TU WIEN Informatics

How to Simulate PLONK: A Formal Security Analysis of a zk-SNARK

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Marek Sefranek, BSc
Matrikelnummer 11779683

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dr. Georg Fuchsbauer

Wien, 20. September 2023

Marek Sefranek	Georg Fuchsbauer

TU WIEN Informatics

How to Simulate PLONK: A Formal Security Analysis of a zk-SNARK

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Marek Sefranek, BSc
Registration Number 11779683

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dipl.-Ing. Dr. Georg Fuchsbauer

Vienna, September 20, 2023

Marek Sefranek	Georg Fuchsbauer

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Marek Sefranek, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 20. September 2023

Marek Sefranek

Acknowledgements

First, I want to thank my supervisor, Georg Fuchsbauer, for his patience and all the invaluable feedback he provided me with throughout this project. I would not have been able to finish this thesis without your support.

I am also very grateful for the financial support I received from the Vienna Science and Technology Fund (WWTF) [10.47379/VRG18002] during the work on this thesis.

Furthermore, I would like to thank Ariel Gabizon for answering questions about PLONK, as well as for discussing the vulnerability in its zero knowledge implementation and how to fix it.

Lastly, I am deeply grateful to my family for all their support and everlasting love, believing in me even during the most challenging moments of this journey.

Kurzfassung

Zero-Knowledge-Beweise ermöglichen es, etwas zu beweisen, ohne dabei Informationen über die Wahrheit der Aussage hinaus preiszugeben. Dieses paradoxe Konzept, welches ursprünglich rein theoretischer Natur war, hat in den letzten Jahrzehnten eine breite Anwendung in der Praxis gefunden. An der Spitze dieser Entwicklung stehen Beweissysteme, die zk-SNARKs genannt werden, was für *Zero-Knowledge Succinct Non-Interactive Argument of Knowledge* steht. Sie vermeiden nicht nur, dass mehrere Runden der Interaktion erforderlich sind, sondern haben auch Beweise, die deutlich kürzer als die bewiesene Aussage selbst sind, wobei einige Konstruktionen sogar Beweise mit konstanter Größe erreichen.

Einer der aktuellsten zk-SNARKs ist "PLONK" von Gabizon, Williamson und Ciobotaru aus dem Jahr 2019. Seine Beweise haben mit nur einem halben Kilobyte konstante Größe und können in sublinearer Zeit verifiziert werden. Darüber hinaus müssen die erforderlichen öffentlichen Parameter nur einmalig aufgesetzt werden, um beliebige Aussagen bis zu einer bestimmten Länge beweisen zu können, was PLONK zu einem universellen und zeiteffizienten zk-SNARK macht. Obwohl PLONK sehr einflussreich ist und in mehreren realen Anwendungen eingesetzt wird, gibt es keinen formalen Sicherheitsbeweis seiner Zero-Knowledge-Eigenschaft.

Im Rahmen dieser Arbeit zeigen wir auf, wie eine von uns gefundene Sicherheitslücke in der Zero-Knowledge-Implementierung von PLONK behoben werden kann. Das PLONK-Protokoll wurde bereits entsprechend ausgebessert. Unser Hauptbeitrag ist ein formaler Sicherheitsbeweis dafür, dass die resultierende Version von PLONK *statistisches Zero-Knowledge* erfüllt. Hierfür zeigen wir, wie Beweise bis auf einen exponentiell kleinen Unterschied simuliert werden können, ohne dabei Zugriff auf die geheimen Informationen des Beweisers zu haben. Gemäß der Standarddefinition von Zero-Knowledge folgt daraus, dass PLONK-Beweise (statistisch) keine Informationen über die Wahrheit der Aussage hinaus preisgeben.

Zudem führen wir eine genaue Sicherheitsanalyse des gesamten PLONK-Protokolls durch, einschließlich des Nachweises der Sicherheit aller seiner Komponenten. Dabei beweisen wir eine präzise obere Schranke für den Knowledge-Soundness-Fehler von PLONK im *algebraischen Gruppenmodell*. Da der ursprüngliche Beweis der Knowledge-Soundness von PLONK ebenfalls auf diesem idealisierten Modell beruht, tragen unsere Ergebnisse zu einem allgemein besseren Verständnis der Sicherheitseigenschaften von PLONK bei.

Abstract

Zero-knowledge proofs enable proving a statement without revealing any information beyond its truth. This paradoxical notion has evolved over the last few decades from a theoretical concept to the wide adoption of highly efficient zero-knowledge proof systems in practice. At the forefront of this development are proof systems called zk-SNARKs, which stands for *zero-knowledge succinct non-interactive argument of knowledge*. Not only do they avoid multiple rounds of interaction, but zk-SNARKs also offer succinct proofs whose length is much shorter than the size of the proved statement, with some constructions even achieving constant-size proofs.

Among the most recent state-of-the-art constructions is the zk-SNARK "PLONK" by Gabizon, Williamson, and Ciobotaru from 2019. It has constant-size proofs of only half a kilobyte and sublinear proof verification time. Furthermore, it only requires a single trusted setup of its public parameters to support proofs of any statement up to a certain size bound, making PLONK a universal and fully succinct zk-SNARK. Although highly influential and implemented in several real-world applications, there is no formal security proof of its zero knowledge property.

In this thesis, we disclose a vulnerability found in PLONK's implementation of zero knowledge and propose how to fix it. As a result, the PLONK protocol has been patched accordingly. Our primary contribution is a formal security proof establishing that the resulting version of PLONK achieves *statistical zero knowledge*. Towards this goal, we show how to simulate proofs up to an exponentially small difference without relying on any secret information used by the prover. Following the standard definition of zero knowledge, this implies that PLONK proofs reveal (statistically) zero information beyond the truth of the statement.

Moreover, we conduct a rigorous security analysis of the entire PLONK protocol, proving the security of all its underlying components. This allows us to show a precise upper bound on PLONK's knowledge soundness error in the *algebraic group model*. Since the original proof given by the authors of PLONK relies on the same idealized model, our results help towards a better understanding of the security guarantees of PLONK in general.

Contents

CHAPTER 1

Introduction

1.1 Motivation

First introduced by Goldwasser, Micali, and Rackoff [GMR85] as a theoretical concept in the 1980s, *zero-knowledge proofs* have become efficient enough to be used in practice over the subsequent decades. Informally speaking, a zero-knowledge proof allows a party (the "prover") to convince another party (the "verifier") that a statement is true, without revealing any information beyond the validity of the statement. More precisely, given a statement represented as a bit string $x \in \{0,1\}^*$ and a formal language $\mathcal{L} \subseteq \{0,1\}^*$, it enables the prover to convince the verifier that $x \in \mathcal{L}$ without leaking any other information. For example, take an NP relation $\mathcal{R}_\mathcal{L}$, where $(x, w) \in \mathcal{R}_\mathcal{L}$ if and only if w is a witness for $x \in \mathcal{L}$. In this case, a prover that knows such a witness is able to perform a zero-knowledge proof of $x \in \mathcal{L}$; however, the verifier learns no information about this witness in the process.

Formally, a zero-knowledge proof is an interactive protocol, where both the prover and the verifier are randomized algorithms. Furthermore, it has to satisfy three central properties:

1. **Completeness:** The protocol works as expected if both parties are honest, i.e., given a statement $x \in \mathcal{L}$ and a witness w, the prover can convince the verifier.

2. **Soundness:** A malicious prover cannot convince the verifier of a false statement $x \notin \mathcal{L}$, except with very small probability.

3. **Zero knowledge:** The proof does not reveal anything beyond the truth of the statement; in particular, it does not reveal the prover's witness w.

But what does it even mean to *"not reveal anything beyond the truth of the statement"*? Intuitively, this is achieved if whatever the verifier gets to see during its interaction

with the prover could have been generated by the verifier alone in the first place. This notion is formally captured through the existence of a *simulator*, which is an efficient algorithm capable of simulating the interaction between prover and verifier only given the same inputs as the verifier. Crucially, it does not rely on any secret information used by the prover, including its witness. This shows that the *view* of the verifier (i.e., its inputs, randomness, and the messages it receives from the prover) can be simulated, which implies that its interaction with the prover reveals zero information beyond the truth of the statement.

zk-SNARKs. One of the currently most promising approaches to make zero-knowledge proofs as practical as possible are proof systems called *zk-SNARKs* [GGPR13]. This acronym stands for ***z**ero-**k**nowledge **s**uccinct **n**on-interactive **ar**gument of **k**nowledge*. An *argument* is a proof system between two efficient parties, i.e., both running in probabilistic polynomial time.[1] Furthermore, the prover is required to know a witness whenever producing a valid proof (hence argument *of knowledge*). *Non-interactive* means that only a single message, "the proof", is sent during the protocol, as opposed to a multi-round exchange of messages. The term *succinct* refers to the proof size, in particular, a zk-SNARK proof is much shorter than the size of the proved statement, in some cases even of constant size.

In an impossibility result, Goldreich and Oren [GO94] showed that a non-interactive proof for a non-trivial language (i.e., one that is not decidable in probabilistic polynomial time) cannot be both sound and zero-knowledge simultaneously. To get around this impossibility, most zk-SNARKs rely on a trusted setup which generates a so-called *common reference string* (CRS), a special public input given to both the prover and the verifier before they start their interaction. Intuitively, a common reference string can be thought of as a long public key which enables zk-SNARKs to have non-interactive proofs as long as neither the prover nor the verifier knows the corresponding secret, called the *trapdoor*. Since the only exchanged message is now the proof, which should not be simulatable without a witness, the definition of zero knowledge for non-interactive proofs changes. Instead, the simulator is allowed to set up the CRS and thus know the trapdoor, which it uses to simulate proofs.

The applications of zk-SNARKs are vast. In principle, they can be used to prove the validity of any statement in the complexity class NP in zero knowledge. This captures any feasible computation, as any polynomial-time algorithm f can be modeled as an NP relation $\mathcal{R}_f$. Suppose $f\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^\ell$ takes a public input x, a private input w, and outputs y, then the corresponding relation $\mathcal{R}_f$ can be defined as $\mathcal{R}_f((x,y),w) := 1$ if and only if $f(x,w) = y$. With this, a zk-SNARK can be used to verify the computation of any polynomial-time computable function without revealing possible private inputs. Specifically, given the (public) description of the function f, a

[1] The difference between a proof and an argument is essentially that in the former the prover is unbounded, whereas in the latter the prover is efficient, i.e, runs in PPT. Throughout this thesis, we are using the term *"proof"* to refer to both concepts interchangeably.

public input x and output y, a zk-SNARK allows proving knowledge of a private input w such that $f((x, y), w) = 1$ without revealing anything about w. For example, given a collision-resistant hash function $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^\ell$, we could be proving knowledge of a preimage w of a certain hash value y such that $y = \mathcal{H}(w)$ without revealing it. In addition, what makes a zk-SNARK proof so powerful is that it can be verified much faster than the time it would take to compute $f(x, w)$ in the first place, even when given access to the private input w.

In summary, the two main benefits obtained by making use of zk-SNARKs are the preservation of *privacy* and better *scalability* of distributed computation. The first goal is achieved by enabling parties to only have to reveal the minimum amount of sensitive information necessary. For example, consider taking out additional health insurance whose price depends on the health status, such as the blood values of the insurance holder. Instead of disclosing one's complete health status to the insurance company, one could prove to them in zero knowledge that the relevant values fall within the required range for the specified risk category. On the other hand, zk-SNARKs can be used to improve the scalability of distributed computation by enabling verifiable computation, i.e., by appending a succinct proof to the result of a computation that proves its correctness without revealing any potentially involved private parameters.

Currently, the application of zk-SNARKs predominantly takes place in blockchain technologies such as the cryptocurrencies Bitcoin and Ethereum. The most prominent of these projects is the anonymous cryptocurrency Zcash [HBHW22]. In contrast to Bitcoin, where the entire transaction and ledger history is transparent, Zcash uses zk-SNARKs to enable transactions which can be verified without revealing the sender, receiver or transaction amount, and thus provide full financial privacy.

1.2 Problem Statement

In this thesis, we conduct a security analysis of the widely used and influential zk-SNARK named *PLONK* by Gabizon et al. [GWC19]. Not only does it have constant-size proofs and sublinear proof verification time, but it or derivatives of it are also implemented in a variety of real-world projects including Zcash[2], Polygon Zero[3], Aztec Network[4], Dusk[5], MatterLabs[6], Astar[7], Mina[8], Anoma[9], and Espresso Systems[10]. The most prominent of these projects is the anonymous cryptocurrency Zcash [HBHW22], which currently uses Halo 2 as its zk-SNARK [Ele22], built on top of the PLONK arithmetization.

[2] https://z.cash/
[3] https://polygon.technology/
[4] https://aztec.network/
[5] https://dusk.network/
[6] https://matter-labs.io/
[7] https://astar.network/
[8] https://minaprotocol.com/
[9] https://anoma.net/
[10] https://www.espressosys.com/

Unfortunately, there has been no formal proof of PLONK's zero knowledge property. In the original paper [GWC19, Sec. 8], this is simply claimed without giving any proof or even defining what notion of zero knowledge the protocol is supposed to satisfy. As a result, when trying to construct our own simulator, we discovered a vulnerability in PLONK's zero knowledge implementation. Subsequently, the PLONK protocol was patched using a fix we proposed in collaboration with Ariel Gabizon, one of the authors of PLONK. On June 30, 2022, Ariel Gabizon disclosed this security issue via Twitter.[11]

The main goal of this thesis is to formally prove that the patched version of PLONK achieves zero knowledge. For this purpose, we will construct an appropriate simulator and show that its outputs are distributed correctly, i.e., that they match the distribution of honestly generated PLONK proofs. Beyond this main objective, we want to do a rigorous analysis of the entire protocol, as we think that some aspects of the security proofs given in the original PLONK paper are a bit vague, and consequently not fully convincing. Moreover, we believe that a formal security analysis of the complete protocol, including a rigorous proof of its knowledge soundness property, will make a contribution towards further strengthening the validity of the security properties of PLONK.

1.3 State of the Art

The currently most efficient zk-SNARKs are constructed using an elliptic-curve group over a prime field $\mathbb{F}_p$, i.e., their proofs are composed of elliptic-curve group elements and field elements. To prove arbitrary statements in NP, they express them in the NP-complete language of *arithmetic circuits*. We explain this generalization of Boolean circuits in more detail in Section 2.5. Furthermore, the proof size of these zk-SNARKs is constant and independent of the size of the proved statement, which is one of the major advances in the last decade pushing zk-SNARKs from a theoretical concept to being implemented and used in practice.

Currently, the zk-SNARK with the shortest proof size is the construction by Groth [Gro16]. Its proofs consist of just 3 group elements ($\approx$ 128 bytes), independent of the size of the associated arithmetic circuit. Nevertheless, it comes with the major drawback of requiring a special common reference string, which has to be computed for each different circuit one wants to prove statements about separately. Since the common reference string either has to be set up by a trusted party or in a distributed manner via a multiparty computation (MPC) protocol, which requires considerable effort and special care, this restriction limits the usability of this zk-SNARK in practice. To solve this issue, the most recent development is to construct zk-SNARKs with a *universal structured reference string* (SRS), i.e., once generated, it can be used to prove statements about any arithmetic circuit up to a certain size bound.

The first promising construction with a universal SRS is Sonic by Maller et al. [MBKM19], featuring a proof size of 20 group elements and 16 field elements ($\approx$ 1152 bytes). Improving

[11] https://twitter.com/rel_Aztec/status/1542474186664210432

Table 1.1: Comparison of succinct non-interactive zero-knowledge proof systems.

Proof system	**Proof size**	**Setup**
Groth16 [Gro16]	≈ 128 bytes	Trusted, circuit-specific CRS
Sonic [MBKM19]	≈ 1152 bytes	Trusted, universal SRS
PLONK [GWC19]	≈ 480 bytes	
Marlin [CHM+20]	≈ 672 bytes	
Bulletproofs [BBB+18]	Logarithmic	No trusted setup
zk-STARKs [BSBHR18]	Polylogarithmic	No trusted setup, post-quantum secure

upon the efficiency of Sonic, the PLONK construction by Gabizon et al. [GWC19], which is the subject of study of this thesis, offers even shorter proofs consisting of only 9 group elements and 6 field elements (≈ 480 bytes). Another prominent universal zk-SNARK is Marlin by Chiesa et al. [CHM+20] with a proof size of 13 group elements and 8 field elements (≈ 672 bytes). However, compared to PLONK, it has not found such a wide adoption in practice.

Beyond that, other promising directions in constructing succinct non-interactive zero-knowledge proofs include Bulletproofs [BBB+18] and zk-STARKs [BSBHR18], short for ***z**ero-**k**nowledge **s**calable **t**ransparent **ar**guments of **k**nowledge*. At the cost of having longer proofs—logarithmic (Bulletproofs) and polylogarithmic (zk-STARKs) in the size of the proved statement—these constructions come with the advantage of not requiring any trusted setup, i.e., they do not use a common reference string.[12] On top of that, zk-STARKs are currently considered post-quantum secure as opposed to proof systems which rely on the hardness of the discrete logarithm problem in elliptic-curve groups.

To summarize the various approaches for constructing succinct non-interactive zero-knowledge proofs discussed in this section, we provide a comparison in Table 1.1.

1.4 Contributions

As the main contribution of this thesis, we give a formal security proof establishing that the current version of PLONK, which includes our suggested fix, achieves *statistical zero knowledge*.[13] Towards this goal, we construct a simulator for the interactive version of PLONK in Section 7.2. By showing that it perfectly simulates the outputs of the prover except for a negligible fraction of possible cases, we prove that the interactive PLONK protocol achieves statistical honest-verifier zero knowledge. Then, in Section 7.3, we extend this to the PLONK zk-SNARK, obtaining the following main result:

Theorem 1.1. *The PLONK zk-SNARK is statistically zero-knowledge.*

[12] More precisely, both rely on a collision-resistant hash function to avoid a trusted setup, with Bulletproofs modeling it as a random oracle [BR93] and zk-STARKs using it directly.

[13] For a precise definition of this notion, see Definition 2.11 and the explanation that follows it.

Furthermore, we conduct a rigorous security analysis of the entire PLONK protocol, giving formal security proofs for all its underlying components. As a result of this modular security analysis, we obtain a precise upper bound on PLONK's knowledge soundness error in the *algebraic group model* of Fuchsbauer et al. [FKL18] in Section 7.3. Since the original proof of PLONK's knowledge soundness by Gabizon et al. [GWC19, Sec. 7] uses the same idealized model, our result helps towards better quantifying the security guarantees provided by PLONK.

1.5 Outline

This thesis is organized as follows. We begin by presenting the necessary background material as well as formally defining all the concepts used throughout this thesis in Chapter 2. Then, in Chapter 3, we explain the constraint system used by PLONK for its internal representation of arithmetic circuits. In Chapter 4, we explore how *polynomial commitments* are used to build a protocol for succinctly proving the evaluations of multiple polynomials. We also give a formal security proof of this first cryptographic building block used in the construction of the PLONK protocol. Next, we describe and provide security proofs for a series of protocols that build on one another to achieve succinct proofs of statements about polynomials in Chapter 5. This culminates in the *permutation argument* in Chapter 6, which is at the heart of the PLONK construction. In Chapter 7, we finally give a detailed description of the complete PLONK protocol. Importantly, we describe the vulnerability found in its zero knowledge implementation and give a formal proof that the fixed version of PLONK satisfies the notion of *statistical zero knowledge* by constructing an appropriate simulator. We conclude with a summary of our contributions and by highlighting remaining open problems in Chapter 8.

CHAPTER 2

Preliminaries

We start by defining the notation used throughout this thesis, as well as giving all the necessary background material the PLONK protocol is built upon.

2.1 Notation

We use $\lambda \in \mathbb{N}_{>0}$ to denote the security parameter in bits. PPT stands for *probabilistic polynomial time*, e.g., PPT algorithm. We use $[n]$ as a shorthand for the set $\{1, \ldots, n\} \subset \mathbb{N}$, and $(m, n]$ for the set $\{m+1, \ldots, n\} \subset \mathbb{N}$. Let $\mathbb{F}_p$ be a finite field of prime order p. We use $\mathbb{F}_p[X_1, \ldots, X_n]$ to denote the set of n-variate polynomials over $\mathbb{F}_p$, and $\mathbb{F}_p^{(\leq d)}[X_1, \ldots, X_n]$ to further restrict this to polynomials of total degree[1] $\leq d$. We write ω to denote a *primitive n-th root of unity* in $\mathbb{F}_p$, i.e., a generator of an order-n subgroup $\langle\omega\rangle := \{\omega^i \mid i \in [n]\}$ of $\mathbb{F}_p^*$. By $\mathsf{negl}(\cdot)$, we mean any negligible function, i.e., a function $\mathsf{negl}\colon \mathbb{N} \to \mathbb{R}^+$ satisfying $\mathsf{negl}(n) < \frac{1}{\mathsf{poly}(n)}$ for all positive polynomials $\mathsf{poly}(\cdot)$ and all sufficiently large $n \in \mathbb{N}$.

$\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order p endowed with a *pairing* $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. A pairing is an efficiently computable, bilinear map with the following two properties:

- Bilinearity: For all generators $\mathsf{g}_1 \in \mathbb{G}_1$, $\mathsf{g}_2 \in \mathbb{G}_2$ and all $a, b \in \mathbb{F}_p$, we have $e(\mathsf{g}_1^a, \mathsf{g}_2^b) = e(\mathsf{g}_1, \mathsf{g}_2)^{ab}$.
- Non-degeneracy: For all generators $\mathsf{g}_1 \in \mathbb{G}_1$, $\mathsf{g}_2 \in \mathbb{G}_2$, $\mathsf{g}_T := e(\mathsf{g}_1, \mathsf{g}_2)$ is a generator of $\mathbb{G}_T$.

A statement of the form $y := \mathcal{A}(x)$ denotes a deterministic assignment, i.e., the variable y is assigned the output of the deterministic algorithm $\mathcal{A}$ on input x, while $y \leftarrow \mathcal{A}(x)$ denotes a probabilistic assignment, where the (now) probabilistic algorithm $\mathcal{A}$ is run

[1] The total degree of a multivariate polynomial is the largest sum of individual degrees in any of its monomials.

with uniform randomness on input x. We also use $x \leftarrow \mathcal{X}$ for sampling x uniformly at random from the set $\mathcal{X}$. We write $(y; z) \leftarrow (\mathcal{A} \parallel \mathcal{E})(x)$ when $\mathcal{A}$ on input x outputs y, and $\mathcal{E}$ on the same input (including $\mathcal{A}$'s randomness) outputs z. When running a stateful algorithm $\mathcal{A}$, we use st to denote its state, e.g., $(y_1, \mathsf{st}) \leftarrow \mathcal{A}(x_1); y_2 \leftarrow \mathcal{A}(\mathsf{st}, x_2)$.

Furthermore, $\Pr[S_1; \ldots; S_n : E]$ refers to the probability of event E after running the experiment given by sequentially executing the statements $S_1, \ldots, S_n$. In an interactive protocol between a prover $\mathcal{P}$ with input x and a verifier $\mathcal{V}$ with input y, we write $b \leftarrow \langle \mathcal{P}(x), \mathcal{V}(y) \rangle$ to denote the output of the verifier; w.l.o.g., this is a bit $b \in \{0, 1\}$ with 1 meaning accept and 0 reject.

An *indexed relation* $\mathcal{R}$ is a set of index-statement-witness triples $(\mathbf{i}, \mathbf{x}, \mathbf{w})$ with the corresponding *indexed language* $\mathcal{L}_{\mathcal{R}} := \{(\mathbf{i}, \mathbf{x}) \mid \exists \mathbf{w} : (\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}\}$. For a size bound $n \in \mathbb{N}$, we denote by $\mathcal{R}_n$ the restriction of $\mathcal{R}$ to triples $(\mathbf{i}, \mathbf{x}, \mathbf{w})$ with $|\mathbf{i}| \leq n$. For example, the indexed relation of satisfiable Boolean circuits consists of the triples where $\mathbf{i}$ is the description of a Boolean circuit, $\mathbf{x}$ is a partial assignment to its input wires, and $\mathbf{w}$ is an appropriate assignment to the remaining wires such that the circuit outputs 1.

Conventions. We implicitly assume all PPT algorithms described in this thesis run in time polynomial in the security parameter λ. Furthermore, we assume there is an efficient group-generation algorithm GGen which, on input 1^λ (i.e., the security parameter λ written in unary), generates appropriate $(\mathbb{F}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{g}_1, \mathsf{g}_2, \mathsf{g}_T, e)$ such that:

- $\mathbb{F}_p$ is a prime field of super-polynomial size $|\mathbb{F}_p| = \lambda^{\omega(1)}$, which admits certain primitive n-th roots of unity (putting a restriction on the factorization of $p - 1$).
- $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order p endowed with a pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.
- $\mathsf{g}_1 \in \mathbb{G}_1$ and $\mathsf{g}_2 \in \mathbb{G}_2$ are uniformly chosen generators, and $\mathsf{g}_T = e(\mathsf{g}_1, \mathsf{g}_2)$.

Furthermore, when instantiating a cryptographic scheme over $(\mathbb{F}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{g}_1, \mathsf{g}_2, \mathsf{g}_T, e)$, we assume that the group-generation algorithm GGen is implicitly run on input 1^λ during the setup phase, and that the outputs are publicly available to all involved parties.

2.2 Cryptographic Hardness Assumptions

For a cyclic group $\mathbb{G}$ of prime order p and a generator $\mathsf{g} \in \mathbb{G}$, the widely used *discrete logarithm assumption* (DLog) states that it is hard to compute $\log_{\mathsf{g}} h$ given a uniformly random element $h \in \mathbb{G}$. More formally, this holds relative to some group-generation algorithm which, on input 1^λ, implicitly outputs an appropriate group description $(\mathbb{G}, p, \mathsf{g})$ at the start of the following experiment:

Definition 2.1 (DLog Assumption)**.** *For all PPT adversaries $\mathcal{A}$:*

$$\Pr\left[\alpha \leftarrow \mathbb{F}_p; \alpha' \leftarrow \mathcal{A}(\mathsf{g}, \mathsf{g}^\alpha) : \alpha' = \alpha\right] \leq \mathsf{negl}(\lambda).$$

Furthermore, in their construction of PLONK, Gabizon et al. rely on the hardness of the d-DLog assumption in bilinear groups, which is a stronger version of discrete logarithm assumption. For groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p and a pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, it states that the discrete logarithm problem remains hard even when given the extended input $(\mathsf{g}_1, \mathsf{g}_1^{\alpha}, \ldots, \mathsf{g}_1^{\alpha^d}, \mathsf{g}_2, \mathsf{g}_2^{\alpha}, \ldots, \mathsf{g}_2^{\alpha^d}) \in \mathbb{G}_1^{d+1} \times \mathbb{G}_2^{d+1}$ for $d = \mathsf{poly}(\lambda)$. We will assume that this assumption is hard relative to GGen, i.e., for any fixed $\lambda \in \mathbb{N}_{>0}$, we implicitly assume $\mathsf{GGen}(1^\lambda)$ is run at the start of the following experiment:

Definition 2.2 (d-DLog Assumption)**.** *For all $d = \mathsf{poly}(\lambda)$ and all PPT adversaries $\mathcal{A}$:*

$$\Pr\left[\begin{array}{c} \alpha \leftarrow \mathbb{F}_p; \\ \alpha' \leftarrow \mathcal{A}(\mathsf{g}_1, \mathsf{g}_1^{\alpha}, \ldots, \mathsf{g}_1^{\alpha^d}, \mathsf{g}_2, \mathsf{g}_2^{\alpha}, \ldots, \mathsf{g}_2^{\alpha^d}) : \\ \alpha' = \alpha \end{array}\right] \leq \mathsf{negl}(\lambda).$$

2.3 Interactive Arguments of Knowledge

Before formally defining what a zk-SNARK is, we start with a weaker primitive called an *interactive argument of knowledge*, which is often used as the underlying scheme of a zk-SNARK. In fact, the PLONK zk-SNARK is obtained by first constructing a *public-coin* interactive argument of knowledge and then compiling it into its non-interactive version via the Fiat–Shamir transform [FS87]. Here, "public-coin" means that the verifier chooses all of its messages independently and uniformly at random from a specified domain, making its randomness public. More precisely, we will define "universal preprocessing zero-knowledge public-coin" interactive arguments of knowledge. In the following, we omit this lengthy nomenclature for the sake of simplicity. Also, note that our definitions are inspired by the definitions given in [CBBZ23, Sec. 2.1] and [CHM+20, Sec. 7].

Definition 2.3. *An interactive argument of knowledge for an indexed relation $\mathcal{R}$ is a tuple of four PPT algorithms $(\mathsf{Setup}, \mathsf{Preprocess}, \mathcal{P}, \mathcal{V})$ with the following syntax:*

- $\mathsf{Setup}(1^\lambda, n) \to \mathsf{srs}$: *A probabilistic algorithm that, given the security parameter 1^λ and a size bound $n \in \mathbb{N}$ on indices in $\mathcal{R}$, outputs a structured reference string srs.*
- $\mathsf{Preprocess}(\mathsf{srs}, \mathbf{i}) =: (\mathsf{pp}, \mathsf{vp})$: *A deterministic algorithm that, given the srs (with implicit size bound n) and an index $\mathbf{i}$ of size $\leq n$, outputs prover and verifier parameters pp and vp, respectively.*
- $\langle \mathcal{P}(\mathsf{pp}, \mathbf{x}, \mathbf{w}), \mathcal{V}(\mathsf{vp}, \mathbf{x}) \rangle \to b$: *An interactive public-coin protocol with common input a statement $\mathbf{x}$ between a prover $\mathcal{P}$ who has prover parameters pp as well as a witness $\mathbf{w}$ and a verifier $\mathcal{V}$ who has verifier parameters vp. The purpose of the protocol is to convince $\mathcal{V}$ that $(\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}$, where the index $\mathbf{i}$ is implicit in pp and vp. At the end of the protocol, $\mathcal{V}$ outputs a bit $b \in \{0, 1\}$, with 1 meaning accept and 0 reject.*

The given definition captures a *universal* interactive argument of knowledge, because the Setup algorithm generates a structured reference string srs which supports proofs of any

statement in the relation $\mathcal{R}$ up to a selected size bound n on the index $\mathbf{i}$. Moreover, an interactive argument of knowledge consists of two phases: a non-interactive preprocessing phase and an interactive online phase between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. In the first phase, anyone can use the deterministic algorithm $\mathsf{Preprocess}$ to publicly preprocess an index $\mathbf{i}$ of size $\leq n$ in the relation $\mathcal{R}$, resulting in prover parameters pp and verifier parameters vp. The purpose of this step is to compress the srs and index $\mathbf{i}$ into a form which contains the minimum amount of information needed by the prover and the verifier, respectively, to check different instances $\mathbf{x}$ in subsequent online phases. This will enable significant efficiency gains compared to directly using the srs and index $\mathbf{i}$, especially for the verifier. For this reason, it is also referred to as a *preprocessing* interactive argument of knowledge.

Furthermore, a zero-knowledge interactive argument of knowledge has to satisfy the following properties.

Completeness. If both parties behave honestly and follow the protocol, then the verifier will always accept a true statement, formalized as follows:

Definition 2.4 (Completeness)**.** *For all* $n = \mathsf{poly}(\lambda)$ *and all* $(\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}_n$*:*

$$\Pr\left[\begin{array}{c} \mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, n); \\ (\mathsf{pp}, \mathsf{vp}) := \mathsf{Preprocess}(\mathsf{srs}, \mathbf{i}); \\ b \leftarrow \langle \mathcal{P}(\mathsf{pp}, \mathbf{x}, \mathbf{w}), \mathcal{V}(\mathsf{vp}, \mathbf{x}) \rangle : \\ b = 1 \end{array}\right] = 1.$$

This is referred to as *perfect completeness*. It is also possible to consider a weaker notion called *statistical completeness*, which is allowed to fail with negligible probability, i.e., the probability above has to be at least $1 - \mathsf{negl}(\lambda)$. In fact, this will be the notion achieved by PLONK as discussed in Section 6.2.

Knowledge soundness. To motivate the stronger notion of *knowledge soundness*, we start by defining *soundness*, which ensures that a malicious prover cannot convince the verifier of a false statement. In the following, we give an adaptive definition of soundness, where the adversary $\mathcal{A}$ that represents the malicious prover gets to choose the index $\mathbf{i}$ and statement $\mathbf{x}$ based on the concrete structured reference string srs it sees.

Definition 2.5 (Soundness)**.** *For all* $n = \mathsf{poly}(\lambda)$ *and all PPT adversaries* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{c} \mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, n); \\ (\mathbf{i}, \mathbf{x}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{srs}); \\ (\mathsf{pp}, \mathsf{vp}) := \mathsf{Preprocess}(\mathsf{srs}, \mathbf{i}); \\ b \leftarrow \langle \mathcal{A}(\mathsf{st}), \mathcal{V}(\mathsf{vp}, \mathbf{x}) \rangle : \\ b = 1 \wedge (\mathbf{i}, \mathbf{x}) \notin \mathcal{L}_{\mathcal{R}_n} \end{array}\right] \leq \mathsf{negl}(\lambda).$$

On the other hand, knowledge soundness additionally requires the prover to know a witness whenever it makes the verifier accept a statement. This is formalized through the existence of an *extractor* $\mathcal{E}$, i.e., a PPT algorithm which is able to extract a witness from the prover whenever the verifier accepts. This extractor is given the prover's randomness and can therefore rewind it.

Definition 2.6 (Knowledge Soundness)**.** *For all $n = \mathsf{poly}(\lambda)$ and every PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that:*

$$\Pr\left[\begin{array}{c}\mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, n);\\ ((\mathbf{i}, \mathbf{x}, \mathsf{st}); \mathbf{w}) \leftarrow (\mathcal{A} \parallel \mathcal{E})(\mathsf{srs});\\ (\mathsf{pp}, \mathsf{vp}) := \mathsf{Preprocess}(\mathsf{srs}, \mathbf{i});\\ b \leftarrow \langle \mathcal{A}(\mathsf{st}), \mathcal{V}(\mathsf{vp}, \mathbf{x})\rangle :\\ b = 1 \wedge (\mathbf{i}, \mathbf{x}, \mathbf{w}) \notin \mathcal{R}_n\end{array}\right] \leq \mathsf{negl}(\lambda).$$

We refer to the above probability as the *knowledge soundness error*. Moreover, we mentioned that knowledge soundness is a stronger property than soundness. To see this, observe that the condition $(\mathbf{i}, \mathbf{x}) \notin \mathcal{L}_{\mathcal{R}_n}$ means $(\mathbf{i}, \mathbf{x}, \mathbf{w}) \notin \mathcal{R}_n$ for any choice of the witness $\mathbf{w}$ extracted by $\mathcal{E}$. Hence, knowledge soundness implies soundness.

Zero knowledge. As mentioned in the introduction, *zero knowledge* is defined through the existence of a *simulator*, i.e., a PPT algorithm capable of simulating the view of the verifier without relying on any secret information (i.e., the witness $\mathbf{w}$) used by the prover. In the context of zk-SNARKs and, by extension, also in our definition of interactive arguments of knowledge which are used as an underlying building block, the simulator is given the additional power of setting up the structured reference string. We formalize this by letting the adversary $\mathcal{A}$ play the role of the verifier in one of two experiments—either being given an honestly generated srs and interacting with the real prover algorithm $\mathcal{P}$, or being given a simulated srs and interacting with the simulator $\mathcal{S}$—and then having to output a decision bit $b \in \{0, 1\}$ for distinguishing the two scenarios.

Note that the first output of the simulator $\mathcal{S}$ includes a *trapdoor* τ, which can contain any information used to construct the srs. This is what gives the simulator the power to simulate correct proofs without knowledge of a witness in the first place. It also shows why the security of these systems fails if used with a corrupted srs, highlighting the importance of a correctly performed Setup.

Definition 2.7 (Zero Knowledge)**.** *There exists a PPT simulator $\mathcal{S}$ such that for all $n = \mathsf{poly}(\lambda)$ and all PPT adversaries $\mathcal{A}$:*

$$\left|\Pr\left[\begin{array}{c}\mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, n);\\ (\mathbf{i}, \mathbf{x}, \mathbf{w}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{srs});\\ (\mathsf{pp}, \mathsf{vp}) := \mathsf{Preprocess}(\mathsf{srs}, \mathbf{i});\\ b \leftarrow \langle \mathcal{P}(\mathsf{pp}, \mathbf{x}, \mathbf{w}), \mathcal{A}(\mathsf{st})\rangle :\\ b = 1 \wedge (\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}_n\end{array}\right] - \Pr\left[\begin{array}{c}(\mathsf{srs}, \tau) \leftarrow \mathcal{S}(1^\lambda, n);\\ (\mathbf{i}, \mathbf{x}, \mathbf{w}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{srs});\\ (\mathsf{pp}, \mathsf{vp}) := \mathsf{Preprocess}(\mathsf{srs}, \mathbf{i});\\ b \leftarrow \langle \mathcal{S}(\tau, \mathsf{pp}, \mathbf{x}), \mathcal{A}(\mathsf{st})\rangle :\\ b = 1 \wedge (\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}_n\end{array}\right]\right| \leq \mathsf{negl}(\lambda).$$

This is also called *computational zero knowledge*, since the adversary runs in PPT. In addition, there are the two increasingly stronger notions of *statistical zero knowledge* and *perfect zero knowledge*, which have to hold even against unbounded adversaries $\mathcal{A}$, and on top of that, for perfect zero knowledge, the above probabilities have to be equal. Moreover, we obtain the weaker notion of *honest-verifier zero knowledge* (HVZK) if the adversary $\mathcal{A}$ is assumed to honestly follow the verifier algorithm $\mathcal{V}$ during the protocol execution, except for its final output $b \in \{0,1\}$ (otherwise, the simulator would just need to produce accepting protocol transcripts whenever $(\mathbf{i},\mathbf{x},\mathbf{w}) \in \mathcal{R}_n$, which is weaker than zero knowledge).

2.4 zk-SNARKs

Now we are ready to define zk-SNARKs, i.e., *zero-knowledge succinct non-interactive arguments of knowledge*. A (universal preprocessing) zk-SNARK is defined analogously to an interactive argument of knowledge $(\mathsf{Setup},\mathsf{Preprocess},\mathcal{P},\mathcal{V})$ from Definition 2.3, except that the interactive algorithms $\mathcal{P}$ and $\mathcal{V}$ are replaced with their non-interactive counterparts Prove and Verify as follows:

Definition 2.8 (zk-SNARK)**.** *A zk-SNARK for an indexed relation $\mathcal{R}$ is a tuple of four PPT algorithms* $(\mathsf{Setup},\mathsf{Preprocess},\mathsf{Prove},\mathsf{Verify})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda, n) \to \mathsf{srs}$: *A probabilistic algorithm that, given the security parameter 1^λ and a size bound $n \in \mathbb{N}$ on indices in $\mathcal{R}$, outputs a structured reference string* srs.
- $\mathsf{Preprocess}(\mathsf{srs},\mathbf{i}) =: (\mathsf{pp},\mathsf{vp})$: *A deterministic algorithm that, given the* srs *(with implicit size bound n) and an index $\mathbf{i}$ of size $\leq n$, outputs prover and verifier parameters* pp *and* vp*, respectively.*
- $\mathsf{Prove}(\mathsf{pp},\mathbf{x},\mathbf{w}) \to \pi$: *A probabilistic algorithm that, given* pp *(with implicit index $\mathbf{i}$), a statement $\mathbf{x}$ and a witness $\mathbf{w}$, outputs a proof π for $(\mathbf{i},\mathbf{x},\mathbf{w}) \in \mathcal{R}$.*
- $\mathsf{Verify}(\mathsf{vp},\mathbf{x},\pi) =: b$: *A deterministic algorithm that, given* vp*, a statement $\mathbf{x}$ and a proof π, outputs a bit $b \in \{0,1\}$, with 1 meaning accept and 0 reject.*

Furthermore, a zk-SNARK has to satisfy the non-interactive analogues of the properties *completeness*, *knowledge soundness*, and *zero knowledge* of an interactive argument of knowledge, as defined in the previous section. For completeness, we state them below.

Definition 2.9 (Completeness)**.** *For all $n = \mathsf{poly}(\lambda)$ and all $(\mathbf{i},\mathbf{x},\mathbf{w}) \in \mathcal{R}_n$:*

$$\Pr\left[\begin{array}{c} \mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, n); \\ (\mathsf{pp},\mathsf{vp}) := \mathsf{Preprocess}(\mathsf{srs},\mathbf{i}); \\ \pi \leftarrow \mathsf{Prove}(\mathsf{pp},\mathbf{x},\mathbf{w}); \\ b := \mathsf{Verify}(\mathsf{vp},\mathbf{x},\pi): \\ b = 1 \end{array}\right] = 1.$$

Definition 2.10 (Knowledge Soundness)**.** *For all $n = \mathsf{poly}(\lambda)$ and every PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that:*

$$\Pr\left[\begin{array}{c}\mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, n);\\ ((\mathbf{i}, \mathbf{x}, \pi); \mathbf{w}) \leftarrow (\mathcal{A} \parallel \mathcal{E})(\mathsf{srs});\\ (\mathsf{pp}, \mathsf{vp}) := \mathsf{Preprocess}(\mathsf{srs}, \mathbf{i});\\ b := \mathsf{Verify}(\mathsf{vp}, \mathbf{x}, \pi):\\ b = 1 \wedge (\mathbf{i}, \mathbf{x}, \mathbf{w}) \notin \mathcal{R}_n\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 2.11 (Zero Knowledge)**.** *There exists a PPT simulator $\mathcal{S}$ such that for all $n = \mathsf{poly}(\lambda)$ and all PPT adversaries $\mathcal{A}$:*

$$\left|\Pr\left[\begin{array}{c}\mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, n);\\ (\mathbf{i}, \mathbf{x}, \mathbf{w}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{srs});\\ (\mathsf{pp}, \mathsf{vp}) := \mathsf{Preprocess}(\mathsf{srs}, \mathbf{i});\\ \pi \leftarrow \mathcal{P}(\mathsf{pp}, \mathbf{x}, \mathbf{w});\\ b \leftarrow \mathcal{A}(\mathsf{st}, \pi):\\ b = 1 \wedge (\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}_n\end{array}\right] - \Pr\left[\begin{array}{c}(\mathsf{srs}, \tau) \leftarrow \mathcal{S}(1^\lambda, n);\\ (\mathbf{i}, \mathbf{x}, \mathbf{w}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{srs});\\ (\mathsf{pp}, \mathsf{vp}) := \mathsf{Preprocess}(\mathsf{srs}, \mathbf{i});\\ \pi \leftarrow \mathcal{S}(\tau, \mathsf{pp}, \mathbf{x});\\ b \leftarrow \mathcal{A}(\mathsf{st}, \pi):\\ b = 1 \wedge (\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}_n\end{array}\right]\right| \leq \mathsf{negl}(\lambda).$$

Again, we make a distinction between *computational*, *statistical*, and *perfect* zero knowledge as explained below Definition 2.7. However, since the creation of a proof no longer involves any interaction with the verifier, the notion of *honest-verifier* zero knowledge becomes obsolete.

Lastly, there is one more essential property that makes zk-SNARKs so appealing in practice: their *succinctness*.

Definition 2.12 (Succinctness)**.** *For all $n = \mathsf{poly}(\lambda)$ and all $(\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}_n$:*

$$\Pr\left[\begin{array}{c}\mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, n);\\ (\mathsf{pp}, \mathsf{vp}) := \mathsf{Preprocess}(\mathsf{srs}, \mathbf{i});\\ \pi \leftarrow \mathsf{Prove}(\mathsf{pp}, \mathbf{x}, \mathbf{w}):\\ |\pi| = o(|\mathbf{i}| + |\mathbf{x}|)\end{array}\right] = 1.$$

Put into words, this states that zk-SNARK proofs are sublinear in the size of the proved statement $(\mathbf{i}, \mathbf{x})$. As discussed in Section 1.3, this can range anywhere from polylogarithmic, i.e., $O(\log^c(|\mathbf{i}| + |\mathbf{x}|))$, to constant-size[2], i.e., $O(1)$, proofs depending on the concrete zk-SNARK. Note that, since we are only concerned with proving statements about languages in NP, we can (asymptotically) ignore the dependence on the size of the corresponding witness $\mathbf{w}$, assuming w.l.o.g. $|\mathbf{w}| = \mathsf{poly}(|\mathbf{i}| + |\mathbf{x}|)$, i.e., its size is polynomial in the size of the index-statement pair.

[2]Note that *constant-size* here means "constant in the size of the statement" ignoring the dependence on the security parameter λ, i.e., the proof is actually of size $\mathsf{poly}(\lambda)$.

Moreover, we say that a zk-SNARK is *fully succinct* if, in addition to the proof length, also the proof verification time is sublinear in the size of the proved statement. Since PLONK achieves this additional property and has a universal SRS, it is a universal and fully succinct zk-SNARK.

2.5 Arithmetic Circuits

Most of the state-of-the-art zk-SNARK constructions, including PLONK, use the concept of *arithmetic circuits*, a generalization of Boolean circuits, as their representation of the NP relation $\mathcal{R}$ they are proving statements in. As discussed later in this section, the language of satisfiable arithmetic circuits is NP-complete and can thus be used to express all of NP.

An arithmetic circuit is a directed acyclic graph whose vertices and edges represent gates and wires for carrying out computations in a finite field $\mathbb{F}_p$, respectively. There are 5 types of gates: input and constant gates of indegree 0 and outdegree ≥ 1, output gates of indegree 1 and outdegree 0, as well as addition and multiplication gates of indegree ≥ 2 and outdegree ≥ 1. Hence, arithmetic circuits provide a framework for computing multivariate polynomials over a finite field $\mathbb{F}_p$. For example, consider the arithmetic circuit $\mathcal{C}\colon \mathbb{F}_p^3 \to \mathbb{F}_p$ that computes the polynomial $f(x_1, x_2, x_3) := (x_1 + x_2) \cdot (x_2 + x_3)$ over $\mathbb{F}_p$ shown in Figure 2.1.

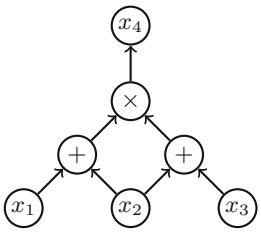

Figure 2.1: Arithmetic circuit for computing $x_4 := (x_1 + x_2) \cdot (x_2 + x_3)$.

To motivate the choice of arithmetic circuits as a model of feasible computation, let us examine them from a complexity-theoretic point of view. Given an arithmetic circuit $\mathcal{C}$ over $\mathbb{F}_p$ with n inputs, the *arithmetic circuit satisfiability* problem asks whether there exist inputs $x_1, \ldots, x_n \in \mathbb{F}_p$ such that $\mathcal{C}(x_1, \ldots, x_n) = 0$, in analogy to roots of a multivariate polynomial. It is well-known that Boolean circuit satisfiability is NP-complete via a reduction from the SAT problem. Since Boolean circuits are just a special case of arithmetic circuits over the field $\mathbb{F}_2$, the same is true for arithmetic circuit satisfiability. Thus, it captures all efficiently computable functions. Moreover, arithmetic circuit satisfiability can be expressed as the indexed relation containing all the triples $(\mathbf{i}, \mathbf{x}, \mathbf{w})$, where the index $\mathbf{i}$ is the description of an arithmetic circuit $\mathcal{C}$, the statement $\mathbf{x}$ is a partial assignment of values to its input gates, and $\mathbf{w}$ is an appropriate assignment to all of its wires such that the circuit outputs 0.

2.6 Polynomial Identity Testing

At multiple points throughout this thesis, we will rely on the well-known Schwartz–Zippel lemma, stated below.

Lemma 2.1 (Schwartz–Zippel)**.** *Let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$ be a non-zero multivariate polynomial of total degree d over $\mathbb{F}_p$ and $S \subseteq \mathbb{F}_p$. Then*

$$\Pr\Big[(\alpha_1, \ldots, \alpha_n) \leftarrow S^n : f(\alpha_1, \ldots, \alpha_n) = 0\Big] \leq \frac{d}{|S|}.$$

In particular, for any point $(\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_p^n$ chosen uniformly at random from the whole field, the above probability is at most n/p. The lemma is based on the fact that any non-zero univariate polynomial of degree d has at most d roots. Informally, its corollary states that two distinct polynomials $f \not\equiv g$ of degree at most d differ almost everywhere, or more precisely, that they agree on at most d points (otherwise, the non-zero, degree-d polynomial $f - g$ would have more than d roots).

Consequently, we can use this lemma for probabilistic *polynomial identity testing*. Given two polynomials $f(X_1, \ldots, X_n)$ and $g(X_1, \ldots, X_n)$ over $\mathbb{F}_p$, it suffices to check at a random point $(\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_p^n$ whether $f(\alpha_1, \ldots, \alpha_n) = g(\alpha_1, \ldots, \alpha_n)$. It $f \equiv g$, then this holds with probability one. Otherwise, we can apply the Schwartz–Zippel lemma to the non-zero polynomial $f - g$, telling us that the check passes with probability at most $\max(d_1, d_2)/p$, where d_1 and d_2 are the degrees of the two polynomials.

2.7 Lagrange Interpolation

Throughout this thesis, we will often be interested in polynomials taking specific values over a set of distinct evaluation points $S := \{x_1, \ldots, x_n\} \subseteq \mathbb{F}_p$. For this purpose, we will use Lagrange interpolation, which allows interpolating a set of n points $(x_i, y_i)_{i \in [n]}$ with the *unique* polynomial $f \in \mathbb{F}_p^{(\leq n-1)}[X]$ of degree at most $n - 1$ such that $f(x_i) = y_i$ for all $i \in [n]$.

Concretely, the interpolated polynomial is computed via the following linear combination:

$$f(X) := \sum_{i \in [n]} y_i \mathcal{L}_i(X), \tag{2.1}$$

where the $\mathcal{L}_i(X) := \prod_{j \in [n], j \neq i} \frac{X - x_j}{x_i - x_j}$ are *Lagrange basis polynomials* (or simply Lagrange polynomials) of degree $n-1$. Notice that a Lagrange polynomial $\mathcal{L}_i$ has the property that $\mathcal{L}_i(x_i) = 1$, but $\mathcal{L}_i(x_j) = 0$ for all $j \neq i \in [n]$, directly yielding the desired polynomial f via Equation (2.1). Since all of these polynomials are independent of the evaluation values $y_1, \ldots, y_n$, the set $\{\mathcal{L}_i\}_{i \in [n]}$ is referred to as a Lagrange basis for S.

Furthermore, we can rewrite the i-th Lagrange polynomial as follows:

Lemma 2.2. *Let* $S := \{x_1, \ldots, x_n\} \subseteq \mathbb{F}_p$ *with* $|S| = n$, $\mathcal{L}_i(X) := \prod_{j \in [n], j \neq i} \frac{X - x_j}{x_i - x_j}$, *and* $Z_S(X) := \prod_{i \in [n]} (X - x_i)$. *Then*

$$\mathcal{L}_i(X) = \frac{Z_S(X)}{Z'_S(x_i)(X - x_i)},$$

where Z'_S *denotes the formal derivative*[3] *of* Z_S.

This is also known as the *barycentric form* of the Lagrange polynomials [BT04]. For a proof why it is equivalent to the original definition, see Appendix A.2.

To see the benefit of this representation, take the set $H := \langle \omega \rangle$, where ω is a primitive n-th root of unity in $\mathbb{F}_p$. Then the polynomial Z_H simply turns into $X^n - 1$, whose formal derivative is $Z'_H(X) = nX^{n-1}$. Evaluating at any ω^i, we get $Z'_H(\omega^i) = n(\omega^i)^{n-1} = n\omega^{-i}$, which plugged into the i-th Lagrange polynomial gives

$$\mathcal{L}_i(X) = \frac{Z_H(X)}{Z'_H(\omega^i)(X - \omega^i)} = \frac{X^n - 1}{n\omega^{-i}(X - \omega^i)} = \frac{\omega^i(X^n - 1)}{n(X - \omega^i)}.$$

As a result, we are now able to evaluate any Lagrange polynomial at an arbitrary $x \in \mathbb{F}_p$ in $O(\log n)$ field operations as opposed to the $O(n)$ field operations required via the original formula. In fact, for a fixed $x \in \mathbb{F}_p$, we only have a constant number of field operations once the value of $Z_H(x) = x^n - 1$ is computed, which allows us to evaluate any interpolated polynomial $f(X) := \sum_{i \in [n]} y_i \mathcal{L}_i(X)$ at the point x in $O(n + \log n)$ field operations.

2.8 KZG Polynomial Commitments

A *commitment scheme* is a cryptographic primitive that allows one to commit to a chosen value while keeping it hidden from others, with the ability to later reveal the committed value. Crucially, it should not be possible to reveal another value than the one originally committed to, which is known as the scheme's *binding* property. Informally, a commitment can be thought of as a digital envelope, which can only be opened by the party that sealed it.

A key ingredient to the PLONK zk-SNARK is a *polynomial commitment scheme* (PCS), which allows committing to a polynomial in a succinct manner and then later efficiently prove evaluations of it without revealing the entire polynomial. A very prominent PCS is the *KZG polynomial commitment scheme* by Kate, Zaverucha, and Goldberg [KZG10]. Although PLONK can be instantiated with any PCS, Gabizon et al. use the KZG scheme for its efficiency. In general, there is a trade-off between efficiency and relying on a structured reference string in current state-of-the-art PCS constructions.

[3] The formal derivative of a polynomial $f(X) := \sum_{i=0}^{d} f_i X^i \in \mathbb{F}_p[X]$ over the finite field $\mathbb{F}_p$ is defined as the polynomial $f'(X) := \sum_{i=1}^{d} i f_i X^{i-1}$, coinciding with the classical derivative from calculus.

KZG is a constant-size polynomial commitment scheme for polynomials $f \in \mathbb{F}_p[X]$ of degree $\leq d$, i.e., a commitment as well as evaluation proofs are of constant size and independent of the degree of the committed polynomial. To support commitments to polynomials up to degree d, KZG requires a structured reference string of the form $(\mathsf{g}_1, \mathsf{g}_1^{\tau}, \mathsf{g}_1^{\tau^2}, \ldots, \mathsf{g}_1^{\tau^d}, \mathsf{g}_2, \mathsf{g}_2^{\tau})$, which can be either computed by a trusted authority or via a decentralized MPC protocol such as [BGM17] that hides the trapdoor τ. The scheme is based on the observation that $(X - x)$ evenly divides the polynomial $f(X) - f(x)$ for any $x \in \mathbb{F}_p$. This algebraic property of polynomials is also known as the *polynomial remainder theorem*, stated and proved below.

Theorem 2.1 (Polynomial Remainder Theorem)**.** *Let $f \in \mathbb{F}_p[X]$ be a polynomial of degree d, and let $x \in \mathbb{F}_p$. Then $f(x) = y$ if and only if there exists a polynomial $Q \in \mathbb{F}_p[X]$ of degree $d - 1$ such that $f(X) - y = Q(X)(X - x)$.*

Proof. The backward direction of the claim follows immediately by substituting x for X in $f(X) - y = Q(X)(X - x)$. For the forward direction, carrying out the polynomial division of f by $(X - x)$ yields

$$f(X) = Q(X)(X - x) + R(X),$$

where Q is the quotient polynomial of degree $d - 1$ and R is the remainder polynomial of degree less than $\deg(X - x) = 1$. So R is constant and can be written as $R(X) = y'$ for some $y' \in \mathbb{F}_p$, leaving us with $f(X) = Q(X)(X - x) + y'$. Plugging in $X = x$, we directly obtain $y' = y$, finishing the proof. □

Committing. Let $f(X) := \sum_{i=0}^{d} f_i X^i \in \mathbb{F}_p^{(\leq d)}[X]$ be a polynomial of degree $\leq d$ with coefficients f_0, f_1, …, f_d in $\mathbb{F}_p$. A KZG commitment to f is a single group element

$$c := \prod_{i=0}^{d} (\mathsf{g}_1^{\tau^i})^{f_i} = \mathsf{g}_1^{\sum_{i=0}^{d} f_i \tau^i} = \mathsf{g}_1^{f(\tau)},$$

where the values $\mathsf{g}_1^{\tau^i}$ are taken from the structured reference string.

Proving evaluations. To prove that the polynomial $f \in \mathbb{F}_p[X]$ committed to in c evaluates to y at the point x, i.e., $f(x) = y$, one simply commits to the unique quotient polynomial $Q(X) := \frac{f(X) - y}{X - x}$ of degree $\deg(f) - 1$, which can be computed in $O(\deg f)$ field operations. The proof $\pi := \mathsf{g}_1^{Q(\tau)}$ can then be verified (in constant time) by checking the following equation via the pairing e:

$$e(c/\mathsf{g}_1^{y}, \mathsf{g}_2) = e(\pi, \mathsf{g}_1^{\tau}/\mathsf{g}_2^{x}) \tag{2.2}$$

$$\iff \quad e(\mathsf{g}_1^{f(\tau)-y}, \mathsf{g}_2) = e(\mathsf{g}_1^{Q(\tau)}, \mathsf{g}_2^{\tau-x}) \tag{2.3}$$

$$\iff \quad e(\mathsf{g}_1, \mathsf{g}_2)^{f(\tau)-y} = e(\mathsf{g}_1, \mathsf{g}_2)^{Q(\tau)(\tau-x)} \tag{2.4}$$

$$\iff \quad f(\tau) - y = Q(\tau)(\tau - x) \pmod{p} \tag{2.5}$$

Importantly, an adversary should not be able to commit to a polynomial $f \in \mathbb{F}_p[X]$, and then later open it at an incorrect evaluation $y^* \neq f(x)$. We will later formalize this notion when dealing with interactive protocols using the KZG scheme as a building block in Chapter 4.

2.9 Algebraic Group Model

For security analysis of the PLONK protocol, Gabizon et al. use the *algebraic group model* (AGM) of Fuchsbauer, Kiltz and Loss [FKL18]. This idealized model lies between the standard model and the *generic group model* [Sho97, Mau05], where algorithms are only able to use group operations in a black-box manner, not being allowed to exploit the internal representation of the group elements. On the other hand, algorithms in the AGM are less restricted in that they are allowed to use the concrete encoding of group elements as long as they can explain any group element in their output as a linear combination of the group elements they have been provided with as input.

More formally, in the context of security proofs in the AGM, we say that an adversary $\mathcal{A}$ is *algebraic* if it runs in probabilistic polynomial time (PPT) and satisfies the following: Whenever it outputs a group element $h \in \mathbb{G}$, it also outputs a vector of coefficients $(a_1, \ldots, a_n) \in \mathbb{F}_p^n$ such that $h = \prod_{i \in [n]} g_i^{a_i}$, where $g_1, \ldots, g_n \in \mathbb{G}$ are all the group elements previously given to $\mathcal{A}$, i.e., the vector "explains" the new group element as a linear combination of the previously seen group elements.

This model is useful to us in that it allows to directly extract a polynomial from a KZG commitment [KZG10], as introduced in Section 2.8. In particular, in all the protocols considered in this thesis, the only group elements given to an adversary will be the ones contained in the structured reference string $(\mathsf{g}_1, \mathsf{g}_1^\tau, \mathsf{g}_1^{\tau^2}, \ldots, \mathsf{g}_1^{\tau^d}, \mathsf{g}_2, \mathsf{g}_2^\tau) \in \mathbb{G}_1^{d+1} \times \mathbb{G}_2^2$. So, whenever an algebraic adversary outputs a KZG commitment $c \in \mathbb{G}_1$, it must also know a representation $f_1, \ldots, f_d \in \mathbb{F}_p$ such that $c = \prod_{i=0}^{d} (\mathsf{g}_1^{\tau^i})^{f_i}$, implicitly defining the corresponding polynomial $f(X) := \sum_{j=0}^{d} f_j X^j \in \mathbb{F}_p^{(\leq d)}[X]$ of degree $\leq d$. This will greatly ease the construction of an appropriate extractor $\mathcal{E}$ when proving that a protocol relying on KZG commitments is knowledge sound (with respect to Definition 2.6) in the AGM.

CHAPTER 3

The PLONK Arithmetization

In this chapter, we explore how PLONK transforms a statement about an arithmetic circuit into an equivalent system of constraints. In the context of zero-knowledge proofs, this preprocessing step is called *arithmetization* and plays an essential part in enabling PLONK's efficiency.

3.1 The PLONK Constraint System

Gabizon et al. introduce the following constraint system [GWC19, Sec. 6] to represent any fan-in-two arithmetic circuit $\mathcal{C}$ of unlimited fan-out over $\mathbb{F}_p$. Let n denote the number of gates (without counting private input/output gates as explained later) and m the number of wires. Let $\mathbf{w} \in \mathbb{F}_p^m$ be the wire assignment. Let the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c} \in [m]^n$ index the left input, right input, and output wire of every gate in $\mathcal{C}$, respectively. For example, the i-th component of $\mathbf{a}$ is defined as $\mathbf{a}_i := j$, where $j \in [m]$ is the index of the value $\mathbf{w}_j$ assigned to the left input wire of the i-th gate.

Gate Constraints. As a first step, Gabizon et al. express the computation carried out by each gate $i \in [n]$ via a constraint of the form

$$(\mathbf{s_L})_i \cdot \mathbf{w}_{\mathbf{a}_i} + (\mathbf{s_R})_i \cdot \mathbf{w}_{\mathbf{b}_i} + (\mathbf{s_O})_i \cdot \mathbf{w}_{\mathbf{c}_i} + (\mathbf{s_M})_i \cdot \mathbf{w}_{\mathbf{a}_i} \cdot \mathbf{w}_{\mathbf{b}_i} + (\mathbf{s_C})_i + (\mathbf{s_{PI}})_i = 0, \tag{3.1}$$

where $\mathbf{s_L}, \mathbf{s_R}, \mathbf{s_O}, \mathbf{s_M}, \mathbf{s_C}, \mathbf{s_{PI}} \in \mathbb{F}_p^n$ are selector vectors taking fixed values based on the gate type as specified in Table 3.1.

It is easy to verify that these constraints ensure the wire assignment $\mathbf{w} \in \mathbb{F}_p^m$ respects each gate. For example, if the i-th gate is an addition gate, then, by the selector vector assignment from Table 3.1, the corresponding constraint becomes $\mathbf{w}_{\mathbf{a}_i} + \mathbf{w}_{\mathbf{b}_i} - \mathbf{w}_{\mathbf{c}_i} = 0$, as required. Accordingly, they are referred to as the *gate constraints*.

Table 3.1: Selector vector assignment, where c_i denotes the value of the respective constant gate and x_i the value of the respective public input/output gate.

Type	$(\mathbf{s_L})_i$	$(\mathbf{s_R})_i$	$(\mathbf{s_O})_i$	$(\mathbf{s_M})_i$	$(\mathbf{s_C})_i$	$(\mathbf{s_{PI}})_i$
Addition gate	1	1	-1	0	0	0
Multiplication gate	0	0	-1	1	0	0
Constant gate	1	0	0	0	$-c_i$	0
Public input/output gate	1	0	0	0	0	$-x_i$

Furthermore, we make a distinction between *public* and *private* input/output gates, where only the former count towards the number of gates n. The motivation behind this is that this constraint system is designed to be used as a relation for proving statements about arithmetic circuits with a zk-SNARK. In this context, the prover who has the complete wire assignment $\mathbf{w} \in \mathbb{F}_p^m$ wants to prove it is consistent with the circuit's gates, including constant gates and "public" input/output gates whose values represent the verifier's public inputs. In particular, the values assigned to all the remaining input/output gates can be chosen freely by the prover, representing its private inputs. In consequence, we do not include these "private" input/output gates in the number of gates n, aligning it with the number of actual gate constraints.

Copy Constraints. Thus far, we are treating the circuit as a collection of independent gates, completely ignoring the wiring. To properly propagate each gate's output along the circuit's wires, we additionally need to enforce certain equalities between $\mathbf{a}, \mathbf{b}, \mathbf{c}$, which we call the *copy constraints*. For example, suppose the i-th gate's output is also the left input to the j-th gate. Then we want the equality $\mathbf{a}_j = \mathbf{c}_i$ to hold. To capture the general case, let $\mathbf{d} := (\mathbf{a} \| \mathbf{b} \| \mathbf{c}) \in [m]^{3n}$ denote the concatenation of the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$, and let $I_1, \ldots, I_\ell$ be a partition of $[3n]$ into ℓ disjoint subsets, each of which satisfies $\mathbf{d}_j = \mathbf{d}_k$ for all $j, k \in I_i$. If we then let $\sigma \colon [3n] \to [3n]$ be a permutation with ℓ cycles, spanning one of the subsets $I_1, \ldots, I_\ell$ each, the copy constraints can be written in a more condensed form as

$$\forall i \in [3n], \mathbf{d}_{\sigma(i)} = \mathbf{d}_i. \tag{3.2}$$

This works because each cycle in σ creates a chain of equalities for the values in its corresponding block I_i, i.e., $\mathbf{d}_j = \mathbf{d}_k$ for all $j, k \in I_i$.

Note that this constraint system is more general than what we have described here. As such, it can be extended to capture gates of higher fan-in or to support custom gates. For example, by setting $\mathbf{s_L}$ and $\mathbf{s_R}$ to different values in an addition gate, we can obtain arbitrary "linear combination gates".

Example. To better visualize the resulting constraint system, consider the arithmetic circuit $\mathcal{C} \colon \mathbb{F}_p^3 \to \mathbb{F}_p$ depicted in Figure 3.1 that computes $x_4 := (x_1 + x_2) \cdot (x_2 + x_3)$, where all three inputs x_1, x_2, x_3 are private and only the output x_4 is public.

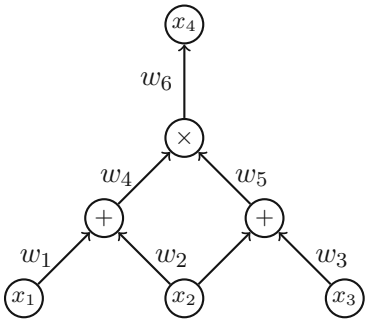

Figure 3.1: Example circuit with private inputs x_1, x_2, x_3 and public input x_4.

Ignoring the 3 private input gates, we have $n = 4$ gates and $m = 6$ wires. Note that even though the input gate x_2 is depicted with two output edges, these correspond to just a single wire in practice which carries the same value w_2. Numbering the 4 gates from bottom left to top right and using the wire assignment $\mathbf{w} := (w_1, \ldots, w_6) \in \mathbb{F}_p^6$ given in Figure 3.1, the indexing vectors $\mathbf{a}, \mathbf{b}, \mathbf{c} \in [6]^4$ take the values

$$\mathbf{a} := (1, 2, 4, 6),\ \mathbf{b} := (2, 3, 5, 6),\ \mathbf{c} := (4, 5, 6, 6).$$

For example, the first gate is the addition gate with left input $\mathbf{w}_{\mathbf{a}_1} = w_1$, right input $\mathbf{w}_{\mathbf{b}_1} = w_2$, and output $\mathbf{w}_{\mathbf{c}_1} = w_4$. Note that the fourth gate is the public output gate, for which, according to the definition of the selector vector assignment in Table 3.1, it is only essential that the left input wire w_6 is indexed correctly by $\mathbf{a}_4 = 6$, while $\mathbf{b}_4$ and $\mathbf{c}_4$ can take arbitrary values (as they are zeroed out in the resulting gate constraint). As a result, we obtain the following 4 gate constraints:

$$w_1 + w_2 - w_4 = 0 \tag{3.3}$$
$$w_2 + w_3 - w_5 = 0 \tag{3.4}$$
$$w_4 \cdot w_5 - w_6 = 0 \tag{3.5}$$
$$w_6 - x_4 = 0 \tag{3.6}$$

As for the copy constraints, we want the following equalities to hold for the gate outputs to be properly propagated along the circuit:

$$\mathbf{a}_3 = \mathbf{c}_1,\ \mathbf{a}_4 = \mathbf{c}_3,\ \mathbf{b}_3 = \mathbf{c}_2$$

Alternatively, let $\mathbf{d} := (\mathbf{a} \| \mathbf{b} \| \mathbf{c}) \in [6]^{12}$ and rewrite the copy constraints as:

$$\mathbf{d}_3 = \mathbf{d}_9,\ \mathbf{d}_4 = \mathbf{d}_{11},\ \mathbf{d}_7 = \mathbf{d}_{10}$$

Then, using cycle notation, define the permutation $\sigma \colon [12] \to [12]$ as

$$\sigma := (1)(2)(3\ 9)(4\ 11)(5)(6)(7\ 10)(8)(12),$$

which consists only of fixed points except for the 2-cycles (3 9), (4 11), (7 10) corresponding to the 3 copy constraints. With this, we obtain the equivalent statement

$$\forall i \in [12], \mathbf{d}_{\sigma(i)} = \mathbf{d}_i$$

for representing the copy constraints.

3.2 Polynomial Representation

An important and very useful property of the PLONK constraint system is that it can be expressed in its entirety via polynomials, which are the actual objects used to construct the PLONK protocol. For this purpose, if n is the number of gates in the circuit $\mathcal{C}$ as defined earlier, then let the polynomials $A, B, C \in \mathbb{F}_p^{(<n)}[X]$ interpolate the values in $\mathbf{w}$ along the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ such that for each $i \in [n]$

$$A(i) = \mathbf{w}_{\mathbf{a}_i}, B(i) = \mathbf{w}_{\mathbf{b}_i}, C(i) = \mathbf{w}_{\mathbf{c}_i}.$$

Also, let $S_{\mathsf{L}}, S_{\mathsf{R}}, S_{\mathsf{O}}, S_{\mathsf{M}}, S_{\mathsf{C}}, S_{\mathsf{PI}} \in \mathbb{F}_p^{(<n)}[X]$ interpolate $\mathbf{s_L}, \mathbf{s_R}, \mathbf{s_O}, \mathbf{s_M}, \mathbf{s_C}, \mathbf{s_{PI}}$ such that for each $i \in [n]$:

$$\begin{aligned} S_{\mathsf{L}}(i) = (\mathbf{s_L})_i,\ \ S_{\mathsf{R}}(i) = (\mathbf{s_R})_i,\ \ S_{\mathsf{O}}(i) = (\mathbf{s_O})_i, \\ S_{\mathsf{M}}(i) = (\mathbf{s_M})_i,\ S_{\mathsf{C}}(i) = (\mathbf{s_C})_i,\ S_{\mathsf{PI}}(i) = (\mathbf{s_{PI}})_i. \end{aligned}$$

By substituting in Equation (3.1), this allows us to rewrite the i-th gate constraint as

$$S_{\mathsf{L}}(i)A(i) + S_{\mathsf{R}}(i)B(i) + S_{\mathsf{O}}(i)C(i) + S_{\mathsf{M}}(i)A(i)B(i) + S_{\mathsf{C}}(i) + S_{\mathsf{PI}}(i) = 0.$$

To obtain an appropriately adapted form of the copy constraints, let us define the following evaluation sequence of the polynomials A, B, C for all $i \in [3n]$:

$$D_{(i)} := \begin{cases} A(i) & \text{if } i \in [n] \\ B(i-n) & \text{if } i \in (n, 2n] \\ C(i-2n) & \text{if } i \in (2n, 3n] \end{cases}$$

Now we can express the copy constraints from Equation (3.2) as

$$\forall i \in [3n], D_{(\sigma(i))} = D_{(i)}. \tag{3.7}$$

This representation is a preview of Chapter 6, where we take a closer look at the permutation argument used by Gabizon et al. to efficiently prove such statements. It is also the namesake of PLONK, which stands for ***P**ermutations over **L**agrange-bases for **O**ecumenical **N**oninteractive arguments of **K**nowledge*.

CHAPTER 4

Polynomial Commitments with Cross-Commitment Proof Aggregation

The PLONK construction relies on a polynomial commitment scheme (PCS) with *cross-commitment proof aggregation*. This primitive allows to efficiently open multiple committed polynomials across several points. For example, the prover might need to convince the verifier about the evaluations of multiple polynomials $f_1, \ldots, f_k \in \mathbb{F}_p[X]$ at a single point $x \in \mathbb{F}_p$. Doing this the trivial way would require a number of evaluation proofs linear in the number of polynomials. To improve on this bound, Gabizon et al. directly extend the KZG polynomial commitment scheme [KZG10] from Section 2.8, achieving the same with only a single aggregated evaluation proof. In general, their construction supports opening k polynomials at ℓ points at a communication cost of ℓ evaluation proofs.

In this chapter, we first precisely define what we mean by a PCS with cross-commitment proof aggregation, before presenting the instantiation by Gabizon et al. and proving it satisfies the necessary security properties.

4.1 Definition

We begin by formally defining the syntax of a polynomial commitment scheme with cross-commitment proof aggregation.

Definition 4.1 (PCS with Cross-Commitment Proof Aggregation)**.** *A polynomial commitment scheme with cross-commitment proof aggregation is a tuple of four PPT algorithms* $(\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda, d) \to \mathsf{srs}$: *A probabilistic algorithm that, given the security parameter* 1^λ *and a degree bound* $d \in \mathbb{N}$*, outputs a structured reference string* srs*.*
- $\mathsf{Commit}(\mathsf{srs}, f) =: c$: *A deterministic algorithm that, given a structured reference string* srs *(with implicit degree bound* d*) and a polynomial* $f \in \mathbb{F}_p^{(\leq d)}[X]$*, outputs a commitment* c *to* f*.*
- $\langle \mathcal{P}(\mathsf{srs}, x, (f_i)_{i\in[k]}), \mathcal{V}(\mathsf{srs}, x, (c_i)_{i\in[k]}, (y_i)_{i\in[k]}) \rangle \to b$: *An interactive public-coin protocol* Open *with common input* srs *and a point* $x \in \mathbb{F}_p$ *between a prover* $\mathcal{P}$ *who has the polynomials* $f_1, \ldots, f_k \in \mathbb{F}_p^{(\leq d)}[X]$ *and a verifier* $\mathcal{V}$ *who is given the alleged commitments* $c_1, \ldots, c_k$ *to these polynomials as well as their alleged evaluations* $y_1, \ldots, y_k \in \mathbb{F}_p$ *at* x*. The purpose of the protocol is to convince* $\mathcal{V}$ *that* $f_i(x) = y_i$ *for all* $i \in [k]$*, i.e., the given polynomial evaluations are correct with respect to their commitments. At the end of the protocol,* $\mathcal{V}$ *outputs a bit* $b \in \{0, 1\}$*, with 1 meaning accept and 0 reject.*

Furthermore, $(\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ has to satisfy the following two properties:

Definition 4.2 (Completeness)**.** *For all* $d, k = \mathsf{poly}(\lambda)$*, all polynomials* $f_1, \ldots, f_k \in \mathbb{F}_p^{(\leq d)}[X]$*, and every point* $x \in \mathbb{F}_p$*:*

$$\Pr\left[\begin{array}{c} \mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, d); \\ (c_1, \ldots, c_k) := (\mathsf{Commit}(\mathsf{srs}, f_1), \ldots, \mathsf{Commit}(\mathsf{srs}, f_k)); \\ b \leftarrow \langle \mathcal{P}(\mathsf{srs}, x, (f_i)_{i\in[k]}), \mathcal{V}(\mathsf{srs}, x, (c_i)_{i\in[k]}, (f_i(x))_{i\in[k]}) \rangle : \\ b = 1 \end{array}\right] = 1.$$

Definition 4.3 (Knowledge Soundness in the AGM)**.** *For all* $d = \mathsf{poly}(\lambda)$ *and every algebraic adversary* $\mathcal{A}$*, there exists a PPT extractor* $\mathcal{E}$ *such that:*

$$\Pr\left[\begin{array}{c} \mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, d); \\ ((c_1, \ldots, c_k, \mathsf{st}_1); (f_1, \ldots, f_k) \in (\mathbb{F}_p^{(\leq d)}[X])^k) \leftarrow (\mathcal{A} \parallel \mathcal{E})(\mathsf{srs}); \\ ((x, y_1, \ldots, y_k) \in \mathbb{F}_p^{k+1}, \mathsf{st}_2) \leftarrow \mathcal{A}(\mathsf{st}_1); \\ b \leftarrow \langle \mathcal{A}(\mathsf{st}_2), \mathcal{V}(\mathsf{srs}, x, (c_i)_{i\in[k]}, (y_i)_{i\in[k]}) \rangle : \\ b = 1 \wedge \exists i \in [k], y_i \neq f_i(x) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Intuitively, the extractor $\mathcal{E}$ should be able to extract correct witness polynomials $f_1, \ldots, f_k \in \mathbb{F}_p^{(\leq d)}[X]$ satisfying $f_i(x) = y_i$ for all $i \in [k]$ from the adversary's commitments $c_1, \ldots, c_k$ whenever the verifier accepts at the end of the protocol run. Since we are in the AGM and thus consider *algebraic* adversaries (as defined in Section 2.9), the extractor does not only get the adversary's randomness, but also a vector of coefficients explaining any group element output by $\mathcal{A}$ as a linear combination of its previously seen group elements. This will greatly ease the construction of an appropriate extractor in the subsequent proofs, where we will be dealing with commitments made up of group elements only.

4.2 The Construction

Extending the KZG scheme from Section 2.8 by an appropriate Open protocol, Gabizon et al. give the following instantiation of this definition [GWC19, Sec. 3.1]:

Construction 4.1: KZG with Cross-Commitment Proof Aggregation.

- $\mathsf{Setup}(1^\lambda, d)$: Choose random $\tau \leftarrow \mathbb{F}_p$ and output

$$\mathsf{srs} := (\mathrm{g}_1, \mathrm{g}_1^\tau, \mathrm{g}_1^{\tau^2}, \ldots, \mathrm{g}_1^{\tau^d}, \mathrm{g}_2, \mathrm{g}_2^\tau).$$

- $\mathsf{Commit}(\mathsf{srs}, f)$: Given $f(X) := \sum_{i=0}^{d} f_i X^i \in \mathbb{F}_p^{(\leq d)}[X]$, output

$$c := \prod_{i=0}^{d} (\mathrm{g}_1^{\tau^i})^{f_i} = \mathrm{g}_1^{f(\tau)}.$$

- $\langle \mathcal{P}(\mathsf{srs}, x, (f_i)_{i\in[k]}), \mathcal{V}(\mathsf{srs}, x, (c_i)_{i\in[k]}, (y_i)_{i\in[k]}) \rangle$:
  1. $\mathcal{V}$ sends a random *opening challenge* $\alpha \leftarrow \mathbb{F}_p$ to $\mathcal{P}$.
  2. $\mathcal{P}$ computes the *aggregated quotient polynomial*

$$Q(X) := \sum_{i\in[k]} \alpha^{i-1} \cdot \frac{f_i(X) - f_i(x)}{X - x},$$

     and sends the commitment $\pi := \mathrm{g}_1^{Q(\tau)}$ to $\mathcal{V}$.
  3. $\mathcal{V}$ accepts iff

$$e\left(\prod_{i\in[k]} (c_i / \mathrm{g}_1^{y_i})^{\alpha^{i-1}}, \mathrm{g}_2\right) \stackrel{?}{=} e(\pi, \mathrm{g}_2^\tau / \mathrm{g}_2^x).$$

4.2.1 Completeness

Completeness of the Open protocol follows directly from the following observation:

$$e\left(\prod_{i\in[k]} (c_i / \mathrm{g}_1^{y_i})^{\alpha^{i-1}}, \mathrm{g}_2\right) = e\left(\pi, \mathrm{g}_2^\tau / \mathrm{g}_2^x\right) \tag{4.1}$$

$$\iff e\left(\prod_{i\in[k]} \mathrm{g}_1^{\alpha^{i-1}(f_i(\tau) - f_i(x))}, \mathrm{g}_2\right) = e\left(\mathrm{g}_1^{Q(\tau)}, \mathrm{g}_2^{\tau - x}\right) \tag{4.2}$$

$$\iff e(\mathrm{g}_1, \mathrm{g}_2)^{\sum_{i\in[k]} \alpha^{i-1}(f_i(\tau) - f_i(x))} = e(\mathrm{g}_1, \mathrm{g}_2)^{Q(\tau)(\tau - x)} \tag{4.3}$$

$$\iff \sum_{i\in[k]} \alpha^{i-1}(f_i(\tau) - f_i(x)) = Q(\tau)(\tau - x) \pmod p \tag{4.4}$$

4.2.2 Knowledge Soundness in the AGM

We will argue knowledge soundness of the Open protocol in the AGM by describing an appropriate PPT extractor $\mathcal{E}$ and analyzing the success probability of any algebraic adversary $\mathcal{A}$ to win the game in Definition 4.3. Unlike the original proof given by Gabizon et al. [GWC19, Sec. 3.1], which relies on a powerful, but informally stated d-DLog-based lemma [GWC19, Lemma 2.2] about interactive protocols in the AGM where the verifier output is determined by a generalization of the pairing check in Construction 4.1, we do this via a direct reduction to the d-DLog assumption in the AGM.

Proof. Let $\mathcal{A}$ be an arbitrary algebraic adversary against the knowledge soundness (as stated in Definition 4.3) of the Open protocol in Construction 4.1. Given srs, $\mathcal{A}$ begins by outputting k commitments $c_1, \ldots, c_k \in \mathbb{G}_1$. Since $\mathcal{A}$ is an algebraic adversary, it also outputs coefficients $f_{i,1}, \ldots, f_{i,d} \in \mathbb{F}_p$ such that $c_i = \prod_{j=0}^{d} (\mathsf{g}_1^{\tau^j})^{f_{i,j}}$ for every $i \in [k]$. We construct a PPT extractor $\mathcal{E}$ which, given these coefficients, simply extracts the k polynomials defined as

$$f_i(X) := \sum_{j=0}^{d} f_{i,j} X^j \in \mathbb{F}_p^{(\leq d)}[X].$$

Moreover, when $\mathcal{A}$ outputs the commitment $\pi \in \mathbb{G}_1$, it must know a representation $q_1, \ldots, q_d \in \mathbb{F}_p$ such that $\pi = \prod_{j=0}^{d} (\mathsf{g}_1^{\tau^j})^{q_j}$, implicitly defining the alleged quotient polynomial

$$Q(X) := \sum_{j=0}^{d} q_j X^j \in \mathbb{F}_p^{(\leq d)}[X].$$

To simplify notation, let

$$F(X) := \sum_{i \in [k]} \alpha^{i-1} (f_i(X) - y_i) \in \mathbb{F}_p^{(\leq d)}[X].$$

Then we can bound the probability that $\mathcal{A}$ wins the security game in Definition 4.3 by:

$$\begin{aligned}
&\Pr[b = 1 \wedge \exists i \in [k], y_i \neq f_i(x)] \\
&\quad = \Pr[b = 1 \wedge \exists i \in [k], y_i \neq f_i(x) \wedge F(X) \equiv Q(X)(X - x)] \\
&\qquad + \Pr[b = 1 \wedge \exists i \in [k], y_i \neq f_i(x) \wedge F(X) \not\equiv Q(X)(X - x)] \qquad (4.5) \\
&\quad \leq \Pr[F(X) \equiv Q(X)(X - x) \mid \exists i \in [k], y_i \neq f_i(x)] \\
&\qquad + \Pr[b = 1 \wedge F(X) \not\equiv Q(X)(X - x)] \qquad (4.6)
\end{aligned}$$

To obtain the first probability in Equation (4.6) we used the fact that $\Pr[A \wedge B] \leq \Pr[A \mid B]$ for any two events A and B with $\Pr[B] \neq 0$. We proceed by showing that

$$\Pr[F(X) \equiv Q(X)(X - x) \mid \exists i \in [k], y_i \neq f_i(x)] \leq \frac{k - 1}{p}, \qquad (4.7)$$

which is negligible in λ. For this purpose, assume there is some $i^* \in [k]$ such that $y_{i^*} \neq f_{i^*}(x)$. Observe that $F(X) \equiv Q(X)(X - x)$ implies $F(x) = 0$ irrespective of $\mathcal{A}$'s

choice of the polynomial Q. This means α is a root of the non-zero (due to $y_{i^*} \neq f_{i^*}(x)$), degree-$(k-1)$ polynomial

$$f(Y) := \sum_{i \in [k]} Y^{i-1}(f_i(x) - y_i),$$

which is the case for at most $k-1$ values of α. Therefore, the probability that $F(x) = 0$, and thus also the probability that $F(X) \equiv Q(X)(X-x)$, is at most $(k-1)/p$, establishing Equation (4.7).

Next, we use a reduction to the d-DLog problem (cf. Definition 2.2) to show that

$$\Pr[b = 1 \wedge F(X) \not\equiv Q(X)(X-x)] \leq \Pr[\mathcal{B} \text{ solves } d\text{-DLog}], \tag{4.8}$$

where $\mathcal{B}$ is a PPT adversary against d-DLog defined as follows:

1. On input $(\mathsf{g}_1, \mathsf{g}_1^{\tau}, \ldots, \mathsf{g}_1^{\tau^d}, \mathsf{g}_2, \mathsf{g}_2^{\tau}, \ldots, \mathsf{g}_2^{\tau^d})$, define $\mathsf{srs} := (\mathsf{g}_1, \mathsf{g}_1^{\tau}, \ldots, \mathsf{g}_1^{\tau^d}, \mathsf{g}_2, \mathsf{g}_2^{\tau})$.
2. Run $\mathcal{A}(\mathsf{srs})$, obtaining $c_1, \ldots, c_k \in \mathbb{G}_1$ and st_1. Since $\mathcal{A}$ is an algebraic adversary, it also outputs coefficients $f_{i,1}, \ldots, f_{i,d} \in \mathbb{F}_p$ such that $c_i = \prod_{j=0}^{d} (\mathsf{g}_1^{\tau^j})^{f_{i,j}}$ for every $i \in [k]$. Define the k polynomials $f_i(X) := \sum_{j=0}^{d} f_{i,j} X^j \in \mathbb{F}_p^{(\leq d)}[X]$.
3. Run $\mathcal{A}(\mathsf{st}_1)$, obtaining $x, y_1, \ldots, y_k \in \mathbb{F}_p$ and st_2.
4. Run the Open protocol $\langle \mathcal{A}(\mathsf{st}_2), \mathcal{B}(\mathsf{srs}, x, (c_i)_{i \in [k]}, (y_i)_{i \in [k]}) \rangle$ with $\mathcal{A}$ and honestly follow the verifier algorithm $\mathcal{V}$ by sending a random $\alpha \leftarrow \mathbb{F}_p$. Obtain a commitment $\pi \in \mathbb{G}_1$ as well as its representation $q_1, \ldots, q_d \in \mathbb{F}_p$ with $\pi = \prod_{j=0}^{d} (\mathsf{g}_1^{\tau^j})^{q_j}$ from $\mathcal{A}$. Define the polynomial $Q(X) := \sum_{j=0}^{d} q_j X^j \in \mathbb{F}_p^{(\leq d)}[X]$.
5. Using the same check as $\mathcal{V}$, i.e., Equation (4.1), compute a bit $b' \in \{0,1\}$, and let $F(X) := \sum_{i \in [k]} \alpha^{i-1}(f_i(X) - y_i) \in \mathbb{F}_p^{(\leq d)}[X]$. If $b' = 1$ but $F(X) \not\equiv Q(X)(X-x)$, factor the non-zero polynomial $F(X) - Q(X)(X-x) \in \mathbb{F}_p^{(\leq d+1)}[X]$ of degree $\leq d+1$ and return a root $\tau' \in \mathbb{F}_p$ with $\mathsf{g}_1^{\tau'} = \mathsf{g}_1^{\tau}$ as the solution to the given d-DLog instance.

Since factoring a polynomial over a finite field $\mathbb{F}_p$ can be done in time polynomial in its degree [vzGG13], $\mathcal{B}$ clearly runs in PPT. Moreover, $\mathcal{B}$ solves d-DLog if and only if $F(\tau) = Q(\tau)(\tau - x)$ and $F(X) \not\equiv Q(X)(X-x)$, which ensures that the non-zero polynomial $F(X) - Q(X)(X-x)$ has a root at $X = \tau$. Note that via the equivalence shown in Equations (4.1) to (4.4), the first of these two conditions corresponds to the case $b' = 1$, i.e., the verifier $\mathcal{V}$ accepting in the Open protocol. Thus, we obtain Equation (4.8). Combined with Equations (4.6) and (4.7), it follows that

$$\Pr[b = 1 \wedge \exists i \in [k], y_i \neq f_i(x)] \leq \frac{k-1}{p} + \Pr[\mathcal{B} \text{ solves } d\text{-DLog}] \tag{4.9}$$

$$\leq \frac{k-1}{p} + \epsilon_{d\text{-DLog}}, \tag{4.10}$$

where $\epsilon_{d\text{-DLog}}$ (with λ implicit) is the hardness of the d-DLog problem, finishing the proof. □

4.3 The Case of Multiple Evaluation Points

In the final PLONK protocol, we will need to evaluate a polynomial not just at a single point but at two different points. The natural way to achieve this is by running the Open protocol of the above scheme multiple times in parallel. To this end, Gabizon et al. include one further optimization to reduce the number of expensive verifier pairing computations. Naively, the verifier needs to evaluate two pairings per run of the Open protocol. Suppose we want to evaluate the polynomials $f_1, \ldots, f_k$ at the ℓ points $x_1, \ldots, x_\ell$. For the i-th run of the protocol, let $y_{i,1}, \ldots, y_{i,k}$ denote the respective polynomial evaluations at x_i. Also, define $C_i := \prod_{j\in[k]} (c_j / \mathrm{g}_1^{y_{i,j}})^{\alpha_i^{j-1}}$ to simplify notation. Then the i-th pairing check (see Equation (4.1)) can be rewritten as:

$$e\left(\prod_{j\in[k]} (c_j/\mathrm{g}_1^{y_{i,j}})^{\alpha_i^{j-1}}, \mathrm{g}_2\right) = e(\pi_i, \mathrm{g}_2^\tau / \mathrm{g}_2^{x_i}) \tag{4.11}$$

$$\iff \quad e(C_i, \mathrm{g}_2) = e(\pi_i, \mathrm{g}_2^\tau)/e(\pi_i, \mathrm{g}_2^{x_i}) \tag{4.12}$$

$$\iff \quad e(C_i \cdot \pi_i^{x_i}, \mathrm{g}_2) = e(\pi_i, \mathrm{g}_2^\tau) \tag{4.13}$$

This enables the following method for batch randomized evaluation of pairing equations. Instead of checking the system of ℓ pairing equations

$$\begin{aligned} e(C_1 \cdot \pi_1^{x_1}, \mathrm{g}_2) &= e(\pi_1, \mathrm{g}_2^\tau) \\ &\vdots \\ e(C_\ell \cdot \pi_\ell^{x_\ell}, \mathrm{g}_2) &= e(\pi_\ell, \mathrm{g}_2^\tau) \end{aligned} \tag{4.14}$$

one by one, the verifier only checks a single randomized pairing equation of the form

$$e\left(\prod_{i\in[\ell]} (C_i \cdot \pi_i^{x_i})^{\zeta^{i-1}}, \mathrm{g}_2\right) = e\left(\prod_{i\in[\ell]} \pi_i^{\zeta^{i-1}}, \mathrm{g}_2^\tau\right) \tag{4.15}$$

over uniformly chosen $\zeta \in \mathbb{F}_p$. Looking at this as a polynomial equation in $X = \zeta$, it follows from the Schwartz–Zippel lemma (Lemma 2.1) that, if any of the original equalities does not hold, then this new check holds with probability at most $(\ell - 1)/p$ over the choice of ζ; in other words, it fails with overwhelming probability.

To analyze the knowledge soundness error of the resulting protocol, let us first consider the variant without the randomized verifier check, where $b = 1$ denotes the event that all ℓ verifier checks in Equation (4.14) pass. Then we have:

$$\begin{aligned} &\Pr[b = 1 \wedge \exists i \in [\ell], j \in [k], y_{i,j} \neq f_j(x_i)] \\ &\quad = \Pr[\ldots \wedge \forall i' \in [\ell], F_{i'}(X) \equiv Q_{i'}(X)(X - x_{i'})] \\ &\qquad + \Pr[\ldots \wedge \exists i' \in [\ell], F_{i'}(X) \not\equiv Q_{i'}(X)(X - x_{i'})] \qquad (4.16) \\ &\quad \leq \Pr[\forall i' \in [\ell], F_{i'}(X) \equiv Q_{i'}(X)(X - x_{i'}) \mid \exists i \in [\ell], j \in [k], y_{i,j} \neq f_j(x_i)] \\ &\qquad + \Pr[b = 1 \wedge \exists i \in [\ell], F_i(X) \not\equiv Q_i(X)(X - x_i)] \qquad (4.17) \end{aligned}$$

$$\leq \Pr\big[F_i(X) \equiv Q_i(X)(X - x_i) \mid \exists i \in [\ell], j \in [k], y_{i,j} \neq f_j(x_i)\big]$$
$$+ \Pr\big[b = 1 \wedge \exists i \in [\ell], F_i(X) \not\equiv Q_i(X)(X - x_i)\big] \tag{4.18}$$
$$\leq \frac{k-1}{p} + \epsilon_{d\text{-DLog}} \tag{4.19}$$

Using an analogous analysis as in the proof with just a single evaluation point $x \in \mathbb{F}_p$ from Section 4.2.2, we are actually able to obtain the same upper bound. So overall, including the randomized verifier check from Equation (4.15), the general variant of the Open protocol is knowledge sound (in the AGM) with probability

$$\Pr\big[b = 1 \wedge \exists i \in [\ell], j \in [k], y_{i,j} \neq f_j(x_i)\big] \leq \frac{k-1}{p} + \frac{\ell-1}{p} + \epsilon_{d\text{-DLog}}, \tag{4.20}$$

where $\epsilon_{d\text{-DLog}}$ is the hardness of the d-DLog problem.

As previously mentioned, the PLONK construction requires the opening of polynomial commitments at no more than two points. For this reason, we describe the general Open protocol only in the case of two distinct evaluation points $x_1, x_2 \in \mathbb{F}_p$ and the index sets $I_1 := \{i_{1,1}, \ldots, i_{1,k_1}\}, I_2 := \{i_{2,1}, \ldots, i_{2,k_2}\}$ with $I_1 \cup I_2 = [k]$, denoting which of the k polynomials are evaluated at x_1 and/or x_2, respectively:

Construction 4.2: Open Protocol with 2 Points.

$\langle \mathcal{P}(\mathsf{srs}, x_1, x_2, I_1, I_2, (f_j)_{j\in[k]}), \mathcal{V}(\mathsf{srs}, x_1, x_2, I_1, I_2, (c_j)_{j\in[k]}, (y_{1,j})_{j\in[k_1]}, (y_{2,j})_{j\in[k_2]})\rangle$:

1. $\mathcal{V}$ sends two random opening challenges $\alpha_1, \alpha_2 \leftarrow \mathbb{F}_p$ to $\mathcal{P}$.
2. $\mathcal{P}$ computes the aggregated quotient polynomials

$$Q_1(X) := \sum_{j\in[k_1]} \alpha_1^{j-1} \cdot \frac{f_{i_{1,j}}(X) - f_{i_{1,j}}(x_1)}{X - x_1},$$
$$Q_2(X) := \sum_{j\in[k_2]} \alpha_2^{j-1} \cdot \frac{f_{i_{2,j}}(X) - f_{i_{2,j}}(x_2)}{X - x_2},$$

   and sends the commitments $\pi_1 := \mathsf{g}_1^{Q_1(\tau)}$, $\pi_2 := \mathsf{g}_1^{Q_2(\tau)}$ to $\mathcal{V}$.
3. $\mathcal{V}$ chooses a random *multipoint evaluation challenge* $\zeta \leftarrow \mathbb{F}_p$, computes

$$C_1 := \prod_{j\in[k_1]} (c_{i_{1,j}} / \mathsf{g}_1^{y_{1,j}})^{\alpha_1^{j-1}}, \quad C_2 := \prod_{j\in[k_2]} (c_{i_{2,j}} / \mathsf{g}_1^{y_{2,j}})^{\alpha_2^{j-1}},$$

   and accepts iff

$$e\Big(C_1 \cdot \pi_1^{x_1} \cdot (C_2 \cdot \pi_2^{x_2})^{\zeta}, \mathsf{g}_2\Big) \stackrel{?}{=} e(\pi_1 \cdot \pi_2^{\zeta}, \mathsf{g}_2^{\tau}).$$

CHAPTER 5

Polynomial Protocols

In this chapter, we present an iterative series of protocols for proving statements about polynomials until obtaining the actual protocol used in PLONK. As a central building block, we rely on polynomial commitments and the interactive Open protocol from the previous chapter to achieve the subsequent constructions.

Since completeness of all the protocols presented in this chapter follows by inspection, we do not argue it explicitly. We also do not discuss whether these protocols are zero-knowledge, as this will be achieved through a specific modification of the main PLONK protocol described in Chapter 7.

5.1 Checking Polynomial Identities

As a first building block, we will consider a succinct protocol for proving polynomial identities. For this purpose, assume there is a set of ℓ public polynomials $f_1, \ldots, f_\ell \in \mathbb{F}_p[X]$ and a set of t private polynomials $f_{\ell+1}, \ldots, f_{\ell+t} \in \mathbb{F}_p[X]$, between which we want to show that certain identities hold. For example, given two public polynomials f_1, f_2, we might want to show that f_2 evenly divides f_1 by proving the identity $f_1(X) \equiv f_2(X) \cdot f_3(X)$ for some private witness polynomial f_3. Rewritten as $f_1(X) - f_2(X) \cdot f_3(X) \equiv 0$, this can be seen as the following polynomial identity:

$$F(X) := G(f_1(X), f_2(X), f_3(X)) \equiv 0,$$

where $G \in \mathbb{F}_p[X_1, X_2, X_3]$ is the multivariate polynomial $G(X_1, X_2, X_3) := X_1 - X_2 \cdot X_3$.

In general, we want to prove k polynomial identities of the form

$$F_i(X) := G_i(X, f_{\mu(i,1)}(v_{i,1}(X)), \ldots, f_{\mu(i,M_i)}(v_{i,M_i}(X))) \equiv 0,$$

where for each $i \in [k]$, there are $M_i = \mathsf{poly}(\lambda)$ polynomials $f_{\mu(i,1)}, \ldots, f_{\mu(i,M_i)}$ selected from $\{f_1, \ldots, f_{\ell+t}\}$ according to a mapping $\mu\colon \{(i,j)\}_{i\in[k],j\in[M_i]} \to [\ell + t]$. Moreover,

the reason for not directly evaluating these polynomials at X, but instead introducing the input selecting polynomials $v_{i,1}, \ldots, v_{i,M_i} \in \mathbb{F}_p[X]$, is to also capture polynomial identities such as $f_1(X) - f_2(X+1) \equiv 0$.

Formally, let $d, D, t, \ell, k = \mathsf{poly}(\lambda)$ and define the following indexed relation to capture this scenario:

$$\mathcal{R}_{(d,D,t,\ell)} := \left\{ (\mathbf{i}, \mathbf{x}, \mathbf{w}) \;\middle|\; \begin{array}{l} \mathbf{i} = \left(\begin{array}{c} (G_1, \ldots, G_k) \in \bigtimes_{i\in[k]} \mathbb{F}_p^{(\leq D)}[X, X_1, \ldots, X_{M_i}], \\ (v_{i,1}, \ldots, v_{i,M_i})_{i\in[k]} \in \bigtimes_{i\in[k]} (\mathbb{F}_p^{(\leq d)}[X])^{M_i}, \\ \mu \colon \{(i,j)\}_{i\in[k], j\in[M_i]} \to [\ell + t] \end{array} \right), \\ \mathbf{x} = (f_1, \ldots, f_\ell) \in (\mathbb{F}_p^{(\leq d)}[X])^{\ell}, \\ \mathbf{w} = (f_{\ell+1}, \ldots, f_{\ell+t}) \in (\mathbb{F}_p^{(\leq d)}[X])^{t}, \\ \forall i \in [k] \colon G_i(X, f_{\mu(i,1)}(v_{i,1}(X)), \ldots, f_{\mu(i,M_i)}(v_{i,M_i}(X))) \equiv 0 \end{array} \right\}$$

With this, we can finally describe the protocol introduced by Gabizon et al. for proving statements in $\mathcal{R}_{(d,D,t,\ell)}$. It relies on the Open protocol from Construction 4.2, or more precisely, its generalization to an arbitrary amount of evaluation points as described at the beginning of Section 4.3.

Construction 5.1: The (d, D, t, ℓ)-Polynomial Protocol.

- Structured reference string: $\mathsf{srs} := (\mathsf{g}_1, \mathsf{g}_1^{\tau}, \mathsf{g}_1^{\tau^2}, \ldots, \mathsf{g}_1^{\tau^d}, \mathsf{g}_2, \mathsf{g}_2^{\tau})$.
- Preprocessed input: The public polynomials $f_1, \ldots, f_\ell \in \mathbb{F}_p^{(\leq d)}[X]$ and their commitments $(c_1, \ldots, c_\ell) := (\mathsf{g}_1^{f_1(\tau)}, \ldots, \mathsf{g}_1^{f_\ell(\tau)})$.
- $\langle \mathcal{P}(\mathsf{srs}, \mathbf{i}, (f_i)_{i\in[\ell+t]}), \mathcal{V}(\mathsf{srs}, \mathbf{i}, (c_i)_{i\in[\ell]}) \rangle$:
  1. $\mathcal{P}$ commits to its polynomials $f_{\ell+1}, \ldots, f_{\ell+t} \in \mathbb{F}_p^{(\leq d)}[X]$ and sends the commitments $(c_{\ell+1}, \ldots, c_{\ell+t}) := (\mathsf{g}_1^{f_{\ell+1}(\tau)}, \ldots, \mathsf{g}_1^{f_{\ell+t}(\tau)})$ to $\mathcal{V}$.
  2. $\mathcal{V}$ chooses a random *evaluation challenge* $\delta \leftarrow \mathbb{F}_p$, computes all the unique evaluation points in $(v_{i,1}(\delta), \ldots, v_{i,M_i}(\delta))_{i\in[k]}$, and sends δ to $\mathcal{P}$.
  3. $\mathcal{P}$ replies with all the unique evaluations contained in $(y_{i,1}, \ldots, y_{i,M_i})_{i\in[k]} := (f_{\mu(i,1)}(v_{i,1}(\delta)), \ldots, f_{\mu(i,M_i)}(v_{i,M_i}(\delta)))_{i\in[k]}$.
  4. Using the Open protocol, $\mathcal{V}$ verifies the correctness of the evaluations received from $\mathcal{P}$ with respect to the commitments $c_1, \ldots, c_{\ell+t}$.
  5. $\mathcal{V}$ accepts iff $G_i(\delta, y_{i,1}, \ldots, y_{i,M_i}) \stackrel{?}{=} 0$ for all $i \in [k]$.

5.1.1 Knowledge Soundness in the AGM

We prove that the protocol from Construction 5.1 is knowledge sound in the AGM by showing existence of an appropriate extractor $\mathcal{E}$ such that Definition 2.6 is satisfied.

Proof. We first argue knowledge soundness of the (d, D, t, ℓ)-polynomial protocol in the case where $\mathcal{V}$ only checks a single polynomial identity

$$F(X) := G(X, f_1(v_1(X)), \ldots, f_M(v_M(X))) \equiv 0.$$

When the algebraic adversary $\mathcal{A}$ outputs the commitments $c_{\ell+1}, \ldots, c_{\ell+t} \in \mathbb{G}_1$, it also returns coefficients $f_{\ell+i,1}, \ldots, f_{\ell+i,d} \in \mathbb{F}_p$ such that $c_{\ell+i} = \prod_{j=0}^{d} (\mathsf{g}_1^{\tau^j})^{f_{\ell+i,j}}$ for every $i \in [t]$, implicitly defining the polynomials $f_{\ell+i}(X) := \sum_{j=0}^{d} f_{\ell+i,j} X^j \in \mathbb{F}_p^{(\leq d)}[X]$. Since the extractor is given $\mathcal{A}$'s randomness, we construct a PPT extractor $\mathcal{E}$ which directly extracts the polynomials $f_{\ell+1}, \ldots, f_{\ell+t} \in \mathbb{F}_p^{(\leq d)}[X]$ from $\mathcal{A}$'s commitments as the witness.

From this point on, we will show that the probability that the verifier accepts but $F(X) \not\equiv 0$ is negligible. Let $A \wedge B$ denote the event in the Open protocol that

- $\mathcal{V}$ accepts $\mathcal{P}$'s alleged evaluations $y_1, \ldots, y_M$ (subevent A),
- while $y_{i^*} \neq f_{i^*}(v_{i^*}(\delta))$ for some $i^* \in [M]$ (subevent B).

By the knowledge soundness of the Open protocol, we have

$$\Pr[A \wedge B] := \epsilon_{\mathsf{Open}} \leq \mathsf{negl}(\lambda). \tag{5.1}$$

Via the Schwartz–Zippel lemma, it follows that for a random $\delta \in \mathbb{F}_p$

$$\Pr[F(\delta) = 0 \mid F(X) \not\equiv 0] \leq \frac{\deg F}{p}. \tag{5.2}$$

With this, we can bound the probability that $\mathcal{V}$ accepts but $F(X) \not\equiv 0$ to finish the proof:

$$\begin{aligned}
\Pr[\mathcal{V} \text{ accepts} \wedge F(X) \not\equiv 0] &= \Pr[(A \wedge G(\delta, y_1, \ldots, y_M) = 0) \wedge F(X) \not\equiv 0] && (5.3)\\
&= \Pr[A \wedge G(\delta, y_1, \ldots, y_M) = 0 \wedge F(X) \not\equiv 0 \wedge B] \\
&\quad + \Pr[A \wedge G(\delta, y_1, \ldots, y_M) = 0 \wedge F(X) \not\equiv 0 \wedge \bar{B}] && (5.4)\\
&\leq \Pr[A \wedge B] + \Pr[F(\delta) = 0 \wedge F(X) \not\equiv 0] && (5.5)\\
&\leq \Pr[A \wedge B] + \Pr[F(\delta) = 0 \mid F(X) \not\equiv 0] && (5.6)\\
&\leq \epsilon_{\mathsf{Open}} + \frac{\deg F}{p} && (5.7)\\
&\leq \epsilon_{\mathsf{Open}} + \frac{d^2 D}{p} && (5.8)
\end{aligned}$$

To obtain the second probability in Equation (5.5), we used the fact that $\bar{B}$, i.e., $y_i = f_i(v_i(\delta))$ for all $i \in [M]$, and $G(\delta, y_1, \ldots, y_M) = 0$ implies $F(\delta) = 0$. Note that the overall probability is negligible in λ.

It is straightforward to adapt this analysis (and the event $A \wedge B$) to the case of checking multiple polynomial identities $F_i(X) \equiv 0$ for $i \in [k]$ in a single run of the protocol, obtaining the same upper bound for knowledge soundness:

$$\begin{aligned}
&\Pr[\mathcal{V} \text{ accepts} \wedge \exists i^* \in [k], F_{i^*}(X) \not\equiv 0] \\
&\quad \leq \Pr[A \wedge B] + \Pr[\forall i \in [k], F_i(\delta) = 0 \wedge \exists i^* \in [k], F_{i^*}(X) \not\equiv 0] && (5.9)\\
&\quad \leq \Pr[A \wedge B] + \Pr[\forall i \in [k], F_i(\delta) = 0 \mid \exists i^* \in [k], F_{i^*}(X) \not\equiv 0] && (5.10)\\
&\quad \leq \Pr[A \wedge B] + \Pr[F_{i^*}(\delta) = 0 \mid \exists i^* \in [k], F_{i^*}(X) \not\equiv 0] && (5.11)\\
&\quad \leq \epsilon_{\mathsf{Open}} + \frac{\max_{i\in[k]} \deg F_i}{p} && (5.12)\\
&\quad \leq \epsilon_{\mathsf{Open}} + \frac{d^2 D}{p} && (5.13)
\end{aligned}$$

□

5.2 Ranged Polynomial Protocol

We next look into the actual primitive needed for proving statements in the PLONK constraint system from Chapter 3. Let $S \subset \mathbb{F}_p$ be a subset of the field $\mathbb{F}_p$ of size $|S| \leq d^2 D = \mathsf{poly}(\lambda)$ and define the *vanishing polynomial* $Z_S(X) := \prod_{a\in S}(X - a)$, whose roots are exactly given by S. Instead of proving polynomial identities of the form

$$F(X) := G(X, f_1(v_1(X)), \ldots, f_M(v_M(X))) \equiv 0,$$

we want to prove that

$$F(a) = G(a, f_1(v_1(a)), \ldots, f_M(v_M(a))) = 0 \text{ for all } a \in S,$$

which we call an *S-ranged polynomial identity*. Note that this is equivalent to F being divisible by Z_S. Thus, one can trivially extend the protocol in the previous section to obtain an S-ranged polynomial protocol by additionally making the prover commit to the quotient polynomial T obtained from dividing F by Z_S. Security of the resulting protocol follows directly from the polynomial protocol, because if the polynomial identity $F(X) - T(X) \cdot Z_S(X) \equiv 0$ holds for some $T \in \mathbb{F}_p[X]$, then $F(a) = 0$ for all $a \in S$.

In the case of checking only a single ranged polynomial identity, the required communication overhead from committing to the resulting quotient polynomial consists of one group element, which can be considered optimal. However, using this approach to prove multiple polynomial identities over the same range S results in a communication overhead linear in the number of identities. To address this issue, Gabizon et al. introduce the following lemma [GWC19, Claim 4.6], enabling a batch randomization approach that reduces the overhead to a single commitment independent of the number of S-ranged identities.

Lemma 5.1 (Claim 4.6)**.** *Let $F_1, \ldots, F_k \in \mathbb{F}_p[X]$, $S \subset \mathbb{F}_p$, and $Z_S(X) := \prod_{a \in S}(X - a)$. Define the polynomial $F(X) := \sum_{i \in [k]} \alpha^{i-1} \cdot F_i(X)$ over a random $\alpha \in \mathbb{F}_p$. Then*

$$\Pr\left[Z_S \mid F \;\middle|\; \exists i^* \in [k], Z_S \nmid F_{i^*}\right] \leq \frac{k-1}{p}$$

over the choice of α.

Proof. Assume $Z_S \nmid F_{i^*}$ for some $i^* \in [k]$. We can write $F_{i^*} = Q \cdot Z_S + R$, where Q is the quotient polynomial and R is the non-zero remainder polynomial of degree $< \deg Z_S$. Let $x \in S$ be such that $Z_S(x) = 0$ but $R(x) \neq 0$ (such an x must exist, since $Z_S \nmid R$). Then $Z_S \mid F$ implies $F(x) = 0$, giving us

$$F(x) = \sum_{i \in [k], i \neq i^*} \alpha^{i-1} \cdot F_i(x) + \alpha^{i^*-1} \cdot R(x) = 0. \tag{5.14}$$

This means α must be a root of the non-zero polynomial

$$f(Y) := \sum_{i \in [k], i \neq i^*} Y^{i-1} \cdot F_i(x) + Y^{i^*-1} \cdot R(x), \tag{5.15}$$

which is the case for at most $k - 1$ values of α, finishing the proof. □

Let $\hat{d} := \max(d, |S|, d^2D - |S|)$. Formally, the S-ranged protocol [GWC19, Sec. 4.2] described in Construction 5.2 allows to prove statements in the following relation:

$$\mathcal{R}^S_{(d,D,t,\ell)} := \left\{ (\mathbf{i}, \mathbf{x}, \mathbf{w}) \;\middle|\; \begin{array}{l} \mathbf{i} = \left(\begin{array}{c} (G_1, \ldots, G_k) \in \bigtimes_{i \in [k]} \mathbb{F}_p^{(\leq D)}[X, X_1, \ldots, X_{M_i}], \\ (v_{i,1}, \ldots, v_{i,M_i})_{i \in [k]} \in \bigtimes_{i \in [k]} (\mathbb{F}_p^{(\leq d)}[X])^{M_i}, \\ \mu \colon \{(i,j)\}_{i \in [k], j \in [M_i]} \to [\ell + t] \end{array} \right), \\ \mathbf{x} = (f_1, \ldots, f_\ell) \in (\mathbb{F}_p^{(\leq d)}[X])^\ell, \\ \mathbf{w} = (f_{\ell+1}, \ldots, f_{\ell+t}) \in (\mathbb{F}_p^{(\leq \hat{d})}[X])^t, \\ \forall i \in [k] \colon \forall a \in S, F_i(a) = 0 \wedge \deg F_i \leq \hat{d} + |S|, \text{where} \\ F_i(X) := G_i(X, f_{\mu(i,1)}(v_{i,1}(X)), \ldots, f_{\mu(i,M_i)}(v_{i,M_i}(X))) \end{array} \right\}$$

The notable difference to the previous relation is that the witness polynomials $f_1, \ldots, f_t$ are now allowed to be of degree up to $\hat{d} := \max(d, |S|, d^2D - |S|)$ as opposed to just d. The reason for this is that we will be committing to Z_S and the quotient polynomial $T := (\sum_{i \in [k]} \alpha^{i-1} \cdot F_i)/Z_S$ over random $\alpha \in \mathbb{F}_p$, which requires an SRS of degree at least $|S|$ and $d^2D - |S|$, respectively, allowing the prover to commit to witness polynomials of the same degree. This, however, can potentially create a situation where the degree of the quotient polynomial lies outside the range of the SRS, i.e, is greater than $\hat{d}$. To obtain a complete protocol with respect to $\mathcal{R}^S_{(d,D,t,\ell)}$, we have the additional condition that $\deg F_i \leq \hat{d} + |S|$ for all $i \in [k]$, ensuring that $\deg T \leq \hat{d}$ no matter the value of α.

Construction 5.2: S-Ranged (d, D, t, ℓ)-Polynomial Protocol.

- Structured reference string: $\mathsf{srs} := (\mathsf{g}_1, \mathsf{g}_1^{\tau}, \mathsf{g}_1^{\tau^2}, \ldots, \mathsf{g}_1^{\tau^{\max(d,|S|,d^2D-|S|)}}, \mathsf{g}_2, \mathsf{g}_2^{\tau})$.
- Preprocessed input: The public polynomials $f_1, \ldots, f_\ell \in \mathbb{F}_p^{(\leq d)}[X]$, the polynomial $Z_S \in \mathbb{F}_p^{(\leq |S|)}[X]$, and their commitments $(c_1, \ldots, c_\ell, c_Z) := (\mathsf{g}_1^{f_1(\tau)}, \ldots, \mathsf{g}_1^{f_\ell(\tau)}, \mathsf{g}_1^{Z_S(\tau)})$.
- $\langle \mathcal{P}(\mathsf{srs}, \mathbf{i}, (f_i)_{i\in[\ell+t]}), \mathcal{V}(\mathsf{srs}, \mathbf{i}, (c_i)_{i\in[\ell]}, c_Z)\rangle$:
  1. $\mathcal{P}$ commits to its polynomials $f_{\ell+1}, \ldots, f_{\ell+t} \in \mathbb{F}_p^{(\leq d)}[X]$ and sends the commitments $(c_{\ell+1}, \ldots, c_{\ell+t}) := (\mathsf{g}_1^{f_{\ell+1}(\tau)}, \ldots, \mathsf{g}_1^{f_{\ell+t}(\tau)})$ to $\mathcal{V}$.
  2. $\mathcal{V}$ sends a random *quotient challenge* $\alpha \leftarrow \mathbb{F}_p$ to $\mathcal{P}$.
  3. $\mathcal{P}$ computes the quotient polynomial $T(X) := \frac{\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(X)}{Z_S(X)} \in \mathbb{F}_p^{(\leq d^2D-|S|)}[X]$ and sends the commitment $c_T := \mathsf{g}_1^{T(\tau)}$ to $\mathcal{V}$.
  4. $\mathcal{V}$ chooses a random evaluation challenge $\delta \leftarrow \mathbb{F}_p$, computes all the unique evaluation points in $(v_{i,1}(\delta), \ldots, v_{i,M_i}(\delta))_{i\in[k]}$, and sends δ to $\mathcal{P}$.
  5. $\mathcal{P}$ replies with $y_T := T(\delta)$, $y_Z := Z_S(\delta)$, and all the unique evaluations in $(y_{i,1}, \ldots, y_{i,M_i})_{i\in[k]} := (f_{\mu(i,1)}(v_{i,1}(\delta)), \ldots, f_{\mu(i,M_i)}(v_{i,M_i}(\delta)))_{i\in[k]}$.
  6. Using the Open protocol, $\mathcal{V}$ verifies the correctness of the evaluations received from $\mathcal{P}$ with respect to the commitments $c_1, \ldots, c_{\ell+t}, c_T, c_Z$.
  7. $\mathcal{V}$ accepts iff $\sum_{i\in[k]} \alpha^{i-1} \cdot G_i(\delta, y_{i,1}, \ldots, y_{i,M_i}) \stackrel{?}{=} y_T \cdot y_Z$.

5.2.1 Knowledge Soundness in the AGM

Proof. This proof follows the same structure as the proof that the (d, D, t, ℓ)-polynomial protocol is knowledge sound given in Section 5.1.1, which is why we only discuss the adapted parts.

The first notable difference is the size of the SRS; instead of only allowing commitments up to degree-d polynomials, the new SRS is of degree $\hat{d} := \max(d, |S|, d^2D - |S|)$ depending on the choice of d, D, and $|S|$. This means a malicious prover $\mathcal{A}$ could commit to witness polynomials of degree $> d$, which is accounted for in the definition of $\mathcal{R}^S_{(d,D,t,\ell)}$. After the extractor $\mathcal{E}$ outputs $f_1, \ldots, f_t \in \mathbb{F}_p^{(\leq \hat{d})}[X]$ as $\mathcal{A}$'s witness, we will instead bound the probability that the verifier accepts but there is an $i^* \in [k]$ such that $Z_S \nmid F_{i^*}$ (which is equivalent to $F_{i^*}(a) \neq 0$ for some $a \in S$) or there is a $j^* \in [k]$ such that $\deg F_{j^*} > \hat{d} + |S|$. Let A and B stand for the same events as before, albeit adjusted with respect to the new protocol. Furthermore, let C denote the event that $\deg F_{j^*} > \hat{d} + |S|$ for some $j^* \in [k]$, and let E denote that $\deg(\sum_{i\in[k]} \alpha^{i-1} \cdot F_i) > \hat{d} + |S|$. Also, since we are in the AGM, let $T \in \mathbb{F}_p^{(\leq \hat{d})}[X]$ be the alleged quotient polynomial committed to by $\mathcal{A}$.

We begin by proving the following claim, bounding a certain probability over the choice

of $\mathcal{V}$'s quotient challenge $\alpha \in \mathbb{F}_p$, which is used later in the proof.

Claim 5.1. *Over the choice of $\alpha \in \mathbb{F}_p$, we have*

$$\Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge C \wedge Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i\Big] \leq \frac{\max(\hat{d}dD, \hat{d}+|S|)+k-1}{p}.$$

Proof. We derive this upper bound as follows:

$$\begin{aligned}
&\Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge C \wedge Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i\Big] \\
&\quad = \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge C \wedge Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i \wedge E\Big] \\
&\qquad + \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge C \wedge Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i \wedge \bar{E}\Big] && (5.16) \\
&\quad \leq \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge \sum_{i\in[k]} \alpha^{i-1} \cdot F_i(X) \not\equiv T(X) \cdot Z_S(X)\Big] \\
&\qquad + \Pr\Big[C \wedge Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i \wedge \bar{E}\Big] && (5.17) \\
&\quad \leq \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \;\Big|\; \sum_{i\in[k]} \alpha^{i-1} \cdot F_i(X) \not\equiv T(X) \cdot Z_S(X)\Big] \\
&\qquad + \Pr\Big[C \;\Big|\; Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i \wedge \bar{E}\Big] && (5.18) \\
&\quad \leq \frac{\deg(\sum_{i\in[k]} \alpha^{i-1} \cdot F_i - T \cdot Z_S)}{p} + \frac{k-1}{p} && (5.19) \\
&\quad \leq \frac{\max(\hat{d}dD, \hat{d}+|S|)+k-1}{p} && (5.20)
\end{aligned}$$

To obtain the first probability in Equation (5.17), we used the fact that E implies $\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(X) \not\equiv T(X) \cdot Z_S(X)$ for any $T \in \mathbb{F}_p^{(\leq \hat{d})}[X]$. To derive the second probability in Equation (5.19), i.e.,

$$\Pr\Big[C \;\Big|\; Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i \wedge \bar{E}\Big] \leq \frac{k-1}{p}, \tag{5.21}$$

write $\alpha^{i-1} \cdot F_i = Q_i \cdot Z_S + R_i$ for each $i \in [k]$, where Q_i and R_i are the respective quotient and remainder polynomials such that $\deg R_i < \deg Z_S$. Then

$$\sum_{i\in[k]} \alpha^{i-1} \cdot F_i = \sum_{i\in[k]} (Q_i \cdot Z_S + R_i) = Z_S \cdot \sum_{i\in[k]} Q_i + \sum_{i\in[k]} R_i. \tag{5.22}$$

Since Z_S divides $\sum_{i\in[k]} \alpha^{i-1} \cdot F_i$ and $\deg(\sum_{i\in[k]} R_i) < \deg Z_S$, it must be the case that $\sum_{i\in[k]} R_i \equiv 0$. Considering what happens if the event C also applies, we see that

$$\deg\Big(\textstyle\sum_{i\in[k]} Q_i\Big) = \deg\Big((\sum_{i\in[k]} \alpha^{i-1} \cdot F_i)/Z_S\Big) \leq \hat{d} < \deg Q_{j^*}, \tag{5.23}$$

where the second-to-last inequality follows from $\bar{E}$. Next, assuming w.l.o.g. that Q_{j^*} has the largest degree, let us write $Q_i(X) = \alpha^{i-1} \cdot \sum_{j=0}^{\deg Q_{j^*}} q_{i,j} X^j$ for every $i \in [k]$. Then, for $\deg(\sum_{i\in[k]} Q_i) \leq \hat{d}$, the following $\deg(Q_{j^*}) - \hat{d}$ equations have to hold over the randomness of $\alpha \in \mathbb{F}_p$:

$$\begin{array}{lllllllll}
q_{1,\hat{d}+1} & + & \alpha \cdot q_{2,\hat{d}+1} & + & \cdots & + & \alpha^{k-1} \cdot q_{k,\hat{d}+1} & = & 0 \\
q_{1,\hat{d}+2} & + & \alpha \cdot q_{2,\hat{d}+2} & + & \cdots & + & \alpha^{k-1} \cdot q_{k,\hat{d}+2} & = & 0 \\
\vdots & & \vdots & & \vdots & & & \vdots & \\
q_{1,\deg Q_{j^*}} & + & \alpha \cdot q_{2,\deg Q_{j^*}} & + & \cdots & + & \alpha^{k-1} \cdot q_{k,\deg Q_{j^*}} & = & 0
\end{array}$$

In either case, at least the final equation has to be satisfied, which again can be seen as a polynomial equation in $X = \alpha$ and bounded by the probability $(k-1)/p$, since $q_{j^*,\deg Q_{j^*}} \neq 0$. □

With this, we are ready to bound our actual probability of interest:

$$\begin{aligned}
&\Pr[\mathcal{V} \text{ accepts} \wedge (\mathbf{i}, \mathbf{x}, \mathbf{w}) \notin \mathcal{R}^S_{(d,D,t,\ell)}] \\
&= \Pr\Big[(A \wedge \sum_{i\in[k]} \alpha^{i-1} \cdot G_i(\delta, y_{i,1}, \ldots, y_{i,M_i}) = y_T \cdot y_Z) \wedge (\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C)\Big] && (5.24) \\
&= \Pr\Big[A \wedge \sum_{i\in[k]} \alpha^{i-1} \cdot G_i(\delta, y_{i,1}, \ldots, y_{i,M_i}) = y_T \cdot y_Z \wedge (\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C) \wedge B\Big] \\
&\quad + \Pr\Big[A \wedge \sum_{i\in[k]} \alpha^{i-1} \cdot G_i(\delta, y_{i,1}, \ldots, y_{i,M_i}) = y_T \cdot y_Z \wedge (\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C) \wedge \bar{B}\Big] \\
& && (5.25) \\
&\leq \Pr[A \wedge B] + \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge (\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C)\Big] && (5.26)
\end{aligned}$$

The second probability in Equation (5.26) follows from the fact that the event $\bar{B}$ and $\sum_{i\in[k]} \alpha^{i-1} \cdot G_i(\delta, y_{i,1}, \ldots, y_{i,M_i}) = y_T \cdot y_Z$ implies $\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta)$.

Continuing, we arrive at the following final upper bound:

$$\Pr[A \wedge B] + \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge \big(\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C\big)\Big]$$

$$\begin{aligned}
&= \Pr[A \wedge B] \\
&\quad + \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge \big(\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C\big) \wedge Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i\Big] \\
&\quad + \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge \big(\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C\big) \wedge Z_S \nmid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i\Big]
\end{aligned} \tag{5.27}$$

$$\begin{aligned}
&\leq \Pr[A \wedge B] \\
&\quad + \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge \exists i^* \in [k], Z_S \nmid F_{i^*} \wedge Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i\Big] \\
&\quad + \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge C \wedge Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i\Big] \\
&\quad + \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge \big(\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C\big) \wedge Z_S \nmid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i\Big]
\end{aligned} \tag{5.28}$$

$$\begin{aligned}
&\leq \Pr[A \wedge B] \\
&\quad + \Pr\Big[Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i \;\Big|\; \exists i^* \in [k], Z_S \nmid F_{i^*}\Big] \\
&\quad + \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge C \wedge Z_S \mid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i\Big] \\
&\quad + \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \;\Big|\; Z_S \nmid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i\Big]
\end{aligned} \tag{5.29}$$

$$\leq \epsilon_{\mathsf{Open}} + \frac{k-1}{p} + \frac{\max(\hat{d}dD, \hat{d} + |S|) + k - 1}{p} + \frac{\deg\big(\sum_{i\in[k]} \alpha^{i-1} \cdot F_i - T \cdot Z_S\big)}{p} \tag{5.30}$$

$$\leq \epsilon_{\mathsf{Open}} + 2 \cdot \frac{\max(\hat{d}dD, \hat{d} + |S|) + k - 1}{p} \tag{5.31}$$

To derive the second, third, and fourth part of the probability bound in Equation (5.30), respectively, we used Lemma 5.1, Claim 5.1, and the fact that the event $Z_S \nmid \sum_{i\in[k]} \alpha^{i-1} \cdot F_i$ is equivalent to $\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(X) - T(X) \cdot Z_S(X) \not\equiv 0$ for any choice of $T \in \mathbb{F}_p[X]$, which means that the probability that $\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta)$ over random $\delta \in \mathbb{F}_p$ can be bounded accordingly by the Schwartz–Zippel lemma. Since the overall probability is negligible in λ, this concludes the proof. □

On a final note, we want to remark that the maximum degree of the protocol's SRS can be even greater than $\max(d, |S|, d^2D - |S|)$ without breaking knowledge soundness (assuming

this bound is adjusted accordingly in the definition of $\mathcal{R}^S_{(d,D,t,\ell)}$). This has implications for the reusability of an already existing, trusted SRS. To be more concrete, in our case the parameters of the relation $\mathcal{R}^S_{(d,D,t,\ell)}$ will depend on the particular arithmetic circuit we want to prove statements about. Nonetheless, once we fix the maximum supported circuit size and generate an appropriate SRS, it can also be used to prove statements about arithmetic circuits of smaller size without impairing security.

5.3 Reducing the Number of Field Elements

In the final PLONK protocol, Gabizon et al. use an optimized version of the S-ranged polynomial protocol, which relies on a clever method suggested by Mary Maller to reduce the number of field elements sent by the prover [GWC19, Sec. 4.2]. In the current version of the protocol, $\mathcal{P}$ has to send the evaluations at δ of all the polynomials needed by $\mathcal{V}$ to check if $\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) \stackrel{?}{=} T(\delta) \cdot Z_S(\delta)$.

To see how we can do better, let us look at an illustrating example. Suppose $\mathcal{V}$ wants to check the identity $f_1(X) - f_2(X) \cdot f_3(X) \equiv 0$. Instead of $\mathcal{P}$ sending the values $f_1(\delta), f_2(\delta), f_3(\delta)$ and $\mathcal{V}$ checking $f_1(\delta) - f_2(\delta) \cdot f_3(\delta) = 0$, we can use the homomorphic property of the KZG polynomial commitment scheme to have $\mathcal{P}$ only send $f_2(\delta)$. $\mathcal{V}$ then checks whether the polynomial $L(X) := f_1(X) - f_2(\delta) \cdot f_3(X)$ evaluates to zero at δ, which can be directly integrated into the Open protocol used in step 6. The important observation is that $\mathcal{P}$ does not have to commit to L, as $\mathcal{V}$ can compute this commitment by itself using the commitments it already knows. In this example, $\mathcal{V}$ can use the commitments $c_1 := \mathsf{g}_1^{f_1(\tau)}$ and $c_3 := \mathsf{g}_1^{f_3(\tau)}$ to compute

$$c_L := c_1 / c_3^{f_2(\delta)} = \mathsf{g}_1^{f_1(\tau)} / \left(\mathsf{g}_1^{f_3(\tau)}\right)^{f_2(\delta)} = \mathsf{g}_1^{f_1(\tau) - f_2(\delta)\cdot f_3(\tau)} = \mathsf{g}_1^{L(\tau)}.$$

We refer to L as the *linearization polynomial*. Note that it is not a prerequisite for this optimization to work that all the selector polynomials are of the form X as in this example. In fact, if we instead had $f_1(X+1) - f_2(X) \cdot f_3(X+1) \equiv 0$, we could still use the same linearization polynomial as before but have $\mathcal{P}$ prove that it evaluates to 0 at $\delta + 1$ rather than at δ.

For simplicity, let us assume $\mathcal{V}$ only checks a single S-ranged identity of the form

$$F(X) := G(X, f_1(v_1(X)), \ldots, f_M(v_M(X))),$$

where $F(a) = 0$ supposedly holds for all $a \in S$. To describe the general method, let $I_L \subseteq [M]$ be the largest subset of $[M]$ such that

- $v_i \equiv v_j$ for all $i, j \in I_L$, and
- each monomial in $G(X, X_1, \ldots, X_M)$ contains at most one variable X_i with $i \in I_L$.

These two conditions ensure that $\mathcal{V}$ can compute the commitment to the resulting linearization polynomial

$$L(X) := G(\delta, f_1(\delta_1), \ldots, f_M(\delta_M)) - Z_S(\delta) \cdot T(X),$$
$$\text{where } \delta_i := \begin{cases} X & \text{if } i \in I_L \\ v_i(\delta) & \text{otherwise} \end{cases}.$$

Note that by maximizing the size of the subset $I_L \subseteq [M]$, we effectively minimize the number of evaluations the prover has to send to $\mathcal{V}$.

In summary, the optimized version of the S-ranged polynomial protocol from Construction 5.2 no longer contains step 7 and modifies steps 5–6 in the following way:

5. $\mathcal{P}$ replies with $y_Z := Z_S(\delta)$ and $(y_i)_{i\in[M]\setminus I_L} := (f_i(v_i(\delta)))_{i\in[M]\setminus I_L}$.
6. $\mathcal{V}$ computes the commitment to the linearization polynomial L and uses the Open protocol to verify the correctness of the evaluations received from $\mathcal{P}$, including $L(v_i(\delta)) = 0$ for an arbitrary $i \in I_L$.

To prove this variant of the S-ranged polynomial protocol remains knowledge sound in the AGM, we need to take into account that $\mathcal{V}$ accepting is now only tied to the event A, i.e., $\mathcal{V}$ accepting in the Open protocol. Thus, we have:

$$\begin{aligned}
&\Pr\left[\mathcal{V} \text{ accepts} \wedge (\mathbf{i}, \mathbf{x}, \mathbf{w}) \notin \mathcal{R}^S_{(d,D,t,\ell)}\right] \\
&= \Pr\left[A \wedge (\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C)\right] && (5.32)\\
&= \Pr\left[A \wedge (\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C) \wedge B\right] + \Pr\left[A \wedge (\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C) \wedge \bar{B}\right] && (5.33)\\
&\leq \Pr[A \wedge B] + \Pr\Big[\sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) = T(\delta) \cdot Z_S(\delta) \wedge (\exists i^* \in [k], Z_S \nmid F_{i^*} \vee C)\Big] && (5.34)\\
&\leq \epsilon_{\mathsf{Open}} + 2 \cdot \frac{\max(\hat{d}dD, \hat{d} + |S|) + k - 1}{p} && (5.35)
\end{aligned}$$

To derive the second probability in Equation (5.34), we used the fact that $\bar{B}$ implies $L(v_j(\delta)) = \sum_{i\in[k]} \alpha^{i-1} \cdot F_i(\delta) - T(\delta) \cdot Z_S(\delta) = 0$ for any $j \in I_L$. This is the case because $\bar{B}$ means that all the evaluations sent by the prover are correct, including $L(v_j(\delta)) = 0$, and because the commitment to L computed by the verifier is the commitment to the correct polynomial. The final bound is then obtained via the same steps as used in the proof of the original protocol's knowledge soundness from Equation (5.26) on.

CHAPTER 6

Permutation Argument

Having established the S-ranged polynomial protocol, proving that the wire polynomials A, B, C from Section 3.2 satisfy all the gate constraints is trivial; in fact, such a proof is meaningless as it treats the circuit as a collection of independent gates. This is where the copy constraints come into play, ensuring that the gate outputs are being properly propagated along the circuit's wires. Rather than directly proving all the necessary equalities between the wire polynomials, recall that we can instead prove the following equivalent statement from Equation (3.7):

$$\forall i \in [3n], D_{(\sigma(i))} = D_{(i)}, \tag{6.1}$$

where $D_{(i)}$ is the evaluation sequence of A, B, C, and $\sigma\colon [3n] \to [3n]$ is a permutation containing certain cycles as explained in Chapter 3. Gabizon et al. describe a more general method to achieve this goal. In the following, we explain their technique, which is at the heart of PLONK and is referred to as the *permutation argument*.

Towards this goal, let $f_1, \ldots, f_k, g_1, \ldots, g_k \in \mathbb{F}_p^{(\leq d)}[X]$ be polynomials and $\sigma\colon [kn] \to [kn]$ a permutation. Set $H := \langle \omega \rangle$, where ω is a primitive n-th root of unity in $\mathbb{F}_p$, i.e., $n \in \mathbb{N}_{>0}$ is the smallest positive integer satisfying $\omega^n = 1$. This makes H an order-n subgroup of $\mathbb{F}_p^*$, i.e, $H = \{\omega^0, \omega^1, \ldots, \omega^{n-1}\}$. Recalling Section 2.7 on *Lagrange interpolation*, for every $i \in [n]$, we denote by $\mathcal{L}_i \in \mathbb{F}_p^{(<n)}[X]$ the Lagrange polynomial with $\mathcal{L}_i(\omega^i) = 1$ and $\mathcal{L}_i(\omega^j) = 0$ for all $j \neq i \in [n]$, i.e., $\{\mathcal{L}_i\}_{i\in[n]}$ is a Lagrange basis for H. Next, define the evaluation sequences $(f_{(1)}, \ldots, f_{(kn)}), (g_{(1)}, \ldots, g_{(kn)}) \in \mathbb{F}_p^{kn}$ of the given polynomials over H by

$$f_{((i-1)n+j)} := f_i(\omega^j),\ g_{((i-1)n+j)} := g_i(\omega^j) \tag{6.2}$$

for all $i \in [k], j \in [n]$. To better visualize this notation, consider the resulting evaluation sequence of the polynomials $f_1, \ldots, f_k$:

$$(f_{(1)}, f_{(2)}, \ldots, f_{(n)}, f_{(n+1)}, \ldots, f_{(kn)}) := (f_1(\omega^1), f_1(\omega^2), \ldots, f_1(\omega^n), f_2(\omega^1), \ldots, f_k(\omega^n)).$$

The goal will be to show that

$$\forall i \in [kn], f_{(\sigma(i))} = g_{(i)}, \tag{6.3}$$

that is, permuting the values in $(f_{(1)}, \ldots, f_{(kn)})$ according to σ results in $(g_{(1)}, \ldots, g_{(kn)})$. Note the resemblance to our original goal of proving that the copy constraints from Equation (6.1) are satisfied.

To achieve this efficiently, we will leverage the following lemma by Gabizon et al. [GWC19, Claim A.1], which forms the foundation of their permutation argument. We provide our own proof of this lemma in Appendix A.1.

Lemma 6.1 (Claim A.1)**.** *Fix n distinct elements $s_1, \ldots, s_n \in \mathbb{F}_p$. Let $\sigma\colon [n] \to [n]$ be a permutation, and let $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in \mathbb{F}_p^n$. If*

$$\Pr\left[\begin{array}{c}\beta, \gamma \leftarrow \mathbb{F}_p\colon \\ \prod_{i\in[n]} (a_i + \beta \cdot s_i + \gamma) = \prod_{i\in[n]} (b_i + \beta \cdot s_{\sigma(i)} + \gamma)\end{array}\right] > \frac{n}{p}, \tag{6.4}$$

then $a_{\sigma(i)} = b_i$ for all $i \in [n]$.

Fixing kn distinct elements $s_1, \ldots, s_{kn} \in \mathbb{F}_p$, and letting our two evaluations sequences $(f_{(i)})_{i\in[kn]}$ and $(g_{(i)})_{i\in[kn]}$ take the place of the vectors $(a_1, \ldots, a_{kn}), (b_1, \ldots, b_{kn}) \in \mathbb{F}_p^{kn}$, respectively, this lemma motivates the following approach: First, we let the verifier challenge the prover with uniformly sampled $\beta, \gamma \in \mathbb{F}_p$, and then the prover has to show that the equality in Equation (6.4) holds, i.e.,

$$\prod_{i\in[kn]} (f_{(i)} + \beta \cdot s_i + \gamma) = \prod_{i\in[kn]} (g_{(i)} + \beta \cdot s_{\sigma(i)} + \gamma), \tag{6.5}$$

implying the originally desired statement from Equation (6.3) with high probability.

To this end, define the following two polynomials for every $i \in [k]$, representing the values $s_1, \ldots, s_{kn} \in \mathbb{F}_p$ in their original order and permuted according to σ, respectively:

$$S_{\mathrm{id},i}(X) := \sum_{j\in[n]} s_{(i-1)n+j} \cdot \mathcal{L}_j(X), \; S_{\sigma,i}(X) := \sum_{j\in[n]} s_{\sigma((i-1)n+j)} \cdot \mathcal{L}_j(X)$$

Also, let

$$f(X) := \prod_{i\in[k]} (f_i(X) + \beta \cdot S_{\mathrm{id},i}(X) + \gamma), \; g(X) := \prod_{i\in[k]} (g_i(X) + \beta \cdot S_{\sigma,i}(X) + \gamma).$$

This allows us to rewrite Equation (6.5) in terms of our initial polynomials $f_1, \ldots, f_k$, $g_1, \ldots, g_k \in \mathbb{F}_p^{(\leq d)}[X]$ as:

$$\prod_{i \in [kn]} (f_{(i)} + \beta \cdot s_i + \gamma) = \prod_{i \in [kn]} (g_{(i)} + \beta \cdot s_{\sigma(i)} + \gamma) \tag{6.6}$$

$$\iff \prod_{i \in [k]} \prod_{j \in [n]} (f_{((i-1)n+j)} + \beta \cdot s_{(i-1)n+j} + \gamma) = \prod_{i \in [k]} \prod_{j \in [n]} (g_{((i-1)n+j)} + \beta \cdot s_{\sigma((i-1)n+j)} + \gamma) \tag{6.7}$$

$$\iff \prod_{i \in [k]} \prod_{j \in [n]} (f_i(\omega^j) + \beta \cdot S_{\mathrm{id},i}(\omega^j) + \gamma) = \prod_{i \in [k]} \prod_{j \in [n]} (g_i(\omega^j) + \beta \cdot S_{\sigma,i}(\omega^j) + \gamma) \tag{6.8}$$

$$\iff \prod_{j \in [n]} \prod_{i \in [k]} (f_i(\omega^j) + \beta \cdot S_{\mathrm{id},i}(\omega^j) + \gamma) = \prod_{j \in [n]} \prod_{i \in [k]} (g_i(\omega^j) + \beta \cdot S_{\sigma,i}(\omega^j) + \gamma) \tag{6.9}$$

$$\iff \prod_{j \in [n]} f(\omega^j) = \prod_{j \in [n]} g(\omega^j) \tag{6.10}$$

Next, Gabizon et al. introduce a special polynomial $\Phi \in \mathbb{F}_p^{(<n)}[X]$, called the *permutation check polynomial*, enabling the verifier to succinctly check the above equation via two H-ranged identities. It is recursively defined as:

$$\begin{aligned} &1.\ \Phi(\omega) = 1, \\ &2.\ \Phi(\omega^{i+1}) = \Phi(\omega^i) \cdot \frac{f(\omega^i)}{g(\omega^i)}, \forall i \in [n]. \end{aligned} \tag{6.11}$$

Unfolding the recursion, we get:

$$\Phi(\omega^{n+1}) = \Phi(\omega) \cdot \prod_{j \in [n]} \frac{f(\omega^j)}{g(\omega^j)}$$

Since $\omega^{n+1} = \omega$ (because ω is a primitive n-root of unity) and $\Phi(\omega) = 1$, it follows that

$$\prod_{j \in [n]} \frac{f(\omega^j)}{g(\omega^j)} = \Phi(\omega^{n+1}) = \Phi(\omega) = 1, \tag{6.12}$$

which, by the previously shown equivalences, implies Equation (6.5).

So all that needs to be checked is that the polynomial $\Phi \in \mathbb{F}_p^{(<n)}[X]$ is of the correct form as defined above. To achieve this, Gabizon et al. express each of the two conditions defining Φ as an H-ranged polynomial identity and then leverage the H-ranged polynomial protocol from Construction 5.2 to obtain the following protocol [GWC19, Sec. 5.1] for proving statements like the one in Equation (6.3):

Construction 6.1: Permutation Check Protocol.

- Structured reference string: $\mathsf{srs} := \left(\mathsf{g}_1, \mathsf{g}_1^{\tau}, \mathsf{g}_1^{\tau^2}, \ldots, \mathsf{g}_1^{\tau^{\max(d,n,kd-1,k(n-1)-1)}}, \mathsf{g}_2, \mathsf{g}_2^{\tau}\right)$.
- Preprocessed input: For every $i \in [k]$, the two permutation polynomials

$$S_{\mathrm{id},i}(X) := \sum_{j\in[n]} s_{(i-1)n+j} \cdot \mathcal{L}_j(X),\ S_{\sigma,i}(X) := \sum_{j\in[n]} s_{\sigma((i-1)n+j)} \cdot \mathcal{L}_j(X),$$

  and their commitments $c_{S_{\mathrm{id},i}} := \mathsf{g}_1^{S_{\mathrm{id},i}(\tau)}, c_{S_{\sigma,i}} := \mathsf{g}_1^{S_{\sigma,i}(\tau)}$.
- $\langle \mathcal{P}(\mathsf{srs}, (S_{\mathrm{id},i}, S_{\sigma,i})_{i\in[k]}, (f_i, g_i)_{i\in[k]}), \mathcal{V}(\mathsf{srs}, (c_{S_{\mathrm{id},i}}, c_{S_{\sigma,i}})_{i\in[k]}) \rangle$:
  1. $\mathcal{P}$ commits to the polynomials $f_1, \ldots, f_k, g_1, \ldots, g_k \in \mathbb{F}_p^{(\leq d)}[X]$ and sends the commitments $(c_1, \ldots, c_k, c'_1, \ldots, c'_k) := (\mathsf{g}_1^{f_1(\tau)}, \ldots, \mathsf{g}_1^{f_k(\tau)}, \mathsf{g}_1^{g_1(\tau)}, \ldots, \mathsf{g}_1^{g_k(\tau)})$ to $\mathcal{V}$.
  2. $\mathcal{V}$ sends the random *permutation challenges* $\beta, \gamma \leftarrow \mathbb{F}_p$ to $\mathcal{P}$.
  3. Define the polynomials

$$f(X) := \prod_{i\in[k]} (f_i(X) + \beta \cdot S_{\mathrm{id},i}(X) + \gamma),\ g(X) := \prod_{i\in[k]} (g_i(X) + \beta \cdot S_{\sigma,i}(X) + \gamma).$$

     $\mathcal{P}$ computes the *permutation check polynomial* $\Phi \in \mathbb{F}_p^{(<n)}[X]$ such that $\Phi(\omega) = 1$ and $\forall i \in [n], \Phi(\omega^{i+1}) = \Phi(\omega^i) \cdot \frac{f(\omega^i)}{g(\omega^i)}$, and sends the commitment $c_\Phi := \mathsf{g}_1^{\Phi(\tau)}$ to $\mathcal{V}$.
  4. Using the H-ranged polynomial protocol from Construction 5.2, $\mathcal{V}$ checks the following two H-ranged identities for all $a \in H$:
     (a) $\Phi(a)f(a) - \Phi(a \cdot \omega)g(a) = 0$,
     (b) $\mathcal{L}_1(a)(\Phi(a) - 1) = 0$.

6.1 Knowledge Soundness in the AGM

Having given the intuition for the security of the protocol, we now formally argue why the *permutation check protocol* is knowledge sound in the AGM. The first check (a) in step 4 of Construction 6.1 ensures that $\Phi(\omega^{i+1}) = \Phi(\omega^i) \cdot \frac{f(\omega^i)}{g(\omega^i)}$ for all $i \in [n]$. The second check (b) is equivalent to $\Phi(\omega) = 1$, as $\mathcal{L}_1(\omega) = 1$ and $\mathcal{L}_1(\omega^i) = 0$ for all $i \neq 1 \in [n]$. As discussed when establishing Equation (6.12), together these two conditions ensure that:

$$\prod_{j\in[n]} \frac{f(\omega^j)}{g(\omega^j)} = 1 \tag{6.13}$$

$$\iff \prod_{i\in[kn]} (f_{(i)} + \beta \cdot s_i + \gamma) = \prod_{i\in[kn]} (g_{(i)} + \beta \cdot s_{\sigma(i)} + \gamma) \tag{6.14}$$

By contraposition of Lemma 6.1, we know that if there exists an $i^* \in [kn]$ such that $f_{(\sigma(i^*))} \neq g_{(i^*)}$, then Equation (6.14) only holds with negligible probability over the choice of β and γ, establishing the soundness of this procedure. More formally, if Equation (6.3) is not satisfied, then the probability that the H-ranged identities (a) and (b) hold is at most kn/p, and if they do not hold, then by the knowledge soundness of the H-ranged polynomial protocol, the probability of $\mathcal{V}$ accepting is negligible:

$$\begin{aligned}
&\Pr\left[\mathcal{V} \text{ accepts} \wedge \exists i^* \in [kn], f_{(\sigma(i))} \neq g_{(i)}\right] \\
&\quad = \Pr\left[\mathcal{V} \text{ accepts} \wedge \exists i^* \in [kn], f_{(\sigma(i))} \neq g_{(i)} \wedge (a) \wedge (b)\right] \\
&\qquad + \Pr\left[\mathcal{V} \text{ accepts} \wedge \exists i^* \in [kn], f_{(\sigma(i))} \neq g_{(i)} \wedge \neg((a) \wedge (b))\right] && (6.15) \\
&\quad \leq \Pr\left[(a) \wedge (b) \mid \exists i^* \in [kn], f_{(\sigma(i))} \neq g_{(i)}\right] + \Pr\left[\mathcal{V} \text{ accepts} \wedge \neg((a) \wedge (b))\right] && (6.16) \\
&\quad \leq \frac{kn}{p} + \epsilon_{H\text{-Ranged}}, && (6.17)
\end{aligned}$$

where $\epsilon_{H\text{-Ranged}}$ is the knowledge soundness error of the H-ranged polynomial protocol from Construction 5.2.

6.2 Statistical Completeness

An important point we have ignored so far, is that it is possible that one of the denominators $g(\omega^{i^*}) = 0$ for some $i^* \in [n]$, which renders the permutation check polynomial Φ undefined and as such impossible to compute. In this case, $\mathcal{P}$ would have to abort, ruining the perfect completeness of the protocol. Fortunately, this event only happens with negligible probability over the choice of $\gamma \in \mathbb{F}_p$. To see this, consider the polynomial

$$G(Y) := \prod_{i \in [n]} g(\omega^i) = \prod_{i \in [n]} \prod_{j \in [k]} (g_j(\omega^i) + \beta \cdot S_{\sigma,j}(\omega^i) + Y). \tag{6.18}$$

Clearly, it has at most kn roots, so the probability that $G(\gamma) = 0$ is at most kn/p, which is negligible in λ. Overall, we attain a protocol with *statistical completeness*.

CHAPTER 7

The Full PLONK Protocol

Having established all the necessary tools, we can finally present the full PLONK protocol. For this purpose, let $\mathcal{C}$ be the arithmetic circuit with fan-in two and unlimited fan-out over $\mathbb{F}_p$ for which we want to prove knowledge of a satisfying wire assignment. Suppose $\mathcal{C}$ has ℓ public inputs/outputs. Not counting the private inputs/outputs as described in Section 3.1, let $n = \mathsf{poly}(\lambda)$ be the number of gates and $m \leq 3n$ the number of wires. Without loss of generality, assume the public inputs are associated with the first ℓ gates. Using the PLONK constraint system from Chapter 3, we can model $\mathcal{C}$ via a combination of gate and copy constraints as written in Equations (3.1) and (3.2). Let $(\mathbf{s_L}, \mathbf{s_R}, \mathbf{s_O}, \mathbf{s_M}, \mathbf{s_C}) := ((s_{\mathsf{L}i})_{i\in[n]}, (s_{\mathsf{R}i})_{i\in[n]}, (s_{\mathsf{O}i})_{i\in[n]}, (s_{\mathsf{M}i})_{i\in[n]}, (s_{\mathsf{C}i})_{i\in[n]}) \in \mathbb{F}_p^{5n}$ denote all the gate constraints, and let $\sigma\colon [3n] \to [3n]$ be a permutation with appropriate cycles capturing all the copy constraints of $\mathcal{C}$. Formally, we want to prove statements of knowledge in the following indexed relation:

$$\mathcal{R}_{\text{PLONK}} :=$$

$$\left\{ (\mathbf{i}, \mathbf{x}, \mathbf{w}) \;\middle|\; \begin{array}{l} \mathbf{i} = \begin{pmatrix} (\mathbf{s_L}, \mathbf{s_R}, \mathbf{s_O}, \mathbf{s_M}, \mathbf{s_C}) \in \mathbb{F}_p^{5n}, \\ \sigma\colon [3n] \to [3n] \end{pmatrix}, \\ \mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{F}_p^{\ell}, \\ \mathbf{w} = (w_1, \ldots, w_{3n}) \in \mathbb{F}_p^{3n}, \\ \forall i \in [\ell]\colon s_{\mathsf{L}i} w_i + s_{\mathsf{R}i} w_{n+i} + s_{\mathsf{O}i} w_{2n+i} + s_{\mathsf{M}i} w_i w_{n+i} + s_{\mathsf{C}i} - x_i = 0, \\ \forall i \in (\ell, n]\colon s_{\mathsf{L}i} w_i + s_{\mathsf{R}i} w_{n+i} + s_{\mathsf{O}i} w_{2n+i} + s_{\mathsf{M}i} w_i w_{n+i} + s_{\mathsf{C}i} = 0, \\ \forall i \in [3n]\colon w_{\sigma(i)} = w_i \end{array} \right\},$$

where the index $\mathbf{i}$ is the specification of the arithmetic circuit $\mathcal{C}$ (expressed in the PLONK constraint system from Section 3.1), the statement $\mathbf{x}$ is the specific values assigned to the public inputs/output gates of $\mathcal{C}$, and the witness $\mathbf{w}$ is an assignment of values to all the wires of the circuit such that they are consistent with $\mathbf{x}$ and all the internal addition/multiplication gates of $\mathcal{C}$.

Note that we have made the following two changes compared to our initial presentation of the PLONK constraint system in Section 3.1:

1. For simplicity of notation, $\mathbf{w} \in \mathbb{F}_p^{3n}$ is now a redundant value assignment to the m wires of $\mathcal{C}$, representing the wires from the point of view of each gate. For example, the output wire of gate i could be the same as the left input wire to another gate j, but we include two separate values w_{2n+i} and w_j in $\mathbf{w}$. This correctly captures the fact that the gate constraints model the circuit as a collection of independent gates. Equivalences such as $w_{2n+i} = w_j$ are exactly what we use the copy constraints for, i.e., an appropriate choice of the permutation σ would in this case include a cycle containing both indices $2n + i$ and j.

2. We have omitted the selector vector s_{PI} in its entirety from the definition of $\mathcal{R}_{\text{PLONK}}$, which is responsible for enforcing the public inputs/outputs as introduced in Section 3.1. Since the concrete values of s_{PI} directly depend on $\mathbf{x}$, it cannot be part of the index $\mathbf{i}$, unlike the other selector vectors. But more importantly, this has the benefit of decoupling the concrete choice of public inputs from the preprocessing phase, i.e., once the circuit $\mathcal{C}$ is preprocessed, $\mathcal{P}$ can prove knowledge of a satisfying wire assignment for any choice of public inputs $(x_i)_{i\in[\ell]} \in \mathbb{F}_p^\ell$.

Next, we will express the relation $\mathcal{R}_{\text{PLONK}}$ via an equivalent set of polynomials as already previewed in Section 3.2. Towards this goal, we assume the existence of a primitive n-th root of unity ω in $\mathbb{F}_p$, so that we can use the multiplicative order-n subgroup $H := \langle\omega\rangle$ for proving all the necessary constraints as H-ranged polynomial identities. The particular choice of H comes with several advantages, the most important of which being the succinct representation of the resulting divisor polynomial Z_H, making it efficiently computable. This allows to shift the evaluations of some simple public polynomials to $\mathcal{V}$, resulting in a shorter proof. In particular, as explained in Section 2.7, the divisor polynomial and the i-th Lagrange polynomial become $Z_H(X) = X^n - 1$ and $\mathcal{L}_i(X) = \frac{\omega^i(X^n-1)}{n(X-\omega^i)}$, respectively, both of which can be evaluated in a polylogarithmic number of field operations in n.

Let $S_{\mathsf{L}}, S_{\mathsf{R}}, S_{\mathsf{O}}, S_{\mathsf{M}}, S_{\mathsf{C}} \in \mathbb{F}_p[X]$ be the constraint polynomials defined for each $i \in [n]$ by

$$S_{\mathsf{L}}(\omega^i) := s_{\mathsf{L}i},\ S_{\mathsf{R}}(\omega^i) := s_{\mathsf{R}i},\ S_{\mathsf{O}}(\omega^i) := s_{\mathsf{O}i},\ S_{\mathsf{M}}(\omega^i) := s_{\mathsf{M}i},\ S_{\mathsf{C}}(\omega^i) := s_{\mathsf{C}i}.$$

In addition, to handle the public inputs/outputs, Gabizon et al. introduce the public input polynomial

$$S_{\mathsf{PI}}(X) := \sum_{i\in[\ell]} -x_i\mathcal{L}_i(X),$$

which will be incorporated as a final summand into each gate constraint taking the place of the corresponding selector vector s_{PI}. Note that this polynomial has the desired property that $S_{\mathsf{PI}}(\omega^i) = -x_i$ for all $i \in [\ell]$ and $S_{\mathsf{PI}}(\omega^i) = 0$ otherwise, just as the vector s_{PI} is defined in Table 3.1.

Let $A, B, C \in \mathbb{F}_p[X]$ be the left input, right input, and output wire polynomials satisfying

$$A(\omega^i) := w_i,\ B(\omega^i) := w_{n+i},\ C(\omega^i) := w_{2n+i}$$

for all $i \in [n]$. Also, define the following two polynomials used in the permutation argument:

$$f(X) := (A(X) + \beta \cdot S_{\mathrm{id},1}(X) + \gamma)(B(X) + \beta \cdot S_{\mathrm{id},2}(X) + \gamma)(C(X) + \beta \cdot S_{\mathrm{id},3}(X) + \gamma),$$
$$g(X) := (A(X) + \beta \cdot S_{\sigma,1}(X) + \gamma)(B(X) + \beta \cdot S_{\sigma,2}(X) + \gamma)(C(X) + \beta \cdot S_{\sigma,3}(X) + \gamma),$$

where $S_{\mathrm{id},1}, S_{\mathrm{id},2}, S_{\mathrm{id},3}, S_{\sigma,1}, S_{\sigma,2}, S_{\sigma,3} \in \mathbb{F}_p[X]$ are the respective permutation polynomials from Chapter 6, and $\beta, \gamma \in \mathbb{F}_p$ are the permutation challenges chosen by the verifier. Lastly, let $\Phi \in \mathbb{F}_p[X]$ be the permutation check polynomial defined in terms of f, g as specified in Equation (6.11).

With this, proving knowledge of a triple $(\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}_{\mathrm{PLONK}}$ is equivalent to proving that the following three H-ranged polynomial identities hold for all $a \in H$:

1. $S_{\mathsf{L}}(a)A(a) + S_{\mathsf{R}}(a)B(a) + S_{\mathsf{O}}(a)C(a) + S_{\mathsf{M}}(a)A(a)B(a) + S_{\mathsf{C}}(a) + S_{\mathsf{PI}}(a) = 0,$
2. $\Phi(a)f(a) - \Phi(a \cdot \omega)g(a) = 0,$
3. $\mathcal{L}_1(a)(\Phi(a) - 1) = 0.$

The first identity represents the gate constraints, and the second and third ensure that the copy constraints hold via the permutation argument from Chapter 6.

Next, we need to fix $3n$ distinct elements from $\mathbb{F}_p$ as required by Lemma 6.1 for the permutation argument. This is where Gabizon et al. use an optimization suggested by Vitalik Buterin [GWC19, Sec. 8] to represent the identity permutation polynomials $S_{\mathrm{id},1}, S_{\mathrm{id},2}, S_{\mathrm{id},3}$ via degree-1 polynomials, so that their evaluations can be directly computed by the verifier. To this end, define the set

$$H' := H \cup (k_1 \cdot H) \cup (k_2 \cdot H),$$

where $k_1, k_2 \in \mathbb{F}_p^*$ are chosen such that H, $k_1 \cdot H$, $k_2 \cdot H$ are distinct cosets of H consisting of $3n$ elements in total. Furthermore, extend the permutation $\sigma\colon [3n] \to [3n]$ onto H' by defining $\sigma^*\colon [3n] \to H'$ as

$$\sigma^*(i) := \begin{cases} \omega^{\sigma(i)} & \text{if } \sigma(i) \in [n] \\ k_1 \cdot \omega^{\sigma(i)} & \text{if } \sigma(i) \in (n, 2n] \\ k_2 \cdot \omega^{\sigma(i)} & \text{if } \sigma(i) \in (2n, 3n] \end{cases}.$$

Then the identity permutation polynomials simply turn out to be

$$S_{\mathrm{id},1}(X) := X, \; S_{\mathrm{id},2}(X) := k_1 X, \; S_{\mathrm{id},3}(X) := k_2 X.$$

A simple method for picking appropriate $k_1, k_2 \in \mathbb{F}_p^*$ is to take an arbitrary value in $\mathbb{F}_p^* \setminus H$ as k_1 and then take any value not contained in $H \cup (k_1 \cdot H)$ as k_2. To see why this works, let g be a generator of $\mathbb{F}_p^*$ such that $\omega = \mathsf{g}^r$ with $r := (p-1)/n$ and recall that H

can be expressed as $\{\mathsf{g}^0, \mathsf{g}^r, \mathsf{g}^{2r}, \ldots, \mathsf{g}^{(n-1)r}\}$. If we then let $k_1 := \mathsf{g}^a$ and $k_2 := \mathsf{g}^b$, we see that as long as both $a, b \neq 0 \pmod r$ and $a \neq b \pmod r$, we have that H, $k_1 \cdot H$, $k_2 \cdot H$ are pairwise disjoint as required. These two conditions are met exactly when choosing k_1, k_2 as described above.

In summary, there are 14 public polynomials in PLONK. Out of these, $\mathcal{V}$ can evaluate $S_{\mathsf{PI}}, S_{\mathrm{id},1}, S_{\mathrm{id},2}, S_{\mathrm{id},3}, Z_H, \mathcal{L}_1$ at any point $x \in \mathbb{F}_p$ in polylogarithmic time. In addition to that, Gabizon et al. save the evaluations of $S_{\mathsf{L}}, S_{\mathsf{R}}, S_{\mathsf{O}}, S_{\mathsf{M}}, S_{\mathsf{C}}, S_{\sigma,3}$ via the optimization by Maller from Section 5.3. This just leaves the evaluation of the two public polynomials $S_{\mathrm{id},1}, S_{\mathrm{id},2}$ to the prover.

As a final step, Gabizon et al. make the PLONK protocol zero-knowledge by adding random multiples of Z_H to the prover's witness polynomials. This does not ruin satisfiability, while creating a situation where all values in the proof are either completely uniform or determined by verifier equations. We defer a detailed discussion of PLONK's zero knowledge, including the demonstration of an adequate simulator, to Section 7.2.

Having established all of this, we are finally ready to present the full PLONK protocol:

Construction 7.1: The PLONK Protocol.

- $\mathsf{Setup}(1^\lambda, n)$: Choose random $\tau \leftarrow \mathbb{F}_p$ and output

$$\mathsf{srs} := (\mathsf{g}_1, \mathsf{g}_1^\tau, \mathsf{g}_1^{\tau^2}, \ldots, \mathsf{g}_1^{\tau^{n+2}}, \mathsf{g}_2, \mathsf{g}_2^\tau).$$

- $\mathsf{Preprocess}(\mathsf{srs}, \mathbf{i})$: Given the circuit size n, the wire permutation σ^*, and the gate constraints $(s_{\mathsf{L}i}, s_{\mathsf{R}i}, s_{\mathsf{O}i}, s_{\mathsf{M}i}, s_{\mathsf{C}i})_{i\in[n]}$, compute the constraint polynomials

$$\begin{aligned} S_{\mathsf{L}}(X) &:= \textstyle\sum_{i\in[n]} s_{\mathsf{L}i}\mathcal{L}_i(X), & S_{\mathsf{R}}(X) &:= \textstyle\sum_{i\in[n]} s_{\mathsf{R}i}\mathcal{L}_i(X), \\ S_{\mathsf{O}}(X) &:= \textstyle\sum_{i\in[n]} s_{\mathsf{O}i}\mathcal{L}_i(X), & S_{\mathsf{M}}(X) &:= \textstyle\sum_{i\in[n]} s_{\mathsf{M}i}\mathcal{L}_i(X), \\ S_{\mathsf{C}}(X) &:= \textstyle\sum_{i\in[n]} s_{\mathsf{C}i}\mathcal{L}_i(X), \end{aligned}$$

  the permutation polynomials

$$\begin{aligned} S_{\sigma,1}(X) &:= \textstyle\sum_{i\in[n]} \sigma^*(i)\mathcal{L}_i(X), \\ S_{\sigma,2}(X) &:= \textstyle\sum_{i\in[n]} \sigma^*(n+i)\mathcal{L}_i(X), \\ S_{\sigma,3}(X) &:= \textstyle\sum_{i\in[n]} \sigma^*(2n+i)\mathcal{L}_i(X), \end{aligned}$$

  and their commitments

$$\begin{aligned} &c_{S_{\mathsf{L}}} := \mathsf{g}_1^{S_{\mathsf{L}}(\tau)}, \quad c_{S_{\mathsf{R}}} := \mathsf{g}_1^{S_{\mathsf{R}}(\tau)}, \quad c_{S_{\mathsf{O}}} := \mathsf{g}_1^{S_{\mathsf{O}}(\tau)}, \quad c_{S_{\mathsf{M}}} := \mathsf{g}_1^{S_{\mathsf{M}}(\tau)}, \\ &c_{S_{\mathsf{C}}} := \mathsf{g}_1^{S_{\mathsf{C}}(\tau)}, \quad c_{S_{\sigma,1}} := \mathsf{g}_1^{S_{\sigma,1}(\tau)}, \quad c_{S_{\sigma,2}} := \mathsf{g}_1^{S_{\sigma,2}(\tau)}, \quad c_{S_{\sigma,3}} := \mathsf{g}_1^{S_{\sigma,3}(\tau)}. \end{aligned}$$

Output the prover and verifier parameters

$$\mathsf{pp} := (\mathsf{srs}, n, \sigma^*, S_{\mathsf{L}}, S_{\mathsf{R}}, S_{\mathsf{O}}, S_{\mathsf{M}}, S_{\mathsf{C}}, S_{\sigma,1}, S_{\sigma,2}, S_{\sigma,3}),$$
$$\mathsf{vp} := (\mathsf{g}_1, \mathsf{g}_2, \mathsf{g}_2^{\tau}, n, c_{S_{\mathsf{L}}}, c_{S_{\mathsf{R}}}, c_{S_{\mathsf{O}}}, c_{S_{\mathsf{M}}}, c_{S_{\mathsf{C}}}, c_{S_{\sigma,1}}, c_{S_{\sigma,2}}, c_{S_{\sigma,3}}).$$

- $\langle \mathcal{P}(\mathsf{pp}, (x_i)_{i\in[\ell]}, (w_i)_{i\in[3n]}), \mathcal{V}(\mathsf{vp}, (x_i)_{i\in[\ell]}) \rangle$:

  1. $\mathcal{P}$ chooses random blinding scalars $\rho_1, \ldots, \rho_6 \leftarrow \mathbb{F}_p$ and computes the wire polynomials

  $$A(X) := (\rho_1 X + \rho_2) Z_H(X) + \sum\nolimits_{i\in[n]} w_i \mathcal{L}_i(X),$$
  $$B(X) := (\rho_3 X + \rho_4) Z_H(X) + \sum\nolimits_{i\in[n]} w_{n+i} \mathcal{L}_i(X),$$
  $$C(X) := (\rho_5 X + \rho_6) Z_H(X) + \sum\nolimits_{i\in[n]} w_{2n+i} \mathcal{L}_i(X).$$

  The first output of $\mathcal{P}$ are the commitments

  $$c_A := \mathsf{g}_1^{A(\tau)},\ c_B := \mathsf{g}_1^{B(\tau)},\ c_C := \mathsf{g}_1^{C(\tau)}.$$

  2. $\mathcal{V}$ sends the random permutation challenges $\beta, \gamma \leftarrow \mathbb{F}_p$ to $\mathcal{P}$.
  3. Define the polynomials

  $$f(X) := (A(X)+\beta S_{\mathrm{id},1}(X)+\gamma)(B(X)+\beta S_{\mathrm{id},2}(X)+\gamma)(C(X)+\beta S_{\mathrm{id},3}(X)+\gamma),$$
  $$g(X) := (A(X)+\beta S_{\sigma,1}(X)+\gamma)(B(X)+\beta S_{\sigma,2}(X)+\gamma)(C(X)+\beta S_{\sigma,3}(X)+\gamma).$$

  $\mathcal{P}$ chooses random blinding scalars $\rho_7, \rho_8, \rho_9 \leftarrow \mathbb{F}_p$ and computes the permutation check polynomial

  $$\Phi(X) := (\rho_7 X^2 + \rho_8 X + \rho_9) Z_H(X) + \sum_{i\in[n]} \mathcal{L}_i(X) \prod_{j=1}^{i-1} \frac{f(\omega^j)}{g(\omega^j)},$$
  $$\text{where } \frac{f(\omega^j)}{g(\omega^j)} = \frac{(w_j+\beta\omega^j+\gamma)(w_{n+j}+\beta k_1\omega^j+\gamma)(w_{2n+j}+\beta k_2\omega^j+\gamma)}{(w_j+\beta\sigma^*(j)+\gamma)(w_{n+j}+\beta\sigma^*(n+j)+\gamma)(w_{2n+j}+\beta\sigma^*(2n+j)+\gamma)}.$$

  The second output of $\mathcal{P}$ is the commitment[a]

  $$c_\Phi := \mathsf{g}_1^{\Phi(\tau)}.$$

  4. $\mathcal{V}$ sends a random quotient challenge $\alpha \leftarrow \mathbb{F}_p$ to $\mathcal{P}$.

5. Define the polynomials

$$\begin{aligned}\mathsf{gates}(X) &:= \begin{pmatrix} S_{\mathsf{L}}(X)A(X) + S_{\mathsf{R}}(X)B(X) + S_{\mathsf{O}}(X)C(X) \\ + S_{\mathsf{M}}(X)A(X)B(X) + S_{\mathsf{C}}(X) + S_{\mathsf{PI}}(X) \end{pmatrix}, \\ \mathsf{copy}_1(X) &:= \Phi(X)f(X) - \Phi(X\omega)g(X), \\ \mathsf{copy}_2(X) &:= \mathcal{L}_1(X)(\Phi(X) - 1).\end{aligned}$$

$\mathcal{P}$ computes the quotient polynomial of degree $\leq 3n + 5$

$$T(X) := \frac{\mathsf{gates}(X) + \alpha \cdot \mathsf{copy}_1(X) + \alpha^2 \cdot \mathsf{copy}_2(X)}{Z_H(X)},$$

and splits it into the three unique polynomials $T'_1, T'_2, T'_3 \in \mathbb{F}_p^{(\leq n+1)}[X]$ satisfying

$$T(X) = T'_1(X) + X^{n+2} \cdot T'_2(X) + X^{2n+4} \cdot T'_3(X).$$

Then, $\mathcal{P}$ chooses random blinding scalars $\rho_{10}, \rho_{11} \leftarrow \mathbb{F}_p$ and defines

$$\begin{aligned}T_1(X) &:= T'_1(X) + \rho_{10}X^{n+2}, \\ T_2(X) &:= T'_2(X) - \rho_{10} + \rho_{11}X^{n+2}, \\ T_3(X) &:= T'_3(X) - \rho_{11}.\end{aligned}$$

Note that these polynomials still satisfy

$$T(X) = T_1(X) + X^{n+2} \cdot T_2(X) + X^{2n+4} \cdot T_3(X).$$

The third output of $\mathcal{P}$ are the commitments[b]

$$c_{T_1} := \mathsf{g}_1^{T_1(\tau)}, \; c_{T_2} := \mathsf{g}_1^{T_2(\tau)}, \; c_{T_3} := \mathsf{g}_1^{T_3(\tau)}.$$

6. $\mathcal{V}$ sends a random evaluation challenge $\delta \leftarrow \mathbb{F}_p$ to $\mathcal{P}$.
7. The fourth output of $\mathcal{P}$ are the evaluations

$$\begin{aligned} a_\delta &:= A(\delta), & b_\delta &:= B(\delta), & c_\delta &:= C(\delta), \\ s_{\sigma_1,\delta} &:= S_{\sigma,1}(\delta), & s_{\sigma_2,\delta} &:= S_{\sigma,2}(\delta), & \phi_{\delta\omega} &:= \Phi(\delta\omega). \end{aligned}$$

8. $\mathcal{V}$ sends a random opening challenge $\varepsilon \leftarrow \mathbb{F}_p$ to $\mathcal{P}$.

9. Define the linearized polynomials

$$
\begin{aligned}
L_{\mathsf{gates}}(X) &:= \begin{pmatrix} S_{\mathsf{L}}(X)A(\delta) + S_{\mathsf{R}}(X)B(\delta) + S_{\mathsf{O}}(X)C(\delta) \\ + S_{\mathsf{M}}(X)A(\delta)B(\delta) + S_{\mathsf{C}}(X) + S_{\mathsf{PI}}(\delta) \end{pmatrix}, \\
L_{\mathsf{copy}_1}(X) &:= \begin{pmatrix} \Phi(X)(A(\delta)+\beta\delta+\gamma)(B(\delta)+\beta k_1\delta+\gamma)(C(\delta)+\beta k_2\delta+\gamma) \\ -\Phi(\delta\omega)(A(\delta)+\beta S_{\sigma,1}(\delta)+\gamma)(B(\delta)+\beta S_{\sigma,2}(\delta)+\gamma)(C(\delta)+\beta S_{\sigma,3}(X)+\gamma) \end{pmatrix}, \\
L_{\mathsf{copy}_2}(X) &:= \mathcal{L}_1(\delta)(\Phi(X) - 1), \\
L_{\mathsf{division}}(X) &:= Z_H(\delta)\big[T_1(X) + \delta^{n+2}T_2(X) + \delta^{2n+4}T_3(X)\big].
\end{aligned}
$$

$\mathcal{P}$ computes the linearization polynomial

$$
L(X) := L_{\mathsf{gates}}(X) + \alpha \cdot L_{\mathsf{copy}_1}(X) + \alpha^2 \cdot L_{\mathsf{copy}_2}(X) - L_{\mathsf{division}}(X),
$$

and the aggregated quotient polynomials

$$
\begin{aligned}
Q_1(X) &:= \frac{1}{X-\delta}\begin{pmatrix} L(X)+\varepsilon\big[A(X)-A(\delta)\big]+\varepsilon^2\big[B(X)-B(\delta)\big]+\varepsilon^3\big[C(X)-C(\delta)\big] \\ +\varepsilon^4\big[S_{\sigma,1}(X)-S_{\sigma,1}(\delta)\big]+\varepsilon^5\big[S_{\sigma,2}(X)-S_{\sigma,2}(\delta)\big] \end{pmatrix}, \\
Q_2(X) &:= \frac{\Phi(X) - \Phi(\delta\omega)}{X - \delta\omega}.
\end{aligned}
$$

The fifth output of $\mathcal{P}$ are the commitments

$$
\pi_1 := \mathsf{g}_1^{Q_1(\tau)}, \ \pi_2 := \mathsf{g}_1^{Q_2(\tau)}.
$$

The full output of $\mathcal{P}$ is

$$
\pi_{\text{PLONK}} := \begin{pmatrix} c_A, c_B, c_C, c_\Phi, c_{T_1}, c_{T_2}, c_{T_3}, \pi_1, \pi_2, \\ a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega} \end{pmatrix} \in \mathbb{G}_1^9 \times \mathbb{F}_p^6.
$$

10. $\mathcal{V}$ computes the commitments

$$
\begin{aligned}
c_{L_{\mathsf{gates}}} &:= {c_{S_{\mathsf{L}}}}^{a_\delta} \cdot {c_{S_{\mathsf{R}}}}^{b_\delta} \cdot {c_{S_{\mathsf{O}}}}^{c_\delta} \cdot {c_{S_{\mathsf{M}}}}^{a_\delta b_\delta} \cdot c_{S_{\mathsf{C}}} \cdot \mathsf{g}_1^{S_{\mathsf{PI}}(\delta)}, \\
c_{L_{\mathsf{copy}_1}} &:= \frac{{c_\Phi}^{(a_\delta+\beta\delta+\gamma)(b_\delta+\beta k_1\delta+\gamma)(c_\delta+\beta k_2\delta+\gamma)}}{\Big({c_{S_{\sigma,3}}}^{\beta} \cdot \mathsf{g}_1^{c_\delta+\gamma}\Big)^{\phi_{\delta\omega}(a_\delta+\beta s_{\sigma_1,\delta}+\gamma)(b_\delta+\beta s_{\sigma_2,\delta}+\gamma)}}, \\
c_{L_{\mathsf{copy}_2}} &:= \big(c_\Phi/\mathsf{g}_1\big)^{\mathcal{L}_1(\delta)}, \\
c_{L_{\mathsf{division}}} &:= \Big(c_{T_1} \cdot {c_{T_2}}^{\delta^{n+2}} \cdot {c_{T_3}}^{\delta^{2n+4}}\Big)^{Z_H(\delta)},
\end{aligned}
$$

where $Z_H(\delta) = \delta^n - 1$, $\mathcal{L}_1(\delta) = \dfrac{\omega(\delta^n - 1)}{n(\delta - \omega)}$, $S_{\mathsf{PI}}(\delta) = \sum_{i\in[\ell]} -x_i\mathcal{L}_i(\delta)$.

With this, $\mathcal{V}$ computes the commitment to the linearization polynomial L as

$$c_L := c_{L_{\mathsf{gates}}} \cdot (c_{L_{\mathsf{copy}_1}})^{\alpha} \cdot (c_{L_{\mathsf{copy}_2}})^{\alpha^2} \cdot (c_{L_{\mathsf{division}}})^{-1}.$$

Then, $\mathcal{V}$ chooses a random multipoint evaluation challenge $\zeta \leftarrow \mathbb{F}_p$, computes

$$C_1 := c_L \cdot \left(\frac{c_A}{\mathsf{g}_1^{a_\delta}}\right)^{\varepsilon} \cdot \left(\frac{c_B}{\mathsf{g}_1^{b_\delta}}\right)^{\varepsilon^2} \cdot \left(\frac{c_C}{\mathsf{g}_1^{c_\delta}}\right)^{\varepsilon^3} \cdot \left(\frac{c_{S_{\sigma,1}}}{\mathsf{g}_1^{s_{\sigma_1,\delta}}}\right)^{\varepsilon^4} \cdot \left(\frac{c_{S_{\sigma,2}}}{\mathsf{g}_1^{s_{\sigma_2,\delta}}}\right)^{\varepsilon^5}, \; C_2 := \frac{c_\Phi}{\mathsf{g}_1^{\phi_{\delta\omega}}},$$

and accepts iff

$$e\Big(C_1 \cdot {\pi_1}^{\delta} \cdot \big(C_2 \cdot {\pi_2}^{\delta\omega}\big)^{\zeta}, \mathsf{g}_2\Big) \stackrel{?}{=} e\big(\pi_1 \cdot {\pi_2}^{\zeta}, \mathsf{g}_2^{\tau}\big).$$

[a] Note that when $\frac{f(\omega^j)}{g(\omega^j)}$ is undefined due to one of its denominators being zero, $\mathcal{P}$ has to abort the protocol. This can only happen with negligible probability, as discussed in Section 6.2.

[b] In a previous version of PLONK, $\mathcal{P}$ used to directly commit to T_1', T_2', T_3', which broke the protocol's zero knowledge and was fixed as a result of this work. We discuss this issue in more detail in Section 7.2.

In this full version of the PLONK protocol, Gabizon et al. employ several optimizations which are not directly part of the general primitives described in the previous chapters. The most prominent modification is that in step 5 of the protocol, $\mathcal{P}$ does not directly commit to the quotient polynomial T but instead splits it into three polynomials T_1, T_2, T_3 such that $T(X) = T_1(X) + X^{n+2} \cdot T_2(X) + X^{2n+4} \cdot T_3(X)$. The reason for this is to maintain a minimal SRS size at the cost of increasing the proof length by two group elements. Without this optimization, the prover would need an SRS of degree $3n+5$ to commit to the resulting quotient polynomial in step 5. Hence, this trade-off effectively reduces the SRS size from $3n+5$ to $n+2$, which can be considered optimal.

To see why this variant of the H-ranged polynomial protocol still remains knowledge sound in the AGM, we can apply a similar analysis as in Section 5.2.1. To this end, let $T_1^*, T_2^*, T_3^* \in \mathbb{F}_p^{(\leq n+2)}[X]$ be the actual polynomials committed to by the adversary and define the degree-$(n+2)$ polynomial

$$T^*(X) := T_1^*(X) + \delta^{n+2} \cdot T_2^*(X) + \delta^{2n+4} \cdot T_3^*(X).$$

Recall that in the proof from Section 5.2.1 we argued over an arbitrary choice of $T \in \mathbb{F}_p^{(\leq \hat{d})}[X]$ by the adversary, where $\hat{d}$ is the largest degree of a polynomial supported by the SRS, i.e., $n+2$ in PLONK. Since the only difference to the original protocol is that in step 10 the verifier uses $\mathsf{g}_1^{T^*(\tau)} = \mathsf{g}_1^{T_1^*(\tau)} \cdot \big(\mathsf{g}_1^{T_2^*(\tau)}\big)^{\delta^{n+2}} \cdot \big(\mathsf{g}_1^{T_3^*(\tau)}\big)^{\delta^{2n+4}}$ instead of just $\mathsf{g}_1^{T(\tau)}$ to compute the commitment to the linearization polynomial L, we can simply replace the quotient polynomial T by T^* in our analysis, resulting in the same upper bound for knowledge soundness as in Equation (5.35).

On a final note, we want to highlight a difference between our description of PLONK and the original by Gabizon et al. [GWC19, Sec. 8.3], which uses a different approach to split

the quotient polynomial T of degree $\leq 3n+5$ into T'_1, T'_2, T'_3. Instead of using the unique degree-$(n+1)$ polynomials satisfying $T(X) = T'_1(X) + X^{n+2} \cdot T'_2(X) + X^{2n+4} \cdot T'_3(X)$, they restrict T'_1 and T'_2 to be of degree $< n$ while letting T'_3 have degree $n+5$ such that $T(X) = T'_1(X) + X^n \cdot T'_2(X) + X^{2n} \cdot T'_3(X)$, resulting in an SRS of degree $n+5$ as opposed to the $n+2$ achieved in our version of PLONK.

7.1 The Vulnerability

Since the authors of PLONK never constructed a simulator to formally verify that PLONK is zero-knowledge, our attempt to devise such a simulator led to the discovery of a vulnerability in their original implementation of step 5 in Construction 7.1. It should be stressed that the version of PLONK presented in Construction 7.1 already includes our suggested fix.

In this step, the prover decomposes the computed quotient polynomial $T \in \mathbb{F}_p[X]$ of degree $\leq 3n+5$ into three lower-degree polynomials as an optimization that keeps the SRS size minimal with respect to the circuit size n. However, in the previous version of PLONK, the prover directly commits to the deterministic polynomials $T'_1, T'_2, T'_3 \in \mathbb{F}_p^{(\leq n+1)}[X]$ instead of the randomized polynomials $T_1, T_2, T_3 \in \mathbb{F}_p^{(\leq n+2)}$. The problem with this approach is that even though a simulator is able to compute the correct evaluation of $T(\tau)$, it does not have enough information to compute the correct values $T'_1(\tau), T'_2(\tau), T'_3(\tau)$ satisfying $T(\tau) = T'_1(\tau) + \tau^{n+2} \cdot T'_2(\tau) + \tau^{2n+4} \cdot T'_3(\tau)$. While there are p^2 possible triples of values in $\mathbb{F}_p^3$ satisfying this equation, only one of them corresponds to the correct distribution of $T'_1(\tau), T'_2(\tau), T'_3(\tau)$ in the execution of the real protocol.

We fixed this issue in collaboration with the authors of PLONK by randomizing these three polynomials in a way that preserves their composition of T while enabling a correct simulation. We discuss this fix in more detail in the next section, where we present our simulator.

7.2 Statistical Zero Knowledge

To show that PLONK is statistically zero-knowledge, we will rely on the following lemma, which explains why the prover's witness polynomials are randomized by adding the product of a random blinding polynomial and Z_H.

Lemma 7.1. *Let $S \subset \mathbb{F}_p$ and $Z_S(X) := \prod_{a \in S}(X - a)$. Fix a polynomial $f \in \mathbb{F}_p[X]$ and any distinct values $x_1, \ldots, x_k \in \mathbb{F}_p \setminus S$. Then the following distribution is uniform in $\mathbb{F}_p^k$:*

1. *Choose random blinding scalars $\rho_0, \ldots, \rho_{k-1} \leftarrow \mathbb{F}_p$ and define the polynomial*

$$\tilde{f}(X) := f(X) + Z_S(X)(\rho_0 + \rho_1 X + \cdots + \rho_{k-1} X^{k-1}).$$

2. *Output $(\tilde{f}(x_1), \ldots, \tilde{f}(x_k)) \in \mathbb{F}_p^k$.*

Proof. Define the blinding polynomial $\rho(X) := \sum_{i=0}^{k-1} \rho_i X^i$. For all $i \in [k]$, we have

$$\tilde{f}(x_i) = f(x_i) + Z_S(x_i)\rho(x_i),$$

where the values $f(x_i), Z_S(x_i)$ are fixed and $Z_S(x_i) \neq 0$ (due to $x_i \in \mathbb{F}_p \setminus S$). Since the product of any fixed $a \in \mathbb{F}_p^*$ and random $b \in \mathbb{F}_p$ is uniform in $\mathbb{F}_p$, all we need to show is that the values $\rho(x_1), \ldots, \rho(x_k)$ are distributed independently and uniformly in $\mathbb{F}_p$, which is a well-known claim for any random degree-$(k-1)$ polynomial such as $\rho \in \mathbb{F}_p^{(\leq k-1)}[X]$. One way to see this, is by fixing any distinct $x_1, \ldots, x_k \in \mathbb{F}_p$ and observing that for any choice of $y_1, \ldots, y_k \in \mathbb{F}_p$ there is a unique degree-$(k-1)$ polynomial interpolating the points $(x_1, y_1), \ldots, (x_k, y_k)$. Formally, there are p^k distinct degree-$(k-1)$ polynomials over $\mathbb{F}_p$, which corresponds to the number of choices for $y_1, \ldots, y_k \in \mathbb{F}_p$. Furthermore, there cannot be any two distinct polynomials $f_1 \not\equiv f_2 \in \mathbb{F}_p^{(\leq k-1)}[X]$ interpolating the same set of points $(x_1, y_1), \ldots, (x_k, y_k)$, since otherwise the non-zero, degree-$(k-1)$ polynomial $f_1 - f_2$ would have at least k roots, which is a contradiction. □

As a consequence, the number of independent and uniform evaluations of a polynomial randomized in this way depends on the degree of the used blinding polynomial $\sum_i \rho_i X^i$, i.e., a constant blinding polynomial allows for just a single evaluation, a linear polynomial for two, and so forth. This also explains why the wire polynomials A, B, C are randomized using linear blinding polynomials, while the permutation check polynomial Φ uses a quadratic blinding polynomial. More specifically, A, B, C, and Φ have to account for their commitments (which equate to evaluations at τ) as well as the openings at the evaluation challenge δ (respectively $\delta\omega$ in the case of Φ), but Φ additionally has to compensate the commitment to the quotient polynomial T (or its decomposition into T_1, T_2, T_3), since the value of $T(\tau)$ depends on $\Phi(\tau\omega)$ (cf. step 5 in Construction 7.1).

An important condition of the lemma is that the evaluation points come from $\mathbb{F}_p \setminus H$, which means that both the SRS trapdoor τ and the evaluation challenge δ must not be in H. Since $H := \langle\omega\rangle = \{\omega^1, \omega^2, \ldots, \omega^n\}$ with $n = \mathsf{poly}(\lambda)$, we have $|H| \,/\, |\mathbb{F}_p| = \mathsf{negl}(\lambda)$, i.e., we will be able to ignore this case when constructing a simulator for PLONK and still obtain statistical zero knowledge. Also, since H is a multiplicative order-n subgroup of $\mathbb{F}_p$, this ensures that the remaining evaluation points $\tau\omega$ and $\delta\omega$ come from $\mathbb{F}_p \setminus H$ as well. To see this, let g be a generator of $\mathbb{F}_p^*$ such that $\omega = \mathsf{g}^r$ with $r := (p-1)/n$. Then the set H can be rewritten as $\{\mathsf{g}^0, \mathsf{g}^r, \mathsf{g}^{2r}, \ldots, \mathsf{g}^{(n-1)r}\}$. Ignoring the case $\tau = 0$, which immediately results in $\tau\omega \notin H$, we have $\tau = \mathsf{g}^s$ for some $s \in \mathbb{Z}_{p-1}$ with $s \neq 0 \pmod r$. Thus, we get $\tau\omega = \mathsf{g}^s \cdot \mathsf{g}^r = \mathsf{g}^{s+r}$, but $s + r \neq 0 \pmod r$, which implies $\tau\omega \notin H$ as required. Analogously, we get $\delta\omega \notin H$. This means that, as long as $\tau, \delta \in \mathbb{F}_p \setminus H$, the commitments to the witness polynomials A, B, C, Φ using evaluations at τ as well as the evaluations of these polynomials at the points δ and $\delta\omega$ are distributed independently and uniformly in $\mathbb{F}_p$, hiding any information about the prover's witness.

A Simulator for PLONK. With all the necessary tools established, we are now ready to present our simulator $\mathcal{S}$ for PLONK. Note that we construct a simulator for the

interactive PLONK protocol described in Construction 7.1. As discussed in more detail in the next section, using the Fiat–Shamir heuristic [FS87] it can be trivially transformed into a simulator for the non-interactive variant in the random oracle model [BR93]. We will assume that both the SRS trapdoor τ and the evaluation challenge δ come from $\mathbb{F}_p \setminus H$. In this case, $\mathcal{S}$ perfectly simulates PLONK when ignoring the aborts caused by denominators being zero in the computation of the permutation check polynomial Φ (cf. step 3 in Construction 7.1) as well as the cases when $\delta = \tau$ or $\delta\omega = \tau$, which is why we attain *statistical* zero knowledge. Furthermore, we only give the second part of the simulator from Definition 2.7, implicitly assuming it runs the Setup algorithm to create a well-formed srs with a uniform trapdoor $\tau \in \mathbb{F}_p \setminus H$ as its first output.

Construction 7.2: Simulator for PLONK.

$\mathcal{S}(\tau, \mathsf{pp}, (x_i)_{i\in[\ell]})$:

1. Choose random $a_\tau, b_\tau, c_\tau \leftarrow \mathbb{F}_p$ and send the commitments $c_A := \mathsf{g}_1^{a_\tau}$, $c_B := \mathsf{g}_1^{b_\tau}$, $c_C := \mathsf{g}_1^{c_\tau}$ to $\mathcal{V}$.
2. Receive the permutation challenges $\beta, \gamma \in \mathbb{F}_p$ from $\mathcal{V}$.
3. Choose random $\phi_\tau \leftarrow \mathbb{F}_p$ and send the commitment $c_\Phi := \mathsf{g}_1^{\phi_\tau}$ to $\mathcal{V}$.
4. Receive the quotient challenge $\alpha \in \mathbb{F}_p$ from $\mathcal{V}$.
5. Choose random $\phi_{\tau\omega} \leftarrow \mathbb{F}_p$ (if $\tau = 0$, set $\phi_{\tau\omega} := \phi_\tau$ instead) and compute the simulated evaluation of the quotient polynomial T at τ as

$$t_\tau := \frac{1}{Z_H(\tau)} \left(\begin{array}{c} \big[S_{\mathsf{L}}(\tau)a_\tau + S_{\mathsf{R}}(\tau)b_\tau + S_{\mathsf{O}}(\tau)c_\tau + S_{\mathsf{M}}(\tau)a_\tau b_\tau + S_{\mathsf{C}}(\tau) + S_{\mathsf{PI}}(\tau)\big] \\ +\alpha\Big[\begin{array}{c} \phi_\tau(a_\tau+\beta\tau+\gamma)(b_\tau+\beta k_1\tau+\gamma)(c_\tau+\beta k_2\tau+\gamma) \\ -\phi_{\tau\omega}(a_\tau+\beta S_{\sigma,1}(\tau)+\gamma)(b_\tau+\beta S_{\sigma,2}(\tau)+\gamma)(c_\tau+\beta S_{\sigma,3}(\tau)+\gamma) \end{array}\Big] \\ +\alpha^2\big[\mathcal{L}_1(\tau)(\phi_\tau - 1)\big] \end{array} \right).$$

   Choose random $t_2, t_3 \leftarrow \mathbb{F}_p$ and compute

$$t_1 := t_\tau - \tau^{n+2} \cdot t_2 - \tau^{2n+4} \cdot t_3$$

   such that $t_\tau = t_1 + \tau^{n+2} \cdot t_2 + \tau^{2n+4} \cdot t_3$. Send the commitments $c_{T_1} := \mathsf{g}_1^{t_1}$, $c_{T_2} := \mathsf{g}_1^{t_2}$, $c_{T_3} := \mathsf{g}_1^{t_3}$ to $\mathcal{V}$.
6. Receive the evaluation challenge $\delta \in \mathbb{F}_p \setminus H$ from $\mathcal{V}$.
7. If $\delta = \tau$ or $\delta\omega = \tau$, abort. Otherwise, choose random $a_\delta, b_\delta, c_\delta, \phi_{\delta\omega} \leftarrow \mathbb{F}_p$ and send the values $a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta} := S_{\sigma,1}(\delta), s_{\sigma_2,\delta} := S_{\sigma,2}(\delta), \phi_{\delta\omega}$ to $\mathcal{V}$.
8. Receive the opening challenge $\varepsilon \in \mathbb{F}_p$ from $\mathcal{V}$.

9. Compute the simulated evaluation of the linearization polynomial L at τ as

$$l_\tau := \begin{pmatrix} \big[a_\delta S_{\mathsf{L}}(\tau)+b_\delta S_{\mathsf{R}}(\tau)+c_\delta S_{\mathsf{O}}(\tau)+a_\delta b_\delta S_{\mathsf{M}}(\tau)+S_{\mathsf{C}}(\tau)+S_{\mathsf{PI}}(\delta)\big] \\ +\alpha\Big[\begin{matrix}\phi_\tau(a_\delta+\beta\delta+\gamma)(b_\delta+\beta k_1\delta+\gamma)(c_\delta+\beta k_2\delta+\gamma) \\ -\phi_{\delta\omega}(a_\delta+\beta S_{\sigma,1}(\delta)+\gamma)(b_\delta+\beta S_{\sigma,2}(\delta)+\gamma)(c_\delta+\beta S_{\sigma,3}(\tau)+\gamma)\end{matrix}\Big] \\ +\alpha^2\big[\mathcal{L}_1(\delta)(\phi_\tau-1)\big]-Z_H(\delta)\big[t_1+\delta^{n+2}t_2+\delta^{2n+4}t_3\big] \end{pmatrix}.$$

Then, compute the simulated opening proofs

$$\pi_1 := \mathsf{g}_1^{\frac{1}{\tau-\delta}\left(\begin{matrix} l_\tau+\varepsilon(a_\tau-a_\delta)+\varepsilon^2(b_\tau-b_\delta)+\varepsilon^3(c_\tau-c_\delta) \\ +\varepsilon^4(S_{\sigma,1}(\tau)-S_{\sigma,1}(\delta))+\varepsilon^5(S_{\sigma,2}(\tau)-S_{\sigma,2}(\delta))\end{matrix}\right)},$$
$$\pi_2 := \mathsf{g}_1^{(\phi_\tau-\phi_{\delta\omega})/(\tau-\delta\omega)},$$

and send π_1, π_2 to $\mathcal{V}$. The full output of $\mathcal{S}$ is

$$\tilde{\pi}_{\text{PLONK}} := \begin{pmatrix} c_A, c_B, c_C, c_\Phi, c_{T_1}, c_{T_2}, c_{T_3}, \pi_1, \pi_2, \\ a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega} \end{pmatrix} \in \mathbb{G}_1^9 \times \mathbb{F}_p^6.$$

Proof. We now argue why $\mathcal{S}$ perfectly simulates PLONK when ignoring the aborts caused by denominators in the computation of the permutation check polynomial Φ (in step 3 of Construction 7.1) being zero, as well as the cases $\delta = \tau$ or $\delta\omega = \tau$. Specifically, we will show that in this case all the elements in a real PLONK proof π_{PLONK} are either uniform random or determined by the verifier equations, and that a proof $\tilde{\pi}_{\text{PLONK}}$ generated by our simulator $\mathcal{S}$ has exactly the same distribution.

To this end, we begin by analyzing the distribution of all the elements contained in a real PLONK proof

$$\pi_{\text{PLONK}} := \begin{pmatrix} \mathsf{g}_1^{A(\tau)}, \mathsf{g}_1^{B(\tau)}, \mathsf{g}_1^{C(\tau)}, \mathsf{g}_1^{\Phi(\tau)}, \mathsf{g}_1^{T_1(\tau)}, \mathsf{g}_1^{T_2(\tau)}, \mathsf{g}_1^{T_3(\tau)}, \mathsf{g}_1^{Q_1(\tau)}, \mathsf{g}_1^{Q_2(\tau)}, \\ A(\delta), B(\delta), C(\delta), S_{\sigma,1}(\delta), S_{\sigma,2}(\delta), \Phi(\delta\omega) \end{pmatrix} \in \mathbb{G}_1^9 \times \mathbb{F}_p^6.$$

Note that the values $S_{\sigma,1}(\delta), S_{\sigma,2}(\delta)$ are just the evaluations of the public permutation polynomials $S_{\sigma,1}, S_{\sigma,1}$ at the evaluation challenge δ, and hence correctly simulated by default. Furthermore, $\mathsf{g}_1^{A(\tau)}, \mathsf{g}_1^{B(\tau)}, \mathsf{g}_1^{C(\tau)}, \mathsf{g}_1^{\Phi(\tau)}, A(\delta), B(\delta), C(\delta), \Phi(\delta\omega)$ are just uniform group/field elements due to the randomization of the polynomials A, B, C, Φ according to Lemma 7.1. More precisely, by our assumption that $\tau, \delta \in \mathbb{F}_p \setminus H$, it follows from Lemma 7.1 as well as the subsequent explanations that the evaluations $A(\tau), B(\tau), C(\tau), \Phi(\tau)$ are distributed independently and uniformly in $\mathbb{F}_p$. This is exactly how $\mathcal{S}$ chooses $a_\tau, b_\tau, c_\tau, \phi_\tau \in \mathbb{F}_p$ in steps 1 and 3 to produce the commitments to the polynomials A, B, C, Φ. The same is true in step 7, where $\mathcal{S}$ again outputs uniform values $a_\delta, b_\delta, c_\delta, \phi_{\delta\omega} \in \mathbb{F}_p$ as the evaluations $A(\delta), B(\delta), C(\delta), \Phi(\delta\omega)$.

Directly simulating the commitment to the quotient polynomial $\mathsf{g}_1^{T(\tau)}$ in step 5 is simple, as $T(\tau)$ is a deterministic function in τ with all the necessary inputs known to the simulator (cf. the computation of t_τ in step 5 of $\mathcal{S}$). On the contrary, simulating the

commitments to the decomposed polynomials T_1, T_2, T_3 satisfying

$$T(X) = T_1(X) + X^{n+2} \cdot T_2(X) + X^{2n+4} \cdot T_3(X)$$

is much more delicate. As explained in the previous section, this is exactly where the specification of PLONK prior to our fix, which instead commits to the deterministic polynomials T'_1, T'_2, T'_3, leaks information about the prover's witness polynomials, and thus cannot be perfectly simulated. But let us instead argue why $\mathcal{S}$ correctly simulates the commitments to the now randomized polynomials T_1, T_2, T_3 in step 5. By the randomness of $\rho_{10}, \rho_{11} \in \mathbb{F}_p$, both $T_2(\tau) = T'_2(\tau) - \rho_{10} + \rho_{11} \cdot \tau^{n+2}$ and $T_3(\tau) = T'_3(\tau) - \rho_{11}$ are distributed independently and uniformly in $\mathbb{F}_p$. The value of $T_1(\tau) = T'_1(\tau) + \rho_{10} \cdot \tau^{n+2}$ is then uniquely determined by the equality $T(\tau) = T_1(\tau) + \tau^{n+2} \cdot T_2(\tau) + \tau^{2n+4} \cdot T_3(\tau)$, which is exactly how $\mathcal{S}$ chooses t_1, t_2, t_3 to compute its commitments $c_{T_1} := \mathsf{g}_1^{t_1}$, $c_{T_2} := \mathsf{g}_1^{t_2}$, $c_{T_3} := \mathsf{g}_1^{t_3}$.

Finally, $\mathcal{S}$ also correctly simulates the opening proofs $\mathsf{g}_1^{Q_1(\tau)}, \mathsf{g}_1^{Q_2(\tau)}$ in step 9 when $\delta \neq \tau$ and $\delta\omega \neq \tau$, since both of them are deterministic functions in τ in this case with all the necessary inputs known to the simulator. For example, see how $\mathcal{S}$ computes the value l_τ, which is the evaluation of the linearization polynomial $L(\tau)$ required for $\mathsf{g}_1^{Q_1(\tau)}$.

Having established that $\mathcal{S}$ perfectly simulates PLONK conditioned on $\tau, \delta \in \mathbb{F}_p \setminus H$, $\tau \neq \delta$, $\tau \neq \delta\omega$, and Φ being well-defined, i.e., $\Phi \in \mathbb{F}_p[X]$, we can turn this into a formal proof of statistical (honest-verifier) zero knowledge as defined in Definition 2.7. Recall from Section 6.2 that the abort probability due to Φ being undefined can be bounded by kn/p, which corresponds to $3n/p$ in the actual PLONK protocol. Let F denote the event in which our simulator $\mathcal{S}$ is unable to produce correctly distributed proofs. This is the case with probability at most

$$\begin{aligned} \Pr[F] &= \Pr\big[\tau \in H \vee \delta \in H \vee \tau = \delta \vee \tau = \delta\omega \vee \Phi \notin \mathbb{F}_p[X]\big] && (7.1)\\ &\leq \frac{n+n+1+1+3n}{p} = \frac{5n+2}{p}. && (7.2) \end{aligned}$$

With this, we can show that the difference of the two probabilities from Definition 2.7 is negligible as well. Using $I_\mathcal{P}$ and $I_\mathcal{S}$ to denote the event $b = 1 \wedge (\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}_{\mathrm{PLONK}n}$ when the adversary interacts with the honest prover $\mathcal{P}$ and with our simulator $\mathcal{S}$, respectively, we have $\Pr[I_\mathcal{P} \wedge \bar{F}] = \Pr[I_\mathcal{S} \wedge \bar{F}]$ by the arguments laid out above. Then:

$$\begin{aligned} \big|\Pr[I_\mathcal{P}] - \Pr[I_\mathcal{S}]\big| &= \big|(\Pr[I_\mathcal{P} \wedge F] + \Pr[I_\mathcal{P} \wedge \bar{F}]) - (\Pr[I_\mathcal{S} \wedge F] + \Pr[I_\mathcal{S} \wedge \bar{F}])\big| && (7.3)\\ &= \big|\Pr[I_\mathcal{P} \wedge F] - \Pr[I_\mathcal{S} \wedge F]\big| && (7.4)\\ &\leq \frac{5n+2}{p} = \mathsf{negl}(\lambda) && (7.5) \end{aligned}$$

The final inequality follows from the fact that both $\Pr[I_\mathcal{P} \wedge F]$ and $\Pr[I_\mathcal{S} \wedge F]$ are at most $\Pr[F]$, and so their difference cannot be any larger. This finishes the proof of PLONK's statistical honest-verifier zero knowledge. □

7.3 The PLONK zk-SNARK

So far, we have only dealt with interactive public-coin protocols, including our description of the PLONK protocol in Construction 7.1. To turn PLONK non-interactive, Gabizon et al. apply the Fiat–Shamir transformation [FS87], allowing the prover to obtain random challenges without interacting with the verifier. For this purpose, let $\mathcal{H}\colon \{0,1\}^* \to \mathbb{F}_p$ be a collision-resistant hash function modeled as a random oracle [BR93] for the sake of proving security of this non-interactive variant. Furthermore, let inputs denote the concatenation of the SRS, the common preprocessed input and all the public inputs $(x_i)_{i\in[\ell]}$. Each challenge is then computed by evaluating $\mathcal{H}$ on the concatenation of inputs and all the proof elements written by the prover up to that point in time. Note that including all of this information in inputs is crucial to prevent certain attacks allowing a malicious prover to forge proofs for statements without knowing a witness as explained in [Mil22], effectively breaking PLONKS's knowledge soundness.

As a result, we finally obtain the PLONK zk-SNARK:

Construction 7.3: The PLONK zk-SNARK.

- $\mathsf{Setup}(1^\lambda, n)$: Same as in Construction 7.1.
- $\mathsf{Preprocess}(\mathsf{srs}, \mathbf{i})$: Same as in Construction 7.1.
- $\mathsf{Prove}(\mathsf{pp}, (x_i)_{i\in[\ell]}, (w_i)_{i\in[3n]})$:
  1. Same as step 1 in Construction 7.1.
  2. Compute the permutation challenges $\beta, \gamma \in \mathbb{F}_p$ as
  $$\beta := \mathcal{H}(\mathsf{inputs}, c_A, c_B, c_C, 0),\ \ \gamma := \mathcal{H}(\mathsf{inputs}, c_A, c_B, c_C, 1).$$
  3. Same as step 3 in Construction 7.1.
  4. Compute the quotient challenge $\alpha \in \mathbb{F}_p$ as
  $$\alpha := \mathcal{H}(\mathsf{inputs}, c_A, c_B, c_C, c_\Phi).$$
  5. Same as step 5 in Construction 7.1.
  6. Compute the evaluation challenge $\delta \in \mathbb{F}_p$ as
  $$\delta := \mathcal{H}(\mathsf{inputs}, c_A, c_B, c_C, c_\Phi, c_{T_1}, c_{T_2}, c_{T_3}).$$
  7. Same as step 7 in Construction 7.1.
  8. Compute the opening challenge $\varepsilon \in \mathbb{F}_p$ as
  $$\varepsilon := \mathcal{H}(\mathsf{inputs}, c_A, c_B, c_C, c_\Phi, c_{T_1}, c_{T_2}, c_{T_3}, a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega}).$$

9. Same as step 9 in Construction 7.1.
10. Output the full proof

$$\pi_{\text{PLONK}} := \begin{pmatrix} c_A, c_B, c_C, c_\Phi, c_{T_1}, c_{T_2}, c_{T_3}, \pi_1, \pi_2, \\ a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega} \end{pmatrix} \in \mathbb{G}_1^9 \times \mathbb{F}_p^6.$$

- $\mathsf{Verify}(\mathsf{vp}, (x_i)_{i\in[\ell]}, \pi_{\text{PLONK}})$: Same as step 10 in Construction 7.1, except that the multipoint evaluation challenge $\zeta \in \mathbb{F}_p$ is computed deterministically as

$$\zeta := \mathcal{H}(\mathsf{inputs}, c_A, c_B, c_C, c_\Phi, c_{T_1}, c_{T_2}, c_{T_3}, a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega}, \pi_1, \pi_2).$$

Regarding the succinctness property stated in Definition 2.12, for a fixed value of the security parameter λ, the PLONK zk-SNARK achieves constant-size proofs composed of only 9 group elements and 6 field elements, independent of the size of the proved statement. In practice, this amounts to approximately 480 bytes, as stated in the introduction.

Note that, unlike the final probabilistic verification check in Construction 7.1, where $\mathcal{V}$ samples a random multipoint evaluation challenge $\zeta \leftarrow \mathbb{F}_p$, this step has now been made deterministic by applying the Fiat–Shamir heuristic one more time to obtain ζ. This has the advantage of keeping the verification algorithm Verify fully deterministic (as required by our definition of zk-SNARK, see Definition 2.8), which is necessary in certain applications such as smart contracts and proofs of verification (e.g., when modeling the verifier algorithm as an arithmetic circuit to obtain a recursive zk-SNARK).

Finally, we are ready to obtain the main result of this thesis, as stated in Theorem 1.1:

The PLONK zk-SNARK is statistically zero-knowledge.

By applying the Fiat–Shamir transform to the simulator $\mathcal{S}$ in Construction 7.2 and then using the same arguments as in Section 7.2, we obtain a simulator which statistically simulates proofs of the PLONK zk-SNARK, yielding the theorem.

As a final contribution, combining the results of all our security proofs in this thesis, we will derive a precise upper bound on PLONK's knowledge soundness error in the algebraic group model (AGM). First, we unfold the knowledge soundness error obtained in Section 6.1 for the general permutation argument protocol from Construction 6.1. Using the results of Sections 4.3 and 5.2.1, it holds that:

$$\frac{kn}{p} + \epsilon_{H\text{-Ranged}} \leq \frac{kn}{p} + 2 \cdot \frac{\max(\hat{d}dD, \hat{d} + |H|) + k' - 1}{p} + \epsilon_{\mathsf{Open}} \tag{7.6}$$

$$\leq \frac{kn}{p} + 2 \cdot \frac{\max(\hat{d}dD, \hat{d} + n) + k' - 1}{p} + \frac{k'' - 1}{p} + \frac{\ell - 1}{p} + \epsilon_{d'\text{-DLog}}, \tag{7.7}$$

where n is the size of the range H and k is the number of polynomials in the permutation argument; $\hat{d}, d, D$ are specified by the relation $\mathcal{R}^{H}_{(d,D,t,\ell)}$ from Section 5.2 and k' is the number of checked H-ranged polynomial identities; k'' and ℓ are the number of evaluated polynomials and the number of distinct evaluation points in the Open protocol, respectively; and $\epsilon_{d'\text{-DLog}}$ is the hardness of the d'-DLog problem, where d' is the degree of the used SRS.

Plugging in all the respective values of the PLONK protocol, i.e.,

$$k = 3, \hat{d} = n + 2, d = 1, D = 4, k' = 3, k'' = 7, \ell = 2, d' = n + 2,$$

we arrive at the following upper bound on the knowledge soundness error of PLONK:

$$\epsilon_{\text{PLONK}} \leq \frac{3n}{p} + 2 \cdot \frac{\max\big(4(n+2), 2n+2\big) + 2}{p} + \frac{6}{p} + \frac{1}{p} + \epsilon_{(n+2)\text{-DLog}} \tag{7.8}$$

$$= \frac{11n + 27}{p} + \epsilon_{(n+2)\text{-DLog}} \tag{7.9}$$

Assuming hardness of the $(n+2)$-DLog problem and $n = \mathsf{poly}(\lambda)$, the overall bound is negligible in the security parameter λ with a linear dependence on the circuit size n. We stress that this bound is obtained in the algebraic group model, and thus does not necessarily reflect the actual level of security offered by PLONK in the real world. In particular, it only holds up against adversarial strategies which behave according to the rules of this model, i.e., *algebraic adversaries* which can only derive new group elements as linear combinations of the group elements contained in PLONK's structured reference string. We leave a more in-depth discussion of this issue as an open problem.

CHAPTER 8

Conclusion

In this thesis, we conducted a formal security analysis of the universal and fully-succinct zk-SNARK PLONK introduced by Gabizon et al. [GWC19]. Even though this influential state-of-the-art construction is implemented in several real-world applications, there was no formal security proof of its zero knowledge property. Consequently, we were able to discover a vulnerability in PLONK's zero knowledge implementation. Our subsequent disclosure and proposed fix of this vulnerability led to a security patch of the PLONK protocol.

As the main contribution of this thesis, we gave a formal security proof establishing that the resulting version of PLONK achieves *statistical zero knowledge*. Towards this goal, we showed how to construct a simulator capable of generating proofs which are distributed identically to PLONK proofs up to a negligible statistical difference.

Furthermore, our modular security analysis of all the building blocks involved in the construction of the PLONK protocol, including formal security proofs for each of them, allowed us to establish a precise upper bound on PLONK's knowledge soundness error in the *algebraic group model* of Fuchsbauer et al. [FKL18]. Since this is the idealized model used in the original proof of PLONK's knowledge soundness, our result helps towards a better understanding of the security guarantees of PLONK. Overall, the work in this thesis highlights the importance of rigorous definitions and formal security proofs.

We gave a positive result by proving that the patched version of PLONK achieves zero knowledge. We leave the negative result of formally disproving that the previous version of PLONK is zero-knowledge to future work. It remains an open problem to devise an attack which exploits the vulnerability we found in PLONK's old zero knowledge implementation. We are confident that one can show it does not even satisfy *witness indistinguishability*. In this weaker notion, the adversary defines a statement with two different witnesses and is given a proof computed under one of them chosen at random. Its goal is then to determine which witness was used.

APPENDIX A

Appendix

A.1 Proof of Permutation Argument

We provide our own proof[1] of Lemma 6.1 from [GWC19, Claim A.1], which is the foundation of PLONK's permutation argument.

Proof. Suppose Equation (6.4) from the lemma is true, i.e.,

$$\prod_{i\in[n]} (a_i + \beta \cdot s_i + \gamma) = \prod_{i\in[n]} (b_i + \beta \cdot s_{\sigma(i)} + \gamma) \tag{A.1}$$

holds with probability greater than n/p over the choice of $\beta, \gamma \in \mathbb{F}_p$. Let us express both sides of the equality as the following two bivariate polynomials, each of total degree n:

$$f(X,Y) := \prod_{i\in[n]} (a_i + X \cdot s_i + Y),\ g(X,Y) := \prod_{i\in[n]} (b_i + X \cdot s_{\sigma(i)} + Y).$$

Observe that their sets of roots are $\{(\beta, -(a_1 + \beta s_1)), \ldots, (\beta, -(a_n + \beta s_n)) \mid \beta \in \mathbb{F}_p\}$ and $\{(\beta, -(b_1 + \beta s_{\sigma(1)})), \ldots, (\beta, -(b_n + \beta s_{\sigma(n)})) \mid \beta \in \mathbb{F}_p\}$, respectively. Then, since Equation (A.1) is assumed to hold with probability greater than n/p, it follows from the Schwartz–Zippel lemma (Lemma 2.1) that these sets of roots must be the same. This means that for any fixed value of β, the values $(a_1 + \beta s_1), \ldots, (a_n + \beta s_n)$ and $(b_1 + \beta s_{\sigma(1)}), \ldots, (b_n + \beta s_{\sigma(n)})$ are equal, but not necessarily in this order.

Now, assume $a_{\sigma(i^*)} \neq b_{i^*}$ for some $i^* \in [n]$. Then, by the previous observation, for every $\beta \in \mathbb{F}_p$ there must be some $j \in [n]$ such that $a_{\sigma(i^*)} + \beta s_{\sigma(i^*)} = b_j + \beta s_{\sigma(j)}$, or equivalently

$$a_{\sigma(i^*)} - b_j + \beta(s_{\sigma(i^*)} - s_{\sigma(j)}) = 0. \tag{A.2}$$

[1] At the time of writing this proof, the original proof of Claim A.1 given by Gabizon et al. was not fully satisfactory. In the most recent version of the PLONK paper [GWC19, Appendix A], a proof by Tohru Kohrita is used instead which is even simpler and more elegant than ours.

Furthermore, we know that $j \neq i^*$, since $a_{\sigma(i^*)} \neq b_{i^*}$. Fixing j and looking at this as a polynomial equation in $X = \beta$, we see that, unless the left-hand side is the zero polynomial (which cannot be the case here due to $j \neq i^*$), there is at most one value of β for which the equality holds. The same argument applies to any of the $n-1$ values of $j \neq \sigma(i^*) \in [n]$, that is, there are at most $n-1$ values of β satisfying Equation (A.2). Since $n-1 < |\mathbb{F}_p|$, this in turn contradicts our assumption that $a_{\sigma(i^*)} \neq b_{i^*}$. □

A.2 Proof of Alternative Lagrange Polynomial Formula

Let $S := \{x_1, \ldots, x_n\} \subseteq \mathbb{F}_p$ be a set of n distinct elements, and $Z_S(X) := \prod_{i\in[n]}(X - x_i)$. In Lemma 2.2, we stated that the i-th Lagrange polynomial $\mathcal{L}_i(X) := \prod_{j\in[n],j\neq i} \frac{X-x_j}{x_i-x_j}$ can then be rewritten as

$$\mathcal{L}_i(X) = \frac{Z_S(X)}{Z'_S(x_i)(X - x_i)},$$

where Z'_S is the *formal derivative* of the polynomial Z_S. Recall that the formal derivative of a polynomial $f(X) := \sum_{i=0}^{d} f_i X^i \in \mathbb{F}_p[X]$ is simply defined as $f'(X) := \sum_{i=1}^{d} i f_i X^{i-1}$.

We now prove Lemma 2.2, showing that the above substitution is indeed correct.

Proof. Since $Z_S(X)/(X - x_i) = \prod_{j\in[n],j\neq i}(X - x_j)$, all that remains to be shown to verify the correctness of this substitution is

$$Z'_S(x_i) = \prod_{j\in[n],j\neq i} (x_i - x_j). \tag{A.3}$$

For this purpose, we will adapt the *general product rule* from real analysis to the setting of formal derivatives. Given n polynomials $f_1, \ldots, f_n \in \mathbb{F}_p[X]$, it states:

$$\left(\prod_{j\in[n]} f_j\right)' = \sum_{j\in[n]} f'_j \cdot \prod_{k\in[n],k\neq j} f_k \tag{A.4}$$

Applying this to $Z_S(X) = \prod_{j\in[n]}(X - x_j)$, we obtain:

$$Z'_S(X) = \left(\prod_{j\in[n]} (X - x_j)\right)' = \sum_{j\in[n]} (X - x_j)' \cdot \prod_{k\in[n],k\neq j} (X - x_k) = \sum_{j\in[n]} Z_{S\setminus\{x_j\}}(X)$$

Observe that when evaluating $\sum_{j\in[n]} Z_{S\setminus\{x_j\}}(X)$ at $X = x_i$, each summand except for $Z_{S\setminus\{x_i\}}(x_i)$ contains the factor $(x_i - x_i) = 0$ and hence vanishes, yielding Equation (A.3) as required. □

Product rule for formal derivatives. For completeness, we also give a proof of the general product rule for formal derivatives stated in Equation (A.4).

Proof. First, we show that the simple product rule $(uv)' = u'v + uv'$ also holds for formal derivatives. Let $u, v \in \mathbb{F}_p[X]$ be two polynomials of degree d_u and d_v, respectively. Then:

$$\begin{aligned}
(uv)' &= \left(\left(\sum_{i=0}^{d_u} u_i X^i\right)\left(\sum_{i=0}^{d_v} v_i X^i\right)\right)' = \left(\sum_{i=0}^{d_u}\sum_{j=0}^{d_v} u_i v_j X^{i+j}\right)' \\
&= \sum_{i=0}^{d_u}\sum_{j=0}^{d_v}(u_i v_j X^{i+j})' = \sum_{i=0}^{d_u}\sum_{j=0}^{d_v}(i+j) u_i v_j X^{i+j-1} \\
&= \left(\sum_{i=1}^{d_u}\sum_{j=0}^{d_v} i u_i v_j X^{i+j-1}\right) + \left(\sum_{i=0}^{d_u}\sum_{j=1}^{d_v} j u_i v_j X^{i+j-1}\right) \\
&= \left(\sum_{i=1}^{d_u} i u_i X^{i-1}\right)\left(\sum_{i=0}^{d_v} v_i X^i\right) + \left(\sum_{i=0}^{d_u} u_i X^i\right)\left(\sum_{i=1}^{d_v} i v_i X^{i-1}\right) \\
&= u'v + uv'
\end{aligned}$$

Having established the simple product rule, we proceed by proving the general product rule for formal derivatives via induction on the number of polynomials n.

- Base case: $n = 2$.

$$\left(\prod_{i\in[2]} f_i\right)' = (f_1 f_2)' = f_1' f_2 + f_1 f_2' = \sum_{i\in[2]} f_i' \cdot \prod_{j\in[2], j\neq i} f_j$$

- Induction step: Suppose Equation (A.4) holds for $n \geq 2$ polynomials.

$$\begin{aligned}
\left(\prod_{i\in[n+1]} f_i\right)' &= \left(\prod_{i\in[n]} f_i \cdot f_{n+1}\right)' \\
&= \left(\prod_{i\in[n]} f_i\right)' f_{n+1} + \left(\prod_{i\in[n]} f_i\right) f_{n+1}' \\
&= \left(\sum_{i\in[n]} f_i' \cdot \prod_{\substack{j\in[n],\\ j\neq i}} f_j\right) f_{n+1} + f_{n+1}' \cdot \prod_{i\in[n]} f_i \\
&= \sum_{i\in[n]} f_i' \cdot \prod_{\substack{j\in[n+1],\\ j\neq i}} f_j + f_{n+1}' \cdot \prod_{\substack{j\in[n+1],\\ j\neq n+1}} f_j \\
&= \sum_{i\in[n+1]} f_i' \cdot \prod_{\substack{j\in[n+1],\\ j\neq i}} f_j
\end{aligned}$$

This finishes the proof by induction. □

List of Figures

List of Tables

Bibliography

[BBB+18] 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*, pages 315–334, 2018. https://doi.org/10.1109/SP.2018.00020.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model. Cryptology ePrint Archive, Paper 2017/1050, 2017. https://eprint.iacr.org/2017/1050.

[BR93] Mihir Bellare and Phillip Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73. Association for Computing Machinery, 1993. https://doi.org/10.1145/168588.168596.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018. https://eprint.iacr.org/2018/046.

[BT04] Jean-Paul Berrut and Lloyd N. Trefethen. Barycentric Lagrange Interpolation. *SIAM Review*, 46(3):501–517, 2004. https://doi.org/10.1137/S0036144502417715.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates. In *Advances in Cryptology – EUROCRYPT 2023*, volume 14005 of *LNCS*, pages 499–530. Springer, 2023. https://doi.org/10.1007/978-3-031-30617-4_17.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS. In *Advances in Cryptology – EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 738–768. Springer, 2020. https://doi.org/10.1007/978-3-030-45721-1_26.

[Ele22] Electric Coin Company. The halo2 Book, 2022. https://zcash.github.io/halo2/index.html.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The Algebraic Group Model and its Applications. In *Advances in Cryptology – CRYPTO 2018*, volume 10992 of *LNCS*, pages 33–62. Springer, 2018. https://doi.org/10.1007/978-3-319-96881-0_2.

[FS87] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology – CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer, 1987. https://doi.org/10.1007/3-540-47721-7_12.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic Span Programs and Succinct NIZKs without PCPs. In *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, 2013. https://doi.org/10.1007/978-3-642-38348-9_37.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof-Systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304. Association for Computing Machinery, 1985. https://doi.org/10.1145/22145.22178.

[GO94] Oded Goldreich and Yair Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology*, 7(1):1–32, 1994. https://doi.org/10.1007/BF00195207.

[Gro16] Jens Groth. On the Size of Pairing-Based Non-interactive Arguments. In *Advances in Cryptology – EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 305–326. Springer, 2016. https://doi.org/10.1007/978-3-662-49896-5_11.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge. Cryptology ePrint Archive, Paper 2019/953, 2019. https://eprint.iacr.org/2019/953.

[HBHW22] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash Protocol Specification. Technical report, Electric Coin Company, 2022. https://zips.z.cash/protocol/protocol.pdf.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-Size Commitments to Polynomials and Their Applications. In *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, 2010. https://doi.org/10.1007/978-3-642-17373-8_11.

[Mau05] Ueli Maurer. Abstract Models of Computation in Cryptography. In *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, 2005. https://doi.org/10.1007/11586821_1.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, pages 2111–2128. Association for Computing Machinery, 2019. https://doi.org/10.1145/3319535.3339817.

[Mil22] Jim Miller. The Frozen Heart vulnerability in PlonK, 2022. https://blog.trailofbits.com/2022/04/18/the-frozen-heart-vulnerability-in-plonk/.

[Sho97] Victor Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Advances in Cryptology – EUROCRYPT '97*, volume 1233 of *LNCS*, pages 256–266. Springer, 1997. https://doi.org/10.1007/3-540-69053-0_18.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. Factoring polynomials over finite fields. In *Modern Computer Algebra*, chapter 14, pages 377–432. Cambridge University Press, 3rd edition, 2013. https://doi.org/10.1017/CBO9781139856065.018.