j+n \notin J$, the values of $(\mathbf{r}'_{(\gamma)})_j \in \mathbb{F}$, $R'_j \in \mathbb{F}^t$ and $R'_j \in \mathbb{F}^t$ are now completely determined, by interpolation.

Finally, $\mathscr{S}$ defines $E_{01}|_{\bar{J}},\ldots,E_\mu|_{\bar{J}}$ to be 0 matrices. It then picks $\mathbf{s}_1,\ldots,\mathbf{s}_\mu$ at random and makes the commitments $\mathbf{c}_1,\ldots,\mathbf{c}_\mu$ as in the protocol. For $j \in J$ we see that all the $\mathbf{c}_i|_j$ commitments are computed as in the real execution from values that have the same distribution as in a real proof. Hence, they will have the correct distribution. The $\mathbf{c}_i|_j$s for $j \notin J$ are commitments to different values than in a real proof. However, by the computational (statistical) hiding property of the commitment scheme, they have a distribution that is computationally (statistically) indistinguishable from the correct distribution. ■

Appendix B

Stand-alone Argument for Arithmetic Circuit Satisfiability with Logarithmic Communication Complexity

We now give the formal description of a stand-alone argument of knowledge for the satisfiability of an arithmetic circuit, which combines the compiled version of the ILC protocol from Section 8.5, and the recursive protocol for inner products from Section 10.1. Both prover and verifier take the move parameter μ as common input. For square root communication complexity, the inner product argument is not used and we set $\mu = 0$. For $\mu > 0$, the common input includes the values $(m_\mu, \ldots, m_1)$ used in the inner product argument.

Common Input: $(ck, C, N, m, n, m'_1, m'_2, n', m_\mu, \ldots, m_1, \mu)$ where ck is a commitment key.The description of an arithmetic circuit

$$\left(\{\mathbf{w}_{q,a,i}, \mathbf{w}_{q,b,i}, \mathbf{w}_{q,c,i}\}_{q\in[Q],i\in[m]}, \{K_q\}_{q\in[Q]}\right)$$

with $N = mn$ multiplication gates. μ is the move parameter and $n = m_\mu \cdots m_1$. Parameters (m'_1, m'_2, n') are set to satisfy both $3m \leq m'_1 n'$ and $4m + 2 \leq m'_2 n'$.

Prover's Witness: Satisfying assignments $\mathbf{a}_i, \mathbf{b}_i$ and $\mathbf{c}_i$ to the wires of C.

Argument:

P: Pick randomness $\alpha_1, \beta_1, \gamma_1, \dots, \alpha_m, \beta_m, \gamma_m, \delta \leftarrow \mathbb{Z}_p$ and blinding vector $\mathbf{d} \leftarrow \mathbb{Z}_p^n$. Compute for $i \in \{1, \dots, m\}$

$$A_i = \mathsf{Commit}(\mathbf{a}_i; \alpha_i) \qquad B_i = \mathsf{Commit}(\mathbf{b}_i; \beta_i)$$

$$C_i = \mathsf{Commit}(\mathbf{c}_i; \gamma_i) \qquad D = \mathsf{Commit}(\mathbf{d}; \delta)$$

Send $A_1, B_1, C_1, \dots, A_m, B_m, C_m, D$ to the verifier.

V: $y \leftarrow \mathbb{Z}_p^*$.

As argued before, the circuit determines vectors of polynomials $\mathbf{w}_{a,i}(Y)$, $\mathbf{w}_{b,i}(Y)$, $\mathbf{w}_{c,i}(Y)$ and $K(Y)$ such that C is satisfiable if and only if

$$\sum_{i=1}^{m} \mathbf{a}_i \cdot (\mathbf{b}_i^T \circ \mathbf{Y}')Y^i + \sum_{i=1}^{m} \mathbf{a}_i \cdot \mathbf{w}_{a,i}(Y) + \sum_{i=1}^{m} \mathbf{b}_i \cdot \mathbf{w}_{b,i}(Y) + \sum_{i=1}^{m} \mathbf{c}_i \cdot \mathbf{w}_{c,i}(Y) = K(Y)$$

where $\mathbf{Y}' = (Y^m, \dots, Y^{mn})$. Given y, both the prover and verifier can compute $K = K(y)$, $\mathbf{w}_{a,i} = \mathbf{w}_{a,i}(y)$, $\mathbf{w}_{b,i} = \mathbf{w}_{b,i}(y)$ and $\mathbf{w}_{c,i} = \mathbf{w}_{c,i}(y)$.

P: Compute Laurent polynomials $\mathbf{r}, \mathbf{s}, \mathbf{r}'$, which have vector coefficients, and Laurent polynomial t, in the indeterminate X

$$\begin{aligned}
\mathbf{r}(X) &= \sum_{i=1}^{m} \mathbf{a}_i y^i X^i + \sum_{i=1}^{m} \mathbf{b}_i X^{-i} + X^m \sum_{i=1}^{m} \mathbf{c}_i X^i + \mathbf{d} X^{2m+1} \\
\mathbf{s}(X) &= \sum_{i=1}^{m} \mathbf{w}_{a,i} y^{-i} X^{-i} + \sum_{i=1}^{m} \mathbf{w}_{b,i} X^i + X^{-m} \sum_{i=1}^{m} \mathbf{w}_{c,i} X^{-i} \\
\mathbf{r}'(X) &= \mathbf{r}(X) \circ \mathbf{y}' + 2\mathbf{s}(X) \\
t(X) &= \mathbf{r}(X) \cdot \mathbf{r}'(X) - 2K = \sum_{k=-3m}^{4m+2} t_k X^k
\end{aligned}$$

When the wires $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$ correspond to a satisfying assignment, the Laurent polynomial $t(X)$ will have constant term $t_0 = 0$.

Commit to $t(X)$ by computing $T_k = \mathsf{Commit}(t_k; \tau_k)$ for each $k \neq 0$, and send all of these commitments to the verifier.

V: $x \leftarrow \mathbb{Z}_p^*$

P: Compute $v = t(x)$, $\tau = \sum_{k=-3m,k\neq 0}^{4m+2} \tau_k x^k$, and

$$\mathbf{r} = \sum_{i=1}^{m} \mathbf{a}_i x^i y^i + \sum_{i=1}^{m} \mathbf{b}_i x^{-i} + x^m \sum_{i=1}^{m} \mathbf{c}_i x^i + \mathbf{d} x^{2m+1}$$

$$\rho = \sum_{i=1}^{m} \alpha_i x^i y^i + \sum_{i=1}^{m} \beta_i x^{-i} + x^m \sum_{i=1}^{m} \gamma_i x^i + \delta x^{2m+1}$$

- **If $\boldsymbol{\mu = 0}$** : the inner product argument is not used. The prover sends $(v, \tau, \mathbf{r}, \rho)$ to the verifier.
- **If $\boldsymbol{\mu > 0}$** : the inner product argument is used. The prover computes $\mathbf{r}' = \mathbf{r}'(x)$ and sends (v, τ, ρ) to the verifier.

Verification: The verifier checks whether $\mathsf{Commit}(v;\tau) \stackrel{?}{=} \prod_{k=-3m,k\neq 0}^{4m+2} T_k^{x^k}$.

- **If $\boldsymbol{\mu = 0}$** : the inner product argument is not used. The verifier computes $\mathbf{r}' = \mathbf{r} \circ \mathbf{y}' + 2\mathbf{s}(x)$, and accepts only if

$$\begin{aligned} \mathbf{r} \cdot \mathbf{r}' - 2K &= v \\ \mathsf{Commit}(\mathbf{r};\rho) &= \left[\prod_{i=1}^m A_i^{x^i y^i}\right] \left[\prod_{i=1}^m B_i^{x^{-i}}\right] \left[\prod_{i=1}^m C_i^{x^{m+i}}\right] D^{x^{2m+1}} \end{aligned}$$

- **If $\boldsymbol{\mu > 0}$** : prover and verifier run the inner product argument with common input

$$(\mathbb{G}, p, \mathbf{g}, R, \mathbf{h}, R', v + 2K, m_\mu, m_{\mu-1}, \ldots, m_1) \qquad \text{where}$$

$$\begin{aligned} ck &= (\mathbb{G}, p, g, \mathbf{g}) & n &= m_\mu m_{\mu-1} \cdots m_1 \\ \mathbf{g} &= (g_1, g_2, \ldots, g_n) & \mathbf{h} &= (g_1^{y^{-m}}, g_2^{y^{-2m}}, \ldots, g_n^{y^{-mn}}) \\ R &= \mathsf{Commit}(0;-\rho) \left[\prod_{i=1}^m A_i^{x^i y^i}\right] & \left[\prod_{i=1}^m B_i^{x^{-i}}\right] & \left[\prod_{i=1}^m C_i^{x^{m+i}}\right] D^{x^{2m+1}} = \mathbf{g}^{\mathbf{r}} \\ R' &= R \cdot \mathbf{h}^{2\mathbf{s}(x)} = \mathbf{h}^{\mathbf{r}'} \end{aligned}$$

and the prover's witness is $\mathbf{r}$ and $\mathbf{r}'$.

The verifier accepts if the inner product argument is accepting.

B.0.1 Security Analysis.

Theorem 11 *The argument for satisfiability of an arithmetic circuit has perfect completeness, perfect special honest verifier zero-knowledge and statistical witness-extended emulation for extracting either a breach of the binding property of the commitment scheme or a witness for the satisfiability of the circuit.*

Proof Perfect completeness follows by inspection and using the fact that the inner product argument also have perfect completeness.

For perfect special honest verifier zero-knowledge we are given $y, x \in \mathbb{Z}_p^*$, which allows us to compute $\mathbf{w}_{a,i}, \mathbf{w}_{b,i}, \mathbf{w}_{c,i}$ and K from the circuit. The simulator picks $\mathbf{r} \leftarrow \mathbb{Z}_p^n$ and $\rho \leftarrow \mathbb{Z}_p$ and random commitments A_i, B_i and C_i. It computes

$$D = \left[\prod_{i=1}^{m} A_i^{x^i y^i} B_i^{x^{-i}} C_i^{x^{m+i}} \mathsf{Commit}(-\mathbf{r}; -\rho)\right]^{-x^{-2m-1}} \qquad v = \mathbf{r} \cdot \mathbf{r}' - 2K$$

The simulator picks random commitments $T_{-3m}, \ldots, T_{4m+2}$ and computes T_{4m+2} from the verification equation $\mathsf{Commit}(v; \tau) \stackrel{?}{=} \prod_{k=-3m, k \neq 0}^{4m+2} T_k^{x^k}$.

To see that the simulated components have the same distribution as a real argument observe that since the commitment scheme is perfectly hiding the commitments A_i, B_i and C_i have the same distribution as in a real argument. Also, in both the simulation and a real argument $\mathbf{r}$ and ρ are uniformly random. Given these values the commitment D is uniquely defined, and similarly for the T_k.

When $\mu > 0$ we simply remove $\mathbf{r}$ from the transcript and execute a fresh run of the inner product argument, given our knowledge of $\mathbf{r}$.

It remains to show that we have witness-extended emulation. We treat the cases $\mu = 0$ and $\mu > 0$ separately.

Square Root Argument. Assume that we have $N + Q$ different challenges $y \in \mathbb{Z}_p^*$ for the same initial message, and for each of these challenges a further $7m + 3$ different challenges $x \in \mathbb{Z}_p^*$ for the same third message, all with valid answers. We begin by showing that from this information we either extract a satisfying assignment to the wires $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$ in the circuit, or encounter a breach of the binding property of the commitment scheme.

Let us first consider a fixed initial transcript $(A_1, \dots, A_m, \dots, C_m, D, y, T_{-3m}, \dots, T_{4m+2})$ and suppose we have valid arguments with $3m+2$ different values of x. Then the vectors $(x^{-m}, \dots, x^{2m+1})$ form the rows of a shifted Vandermonde matrix and we can obtain any unit vector $(0, \dots, 1, \dots, 0)$ by taking an appropriate linear combination of these vectors. By taking the correct linear combinations of the $3m+2$ verification equations $D^{x^{2m+1}} \prod_{i=1}^{m} A_i^{x^i y^i} B_i^{x^{-i}} C_i^{x^{m+i}} = \mathsf{Commit}(\mathbf{r}; \rho)$, we can then extract openings to each A_i, B_i and C_i, since $y \in \mathbb{Z}_p^*$.

We have valid arguments for $7m+3$ different challenges $x \in \mathbb{Z}_p^*$. Similar arguments to the above show that we can extract a Laurent polynomial $t(X) = \sum_{k=-3m, k\neq 0}^{4m+2} t_k x^k$ that is consistent with respect to all $3m+2$ evaluations of $\mathbf{r}(X) \cdot \mathbf{r}'(X) - 2K$. This directly implies that Equation 8.2 holds for $Y = y$.

Finally, suppose that this holds for $N+Q$ different challenges $y \in \mathbb{Z}_p^*$. Then, we have equality of polynomials in Equation 8.2, since a non-zero polynomial of degree $N+Q-1$ cannot have $N+Q$ roots. This means that the circuit is satisfied.

Now we can apply the forking lemma to get witness-extended emulation, as the tree formed by the transcripts has size $(N+Q) \cdot (7m+3)$ which is polynomial in λ.

Inner Product Variant Assume that we have $(N+Q) \cdot (7m+3) \cdot (2m_\mu - 1) \dots (2m_2 - 1)$ accepting transcripts for the same statement arranged in a tree as follows:

- The root is labeled with the statement.
- Each of the $(N+Q)$ depth 1 nodes is labeled with a different challenge y and has $7m+3$ children labeled x.
- The children are subtrees of size $(2m_\mu - 1) \dots (2m_2 - 1)$
- Each level has nodes labeled with the challenges x_i used in the i-th move of the recursive argument, and of degree $2m_{\mu-i+1} - 1$.

Given the above tree of transcripts, we are able to do a two-stage extraction: First, we invoke the witness-extended emulation of the recursive inner product argument. At this point, we either have a a non-trivial discrete logarithm relation, in which case we are done, or we have an accepting $\mathbf{r}$ for each y,x pair. In this case, we proceed with the second stage and repeat the extraction procedure for $\mu = 0$ to obtain either a witness for the original statement or a breach of the binding property of the commitment scheme.

We now point out that the size of the tree will be $O(N \cdot (2m)^{\mu}) \approx O(N^{2+\log_m 2})$ which is polynomial in the security parameter λ and invoke the forking lemma to complete the proof. ■

B.0.2 Efficiency

Square Root Communication. When we set $\mu = 0$, the argument above has a communication cost of $10m+1$ commitments and $n+3$ field elements. Setting $m \approx \sqrt{\frac{N}{10}}$, $n \approx \sqrt{10N}$, we get a total communication complexity where the total number of group and field elements sent is as low as possible and approximately $3.2\sqrt{N}$ each. The main computational cost for the prover is computing the initial commitments, corresponding to $\frac{3mn}{\log n}$ group exponentiations. The prover can compute $t(X)$ using FFT-based techniques. Assuming that p is of a suitable form for the Fast Fourier transform, the dominant number of multiplications for this process is $\frac{3}{2}mn\log m$. The main cost in the verification is computing $\mathbf{s}(X)$ given the description of the circuit which requires in the worst case Qn multiplications in $\mathbb{Z}_p$, considering arbitrary fan-in addition gates. In case of $O(N)$-size circuits with fan-in 2 gates, computing $\mathbf{s}(X)$ requires $O(N)$ multiplications. Evaluating $\mathbf{s}(x)$ requires $3N$ multiplications. The last verification equation costs roughly $\frac{(n+3m)}{\log n+3m}$ group exponentiations to the verifier.

$(\mu+1)$-Root Communication. We can reduce communication by using $\mu = O(1)$ iterations of the inner product argument. Choosing $m = N^{\frac{1}{\mu+1}}$, $n = N^{\frac{\mu}{\mu+1}}$ and $m_i = (\frac{N}{m})^{\frac{1}{\mu}}$ will give us a communication complexity of $4\mu N^{\frac{1}{\mu+1}}$ group elements and $2\mu N^{\frac{1}{\mu+1}}$ field elements. The prover's complexity is dominated by $\frac{6\mu N}{\log N}$ group exponentiations and fewer than $\frac{3N}{2\mu}\log N$ field multiplications. The verifier's cost is dominated by $\frac{2\mu N}{\log N}$ group exponentiations and $O(N)$ field multiplications.

Logarithmic Communication. By increasing the number of iteration of the inner product argument we can further reduce the communication complexity.

To minimize the communication, we set $\mu = \log N - 1$, $n = \frac{N}{2}$, $m = m_i = 2$ in the above argument gives us $2\log N + 1$ moves. The total communication amounts to approximately $4\log N$ group elements and $2\log N$ field elements. The prover computational cost is dominated by $12N$ group exponentiations, and $O(N)$ multiplications in $\mathbb{Z}_p$.The main verification cost is bounded by $4N$ group exponentiations and $O(N)$ multiplications in $\mathbb{Z}_p$.

Appendix C

Exposition of an Efficient Multi-Exponentiation Algorithm

This chapter explains a special case of Pippenger's algorithm [124] for efficient multi-exponentiation, which explains why the cost of computing a Pedersen commitment of length N is $O(N/\log N)$ group operations rather than $O(N)$.

C.1 Goal

Let $\mathbb{G}$ be a group of prime order $p \approx 2^\lambda$. Let $g_0, \dots, g_{N-1}$ be elements of $\mathbb{G}$ and let $e_0, \dots, e_{N-1}$ be elements of $\mathbb{Z}_p$. Assume that $\lambda \geq N$.

Let $G = \prod_{i=0}^{N-1} g_i^{e_i}$.

Problem Given $g_0, \dots, g_{N-1}$, and $e_0, \dots, e_{N-1}$, compute G.

C.2 Reduction to Multi-Products

We call the case where $e_0, \dots, e_{N-1} \in \{0, 1\}$ a multi-product rather than a multi-exponentiation. The first step will be to reduce the computation of G to the computation of many multi-products.

Set $s \approx \sqrt{\frac{\lambda}{N}}$ and $t \approx \sqrt{\lambda N}$. Let $e_{i,l}$ be the binary digits of e_i.

$$\begin{aligned}
G &= \prod_{i=0}^{N-1} g_i^{e_i} \\
&= \prod_{i=0}^{N-1} \prod_{l=0}^{\lambda-1} g_i^{e_{i,l}2^l} \\
&= \prod_{i=0}^{N-1} \prod_{j=0}^{s-1} \prod_{k=0}^{t-1} g_i^{e_{i,j+sk}2^{j+sk}} \\
&= \prod_{k=0}^{t-1} \left(\prod_{i=0}^{N-1} \prod_{j=0}^{s-1} g_i^{e_{i,j+sk}2^k} \right)^{2^{sk}}
\end{aligned}$$

Set $g'_{i,j} = g_i^{2^j}$ for $0 \le j \le s-1$.

Set $e'_{i,j,k} = e_{i,j+sk}$.

Set $G'_k = \prod_{i=0}^{N-1} \prod_{j=0}^{s-1} g_i^{e_{i,sk+j}2^j}$ for $0 \le k \le t-1$.

Then, we have

$$\begin{aligned}
G &= \prod_{k=0}^{t-1} G'^{2^{sk}}_k \\
G'_k &= \prod_{i=0}^{N-1} \prod_{j=0}^{s-1} g'^{e_{i,j+sk}}_{i,j}
\end{aligned}$$

We will now consider the new multi-product problem:

New Problem Given $\{g'_{i,j}\}$, $\{e'_{i,j,k}\}$, compute $\{G'_k\}$.

The new problem has $Ns = t$ input group elements $g'_{i,j}$ and t output group elements G'_k.

C.2.1 Visualisation

This approach to computing G can be visualised by arranging the binary digits in a matrix.

$$\begin{array}{c|ccccc|c}
 & e_{0,0} & e_{0,s} & e_{0,2s} & \cdots & e_{0,(t-1)s} & g_0 \\
e_0 & e_{0,1} & e_{0,s+1} & e_{0,2s+1} & \cdots & e_{0,(t-1)s+1} & g_0^2 \\
 & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 & e_{0,s-1} & e_{0,2s-1} & e_{0,3s-1} & \cdots & e_{0,\lambda-1} & g_0^{2^{s-1}} \\
\hline
 & e_{1,0} & e_{1,s} & e_{1,2s} & \cdots & e_{1,(t-1)s} & g_1 \\
e_1 & e_{1,1} & e_{1,s+1} & e_{1,2s+1} & \cdots & e_{1,(t-1)s+1} & g_1^2 \\
 & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 & e_{1,s-1} & e_{1,2s-1} & e_{1,3s-1} & \cdots & e_{1,\lambda-1} & g_1^{2^{s-1}} \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
 & e_{N-1,0} & e_{N-1,s} & e_{N-1,2s} & \cdots & e_{N-1,(t-1)s} & g_{N-1} \\
e_{N-1} & e_{N-1,1} & e_{N-1,s+1} & e_{N-1,2s+1} & \cdots & e_{N-1,(t-1)s+1} & g_{N-1}^2 \\
 & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 & e_{N-1,s-1} & e_{N-1,2s-1} & e_{N-1,3s-1} & \cdots & e_{N-1,\lambda-1} & g_{N-1}^{2^{s-1}} \\
 & & & & & & \\
 & G'_0 & G'_1 & G'_2 & \cdots & G'_{t-1} &
\end{array}$$

The input values for the new problem are shown to the right of the matrix in the same row as the binary digits that they correspond to. The output values are shown below the matrix in the same column as the binary digits that they correspond to.

Computing the multi-exponentiation of the inputs with a column of the matrix gives the output below that column.

C.2.2 Efficiency

The simplest method of computing the new inputs $g'_{i,j}$ is using s squarings of g_i, for each $0 \leq i \leq N-1$, which gives a cost of $\sqrt{\lambda N}$ group operations.

C.3 Computing the Multi-Products

The new problem has the same number of inputs and outputs, so we relabel to simplify notation. Set $M = \sqrt{\lambda N} = sN = t$.

Problem Given $\{g'_i\}_{i=0}^{M-1}$, $\{e'_{i,j}\}_{i,j=0}^{M-1}$, compute $G'_j = \prod_{i=0}^{M-1} g'^{e'_{i,j}}_i$.

Let b be some parameter to be determined later. We partition the input group elements into sets $S_0, \ldots, S_{M/b-1}$, each consisting of at most b elements. Then, for each set S_i, we compute the set T_i, containing all possible multi-products of elements in S_i. For example, if $S_0 = \{g_0, g_1, g_2\}$, then $T_0 = \{g_0, g_1, g_2, g_0g_1, g_0g_2, g_1g_2, g_0g_1g_2\}$.

Now, we use the elements of the T_i to compute the G'_j. Note that in order to compute the G'_i, we only need to use one element from each T_i.

C.3.1 Visualisation

$$\begin{array}{cccc|ccc|c|ccc}
 & S_0 & & & & S_1 & & \cdots & & S_{M/b-1} & \\
g'_0 & g'_1 & \cdots & g'_{b-1} & g'_b & \cdots & g'_{2b-1} & \cdots & g'_{M-b-1} & \cdots & g'_{M-1}
\end{array}$$

$$\begin{array}{cccc|ccc|c|ccc}
 & T_0 & & & & T_1 & & \cdots & & T_{M/b-1} & \\
g'_0 & g'_1 & \cdots & \prod_{i=0}^{b-1} g'_i & g'_b & \cdots & \prod_{i=b}^{2b-1} g'_i & \cdots & g'_{M-b-1} & \cdots & \prod_{i=M-b-1}^{M-1} g'_i
\end{array}$$

$$G'_0 \qquad G'_1 \qquad \cdots \qquad G'_{M-1}$$

C.3.2 Efficiency

Given S_i, which contains b elements, we can compute all possible multi-products using 2^b group operations. There are M/b sets S_i, so computing all of the T_i costs at most $2^bM/b$ group operations.

Given all of the T_i, each G'_j uses at most one element from each, so it costs at most M/b group operations. There are M of the G'_j, so computing all of them costs at most M^2/b group operations.

C.4 Recombining Inputs

Given the outputs of the multi-product step, we can now compute the final output G. Recall that $G = \prod_{k=0}^{t-1} G'^{2^{sk}}_k$

This can be done using $st = \lambda$ squarings.

C.5 Efficiency Analysis

This approach can be used to compute $\prod_{i=0}^{N-1} g_i^{e_i}$ using $\lambda + M + 2^b \frac{M}{b} + \frac{M^2}{b}$, where $M = \sqrt{\lambda N}$.

Set $b = \log M - \log\log M$. This becomes

$$\lambda + M + \frac{M^2}{(\log M - \log\log M)(\log M)} + \frac{M^2}{\log M - \log\log M}$$

which is

$$\lambda + \frac{M^2}{\log M} + o\left(\frac{M^2}{\log M}\right)$$

Since $M = \sqrt{\lambda N}$, we arrive at a cost of

$$\lambda + \frac{2\lambda N}{\log \lambda N} + o\left(\frac{\lambda N}{\log \lambda N}\right)$$

C.5.1 Generalisation

In general, one can obtain a multiexponentiation algorithm which uses

$$\lambda + \frac{2\lambda MN}{\log \lambda MN} + o\left(\frac{\lambda MN}{\log \lambda MN}\right)$$

group operations to compute M multiexponentiations from N group elements.

Appendix D

Computing Sums from [2] and [3] for Low-Depth Circuit Protocols

We give an explicit method for computing the sum in 8.3.4.1, used by the verifier, for a zero-knowledge proof that a committed value lies in a public list. The task is to compute $\sum_{i=0}^{N-1} \lambda_i \prod_{j=0}^{m-1} f_{j,i_j}$, with $N = n^m$.

First, rewrite the index i in terms of its n-ary digits $i_0, \ldots, i_{m-1}$, where $i = \sum_{j=0}^{m-1} i_j n^j$.

$$\sum_{i_0,\ldots,i_{m-1}=0}^{n-1} \lambda_{i_0,\ldots,i_{m-1}} \prod_{j=0}^{m-1} f_{j,i_j} = \sum_{i_{m-1}=0}^{n-1} f_{m-1,i_{m-1}} \left[\sum_{i_0,\ldots,i_{m-2}} \lambda_{i_0,\ldots,i_{m-1}} \prod_{j=0}^{m-2} f_{ji_j} \right]$$

Observe, that in brackets on the right hand side, we have n copies of the original sum, but with $m-1$ digits i_j instead of m. This suggests a recursive method for computing the sum. The base case is

$$\sum_{i_0=0}^{n-1} \lambda_{i_0} f_{0i_0}$$

which costs n multiplications to compute. Denoting by $M(m)$ the number of multiplications required with m digits, we have $M(1) = n$ and $M(m) \leq n + n\,M(m-1)$. From this, it is easy to see that $M(m) \leq \frac{n}{n-1} N \leq 2N$.

Now, to compute $\sum_{i=0}^{N-1} \lambda_i \prod_{j=0}^{m-1} f_{j,i_j}(X)$, with $N = n^m$ used by the prover in the same protocol, note that since each $f_{ji_j}(X)$ is a linear polynomial, the result has degree m. This means that if we evaluate each $f_{ji_j}(X)$ at $m+1$ points, and compute the sum for each evaluation point, we can interpolate to get the resulting polynomial. When we choose the points to be roots of unity, the dominant cost for computing this polynomial is $O(mN)$ multiplications using fast Fourier transform techniques.

The techniques used to compute expressions like the sums appearing in 8.3.4.2, in arguments for polynomial evaluation using our low-depth circuit argument, are very similar.

Bibliography

[1] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334, 2018.

[2] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 253–280, 2015.

[3] Stephanie Bayer and Jens Groth. Zero-knowledge argument for polynomial evaluation with application to blacklists. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 646–663, 2013.

[4] 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.

[5] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[6] Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 424–441, 1998.

[7] Jens Groth. Linear algebra with sub-linear zero-knowledge arguments. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 192–208, 2009.

[8] Jae Hong Seo. Round-efficient sub-linear zero-knowledge arguments for linear algebra. In *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, pages 387–402, 2011.

[9] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 327–357, 2016.

[10] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*, pages 243–265, 2015.

[11] Jonathan Bootle and Jens Groth. Efficient batch zero-knowledge arguments for low degree polynomials. In *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key*

Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II, pages 561–588, 2018.

[12] Stephanie Bayer. *Practical zero-knowledge protocols based on the discrete logarithm assumption*. PhD thesis, University College London, UK, 2014.

[13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 238–252, 2013.

[14] Riad S. Wahby, Srinath T. V. Setty, Zuocheng Ren, Andrew J. Blumberg, and Michael Walfish. Efficient RAM and control flow in verifiable outsourced computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.

[15] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 2087–2104, 2017.

[16] 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.

[17] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptology ePrint Archive*, 2018:46, 2018.

[18] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 263–280, 2012.

[19] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, pages 117–136, 2016.

[20] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, pages 136–153, 2005.

[21] Aggelos Kiayias and Moti Yung. Secure scalable group signature with dynamic joins and separable authorities. *IJSN*, 1(1/2):24–45, 2006.

[22] Jonathan Bootle, Mehdi Tibouchi, and Keita Xagawa. Cryptanalysis of Compact-LWE. In *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, pages 80–97, 2018.

[23] Dongxi Liu, Nan Li, Jongkil Kim, and Surya Nepal. Compact-lwe: Enabling practically lightweight public key encryption for leveled iot device authentication. *IACR Cryptology ePrint Archive*, 2017:685, 2017.

[24] Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi. LWE without modular reduction and improved side-channel attacks against BLISS. 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 494–524, 2018.

[25] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 40–56, 2013.

[26] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1857–1874, 2017.

[27] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, and Jens Groth. Efficient zero-knowledge proof systems. In *Foundations of Security Analysis and Design VIII - FOSAD 2014/2015/2016 Tutorial Lectures*, pages 1–31, 2016.

[28] Jens Groth. Short non-interactive zero-knowledge proofs. 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 341–358, 2010.

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

[30] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pages 325–341, 2005.

[31] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. 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 321–340, 2010.

[32] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 169–189, 2012.

[33] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 326–349, 2012.

[34] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[35] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 111–120, 2013.

[36] 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.

[37] 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.

[38] Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execu-

tion. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 595–626, Cham, 2018. Springer International Publishing.

[39] 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.

[40] Ronald Cramer, Ivan Damgård, and Valerio Pastro. On the amortized complexity of zero knowledge protocols for multiplicative relations. In *Information Theoretic Security - 6th International Conference, ICITS 2012, Montreal, QC, Canada, August 15-17, 2012. Proceedings*, pages 62–79, 2012.

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

[42] 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.

[43] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 174–187, 1986.

[44] Russell Impagliazzo and Moti Yung. Direct minimum-knowledge computations. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, pages 40–51, 1987.

[45] Adi Shamir. Ip=pspace. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 11–15, 1990.

[46] Uriel Feige, Amos Fiat, and Adi Shamir. Zero knowledge proofs of identity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 210–217, 1987.

[47] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 103–112, 1988.

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

[49] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[50] Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam Smith. Using fully homomorphic hybrid encryption to minimize non-interactive zero-knowledge proofs. *Journal of Cryptology*, pages 1–24, 2014.

[51] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Information Processing Letters*, 67(4):205–214, 1998.

[52] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1-2):1–53, 2002.

[53] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 723–732, 1992.

[54] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short pcps. In *22nd Annual IEEE Conference on Computational*

Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA, pages 278–291, 2007.

[55] Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. The (true) complexity of statistical zero knowledge. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 494–502, 1990.

[56] Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Interactive hashing simplifies zero-knowledge protocol design. In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, pages 267–273, 1993.

[57] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. *J. Comput. Syst. Sci.*, 60(1):47–108, 2000.

[58] Ivan Damgård. Interactive hashing can simplify zero-knowledge protocol design without computational assumptions (extended abstract). In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 100–109, 1993.

[59] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 399–408, 1998.

[60] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 418–430, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[61] Jens Groth. Honest verifier zero-knowledge arguments applied. Technical Report DS-04-3, 2004. PhD thesis. xii+119 pp.

[62] Daniele Micciancio and Erez Petrank. Simulatable commitments and efficient concurrent zero-knowledge. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 140–159, 2003.

[63] 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.

[64] 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.

[65] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[66] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 102–113, 2003.

[67] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, pages 123–128, 1988.

[68] Jens Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. 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 431–448, 2011.

[69] 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.

[70] 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.

[71] Sean Bowe and Ariel Gabizon. Making groth's zk-snark simulation extractable in the random oracle model. *IACR Cryptology ePrint Archive*, 2018:187, 2018.

[72] 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.

[73] 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.

[74] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 287–304, 2015.

[75] Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-snark. *IACR Cryptology ePrint Archive*, 2017:602, 2017.

[76] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 21–30, 2007.

[77] 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.

[78] 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.

[79] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 955–966, 2013.

[80] Melissa Chase, Chaya Ganesh, and Payman Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 499–530, 2016.

[81] Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. In *Advances in Cryptology*

- CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III, pages 643–673, 2018.

[82] 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.

[83] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. How to prove a secret: Zero-knowledge proofs on distributed data via fully linear pcps. *IACR Cryptology ePrint Archive*, 2019:188, 2019.

[84] Emmanuel Bresson and Jacques Stern. Efficient revocation in group signatures. In *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, pages 190–206, 2001.

[85] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings*, pages 400–415, 2007.

[86] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 16–30, 1997.

[87] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, pages 431–444, 2000.

[88] Helger Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In *Advances in Cryptology - ASIACRYPT 2003, 9th International*

Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings, pages 398–415, 2003.

[89] Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 80–91, 2003.

[90] Javier Herranz. Attribute-based versions of schnorr and elgamal. *Appl. Algebra Eng. Commun. Comput.*, 27(1):17–57, 2016.

[91] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, pages 234–252, 2008.

[92] Prastudy Fauzi, Helger Lipmaa, and Bingsheng Zhang. Efficient non-interactive zero knowledge arguments for set operations. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, pages 216–233, 2014.

[93] Geoffroy Couteau, Thomas Peters, and David Pointcheval. Removing the strong RSA assumption from arguments over the integers. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, pages 321–350, 2017.

[94] Jens Groth. Cryptography in subgroups of $\mathrm{z_n}$. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pages 50–65, 2005.

[95] Rafik Chaabouni, Helger Lipmaa, and Abhi Shelat. Additive combinatorics and discrete logarithm based range protocols. In *Information Security and*

Privacy - 15th Australasian Conference, ACISP 2010, Sydney, Australia, July 5-7, 2010. Proceedings, pages 336–351, 2010.

[96] Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. 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 1–20, 2013.

[97] Charanjit S. Jutla and Arnab Roy. Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 295–312, 2014.

[98] Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 101–128, 2015.

[99] Kun Peng. A general, flexible and efficient proof of inclusion and exclusion. In *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, pages 33–48, 2011.

[100] Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, pages 274–285, 1993.

[101] Lan Nguyen. Accumulators from bilinear pairings and applications. In *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Con-*

ference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings, pages 275–292, 2005.

[102] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, pages 481–500, 2009.

[103] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 61–76, 2002.

[104] Ivan Damgård and Nikos Triandopoulos. Supporting non-membership proofs with bilinear-map accumulators. *IACR Cryptology ePrint Archive*, 2008:538, 2008.

[105] Jiangtao Li, Ninghui Li, and Rui Xue. Universal accumulators with efficient nonmembership proofs. In *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*, pages 253–269, 2007.

[106] 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, 2010.

[107] 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.

[108] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vsql: Verifying arbitrary SQL queries over dynamic outsourced databases. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 863–880, 2017.

[109] Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 30:1–30:14, 2016.

[110] Rosario Gennaro, Darren Leigh, Ravi Sundaram, and William S. Yerazunis. Batching schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In *Advances in Cryptology - ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings*, pages 276–292, 2004.

[111] Mihir Bellare, Juan A. Garay, and Tal Rabin. Batch verification with applications to cryptography and checking. In *LATIN '98: Theoretical Informatics, Third Latin American Symposium, Campinas, Brazil, April, 20-24, 1998, Proceedings*, pages 170–191, 1998.

[112] Kun Peng and Feng Bao. Batch ZK proof and verification of OR logic. In *Information Security and Cryptology, 4th International Conference, Inscrypt 2008, Beijing, China, December 14-17, 2008, Revised Selected Papers*, pages 141–156, 2008.

[113] Ryan Henry and Ian Goldberg. Batch proofs of partial knowledge. In *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, pages 502–517, 2013.

[114] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical Report 260, ETH Zurich, 1997.

[115] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 379–396, 2008.

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

[117] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 129–140, 1991.

[118] Oliver Schirokauer. Using number fields to compute logarithms in finite fields. *Math. Comput.*, 69(231):1267–1283, 2000.

[119] Dominique Unruh. Post-quantum security of fiat-shamir. 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 I*, pages 65–95, 2017.

[120] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pages 201–215, 1996.

[121] Stefan Lucks. Design principles for iterated hash functions. *IACR Cryptology ePrint Archive*, 2004:253, 2004.

[122] Mridul Nandi and Souradyuti Paul. Speeding up the wide-pipe: Secure and fast hashing. In *Progress in Cryptology - INDOCRYPT 2010 - 11th International*

Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings, pages 144–162, 2010.

[123] Moni Naor and Benny Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35(5):1254–1281, 2006.

[124] Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM J. Comput.*, 9(2):230–250, 1980.