Plonk Without Random Oracles

Georg Fuchsbauer and Marek Sefranek

TU Wien, Austria
firstname.lastname@tuwien.ac.at

Abstract. Plonk is one of the most influential and widely used zk-SNARKs, with proofs of constant size (0.5 kB), sublinear verification time, and circuit-independent setup. All prior security analyses of Plonk—of both knowledge soundness and zero knowledge (ZK)—are in the random-oracle model (ROM), which recent work has shown to be especially problematic in the context of proof systems. Moreover, a security proof in the ROM does not justify using a system *recursively*, a powerful technique currently used e.g. in "zk-rollups."

We show that for analyzing Plonk, the ROM can be avoided when making the "zero-testing" (ZT) assumption on the used hash function (and the commitment scheme). ZT is a non-interactive, falsifiable hardness assumption. Assuming ZT, we prove Plonk knowledge-sound in the algebraic group model (AGM) and computationally ZK in the standard model. Using Plonk recursively can now be justified in the recursive AGM.

To increase confidence in ZT, we prove that it holds in the ROM assuming the used commitment scheme is binding. Moreover, when using KZG commitments in the generic group model we show that collision resistance of the hash function implies ZT.

Keywords: Plonk · zk-SNARKs · Fiat–Shamir transformation · zero-testing assumption

1 Introduction

First introduced by Goldwasser, Micali, and Rackoff [GMR85], *zero-knowledge proofs* have transitioned from a purely theoretical concept to seeing wide adoption in practice. A zero-knowledge proof system 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. The verifier should accept honestly generated proofs (*completeness*), whereas a malicious prover should not be able to convince the verifier of a false statement (*soundness*).

The most widely used type of zero-knowledge proof systems are *zk-SNARKs* (***s**uccinct **n**on-interactive **ar**guments of **k**nowledge*) [Gro10,BCCT12,GGPR13]. An *argument* is a proof system with computational soundness, i.e., both prover and verifier are probabilistic polynomial-time algorithms. Furthermore, the prover must know a witness when producing a valid proof (hence argument *of knowledge*). *Non-interactive* means that only a single message, "the proof," is sent during the protocol. The term *succinct* refers to the proof size, in particular, SNARKs have proofs that are much shorter than the proved statement, in some cases even of constant size. Moreover, SNARKs support proofs of arbitrary NP statements, typically expressed as circuit satisfiability.

Plonk. This work focuses on the influential zk-SNARK Plonk by Gabizon, Williamson, and Ciobotaru [GWC19]. It achieves constant-size proofs, verification time sublinear in the size of the witness, and a circuit-independent setup which produces a universal and updatable *structured reference string* (SRS). Updatability means that the SRS can be publicly rerandomized, so that the honesty of one of the updaters suffices for soundness.

The original Plonk construction employs the KZG *polynomial commitment scheme* by Kate, Zaverucha, and Goldberg [KZG10] due to its constant-size commitments and opening proofs, each a single group element. This comes at the cost of a trusted setup, which produces a structured reference string of linear size $\mathcal{O}(n)$ group elements, where n is

the maximum supported degree of the committed polynomials. For Plonk, n corresponds to the maximum number of gates in the proved circuit.

Plonk and its derivatives are deployed in a variety of real-world projects, including Zcash,[1] Polygon Zero,[2] Aztec Network,[3] Dusk,[4] MatterLabs,[5] Astar,[6] Mina,[7] Anoma,[8] and Espresso Systems.[9]

Motivation. Consider a scenario where you want to prove (the conjunction of) n statements $x_1, \ldots, x_n$. Suppose that you want to start proving x_1 before x_2 is known, or that the size of the circuit expressing the conjunction is not supported by the proof system. Both scenarios can be addressed by using *recursion*: first, prove statement x_1, yielding a proof π_1; then prove the statement "x_2 is true and π_1 passes the SNARK verification," yielding again a single proof π_2. Repeat this approach until the final output is just a single proof π_n that proves all n statements simultaneously.

This concept is also known as *recursive proofs* and requires a SNARK for statements that involve one or several copies of its own SNARK verifier. A side-effect is that the circuit size (and thus prover memory and parameters) of the statement proved by the SNARK can be kept small. Recursive proofs enable proof composition, incrementally verifiable computation [Val08], and proof-carrying data [BCMS20]. They have many real-world applications like scalable rollups [TSH22] or even succinct blockchains like *Mina* [BMRS20]. In the latter, the blockchain is simply a proof of knowledge of a block and of a proof attesting to the validity of the contained transactions.

To prove the security of a recursively applied proof system, we need to recursively apply the reasoning in the security proof of the original system. Many of the widely used SNARK systems are constructed as interactive protocols that are then made non-interactive using the Fiat–Shamir transformation [FS87]. Security of such schemes is then proved in the random-oracle model (ROM) [BR93], where the used hash function is replaced by a truly random function to which the adversary gets oracle access.[10]

Problems with the random-oracle model have been known for a long time [CGH98]. While these results relied on contrived examples, recent results have put serious doubts on proofs in the ROM especially in the context of proof systems. Khovratovich, Rothblum, and Soukhanov [KRS25] show attacks on the soundness of variants of the GKR protocol [GKR15], which has a security proof in the ROM. A security proof that does not rely on this heuristic would thus largely increase our confidence in the system.

Another issue with random-oracle proofs of zk-SNARK soundness is that the guarantees become vacuous if the proof system is used in a recursive way. Since the verification circuit needs to evaluate the used hash function, the security of the instance of the proof system on which the outer layer recurses cannot be analyzed in the random-oracle model anymore. We thus need SNARK systems whose security analysis does not rely on the ROM.

Security of Plonk. Originally, Plonk was shown to be knowledge-sound in the combined model AGM+ROM [GWC19]. The algebraic group model (AGM) [FKL18] assumes that the adversary computes all group elements it outputs *algebraically*, that is, as a combination of the received group elements. Lipmaa, Parisella, and Siim [LPS25] proved Plonk knowledge-sound in the "pure" ROM under a loose reduction to a new falsifiable assumption. Building on work by Faonio, Fiore, and Russo [FFR24], Lipmaa [Lip25] recently extended this result to *simulation extractability*, a stronger notion of knowledge soundness where the adversary

[1] https://z.cash/ [2] https://polygon.technology/ [3] https://aztec.network/
[4] https://dusk.network/ [5] https://matter-labs.io/ [6] https://astar.network/
[7] https://minaprotocol.com/ [8] https://anoma.net/ [9] https://www.espressosys.com/
[10] A notable exception is Groth16 [Gro16], which is not a Fiat–Shamir transform and is proven secure in the algebraic group model. A limitation is that its setup is specific to the circuit expressing its language.

has access to a proof-simulation oracle. While the motivation in the works by Lipmaa et al. was to remove reliance on the AGM, our focus takes the opposite direction: we want to prove knowledge soundness of Plonk without idealizing the hash function as a random oracle, instead relying on a falsifiable property of it. This leads us to the following question:

Can Plonk be proved secure without random oracles?

1.1 Our Contributions

Plonk without random oracles. We answer this question in the affirmative.

We prove that Plonk is knowledge-sound in the algebraic group model (AGM) under the *zero-testing* assumption [LS24] (discussed below) and under the *power-discrete-logarithm* (PDL) assumption (Theorem 5). Zero-testing is a falsifiable assumption on the underlying hash function (and commitment scheme). Our proof of Plonk is tight since it avoids rewinding. To the best of our knowledge, this makes Plonk the first practical Fiat–Shamir-based zk-SNARK that is tightly knowledge-sound under a falsifiable assumption in the AGM. Our reliance on some from of idealization, the AGM, is expected because of an impossibility result [GW11] that makes the existence of succinct non-interactive proof systems in the standard model implausible.

We moreover show that *both* assumptions are in some sense necessary. While breaking PDL amounts to obtaining the simulation trapdoor, we show how an adversary, which breaks ZT, can be used to construct an adversary that outputs a proof that verifies, but from which the extractor fails to return a valid witness (Section 4.3).

Together with the analyses by Lipmaa et al. [LPS25,Lip25], our results establish that Plonk can be proved knowledge-sound by separately idealizing different aspects: the hash function (in the ROM) or the underlying bilinear group (in the AGM). In contrast, the original AGM+ROM proof required simultaneously idealizing both. We believe that this is a significant step towards justifying the security of a cryptographic scheme for which standard-model proofs are implausible.

Replacing the ROM by the AGM is moreover a significant improvement for recursive proof systems, since their soundness can be argued in the *recursive AGM* [LS24], as previously done by Eagen et al. [EGS+24]. Consider a cryptographic scheme defined over a group $(\mathbb{G}, +)$ of prime order p and let $\mathbb{F}$ be the field of order p. The AGM assumes that from an adversary that is given elements $h_1, h_2, \ldots \in \mathbb{G}$ and returns $h \in \mathbb{G}$, one can extract a *representation* $a_1, a_2, \ldots \in \mathbb{F}$ s.t. $h = \sum a_i \cdot h_i$. For elliptic-curve groups, the *recursive* AGM further assumes that if some (a_i, a_j) is a point on the curve in $\mathbb{G}$, then a representation can again be extracted for this group element; and if elements from these representation lead to further group elements, a representation can again be extracted for them, etc.

Regarding zero knowledge (ZK), Sefranek [Sef24] showed that Plonk is statistically ZK (i.e., even against unbounded adversaries) in the ROM. Again making the zero-testing assumption, we prove Plonk computationally ZK in the *standard* model (Theorem 6).

Analysis of the zero-testing assumption. We view our results as an important basis for future zk-SNARK cryptanalysis. Whereas previous proofs in the ROM are independent of the used hash function, our results motivate the zero-testing assumption as a clearly specified target for cryptanalysis to raise confidence in the use of a concrete hash function in proof systems.

To bolster our confidence in the recent zero-testing (ZT) assumption, we provide various justifications. When introducing it, Lee and Seo [LS24] argued that it holds when modeling the hash function as a random oracle and assuming the commitment scheme is binding.

We prove this formally by extending the forking lemma [PS96,BN06] and showing how to actually use a ZT adversary to break binding (Theorem 3).

To completely remove the ROM from our overall analysis of Plonk, we prove that when using KZG commitments (as done by Plonk) and modeling them in the generic group model [Sho97], then collision resistance, a basic property of the hash function, suffices for ZT (Theorem 4). Finally, we show that ZT can be framed as an instance of hash-function correlation intractability [CGH98] for an efficiently decidable relation (Section 3.4).

1.2 Technical Overview

To analyze the security of Plonk when instantiated with a hash function $H\colon \{0,1\}^* \to \mathbb{F}$, we require a sufficient property of H for achieving knowledge soundness. Unfortunately, basic hash function properties such as collision resistance seem too weak, so we resort to the following assumption on H.

Zero-testing. To motivate the assumption, let us first look at the following simplification: let $\mathbb{F}$ be a large field that is the range of H; consider the problem of finding a low-degree non-zero polynomial f over $\mathbb{F}$ such that $f(H(f)) = 0$, i.e., finding a polynomial whose hash value is one of its roots. Hardness of this problem can be regarded as a form of *correlation intractability* [CGH98] for the relation $\mathcal{R} = \{(f, x) \mid f(x) = 0\}$, which for low-degree polynomials is sparse. Correlation-intractable hash functions have, for example, been used to construct multi-theorem adaptive NIZKs [CX23].

We will assume hardness of the following variant of this problem: instead of hashing the polynomial f directly, we first commit to it, using a suitable commitment scheme Com, and then hash the resulting commitment. More specifically, the task is to find a message $m \in \mathcal{M}$, which is interpreted as a (bounded-degree) polynomial by a fixed function $D\colon \mathcal{M} \to \mathbb{F}[X]$, so that hashing the commitment of m results in a root of the polynomial $D(m)$.

Hardness of the problem

$$\text{“Find} \quad m \quad \text{s.t.} \quad f\big(H(\mathsf{Com}(m))\big) = 0 \quad \text{for non-zero} \quad f \coloneqq D(m)\text{”} \tag{1}$$

is known as the *zero-testing (ZT) assumption*, which was first introduced by Lee and Seo [LS24] to analyze the security of the recursive proof system Nova [KST22], and subsequently used to build a privacy-preserving smart contract system [EGS+24].

A toy example. To demonstrate the power of the zero-testing assumption, consider the following scenario: a prover $\mathcal{P}$ holds two polynomials $f_1, f_2 \in \mathbb{F}[X]$ and wants to convince the verifier $\mathcal{V}$ that they are equivalent, i.e., $f_1 \equiv f_2$. Instead of sending the polynomials, the standard approach for succinctly proving a polynomial identity is to evaluate f_1 and f_2 at a random “challenge” point $\alpha \in \mathbb{F}$ chosen by the verifier, and then check that $f_1(\alpha) = f_2(\alpha)$. By the Schwartz–Zippel lemma, if $f_1 \not\equiv f_2$, this check passes with probability at most $\max\{\deg f_1, \deg f_2\}/|\mathbb{F}|$. Using a *polynomial commitment scheme* (PCS) to remove the honesty assumption that the prover actually sends the correct evaluations, we arrive at the following interactive protocol:

$\mathcal{P}(f_1, f_2)$:		$\mathcal{V}$:
Compute commitments $\mathsf{cm}_1, \mathsf{cm}_2$ to f_1, f_2	$\xrightarrow{\mathsf{cm}_1, \mathsf{cm}_2}$	
$y_1 \coloneqq f_1(\alpha),\ y_2 \coloneqq f_2(\alpha)$	$\xleftarrow{\alpha}$	Sample $\alpha \in \mathbb{F}$
Compute “opening proofs” π_1, π_2	$\xrightarrow{y_1, y_2, \pi_1, \pi_2}$	Verify π_1, π_2
		Return $y_1 \stackrel{?}{=} y_2$

To turn this into a *non-interactive* argument, one applies the Fiat–Shamir transformation [FS87], which lets both parties derive the verifier challenge locally by hashing the partial protocol transcript up to that point using a function $H\colon \{0,1\}^* \to \mathbb{F}$. For the protocol above, $\mathcal{P}$ and $\mathcal{V}$ thus derive the challenge as $\alpha \coloneqq H(\mathsf{cm}_1, \mathsf{cm}_2)$. To argue soundness of this approach, one then usually models H as a random oracle [BR93].

Instead, we will (informally) argue soundness in the standard model relying on the zero-testing assumption and assuming that the polynomial commitment scheme is evaluation-binding.[11] We parametrize ZT as follows: set $\mathcal{M} \coloneqq \mathbb{F}[X] \times \mathbb{F}[X]$ and let Com commit component-wise; define $D\colon \mathcal{M} \to \mathbb{F}[X]$ as $D(f_1, f_2) \coloneqq f_1 - f_2$. Now suppose a malicious prover committed to two polynomials $f_1 \not\equiv f_2$. By evaluation binding of the PCS, the verifier accepts iff

$$f_1(\alpha) = f_2(\alpha) \quad \text{for} \quad \alpha \coloneqq H(\mathsf{cm}_1, \mathsf{cm}_2). \tag{2}$$

In this case, the message $m \coloneqq (f_1, f_2)$ directly breaks ZT: by (2), $\alpha \coloneqq H(\mathsf{cm}_1, \mathsf{cm}_2) = H(\mathsf{Com}(m))$ is a root of the non-zero polynomial $f \coloneqq D(m) = f_1 - f_2$, which shows (1).

To make this intuition work, the reduction to ZT has to obtain the polynomials f_1 and f_2 from the prover to construct m. This is possible if the polynomial commitment scheme is *extractable*, that is, from an adversary computing a commitment, the reduction can extract a committed value.

Note that in the above example the ZT assumption is also *necessary* for soundness: a solution $m \coloneqq (f_1, f_2)$ with $f_1 \not\equiv f_2$ directly leads to an attack on the Fiat–Shamir transform of the protocol.

Multi-challenge protocols. Unfortunately, this approach cannot be directly extended to multi-challenge protocols such as Plonk, which has 5 challenge rounds. The problem is that each challenge is computed by evaluating H on only a prefix of the entire protocol transcript. Thus, the first challenge contains less information than the next one, which can be problematic when trying to define an appropriate function D for ZT.

For this, we first identify suitable polynomials by analyzing Plonk in a round-by-round manner, leading to a modular proof structure. The function D we define for ZT returns different polynomials depending on the protocol round to which the message m corresponds. For example, for the final challenge we can simply express Plonk's verification equation as a polynomial identity: if a cheating prover menages to find a proof that verifies, i.e., the identity holds at the final challenge, but the verification polynomial is non-zero, then the final verifier challenge can be used to break ZT. However, since this is not the only way a prover can cheat, we require more carefully chosen polynomials for the earlier rounds.

Another problem is that Plonk involves polynomial identity checks that cannot be expressed via a single univariate polynomial. For example, when batching multiple univariate polynomial identities, instead of checking $f_1(X) \equiv 0$ and $f_2(X) \equiv 0$ separately, the verifier sends a challenge α, and then the single identity $f_1(X) + \alpha f_2(X) \equiv 0$ is checked. If this is satisfied, but $f_1(X) \not\equiv 0$ and $f_2(X) \not\equiv 0$, we need to use the hash value α to break ZT. In particular, for $F(X, Y) \coloneqq f_1(Y) + X f_2(Y)$, α is such that $F(\alpha, Y) \equiv 0$ while the bivariate polynomial $F(X, Y) \not\equiv 0$.

To cover such cases, we introduce a generalization of the zero-testing assumption to *multivariate* polynomials, where the adversary has to specify a non-zero polynomial $F(X, Y)$ so that α, the hash value of its commitment, yields $F(\alpha, Y) \equiv 0$. Importantly, we justify this variant of ZT by showing that it is implied by the original ZT assumption for a closely related family of functions D (Theorem 2).

[11] That is, it is infeasible to commit to a polynomial and then prove an inconsistent evaluation.

Multivariate ZT also lets us handle the first round of Plonk, the *permutation argument*. Here the verifier sends two challenges β, γ that are used to check a bivariate polynomial identity

$$P(X, Y) \equiv 0. \tag{3}$$

In the Fiat–Shamir transform (ignoring public inputs), the challenges are computed as $\beta := H(\mathsf{msg}_1)$ and $\gamma := H(\mathsf{msg}_1, \beta)$, where msg_1 is the first-round prover message. Again, we can resort to the multivariate ZT assumption: if the identity in Eq. (3) does not hold, but $P(\beta, \gamma) = 0$, there are two cases, yielding different ZT solutions: either the first challenge β already makes the bivariate polynomial P vanish, i.e., $P(\beta, Y) \equiv 0$, or γ is a root of the univariate polynomial $P(\beta, X)$.

As for how our reduction obtains the polynomials it needs to break ZT from a knowledge-soundness adversary, we proceed like in the original security analysis of Plonk [GWC19]. In the algebraic group model, KZG polynomial commitments are straight-line extractable. The adversary's proof contains commitments to polynomials that interpolate the witness; so extracting these, the extractor obtains a candidate witness in the knowledge-soundness experiment. Extracting further polynomials whose commitments are contained in the proof allows us to construct the message m for breaking ZT.

Our proof of Plonk's knowledge soundness proceeds in 2 game hops. First, we deal with all the parts that correspond to solving the *power-discrete-logarithm problem* (which amounts to recovering the trapdoor used in the setup and is the underlying assumption in the original security analysis in the AGM+ROM [GWC19]). The most involved part of our proof is then to leverage the special structure of the polynomials involved in the Plonk protocol to make ZT applicable here.

We believe that our techniques can lay the foundation for a generalization of our results that applies to a wide class of protocols and zk-SNARKs based on the Fiat–Shamir transformation.

2 Preliminaries

Notation. We use $\lambda \in \mathbb{N}$ to denote the security parameter, and $[n]$ as a shorthand for the set $\{1, \ldots, n\} \subset \mathbb{N}$. Moreover, $\mathsf{negl}(\cdot)$ denotes a negligible function $\mathsf{negl}(\lambda) = \lambda^{-\omega(1)}$. Let $\mathbb{F}[X_1, \ldots, X_n]$ be the set of polynomials in the indeterminates $X_1, \ldots, X_n$ over $\mathbb{F}$, and $\mathbb{F}^{(\leq d)}[X_1, \ldots, X_n]$ its restriction to polynomials of (total) degree $\leq d$. Moreover, $\mathbb{F}^{(\leq d)}[X][Y]$ denotes the set of univariate polynomials in Y with coefficients in $\mathbb{F}^{(\leq d)}[X]$.

A statement of the form $y := \mathcal{A}(x)$ denotes a deterministic assignment, while $y \leftarrow \mathcal{A}(x)$ denotes a probabilistic assignment, i.e., running the algorithm $\mathcal{A}$ with uniform randomness on input x. We also use $y \leftarrow S$ for sampling y uniformly at random from the set S. We write $(y; z) \leftarrow (\mathcal{A} \parallel \mathcal{E}_{\mathcal{A}})(x)$ when $\mathcal{A}$ on input x outputs y, and $\mathcal{E}_{\mathcal{A}}$ on the same input (including $\mathcal{A}$'s randomness) outputs z.

An *indexed relation* $\mathcal{R}$ is a set of index-statement-witness triples $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ with the corresponding *indexed language* $\mathcal{L}_{\mathcal{R}} := \{(\mathbb{i}, \mathbb{x}) \mid \exists \mathbb{w} \colon (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$. For a size bound $n \in \mathbb{N}$, we denote by $\mathcal{R}_n$ the restriction of $\mathcal{R}$ to triples $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ with $|\mathbb{i}| \leq n$.

We assume all PPT (probabilistic polynomial-time) algorithms run in time polynomial in the security parameter λ. Furthermore, let GGen be a PPT algorithm that, on input 1^λ, generates the description of a bilinear group $\mathcal{G} := (p, \mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, g_T)$ such that:

- $\mathbb{F}$ is a prime field of super-polynomial size $p = \lambda^{\omega(1)}$, which admits primitive n-th roots of unity, i.e., $n \mid (p - 1)$.
- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of order p endowed with a pairing $e \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.
- $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and $g_T := e(g_1, g_2) \in \mathbb{G}_T$ are group generators.

We leave the dependence on λ implicit, i.e., simply write $\mathbb{F}$ instead of $\mathbb{F}(\lambda)$. Furthermore, we write the group operations in $\mathbb{G}_1, \mathbb{G}_2$ additively and use the shorthands $[x]_1 := x \cdot g_1$ and $[x]_2 := x \cdot g_2$ for $x \in \mathbb{F}$.

We use the code-based game-playing framework of Bellare and Rogaway [BR06], and denote the probability that game $\mathbf{G}_i$ returns 1 by $\Pr[\mathbf{G}_i]$.

Power-DL Assumption. The security of Plonk necessarily relies on the following variant of the discrete-logarithm (DL) assumption. (Solving it corresponds to recovering the secret "trapdoor," which lets one simulate proofs, from a structured reference string for Plonk.)

Definition 1 (n-DL Assumption). *For all $n = \mathsf{poly}(\lambda)$ and all PPT adversaries $\mathcal{A}$:*

$$\mathsf{Adv}^{n\text{-}\mathsf{DL}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \Pr\left[\begin{matrix}\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda);\ \tau \leftarrow \mathbb{F};\\ \tau' \leftarrow \mathcal{A}\left(\mathcal{G}, [\tau]_1, \ldots, [\tau^n]_1, [\tau]_2\right)\end{matrix} \ :\ \tau' = \tau\right] \leq \mathsf{negl}(\lambda).$$

Algebraic Group Model. The AGM [FKL18,Zha22] assumes that adversaries compute group elements *algebraically*. In our context, this means that an *algebraic* adversary $\mathcal{A}$, given input $\left(\mathcal{G}, [\tau]_1, \ldots, [\tau^n]_1, [\tau]_2\right)$, must provide a *representation* $(a_i)_{i=0}^n \in \mathbb{F}^{n+1}$ for every group element $c \in \mathbb{G}_1$ it outputs, such that $c = \sum_{i=0}^n a_i \cdot [\tau^i]_1$. We treat this representation as an implicit output (available e.g. to the security reduction) and simply write $c \leftarrow \mathcal{A}(\mathsf{input})$.

2.1 Deterministic Commitment Schemes

We define *deterministic* commitment schemes with a message space $\mathcal{M}$ that can implicitly depend on the security parameter $\lambda \in \mathbb{N}$ and a size bound $n \in \mathbb{N}$.

Definition 2 (Commitment Scheme). *A (deterministic) commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{Com})$ with message space $\mathcal{M}$ consists of two PPT algorithms, where Setup is randomized and Com deterministic, with the following syntax:*

- $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda, n)$*, given the security parameter 1^λ and a size bound $n \in \mathbb{N}$ on the messages, returns public parameters par which implicitly define a message space $\mathcal{M}_n$.*
- $\mathsf{cm} := \mathsf{Com}_{\mathsf{par}}(m)$*, given the public parameters par (with implicit bound n) and a message $m \in \mathcal{M}_n$, returns a commitment cm to m.*

We require that Γ satisfies the following form of computational binding (which corresponds to collision resistance of keyed hash functions).

Definition 3 (Binding). *For all $n = \mathsf{poly}(\lambda)$ and every PPT adversary $\mathcal{A}$:*

$$\mathsf{Adv}^{\mathsf{bind},n}_{\mathcal{A},\Gamma}(\lambda) := \Pr\left[\begin{matrix}\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda, n);\ (m, m') \leftarrow \mathcal{A}(\mathsf{par}):\\ m \neq m' \ \wedge\ \mathsf{Com}_{\mathsf{par}}(m) = \mathsf{Com}_{\mathsf{par}}(m')\end{matrix}\right] \leq \mathsf{negl}(\lambda).$$

Plonk employs the widely used KZG polynomial commitment scheme [KZG10], given in Constr. 1 (we omit the algorithms for proving and verifying evaluations of committed polynomials). KZG is binding under the n-DL assumption [KZG10] (using the fact that the difference of the two colliding polynomials has the trapdoor τ as a root).

Construction 1: The KZG polynomial commitment scheme.

$\underline{\mathsf{Setup}(1^\lambda, n)}$: Run $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$. Sample $\tau \leftarrow \mathbb{F}$ and return

$$\mathsf{srs} := \left(\mathcal{G}, [\tau]_1, \left[\tau^2\right]_1, \ldots, [\tau^n]_1, [\tau]_2\right).$$

This defines the message space $\mathcal{M}_n := \mathbb{F}^{(\leq n)}[X]$.

$\underline{\mathsf{Com}_{\mathsf{srs}}(m)}$: Parse m as a degree-n polynomial $f(X) = \sum_{i=0}^{n} f_i X^i \in \mathbb{F}^{(\leq n)}[X]$, and return

$$\mathsf{cm} := \sum_{i=0}^{n} f_i \left[\tau^i\right]_1 = [f(\tau)]_1 .$$

2.2 zk-SNARKs

We define universal, preprocessing zk-SNARKs, an example of which is Plonk.

Definition 4 (zk-SNARK). *A zk-SNARK* $\Pi = (\mathsf{Setup}, \mathsf{Index}, \mathsf{Prove}, \mathsf{Verify})$ *for an indexed relation* $\mathcal{R}$ *consists of four PPT algorithms with the following syntax:*

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

This definition captures *universal* zk-SNARKs, because the structured reference string srs generated by Setup supports proofs of any $(\mathtt{i}, \mathtt{x}, \mathtt{w}) \in \mathcal{R}_n$ up to a selected size bound n, as opposed to an index-specific reference string.

Moreover, this defines *preprocessing* zk-SNARKs, since Prove and Verify are not directly run on the instance $(\mathtt{i}, \mathtt{x}) \in \mathcal{L}_\mathcal{R}$. Instead, there is a non-interactive preprocessing phase where anyone can use the deterministic algorithm Index to publicly compute prover and verifier parameters pp, vp for a given index $\mathtt{i}$. This allows to compress the srs and $\mathtt{i}$ significantly, enabling the prover and, especially, the verifier to check different statements $\mathtt{x}$ in subsequent online phases more efficiently.

Furthermore, zk-SNARKs have to satisfy the following properties.

Definition 5 (Completeness). *For all* $n = \mathsf{poly}(\lambda)$ *and all* $(\mathtt{i}, \mathtt{x}, \mathtt{w}) \in \mathcal{R}_n$*:*

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

We define *statistical* completeness, which is the notion achieved by Plonk (see the abort in step 3 of Prove in Constr. 7). For *perfect* completeness, the probability must be 1.

Definition 6 (Knowledge Soundness). *For all* $n = \mathsf{poly}(\lambda)$ *and every PPT adversary* $\mathcal{A}$*, there exists a PPT extractor* $\mathcal{E}_\mathcal{A}$ *such that:*

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

This notion is adaptive, since the malicious prover $\mathcal{A}$ can choose the statement $(\mathbb{i}, \mathbb{x})$ based on the structured reference string srs. Note that in the AGM, the extractor gets the representations for all group elements output by $\mathcal{A}$.

Definition 7 (Succinctness). *The proof size is sublinear in the size of the proved statement* $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_n$*:* $|\pi| = o(|\mathbb{i}| + |\mathbb{x}| + |\mathbb{w}|)$.

Definition 8 (Zero Knowledge). *There exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ *such that for all* $n = \mathsf{poly}(\lambda)$ *and every PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\mathsf{Adv}^{\mathsf{czk},n}_{\mathcal{A},\mathcal{S},\Pi}(\lambda) \coloneqq \left| \Pr\left[\begin{array}{c} \mathsf{srs} \leftarrow \mathsf{Setup}(1^\lambda, n); \\ (\mathbb{i}, \mathbb{x}, \mathbb{w}, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{srs}); \\ (\mathsf{pp}, \mathsf{vp}) \coloneqq \mathsf{Index}(\mathsf{srs}, \mathbb{i}); \\ \pi \leftarrow \mathsf{Prove}(\mathsf{pp}, \mathbb{x}, \mathbb{w}); \\ b \leftarrow \mathcal{A}_1(\mathsf{st}, \pi) : \\ b = 1 \wedge (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_n \end{array}\right] - \Pr\left[\begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathcal{S}_0(1^\lambda, n); \\ (\mathbb{i}, \mathbb{x}, \mathbb{w}, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{srs}); \\ (\mathsf{pp}, \mathsf{vp}) \coloneqq \mathsf{Index}(\mathsf{srs}, \mathbb{i}); \\ \tilde{\pi} \leftarrow \mathcal{S}_1(\tau, \mathsf{pp}, \mathbb{x}); \\ b \leftarrow \mathcal{A}_1(\mathsf{st}, \tilde{\pi}) : \\ b = 1 \wedge (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_n \end{array}\right] \right| \leq \mathsf{negl}(\lambda).$$

This captures *computational* zero knowledge. The stronger notions of *statistical* and *perfect* zero knowledge consider unbounded adversaries, and the latter additionally requires the two probabilities to be equal.

2.3 Hash Functions

Bellare and Kohno [BK04] define the *balance* of a hash function as a measure of its regularity—that is, the extent to which all preimage sets are of equal size. Taking a value between 0 and 1, a balance of 1 means the hash function is regular, whereas 0 means it is constant. We generalize this concept to a computational notion we call *balancedness*.

Intuitively, we say a hash function $H\colon \{0,1\}^* \to R$ is *balanced* if for any efficient sampling algorithm $\mathcal{A}$ with sufficient min-entropy, evaluating H on $\mathcal{A}$'s outputs still has good min-entropy; in particular, there is no value $y^* \in R$ such that evaluating H on the output of $\mathcal{A}$ results in y^* with noticeable probability.

Definition 9 (Balancedness). *A family* $\mathcal{H} = \{H_\lambda\}_\lambda$ *of hash functions* $H_\lambda\colon \{0,1\}^* \to R_\lambda$ *is* balanced *if for every PPT adversary* $\mathcal{A}$ *with*

$$\mathsf{MaxPr}_{\mathcal{A}}(\lambda) \coloneqq \max_{x \in \{0,1\}^*} \Pr\left[m \leftarrow \mathcal{A}(1^\lambda) : m = x\right] \leq \frac{1}{|R_\lambda|}, \tag{4}$$

it holds that

$$\mathsf{Adv}^{\mathsf{bal}}_{\mathcal{A},\mathcal{H}}(\lambda) \coloneqq \max_{y \in R_\lambda} \Pr\left[m \leftarrow \mathcal{A}(1^\lambda) : H_\lambda(m) = y\right] \leq \mathsf{negl}(\lambda). \tag{5}$$

Remark 1. Note that (5) implies $|R_\lambda| = \lambda^{\omega(1)}$, i.e., the size of the hash function range must be super-polynomial in λ. Further note that our notion of balancedness does not hold against unbounded adversaries: consider a value y whose preimage is larger than $|R_\lambda|$; then an adversary that samples uniformly from this set breaks balancedness.

Remark 2. The bound $\mathsf{MaxPr}_{\mathcal{A}}(\lambda) \leq 1/|R_\lambda|$ in condition (4) might seem arbitrary. In fact, since $\mathsf{Adv}^{\mathsf{bal}}_{\mathcal{A},\mathcal{H}}(\lambda) \geq \mathsf{MaxPr}_{\mathcal{A}}(\lambda)$ for any $\mathcal{A}$, only $\mathsf{MaxPr}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$ is necessary (but not sufficient) for (5). However, we chose the concrete bound $1/|R_\lambda|$ to make the definition useful beyond asymptotic security.

Remark 3. The above definition readily extends to families of *keyed* hash functions. Since all results in this work remain unchanged under such an extension, we restrict attention to the unkeyed setting for simplicity.

A balance [BK04] of $1 - \mathsf{negl}(\lambda)$ is a necessary but not sufficient condition for a hash function (family) to satisfy our notion of balancedness. To argue that our definition is a very mild assumption on the hash function, we show that it is implied by collision resistance (CR)—the de facto standard property of cryptographic hash functions.

Definition 10 (Collision Resistance). *A family $\mathcal{H} = \{H_\lambda\}_\lambda$ of hash functions is* collision-resistant *if for every (uniform) PPT adversary $\mathcal{A}$:*

$$\mathsf{Adv}^{\mathsf{cr}}_{\mathcal{A},\mathcal{H}}(\lambda) \coloneqq \Pr\left[(m, m') \leftarrow \mathcal{A}(1^\lambda) \;:\; H_\lambda(m) = H_\lambda(m') \;\wedge\; m \neq m'\right] \leq \mathsf{negl}(\lambda).$$

Theorem 1 (CR ⇒ Balancedness). *Fix a family $\mathcal{H} = \{H_\lambda\}_{\lambda\in\mathbb{N}}$ of hash functions $H_\lambda \colon \{0,1\}^* \to R_\lambda$. For every adversary $\mathcal{A}$ satisfying condition (4), there exists an adversary $\mathcal{B}$ running in time twice that of $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\mathsf{bal}}_{\mathcal{A},\mathcal{H}}(\lambda) \leq \sqrt{\mathsf{Adv}^{\mathsf{cr}}_{\mathcal{B},\mathcal{H}}(\lambda) + \mathsf{MaxPr}_{\mathcal{A}}(\lambda)}.$$

Proof. Let $\mathcal{A}$ be an adversary against balancedness of $\mathcal{H}$, and let $\mathsf{MaxPr}_{\mathcal{A}}(\lambda)$ as in Eq. (4). Define $\mathcal{B}$ against collision resistance of $\mathcal{H}$ as the adversary that runs $\mathcal{A}$ twice (with independent randomness) and returns its two output values, given below.

Construction 2: Reduction to collision resistance of $\mathcal{H}$.

$\underline{\mathcal{B}(1^\lambda)}$:

1. $m \leftarrow \mathcal{A}(1^\lambda)$.
2. $m' \leftarrow \mathcal{A}(1^\lambda)$.
3. Return (m, m').

Let $y^* \in R_\lambda$ be such that $\Pr[m \leftarrow \mathcal{A}(1^\lambda) : H(m) = y^*]$ is maximal. Then, we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{cr}}_{\mathcal{B},\mathcal{H}}(\lambda) &= \Pr\left[m \leftarrow \mathcal{A}(1^\lambda);\; m' \leftarrow \mathcal{A}(1^\lambda) \;:\; H(m) = H(m') \wedge m \neq m'\right] \\
&\geq \Pr\left[m \leftarrow \mathcal{A}(1^\lambda);\; m' \leftarrow \mathcal{A}(1^\lambda) \;:\; H(m) = H(m') = y^* \wedge m \neq m'\right] \\
&\geq \Pr\left[m \leftarrow \mathcal{A}(1^\lambda);\; m' \leftarrow \mathcal{A}(1^\lambda) \;:\; H(m) = H(m') = y^*\right] \\
&\quad - \Pr\left[m \leftarrow \mathcal{A}(1^\lambda);\; m' \leftarrow \mathcal{A}(1^\lambda) \;:\; m = m'\right] \\
&= \mathsf{Adv}^{\mathsf{bal}}_{\mathcal{A},\mathcal{H}}(\lambda)^2 - \sum_{x\in\{0,1\}^*} \Pr\left[m' \leftarrow \mathcal{A}(1^\lambda) : m' = x\right] \cdot \Pr\left[m \leftarrow \mathcal{A}(1^\lambda) : m = x\right] \\
&\geq \mathsf{Adv}^{\mathsf{bal}}_{\mathcal{A},\mathcal{H}}(\lambda)^2 - \sum_{x\in\{0,1\}^*} \mathsf{MaxPr}_{\mathcal{A}}(\lambda) \cdot \Pr\left[m \leftarrow \mathcal{A}(1^\lambda) : m = x\right] \\
&= \mathsf{Adv}^{\mathsf{bal}}_{\mathcal{A},\mathcal{H}}(\lambda)^2 - \mathsf{MaxPr}_{\mathcal{A}}(\lambda),
\end{aligned}$$

which yields the bound in the theorem. □

3 Zero-Testing Assumption

Instead of relying on the random-oracle model, we analyze Plonk under the *zero-testing (ZT) assumption*, a falsifiable property of a hash function (together with a commitment scheme)

first introduced by Lee and Seo [LS24] to analyze the security of the recursive proof system Nova [KST22]. ZT is parametrized by a family of polynomial-time (PT) functions mapping "messages" to low-degree polynomials.

Definition 11 (Zero-Testing Assumption). *Let $\mathcal{H} = \{H_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of hash functions $H_\lambda\colon \{0,1\}^* \to \mathbb{F}$ and let $\Gamma = (\mathsf{Setup}, \mathsf{Com})$ be a commitment scheme with message space $\mathcal{M}$. Fix a family $\mathcal{D} = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ of PT functions $D_\lambda\colon \mathcal{M} \to \mathbb{F}^{(\leq d)}[X]$ with degree bound $d = \mathsf{poly}(\lambda)$. Then $(\mathcal{H}, \Gamma)$* has the zero-testing property for $\mathcal{D}$ *if for all $n = \mathsf{poly}(\lambda)$ and every PPT adversary $\mathcal{A}$:*

$$\mathsf{Adv}^{(n,\mathcal{D})\text{-}\mathsf{ZT}}_{\mathcal{A},(\mathcal{H},\Gamma)}(\lambda) \coloneqq \Pr\left[\begin{array}{c} \mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda, n);\\ m \leftarrow \mathcal{A}(\mathsf{par});\\ f \coloneqq D_\lambda(m)\colon \\ f \not\equiv 0 \ \wedge\ f\big(H_\lambda(\mathsf{Com}_{\mathsf{par}}(m))\big) = 0 \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 4. To better reflect practice, we define ZT for *unkeyed* hash functions, as done in prior work [LS24,EGS+24]. In our setting, however, commitments $\mathsf{Com}_{\mathsf{par}}(m)$ correspond to protocol transcripts that include all public parameters (see Constr. 8), and are of the form $\mathsf{Com}_{\mathsf{par}}(m) = (\mathsf{par}, \mathsf{Com}'_{\mathsf{par}}(m))$. This implicitly "keys" H in Def. 11 via $H_\lambda(\mathsf{par}, \mathsf{Com}'_{\mathsf{par}}(m))$.

Remark 5. For ZT to be plausible for a family $\mathcal{D}$, the functions D_λ should be independent of the hash functions H_λ. In particular, if D_λ can invoke H_λ, this can lead to trivial counterexamples: $D_\lambda(m)$ could compute

$$\alpha \coloneqq H_\lambda(\mathsf{Com}_{\mathsf{par}}(m)) \tag{6}$$

and output the polynomial $X - \alpha$. Note that while D_λ is independent of par, the message m could encode par, so that D_λ can compute Eq. (6). Consequently, in all our applications of ZT, the used functions will not have access to H_λ.

3.1 Multivariate Zero-Testing Assumption

To simplify our proofs for Plonk, we define the following generalization of ZT, where D_λ returns a bivariate polynomial f, and the adversary wins if setting the first indeterminate of f to the hash value of (the commitment of) the adversary's output m results in the zero polynomial. We show that this generalization is implied by the standard ZT assumption for a closely related function family $\mathcal{D}$.

Definition 12 (Multivariate ZT Assumption). *Let $\mathcal{H} = \{H_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of hash functions $H_\lambda\colon \{0,1\}^* \to \mathbb{F}$ and let $\Gamma = (\mathsf{Setup}, \mathsf{Com})$ be a commitment scheme with message space $\mathcal{M}$. Fix a family $\mathcal{D} = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ of PT functions $D_\lambda\colon \mathcal{M} \to \mathbb{F}^{(\leq d)}[X][Y]$ with degree bound $d = \mathsf{poly}(\lambda)$. Then $(\mathcal{H}, \Gamma)$* has the zero-testing property for $\mathcal{D}$ *if for all $n = \mathsf{poly}(\lambda)$ and every PPT adversary $\mathcal{A}$:*

$$\mathsf{Adv}^{(n,\mathcal{D})\text{-}\mathsf{mZT}}_{\mathcal{A},(\mathcal{H},\Gamma)}(\lambda) \coloneqq \Pr\left[\begin{array}{c} \mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda, n);\\ m \leftarrow \mathcal{A}(\mathsf{par});\\ f \coloneqq D_\lambda(m)\colon \\ f \not\equiv 0 \ \wedge\ f\big(H_\lambda(\mathsf{Com}_{\mathsf{par}}(m)), Y\big) \equiv 0 \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 6. An equivalent univariate formulation is to have D output a set of univariate degree-d polynomials, and then require that at least one polynomial is non-zero while all of them vanish at the point $H_\lambda(\mathsf{Com}_{\mathsf{par}}(m))$.

Intuitively, breaking multivariate ZT (mZT) is at least as hard as breaking univariate ZT, since finding a value $\alpha \in \mathbb{F}$ that makes a non-zero bivariate polynomial $f \in F[X,Y]$ vanish, i.e., $f(\alpha, Y) \equiv 0$, means that that all coefficient polynomials in $\mathbb{F}^{(\leq d)}[X]$ have a root at α. We make this argument formal by proving that multivariate ZT is implied by univariate ZT in the following theorem.

Theorem 2 (ZT ⇒ mZT). *Fix a family $\mathcal{H} = \{H_\lambda \colon \{0,1\}^* \to \mathbb{F}\}_{\lambda \in \mathbb{N}}$ of hash functions, a commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{Com})$, and a family $\mathcal{D} = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ of PT functions $D_\lambda \colon \mathcal{M} \to \mathbb{F}^{(\leq d)}[X][Y]$. For all $n \in \mathbb{N}$ and every adversary $\mathcal{A}$, there exists a family $\mathcal{D}' = \{D'_\lambda\}_{\lambda \in \mathbb{N}}$ of PT functions $D'_\lambda \colon \mathcal{M} \to \mathbb{F}^{(\leq d)}[X]$, where $D'_\lambda(m)$ returns the leading coefficient of $D_\lambda(m)$, such that*

$$\mathsf{Adv}^{(n,\mathcal{D})\text{-}\mathsf{mZT}}_{\mathcal{A},(\mathcal{H},\Gamma)}(\lambda) \leq \mathsf{Adv}^{(n,\mathcal{D}')\text{-}\mathsf{ZT}}_{\mathcal{A},(\mathcal{H},\Gamma)}(\lambda).$$

Proof. Let $f \in \mathbb{F}[X][Y]$ be a non-zero polynomial returned by D_λ. Viewing f as a univariate polynomial in Y with coefficients in $\mathbb{F}[X]$, write it as

$$f(X,Y) = \sum_{i=0}^{\deg_Y(f)} f_i(X) \cdot Y^i,$$

where $\deg_Y(f) \geq 0$ is the degree of Y in f. Since f is non-zero, its leading coefficient $f_{\deg_Y(f)}(X)$ must be a non-zero univariate polynomial. Define D'_λ as follows:

$$D'_\lambda(m) \coloneqq \begin{cases} f_{\deg_Y(f)}(X) & \text{if } f \coloneqq D_\lambda(m) \not\equiv 0, \\ 0 & \text{otherwise.} \end{cases}$$

The claim of the theorem then follows from the fact that any solution $m^* \leftarrow \mathcal{A}(\mathsf{par})$ to mZT for (n, D_λ) is also a solution to ZT for (n, D'_λ): let $f^* \coloneqq D_\lambda(m^*)$, $f^*_{\deg_Y(f^*)} \coloneqq D'_\lambda(m^*)$, and $\alpha \coloneqq H_\lambda(\mathsf{Com}_{\mathsf{par}}(m^*))$; then $f^*(X,Y) \not\equiv 0$ implies $f^*_{\deg_Y(f^*)}(X) \not\equiv 0$, and $f(\alpha, Y) \equiv 0$ implies $f^*_{\deg_Y(f^*)}(\alpha) = 0$. □

In the remainder of this work, we omit the security parameter λ and write H and D instead of $\mathcal{H} = \{H_\lambda\}_\lambda$ and $\mathcal{D} = \{D_\lambda\}_\lambda$, respectively, simplifying the presentation.

3.2 ZT in the ROM

As a sanity check, Lee and Seo [LS24, Lemma 2] give a proof sketch that the zero-testing assumption holds when the hash function $H \colon \{0,1\}^* \to R$ is modeled as a random oracle and the commitment scheme is binding. We prove this formally by giving a reduction that, from a ZT adversary, extracts two messages that break binding.

For this, we extend the *general forking lemma* by Bellare and Neven [BN06]. This lemma is used in security proofs in the ROM that *fork* the adversary's execution, as is done, for example, to prove security of Schnorr signatures [PS96]: the reduction runs the adversary twice with the same randomness; however, from a specific random-oracle (RO) query on, the reduction samples fresh answers in the second run. The (general) forking lemma gives a lower bound on the probability that the adversary succeeds in both runs and that both solutions are related to the RO call where the fork happened.

For the reduction for Schnorr signatures to succeed, the RO replies in both runs must be different (which happens with overwhelming probability). In our proof of hardness of ZT we require something more general: every run defines a set S of size at most d (which will

be the roots of the solution polynomial), and we require the RO reply in the second run not to fall in S (so we can argue that the polynomials in the two runs are different). The term $1/|R|$, where R is the range of the RO, in the original bound [BN06] thus becomes $d/|R|$ and setting $S(\mathsf{out}) := \{y_I\}$ recovers the original lemma.

We give a proof of the following in Appendix A.1.

Lemma 1 (d-General Forking Lemma). *Let $d, q \in \mathbb{N}$ and $R \neq \emptyset$ be a finite set. Let IG be a randomized algorithm that generates an input x. Let $\mathcal{A}$ be a randomized algorithm that takes as input x and $y_1, \ldots, y_q \in R$, and outputs an index $I \in [q] \cup \{\bot\}$ and side output out. Let S be a PT function that takes as input out and outputs a set $S(\mathsf{out}) \subset R$ of size $|S(\mathsf{out})| \leq d$. Let $\mathsf{acc}(\mathcal{A})$ be the probability that $\mathcal{A}$ outputs an index $I \neq \bot$, i.e.*

$$\mathsf{acc}(\mathcal{A}) := \Pr\left[x \leftarrow \mathsf{IG};\ y_1, \ldots, y_q \leftarrow R;\ (I, \mathsf{out}) \leftarrow \mathcal{A}(x, y_1, \ldots, y_q)\ :\ I \neq \bot\right].$$

Let $\mathcal{F}_{\mathcal{A}}$ be the forking algorithm associated to $\mathcal{A}$ defined in Constr. 3, and

$$\mathsf{fork}(\mathcal{F}_{\mathcal{A}}) := \Pr\left[x \leftarrow \mathsf{IG};\ \mathsf{sol} \leftarrow \mathcal{F}_{\mathcal{A}}(x)\ :\ \mathsf{sol} \neq \bot\right].$$

Then

$$\mathsf{acc}(\mathcal{A}) \leq \sqrt{q \cdot \mathsf{fork}(\mathcal{F}_{\mathcal{A}})} + \frac{qd}{|R|}. \tag{7}$$

Construction 3: Forking algorithm associated to algorithm $\mathcal{A}$.

$\underline{\mathcal{F}_{\mathcal{A}}(x)}$:

1. Pick randomness ρ for $\mathcal{A}$.
2. $y_1, \ldots, y_q \leftarrow R$.
3. $(I, \mathsf{out}) := \mathcal{A}(x, y_1, \ldots, y_q; \rho)$.
4. If $I = \bot$, return $\bot$.
5. $y'_I, \ldots, y'_q \leftarrow R$.
6. $(I', \mathsf{out}') := \mathcal{A}(x, y_1, \ldots, y_{I-1}, y'_I, \ldots, y'_q; \rho)$.
7. If $I' \neq I$ or $y'_I \in S(\mathsf{out})$, return $\bot$.
8. Return $(\mathsf{out}, \mathsf{out}')$.

Theorem 3 (Binding $\Rightarrow$ ZT in ROM). *Let $H\colon \{0,1\}^* \to \mathbb{F}$ be a hash function and $\Gamma = (\mathsf{Setup}, \mathsf{Com})$ be a commitment scheme with message space $\mathcal{M}$. If H is modeled as a random oracle, then for all $n \in \mathbb{N}$, every function (family) $D\colon \mathcal{M} \to \mathbb{F}^{(\leq d)}[X]$, and every adversary $\mathcal{A}$ making at most q random-oracle queries, there exists an adversary $\mathcal{B}$ running in time roughly twice that of $\mathcal{A}$ such that*

$$\mathsf{Adv}^{(n,D)\text{-}\mathsf{ZT},\mathsf{ROM}}_{\mathcal{A},\Gamma}(\lambda) \leq \sqrt{(q+1) \cdot \mathsf{Adv}^{\mathsf{bind},n}_{\mathcal{B},\Gamma}(\lambda)} + \frac{(q+1)d}{|\mathbb{F}|}.$$

Remark 7. Note that for ZT to possibly hold (also in the standard model), the commitment scheme must satisfy at least the following weak form of binding: given a message m, let $\alpha := H(\mathsf{Com}_{\mathsf{par}}(m))$; then it must be hard to find a *second preimage* m' (i.e., $\mathsf{Com}_{\mathsf{par}}(m') = \mathsf{Com}_{\mathsf{par}}(m)$) with the additional property that $D(m')$ is non-zero and has a root at α.

Proof. Let $\mathcal{A}$ be a ZT adversary, which, on input commitment parameters par, returns $m \in \mathcal{M}$. We assume that $\mathcal{A}$'s random-oracle queries are all distinct and that it queries $H(\mathsf{Com}_{\mathsf{par}}(m))$ prior to returning m. This is w.l.o.g. when we increase the value of q by 1. Let $\mathcal{A}'$ be the wrapper around adversary $\mathcal{A}$ defined in Constr. 4.

Construction 4: Wrapper around adversary $\mathcal{A}$.

$\underline{\mathcal{A}'(\mathsf{par}, y_1, \ldots, y_{q+1}; \rho)}$:

1. Run $m := \mathcal{A}^H(\mathsf{par}; \rho)$, answering the i-th query to H with y_i.
2. $f := D(m)$.
3. If $f \equiv 0$ or $f(H(\mathsf{Com}_{\mathsf{par}}(m))) \neq 0$, return $(\bot, m)$.
4. Let $I \in [q+1]$ be the index of the query to H on input $\mathsf{Com}_{\mathsf{par}}(m)$.
5. Return (I, m).

Set $R := \mathbb{F}$ and $\mathsf{IG} := \mathsf{Setup}$; since $\mathcal{A}'$ returns $I \neq \bot$ iff $\mathcal{A}$ breaks ZT w.r.t. D, we have

$$\mathsf{Adv}^{(n,D)\text{-}\mathsf{ZT},\mathsf{ROM}}_{\mathcal{A},\Gamma}(\lambda) = \mathsf{acc}(\mathcal{A}'). \tag{8}$$

Next, define S as the function that maps m to the set of roots of the polynomial $D(m)$, namely $S\colon m \mapsto \{x \in \mathbb{F} \mid f(x) = 0 \text{ for } f := D(m)\}$. We now use the forking algorithm $\mathcal{B} := \mathcal{F}_{\mathcal{A}'}$ (Constr. 3) with function S as an adversary against the binding property of the commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{Com})$, written out in Constr. 5.

Construction 5: Reduction to the binding property of Γ.

$\underline{\mathcal{B}(\mathsf{par})}$:

1. Pick randomness ρ for $\mathcal{A}'$.
2. $y_1, \ldots, y_q \leftarrow \mathbb{F}$.
3. $(I, m) := \mathcal{A}'(\mathsf{par}, y_1, \ldots, y_q; \rho)$.
4. If $I = \bot$, return $\bot$.
5. $y'_I, \ldots, y'_q \leftarrow \mathbb{F}$.
6. $(I', m') := \mathcal{A}'(\mathsf{par}, y_1, \ldots, y_{I-1}, y'_I, \ldots, y'_q; \rho)$.
7. If $I' \neq I$ or $y'_I \in S(m)$, return $\bot$.
8. Return (m, m').

We claim that

$$\mathsf{Adv}^{\mathsf{bind},n}_{\mathcal{B},\Gamma}(\lambda) = \mathsf{fork}(\mathcal{F}_{\mathcal{A}'}). \tag{9}$$

By the checks in lines 4 and 7, $\mathcal{B} = \mathcal{F}_{\mathcal{A}'}$ returns $(m, m') \neq \bot$ (i.e., fork holds) if and only if (i) $I' = I \neq \bot$ and (ii) $y'_I \notin S(m)$. Considering the definition of $\mathcal{A}'$ (Constr. 4), I (resp., I') is the index of the H-query on input $\mathsf{Com}_{\mathsf{par}}(m)$ (resp., $\mathsf{Com}_{\mathsf{par}}(m')$) in the first (resp., second) run of the ZT adversary $\mathcal{A}$. From (i), and since the two runs are identical up to this query, we have

$$\mathsf{Com}_{\mathsf{par}}(m) = \mathsf{Com}_{\mathsf{par}}(m'). \tag{10}$$

Moreover, from the second successful run of $\mathcal{A}'$ we get that y'_I is a root of the non-zero polynomial $f' := D(m')$. However, by (ii), y'_I is not a root of the polynomial $f := D(m)$. Thus, $f \not\equiv f'$, which implies their preimages under D must also be distinct, i.e., $m \neq m'$. Together with (10), this shows that $\mathcal{B}$ breaks binding of Γ iff fork holds, establishing (9).

Applying Lemma 1 together with Eqs. (8) and (9) now directly yields the bound in the theorem. □

3.3 ZT for KZG in the GGM

Instead of idealizing the hash function as a random oracle, we next turn to justifying the hardness of ZT for a real hash function H, now idealizing the used commitment scheme.

In particular, motivated by its use in Plonk, we consider KZG commitments (Constr. 1), which are defined over a bilinear group. For simplicity of exposition, we restrict KZG to $\mathbb{G}_1$, which we idealize using Shoup's generic group model (GGM) [Sho97]. Since a solution for ZT with KZG does not depend on $\mathbb{G}_2$ and $\mathbb{G}_T$ elements, our proof can easily be extended to the bilinear GGM [BBG05].

In the GGM security game, the adversary is given random encodings instead of actual group elements, defined by an injective function σ mapping their discrete logarithms to bitstrings of length $\lceil \log p \rceil$. The adversary has access to an oracle which, given the encodings of two group elements, returns the encoding of their sum. We show that balancedness (which is implied by collision resistance) of the hash function implies ZT in the GGM.

Theorem 4 (KZG in GGM & Balancedness ⇒ ZT). *Let $H\colon \{0,1\}^* \to \mathbb{F}$ be a hash function. If the group $\mathbb{G}_1$ used in the* KZG *commitment scheme is modeled as a generic group, then there exits a PPT adversary $\mathcal{B}$ such that for all $n \in \mathbb{N}$, every function (family) $D\colon \mathbb{F}^{(\leq n)}[X] \to \mathbb{F}^{(\leq d)}[X]$, and every adversary $\mathcal{A}$ making at most q group-operation queries:*

$$\mathsf{Adv}^{(n,D)\text{-}\mathsf{ZT},\mathsf{GGM}}_{\mathcal{A},(H,\mathsf{KZG})}(\lambda) \leq (q+n+2) \cdot \left(d \cdot \mathsf{Adv}^{\mathsf{bal}}_{\mathcal{B},H}(\lambda) + \frac{(q+n+1)(n+1)}{2p} \right).$$

For simplicity, instead of providing the GGM adversary with a group-operation oracle that takes as input previously generated group-element labels, its oracle GroupOp takes coefficients and returns the (label of the) corresponding linear combination of the group elements generated by the experiment. Any adversary against our formulation is easily converted into an adversary for the traditional GGM formalization. (Note that $\mathsf{GroupOp}((a_i)_{i=1}^n)$ thus corresponds to the KZG commitment of the polynomial with coefficients a_i.)

Proof. Let $\mathcal{A}$ be an adversary against ZT with H and KZG in the GGM. We assume (without loss of generality) that $\mathcal{A}$ queries GroupOp on its output m and makes at most $q+1$ (rather than q) queries to GroupOp.

We define a sequence of games $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$ in Fig. 1, where $\mathbf{G}_0$ corresponds to the interaction of $\mathcal{A}$ with the ZT experiment in the GGM (which runs GGen to obtain the order p of the generic group). We have

$$\Pr[\mathbf{G}_0] = \mathsf{Adv}^{(n,D)\text{-}\mathsf{ZT},\mathsf{GGM}}_{\mathcal{A},(H,\mathsf{KZG})}(\lambda). \tag{11}$$

$\mathbf{G}_0 \to \mathbf{G}_1$. In $\mathbf{G}_1$, we change the sampling of group element labels: they are now sampled uniformly from $\{0,1\}^{\lceil \log p \rceil}$, and $\mathbf{G}_1$ aborts if a label is sampled twice (line 3 of Label). For the i-th call to Label, the probability that $\xi \in \mathrm{ran}(\sigma)$ is at most $(i-1)/p$. By a union bound over at most $q+n+2$ calls to Label throughout $\mathbf{G}_1$, and because $\mathbf{G}_0$ and $\mathbf{G}_1$ are equivalent until the abort, we get

$$\Pr[\mathbf{G}_0] \leq \Pr[\mathbf{G}_1] + \sum_{i=0}^{q+n+1} \frac{i}{p} = \Pr[\mathbf{G}_1] + \frac{(q+n+2)(q+n+1)}{2p}. \tag{12}$$

$\mathbf{G}_1 \to \mathbf{G}_2$. Next, in $\mathbf{G}_2$, we replace τ with an indeterminate X. Thus, $\mathbf{G}_2$ is equivalent to $\mathbf{G}_1$ if none of the formal polynomials on which Label is evaluated throughout the game agree at $\tau \leftarrow \mathbb{F}$. Since there are at most $q+n+2$ such formal polynomials, each of degree at most n, we get

$$\Pr[\mathbf{G}_1] \leq \Pr[\mathbf{G}_2] + \binom{q+n+2}{2} \cdot \frac{n}{p} = \Pr[\mathbf{G}_2] + \frac{(q+n+2)(q+n+1)}{2} \cdot \frac{n}{p}. \tag{13}$$

$\mathbf{G}_0(\lambda, n)$, $\boxed{\mathbf{G}_1(\lambda, n)}$, $\boxed{\boxed{\mathbf{G}_2(\lambda, n)}}$:

1. $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$, where $|\mathbb{F}| = p$.
2. $\sigma := \emptyset$.
3. $\tau \leftarrow \mathbb{F}$.
4. $\mathsf{srs} \leftarrow \big(\mathsf{Label}(1), \mathsf{Label}(\tau), \ldots, \mathsf{Label}(\tau^n)\big)$.
5. $\boxed{\boxed{\mathsf{srs} \leftarrow \big(\mathsf{Label}(1), \mathsf{Label}(X), \ldots, \mathsf{Label}(X^n)\big).}}$
6. $m = (m_i)_{i=0}^n \in \mathbb{F}^{n+1} \leftarrow \mathcal{A}^{\mathsf{GroupOp}}(\mathsf{srs})$
7. $f := D(m)$.
8. If $f \not\equiv 0$ and $f\big(H(\mathsf{GroupOp}(m))\big) = 0$, return 1. Else, return 0.

$\mathsf{Label}(x \in \mathbb{F}^{(\leq n)}[X])$:

1. If $x \in \mathrm{dom}(\sigma)$, return $\sigma(x)$.
2. $\xi \leftarrow \{0,1\}^{\lceil \log p \rceil} \setminus \mathrm{ran}(\sigma)$.
3. $\boxed{\xi \leftarrow \{0,1\}^{\lceil \log p \rceil}.$ If $\xi \in \mathrm{ran}(\sigma)$, abort.$}$
4. $\sigma := \sigma \cup \{(x, \xi)\}$.
5. Return ξ.

$\mathsf{GroupOp}\big((a_i)_{i=0}^n \in \mathbb{F}^{n+1}\big)$:

1. $\xi \leftarrow \mathsf{Label}(\sum_{i=0}^n a_i \cdot \tau^i)$.
2. $\boxed{\boxed{\xi \leftarrow \mathsf{Label}(\sum_{i=0}^n a_i \cdot X^i).}}$
3. Return ξ.

Fig. 1. The games $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$ for the proof of Theorem 4. $\mathbf{G}_0$ ignores all boxed code, $\mathbf{G}_1$ ignores only double-boxed code, and $\mathbf{G}_2$ includes all code.

$\mathbf{G}_2$. To analyze $\mathcal{A}$'s success probability in $\mathbf{G}_2$, we define the following adversary $\mathcal{B}$ against balancedness of $H\colon \{0,1\}^* \to \mathbb{F}$ in Constr. 6.

Construction 6: Adversary against balancedness of H.

$\mathcal{B}(1^\lambda)$:

1. $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$, where $|\mathbb{F}| = p$.
2. Return $\xi \leftarrow \{0,1\}^{\lceil \log p \rceil}$.

Note that $\mathcal{B}$ satisfies the precondition in Def. 9:

$$\mathsf{MaxPr}_{\mathcal{B}}(\lambda) := \max_{x \in \{0,1\}^*} \Pr\left[\xi \leftarrow \mathcal{B}(1^\lambda) : \xi = x\right] = \frac{1}{2^{\lceil \log p \rceil}} \leq \frac{1}{p} = \frac{1}{|\mathbb{F}|}.$$

Moreover, recall that

$$\mathsf{Adv}^{\mathsf{bal}}_{\mathcal{B},H}(\lambda) := \max_{y \in \mathbb{F}} \Pr\left[\xi \leftarrow \mathcal{B}(1^\lambda) : H(\xi) = y\right].$$

Let $\mathbf{a}_i \in \mathbb{F}^{n+1}$ denote $\mathcal{A}$'s i-th input to $\mathsf{GroupOp}$. Moreover, for $i \in [n+1]$, define $\mathbf{e}_i := (\ldots, 0, 1, 0, \ldots) \in \mathbb{F}^{n+1}$ as the i-th standard basis vector (these correspond to the polynomials $1, X, \ldots, X^n$ whose KZG commitments are contained in the srs given to $\mathcal{A}$). Since we are assuming that $\mathcal{A}$ queries $\mathsf{GroupOp}$ on its output m, its success probability

in $\mathbf{G}_2$ is bounded by the probability that there exists a ZT solution m^* in the set $S \coloneqq \{\mathbf{a}_i\}_{i\in[q+1]} \cup \{\mathbf{e}_i\}_{i\in[n+1]}$, i.e., for which $f \coloneqq D(m^*)$ is non-zero and $f(H(\mathsf{GroupOp}(m^*))) = 0$.

Note that for each element $m \in S$, its KZG commitment (the output of $\mathsf{GroupOp}(m)$) is independently sampled by $\mathbf{G}_2$ as $\xi \leftarrow \{0,1\}^{\lceil \log p \rceil}$, which is how $\mathcal{B}$ samples its output. Thus, the probability that $H(\mathsf{GroupOp}(m))$ is a root of the degree-d polynomial $D(m)$ is at most $d \cdot \mathsf{Adv}^{\mathsf{bal}}_{\mathcal{B},H}(\lambda)$. Since $|S| \le q+n+2$, we get

$$\Pr[\mathbf{G}_2] \le (q+n+2)d \cdot \mathsf{Adv}^{\mathsf{bal}}_{\mathcal{B},H}(\lambda). \tag{14}$$

Finally, combining Eqs. (11) to (14) finishes the proof. □

Together with Theorem 1 and $\mathsf{MaxPr}_{\mathcal{B}}(\lambda) \le \frac{1}{p}$, we also get the following corollary.

Corollary 1. *Let $H\colon \{0,1\}^* \to \mathbb{F}$ be a hash function. If the group $\mathbb{G}_1$ used in the KZG commitment scheme is modeled as a generic group, then there exits a PPT adversary $\mathcal{C}$ such that for all $n \in \mathbb{N}$, every function (family) $D\colon \mathbb{F}^{(\le n)}[X] \to \mathbb{F}^{(\le d)}[X]$, and every adversary $\mathcal{A}$ making at most q group-operation queries:*

$$\mathsf{Adv}^{(n,D)\text{-}\mathsf{ZT},\mathsf{GGM}}_{\mathcal{A},(H,\mathsf{KZG})}(\lambda) \le (q+n+2) \cdot \left(d \cdot \sqrt{\mathsf{Adv}^{\mathsf{cr}}_{\mathcal{C},\mathcal{H}}(\lambda) + \frac{1}{p}} + \frac{(q+n+1)(n+1)}{2p} \right).$$

3.4 Relationship to Correlation Intractability

A keyed function H is *correlation-intractable* [CGH98] w.r.t. a relation $\mathcal{R}$ if it is hard, given a random key k, to find x such that $(x, H_k(x)) \in \mathcal{R}$. Correlation intractability (CI) is only possible for *sparse* relations, where for any x, only a negligible fraction of y satisfy $(x,y) \in \mathcal{R}$. The zero-testing assumption (Def. 11) for (H, Com) and a function D can be cast as CI of

$$H'_k(m) \coloneqq H(\mathsf{Com}_k(m)) \quad \text{for} \quad \mathcal{R}_{\mathsf{ZT}} \coloneqq \{(m,y) \mid f \coloneqq D(m) \not\equiv 0 \text{ and } f(y) = 0\}.$$

If D has low-degree outputs, $\mathcal{R}_{\mathsf{ZT}}$ is sparse and efficiently decidable. We can instantiate H' for ZT using a (keyed) correlation-intractable hash function Com that supports $\mathcal{R}_{\mathsf{ZT}}$ by defining

$$H'_k(m) \coloneqq H(\mathsf{Com}_k(m)) \quad \text{with} \quad H \coloneqq \mathrm{id}\,.$$

4 Plonk

We give a condensed summary of Plonk and its underlying constraint system. For more details and intuition, refer to the original paper by Gabizon et al. [GWC19].

4.1 Specification

Let $\mathcal{C}$ be an arithmetic circuit over $\mathbb{F}$ with fan-in two, unlimited fan-out, n gates, and ℓ public inputs $(x_i)_{i\in[\ell]} \in \mathbb{F}^\ell$ (which are also counted as gates). For simplicity, assume the public inputs are associated with the first ℓ gates. Let $(w_i)_{i\in[3n]} \in \mathbb{F}^{3n}$ be a (redundant) wire assignment, where w_i, w_{n+i}, and w_{2n+i} represent the values assigned to the left input, right input, and output wire of the i-th gate, respectively. Using the Plonk constraint system [GWC19, Section 6], $\mathcal{C}$ is then modeled as a combination of gate and copy constraints:

Table 1. Selector vector assignment of Plonk.

Gate type	$s_{\mathsf{L}i}$	$s_{\mathsf{R}i}$	$s_{\mathsf{O}i}$	$s_{\mathsf{M}i}$	$s_{\mathsf{C}i}$
Addition	1	1	-1	0	0
Multiplication	0	0	-1	1	0
Constant c_i	1	0	0	0	$-c_i$
Public input	1	0	0	0	0

1. The *gate constraints* ensure correctness of the individual gate operations, that is, the values of w_i, w_{n+i} and w_{2n+i} are consistent with gate i. Concretely, the i-th gate is associated with a constraint of the form

$$s_{\mathsf{L}i} \cdot w_i + s_{\mathsf{R}i} \cdot w_{n+i} + s_{\mathsf{O}i} \cdot w_{2n+i} + s_{\mathsf{M}i} \cdot w_i \cdot w_{n+i} + s_{\mathsf{C}i} = x_i,$$

   where $x_i \in \mathbb{F}$ is a public input if $i \leq \ell$ and 0 otherwise, and the values of the selector vectors $\mathbf{s_L}, \mathbf{s_R}, \mathbf{s_O}, \mathbf{s_M}, \mathbf{s_C} \in \mathbb{F}^n$ depend on the gate type as specified in Table 1.
2. The *copy constraints* enforce that gate outputs are propagated correctly, e.g., if the output of gate i is wired to the left input of gate j, then w_{2n+i} and w_j must be assigned the same value. This is achieved by choosing a permutation $\sigma\colon [3n] \to [3n]$ with appropriate cycles such that "$w_i = w_{\sigma(i)}$ for all $i \in [3n]$" captures all the values in $(w_i)_{i\in[3n]}$ that must be equal. In the previous example, σ would contain the cycle $(2n+i, j)$ such that $w_{2n+i} = w_j$.

Together, this results in the indexed relation $\mathcal{R}_{\mathsf{Plonk}} :=$

$$\left\{ (\mathtt{i}, \mathbb{x}, \mathbb{w}) \;\middle|\; \begin{array}{l} \mathtt{i} = \big((\mathbf{s_L}, \mathbf{s_R}, \mathbf{s_O}, \mathbf{s_M}, \mathbf{s_C}) \in \mathbb{F}^{5n},\ \sigma\colon [3n] \to [3n]\big), \\ \mathbb{x} = (x_i)_{i\in[\ell]} \in \mathbb{F}^{\ell},\ \mathbb{w} = (w_i)_{i\in[3n]} \in \mathbb{F}^{3n},\ (x_i)_{i=\ell+1}^{n} := \mathbf{0}^{n-\ell}, \\ \forall i \in [n]\colon s_{\mathsf{L}i} w_i + s_{\mathsf{R}i} w_{n+i} + s_{\mathsf{O}i} w_{2n+i} + s_{\mathsf{M}i} w_i w_{n+i} + s_{\mathsf{C}i} = x_i, \\ \forall i \in [3n]\colon w_i = w_{\sigma(i)} \end{array} \right\}, \tag{15}$$

where the index $\mathtt{i}$ is (the Plonk representation of) the arithmetic circuit $\mathcal{C}$, the statement $\mathbb{x}$ its public inputs, and the witness $\mathbb{w}$ a consistent wire assignment to $\mathcal{C}$.

Let ω be a primitive n-th root of unity in $\mathbb{F}$ and define $\mathbb{H} := \langle\omega\rangle$ as the subgroup generated by ω. The core of the Plonk proof system is to express a statement $(\mathtt{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{Plonk}}$ as an equivalent polynomial divisibility of the form

$$Z_{\mathbb{H}}(X) \;\mid\; \mathsf{Gates}(X) + \alpha \cdot \mathsf{Copy}_1(X) + \alpha^2 \cdot \mathsf{Copy}_2(X),$$

where $Z_{\mathbb{H}}(X) := \prod_{i\in[n]}(X - \omega^i) = X^n - 1$ is the vanishing polynomial over $\mathbb{H}$, the polynomials $\mathsf{Gates}(X)$, $\mathsf{Copy}_1(X)$, $\mathsf{Copy}_2(X)$ depend on the given statement, and $\alpha \in \mathbb{F}$ is a randomly chosen challenge from the verifier. Leveraging the Schwartz–Zippel lemma, the prover then convinces the verifier that this divisibility holds at a random evaluation challenge $\delta \in \mathbb{F}$ selected by the verifier.

To obtain non-interactive proofs, Plonk applies the Fiat–Shamir transformation [FS87]. For this purpose, let $H\colon \{0,1\}^* \to \mathbb{F}$ be a hash function, and let trans_0 denote all public information, i.e., the SRS, the common preprocessed input, and all the public circuit inputs $(x_i)_{i\in[\ell]}$.[12] Each verifier challenge is then computed by evaluating H on the concatenation of trans_0 and all protocol messages exchanged between the prover and the verifier up to that point.

In the following, let $\mathcal{L}_i \in \mathbb{F}^{(\leq n-1)}[X]$ denote the i-th Lagrange basis polynomial over $\mathbb{H} = \langle\omega\rangle$, i.e., $\mathcal{L}_i(\omega^i) = 1$ and $\mathcal{L}_i(\omega^j) = 0$ for all $j \neq i \in [n]$. Moreover, $k_1, k_2 \in \mathbb{F}^*$ are

[12] Note that including all of this information in trans_0 is crucial to prevent certain attacks on knowledge soundness [Mil22,DMWG23].

chosen such that $\mathbb{H}$, $k_1\mathbb{H}$, $k_2\mathbb{H}$ are distinct cosets of $\mathbb{H}$, and the permutation $\sigma\colon [3n] \to [3n]$ is extended onto $\mathbb{H}' := \mathbb{H} \cup k_1\mathbb{H} \cup k_2\mathbb{H}$ by defining the wire permutation $\sigma^*\colon [3n] \to \mathbb{H}'$ as

$$\sigma^*(i) := \begin{cases} \omega^{\sigma(i)} & \text{if } \sigma(i) \in [n], \\ k_1 \cdot \omega^{\sigma(i)} & \text{if } \sigma(i) \in (n, 2n], \\ k_2 \cdot \omega^{\sigma(i)} & \text{if } \sigma(i) \in (2n, 3n]. \end{cases} \tag{16}$$

We give a detailed description of the Plonk protocol in Constr. 7.

Construction 7: The zk-SNARK Plonk.

$\underline{\mathsf{Setup}(1^\lambda, n)}$: Given a bound n on the circuit size, run $\mathsf{srs} \leftarrow \mathsf{KZG.Setup}(1^\lambda, n+2)$ and select a hash function $H\colon \{0,1\}^* \to \mathbb{F}$, given implicitly to all other algorithms. Return

$$\mathsf{srs} := \left(\mathcal{G}, [\tau]_1, \left[\tau^2\right]_1, \ldots, \left[\tau^{n+2}\right]_1, [\tau]_2\right).$$

$\underline{\mathsf{Index}(\mathsf{srs}, \mathbb{i})}$: Given $\mathbb{i}$, consisting of the gate constraints $(s_{\mathsf{L}i}, s_{\mathsf{R}i}, s_{\mathsf{O}i}, s_{\mathsf{M}i}, s_{\mathsf{C}i})_{i\in[n]}$ and the wire permutation σ^*, compute the polynomials

$$\begin{aligned} S_{\mathsf{L}}(X) &:= \sum\nolimits_{i\in[n]} s_{\mathsf{L}i}\mathcal{L}_i(X), & S_{\mathsf{R}}(X) &:= \sum\nolimits_{i\in[n]} s_{\mathsf{R}i}\mathcal{L}_i(X), \\ S_{\mathsf{O}}(X) &:= \sum\nolimits_{i\in[n]} s_{\mathsf{O}i}\mathcal{L}_i(X), & S_{\mathsf{M}}(X) &:= \sum\nolimits_{i\in[n]} s_{\mathsf{M}i}\mathcal{L}_i(X), \\ S_{\mathsf{C}}(X) &:= \sum\nolimits_{i\in[n]} s_{\mathsf{C}i}\mathcal{L}_i(X), & S_{\sigma,1}(X) &:= \sum\nolimits_{i\in[n]} \sigma^*(i)\mathcal{L}_i(X), \\ S_{\sigma,2}(X) &:= \sum\nolimits_{i\in[n]} \sigma^*(n+i)\mathcal{L}_i(X), & S_{\sigma,3}(X) &:= \sum\nolimits_{i\in[n]} \sigma^*(2n+i)\mathcal{L}_i(X), \end{aligned}$$

and their KZG commitments with respect to srs,

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

Return the prover and verifier parameters

$$\begin{aligned} \mathsf{pp} &:= (\mathsf{srs}, S_{\mathsf{L}}, S_{\mathsf{R}}, S_{\mathsf{O}}, S_{\mathsf{M}}, S_{\mathsf{C}}, S_{\sigma,1}, S_{\sigma,2}, S_{\sigma,3}), \\ \mathsf{vp} &:= (\mathsf{srs}, \mathsf{cm}_{S_{\mathsf{L}}}, \mathsf{cm}_{S_{\mathsf{R}}}, \mathsf{cm}_{S_{\mathsf{O}}}, \mathsf{cm}_{S_{\mathsf{M}}}, \mathsf{cm}_{S_{\mathsf{C}}}, \mathsf{cm}_{S_{\sigma,1}}, \mathsf{cm}_{S_{\sigma,2}}, \mathsf{cm}_{S_{\sigma,3}}). \end{aligned}$$

$\underline{\mathsf{Prove}\big(\mathsf{pp}, (x_i)_{i\in[\ell]}, (w_i)_{i\in[3n]}\big)}$:

1. Sample blinding scalars $\rho_1, \ldots, \rho_6 \leftarrow \mathbb{F}$ and compute the wire polynomials

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

   The first part of the proof are the commitments

$$\mathsf{cm}_A := [A(\tau)]_1, \ \mathsf{cm}_B := [B(\tau)]_1, \ \mathsf{cm}_C := [C(\tau)]_1.$$

2. Let $\mathsf{trans}_0 := \mathsf{vp}\|\mathbb{x}$ with $\mathbb{x} := (x_i)_{i\in[\ell]}$. Compute the permutation challenges $\beta, \gamma \in \mathbb{F}$:

$$\begin{aligned} \beta &:= H(\mathsf{trans}_1), \ \text{where } \mathsf{trans}_1 := \mathsf{trans}_0\|\mathsf{cm}_A\|\mathsf{cm}_B\|\mathsf{cm}_C, \\ \gamma &:= H(\mathsf{trans}_2), \ \text{where } \mathsf{trans}_2 := \mathsf{trans}_1\|\beta. \end{aligned}$$

3. Define the polynomials

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

where $S_{\mathrm{id},1}(X) \coloneqq X$, $S_{\mathrm{id},2}(X) \coloneqq k_1 X$, $S_{\mathrm{id},3}(X) \coloneqq k_2 X$ are polynomials interpolating the identity permutation $\mathrm{id}^* \colon [3n] \to \mathbb{H}'$ over $\mathbb{H}$.

Sample blinding scalars $\rho_7, \rho_8, \rho_9 \leftarrow \mathbb{F}$ and compute the permutation-check polynomial

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

If $g(\omega^j) = 0$ for some $j \in [n]$, abort. The second part of the proof is the commitment

$$\mathsf{cm}_\Phi \coloneqq [\Phi(\tau)]_1 .$$

4. Compute the quotient challenge $\alpha \in \mathbb{F}$ as

$$\alpha \coloneqq H(\mathsf{trans}_3), \text{ where } \mathsf{trans}_3 \coloneqq \mathsf{trans}_2 \| \gamma \| \mathsf{cm}_\Phi.$$

5. Define the public-input polynomial $S_{\mathsf{PI}}(X) \coloneqq \sum_{i \in [\ell]} x_i \mathcal{L}_i(X)$, and

$$\mathsf{Gates}(X) \coloneqq \begin{pmatrix} S_{\mathsf{L}}(X)A(X) + S_{\mathsf{R}}(X)B(X) + S_{\mathsf{O}}(X)C(X) \\ + S_{\mathsf{M}}(X)A(X)B(X) + S_{\mathsf{C}}(X) - S_{\mathsf{PI}}(X) \end{pmatrix},$$
$$\mathsf{Copy}_1(X) \coloneqq \Phi(X)f(X) - \Phi(X\omega)g(X),$$
$$\mathsf{Copy}_2(X) \coloneqq \mathcal{L}_1(X)(\Phi(X) - 1).$$

Compute the quotient polynomial of degree $\leq 3n + 5$

$$T(X) \coloneqq \frac{\mathsf{Gates}(X) + \alpha \cdot \mathsf{Copy}_1(X) + \alpha^2 \cdot \mathsf{Copy}_2(X)}{Z_{\mathbb{H}}(X)},$$

and split it into 3 unique polynomials $T'_1, T'_2, T'_3 \in \mathbb{F}^{(\leq n+1)}[X]$ such that

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

Then, sample blinding scalars $\rho_{10}, \rho_{11} \leftarrow \mathbb{F}$ and define

$$T_1(X) \coloneqq T'_1(X) + \rho_{10} X^{n+2},$$
$$T_2(X) \coloneqq T'_2(X) - \rho_{10} + \rho_{11} X^{n+2},$$
$$T_3(X) \coloneqq T'_3(X) - \rho_{11}.$$

(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 part of the proof are the commitments

$$\mathsf{cm}_{T_1} \coloneqq [T_1(\tau)]_1 , \ \mathsf{cm}_{T_2} \coloneqq [T_2(\tau)]_1 , \ \mathsf{cm}_{T_3} \coloneqq [T_3(\tau)]_1 .$$

6. Compute the evaluation challenge $\delta \in \mathbb{F}$ as

$$\delta \coloneqq H(\mathsf{trans}_4), \text{ where } \mathsf{trans}_4 \coloneqq \mathsf{trans}_3 \| \alpha \| \mathsf{cm}_{T_1} \| \mathsf{cm}_{T_2} \| \mathsf{cm}_{T_3}.$$

7. The fourth part of the proof are the evaluations

$$a_\delta \coloneqq A(\delta), \qquad b_\delta \coloneqq B(\delta), \qquad c_\delta \coloneqq C(\delta),$$
$$s_{\sigma_1,\delta} \coloneqq S_{\sigma,1}(\delta), \quad s_{\sigma_2,\delta} \coloneqq S_{\sigma,2}(\delta), \quad \phi_{\delta\omega} \coloneqq \Phi(\delta\omega).$$

8. Compute the opening challenge $\varepsilon \in \mathbb{F}$ as

$$\varepsilon := H(\mathsf{trans}_5), \text{ where } \mathsf{trans}_5 := \mathsf{trans}_4 \| \delta \| a_\delta \| b_\delta \| c_\delta \| s_{\sigma_1,\delta} \| s_{\sigma_2,\delta} \| \phi_{\delta\omega}.$$

9. Define the linearized polynomials

$$L_{\mathsf{Gates}}(X) := \begin{pmatrix} S_{\mathsf{L}}(X)A(\delta) + S_{\mathsf{R}}(X)B(\delta) + S_{\mathsf{O}}(X)C(\delta) \\ + S_{\mathsf{M}}(X)A(\delta)B(\delta) + S_{\mathsf{C}}(X) - S_{\mathsf{PI}}(\delta) \end{pmatrix},$$

$$L_{\mathsf{Copy}_1}(X) := \begin{pmatrix} \Phi(X)(A(\delta) + \beta\delta + \gamma)(B(\delta) + \beta k_1 \delta + \gamma)(C(\delta) + \beta k_2 \delta + \gamma) \\ -\Phi(\delta\omega)(A(\delta)+\beta S_{\sigma,1}(\delta)+\gamma)(B(\delta)+\beta S_{\sigma,2}(\delta)+\gamma)(C(\delta)+\beta S_{\sigma,3}(X)+\gamma) \end{pmatrix},$$

$$L_{\mathsf{Copy}_2}(X) := \mathcal{L}_1(\delta)(\Phi(X) - 1),$$

$$L_{\mathsf{div}}(X) := Z_{\mathbb{H}}(\delta)\big(T_1(X) + \delta^{n+2} \cdot T_2(X) + \delta^{2n+4} \cdot T_3(X)\big),$$

$$L(X) := L_{\mathsf{Gates}}(X) + \alpha \cdot L_{\mathsf{Copy}_1}(X) + \alpha^2 \cdot L_{\mathsf{Copy}_2}(X) - L_{\mathsf{div}}(X).$$

Compute the (aggregated) quotient polynomials

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

$$Q_2(X) := \frac{\Phi(X) - \Phi(\delta\omega)}{X - \delta\omega}.$$

The fifth part of the proof are the commitments (KZG opening proofs)

$$\mathsf{cm}_{Q_1} := [Q_1(\tau)]_1, \ \mathsf{cm}_{Q_2} := [Q_2(\tau)]_1.$$

Return the full proof

$$\pi := \begin{pmatrix} \mathsf{cm}_A, \mathsf{cm}_B, \mathsf{cm}_C, \mathsf{cm}_\Phi, \mathsf{cm}_{T_1}, \mathsf{cm}_{T_2}, \mathsf{cm}_{T_3}, \mathsf{cm}_{Q_1}, \mathsf{cm}_{Q_2}, \\ a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega} \end{pmatrix} \in \mathbb{G}_1^9 \times \mathbb{F}^6.$$

$\underline{\mathsf{Verify}\big(\mathsf{vp}, (x_i)_{i\in[\ell]}, \pi\big)}$: Parse verifier parameters vp as $(\mathsf{srs}, \mathsf{cm}_{S_\mathsf{L}}, \mathsf{cm}_{S_\mathsf{R}}, \mathsf{cm}_{S_\mathsf{O}}, \mathsf{cm}_{S_\mathsf{M}}, \mathsf{cm}_{S_\mathsf{C}}, \mathsf{cm}_{S_{\sigma,1}}, \mathsf{cm}_{S_{\sigma,2}}, \mathsf{cm}_{S_{\sigma,3}})$. Compute the challenges $\beta, \gamma, \alpha, \delta, \varepsilon \in \mathbb{F}$ the same way as the prover, and the multipoint evaluation challenge $\zeta \in \mathbb{F}$ as

$$\zeta := H(\mathsf{trans}_6), \text{ where } \mathsf{trans}_6 := \mathsf{trans}_5 \| \varepsilon \| \mathsf{cm}_{Q_1} \| \mathsf{cm}_{Q_2}.$$

Using $Z_{\mathbb{H}}(\delta) = \delta^n - 1$, $\mathcal{L}_i(\delta) = \frac{\omega^i(\delta^n - 1)}{n(\delta - \omega^i)}$, and $S_{\mathsf{PI}}(\delta) = \sum_{i\in[\ell]} x_i \mathcal{L}_i(\delta)$, compute the commitments

$$\mathsf{cm}_{L_{\mathsf{Gates}}} := a_\delta \cdot \mathsf{cm}_{S_\mathsf{L}} + b_\delta \cdot \mathsf{cm}_{S_\mathsf{R}} + c_\delta \cdot \mathsf{cm}_{S_\mathsf{O}} + a_\delta b_\delta \cdot \mathsf{cm}_{S_\mathsf{M}} + \mathsf{cm}_{S_\mathsf{C}} - [S_{\mathsf{PI}}(\delta)]_1,$$

$$\mathsf{cm}_{L_{\mathsf{Copy}_1}} := \begin{pmatrix} (a_\delta + \beta\delta + \gamma)(b_\delta + \beta k_1 \delta + \gamma)(c_\delta + \beta k_2 \delta + \gamma) \cdot \mathsf{cm}_\Phi \\ - \phi_{\delta\omega}(a_\delta + \beta s_{\sigma_1,\delta} + \gamma)(b_\delta + \beta s_{\sigma_2,\delta} + \gamma) \cdot \big([c_\delta]_1 + \beta \cdot \mathsf{cm}_{S_{\sigma,3}} + [\gamma]_1\big) \end{pmatrix},$$

$$\mathsf{cm}_{L_{\mathsf{Copy}_2}} := \mathcal{L}_1(\delta) \cdot \big(\mathsf{cm}_\Phi - [1]_1\big),$$

$$\mathsf{cm}_{L_{\mathsf{div}}} := Z_{\mathbb{H}}(\delta) \cdot \big(\mathsf{cm}_{T_1} + \delta^{n+2} \cdot \mathsf{cm}_{T_2} + \delta^{2n+4} \cdot \mathsf{cm}_{T_3}\big),$$

$$\mathsf{cm}_L := \mathsf{cm}_{L_{\mathsf{Gates}}} + \alpha \cdot \mathsf{cm}_{L_{\mathsf{Copy}_1}} + \alpha^2 \cdot \mathsf{cm}_{L_{\mathsf{Copy}_2}} - \mathsf{cm}_{L_{\mathsf{div}}},$$

$$\mathsf{cm}_1 := \begin{pmatrix} \mathsf{cm}_L + \varepsilon \cdot \big(\mathsf{cm}_A - [a_\delta]_1\big) + \varepsilon^2 \cdot \big(\mathsf{cm}_B - [b_\delta]_1\big) + \varepsilon^3 \cdot \big(\mathsf{cm}_C - [c_\delta]_1\big) \\ + \varepsilon^4 \cdot \big(\mathsf{cm}_{S_{\sigma,1}} - [s_{\sigma_1,\delta}]_1\big) + \varepsilon^5 \cdot \big(\mathsf{cm}_{S_{\sigma,2}} - [s_{\sigma_2,\delta}]_1\big) \end{pmatrix},$$

$$\mathsf{cm}_2 := \mathsf{cm}_\Phi - [\phi_{\delta\omega}]_1.$$

Return

$$e\big(\mathsf{cm}_1 + \delta \cdot \mathsf{cm}_{Q_1} + \zeta \cdot (\mathsf{cm}_2 + \delta\omega \cdot \mathsf{cm}_{Q_2}), [1]_2\big) \stackrel{?}{=} e\big(\mathsf{cm}_{Q_1} + \zeta \cdot \mathsf{cm}_{Q_2}, [\tau]_2\big).$$

4.2 Knowledge Soundness of Plonk from ZT

Before stating our main theorem, we define extKZG (Constr. 8), an extension of the KZG polynomial commitment scheme (Constr. 1) that fits the syntax of ZT in our security proof. The scheme lets one commit to *vectors* of polynomials by using KZG component-wise. For a degree bound n fixed by $\mathsf{KZG.Setup}$, we associate degree-n polynomials with their coefficient vectors in $\mathbb{F}^{n+1}$ (coefficients above the leading one are set to 0). Moreover, we allow inclusion of single field elements, which are simply copied to the commitment vector. Finally, extKZG prepends the SRS to the output commitment.

Construction 8: Extension extKZG of the KZG commitment scheme.

$\underline{\mathsf{Setup}(1^\lambda, n)}$: Run $\mathsf{srs} \leftarrow \mathsf{KZG.Setup}(1^\lambda, n)$ and return srs.

$\underline{\mathsf{Com}_{\mathsf{srs}}(m)}$:

1. Parse m as $(m_1, \ldots, m_k)$.
2. For $i \in [k]$, set $\mathsf{cm}_i \coloneqq \begin{cases} m_i & \text{if } m_i \in \mathbb{F}, \\ \mathsf{KZG.Com}_{\mathsf{srs}}(m_i) & \text{if } m_i = (m_{i,0}, \ldots, m_{i,n}) \in \mathbb{F}^{n+1}. \end{cases}$
3. Return $\mathsf{cm} \coloneqq (\mathsf{srs}, \mathsf{cm}_1, \ldots, \mathsf{cm}_k)$.

This abstraction captures "committed" protocol transcripts of Plonk. For example, consider the first challenge $\beta \coloneqq H(\mathsf{trans}_1)$ with

$$\mathsf{trans}_1 \coloneqq (\mathsf{srs}, \mathsf{cm}_{S_\mathsf{L}}, \mathsf{cm}_{S_\mathsf{R}}, \mathsf{cm}_{S_\mathsf{O}}, \mathsf{cm}_{S_\mathsf{M}}, \mathsf{cm}_{S_\mathsf{C}}, \mathsf{cm}_{S_{\sigma,1}}, \mathsf{cm}_{S_{\sigma,2}}, \mathsf{cm}_{S_{\sigma,3}}, \mathbb{x}, \mathsf{cm}_A, \mathsf{cm}_B, \mathsf{cm}_C),$$

and recall that Plonk uses a KZG degree bound of $n+2$. Then the message

$$m \coloneqq (S_\mathsf{L}, S_\mathsf{R}, S_\mathsf{O}, S_\mathsf{M}, S_\mathsf{C}, S_{\sigma,1}, S_{\sigma,2}, S_{\sigma,3}, \mathbb{x}, A, B, C),$$

which, associating polynomials with their coefficient vectors, is in $(\mathbb{F}^{n+3})^8 \times \mathbb{F}^\ell \times (\mathbb{F}^{n+3})^3$, satisfies $\mathsf{extKZG.Com}_{\mathsf{srs}}(m) = \mathsf{trans}_1$.

It is straightforward to see that extKZG is binding if and only if the underlying KZG commitment scheme is. Furthermore, we define the following notion of compatibility:

Definition 13 (Compatibility with Plonk Transcripts). *A message m is* compatible *with trans_j for $j \in [6]$ if and only if the following conditions are met:*

1. *m parsed as $(m_1, \ldots, m_k)$ and trans_j parsed as $(\mathsf{srs}, \mathsf{cm}_1, \ldots, \mathsf{cm}_{k'})$ satisfy $k = k'$.*
2. *If $m_i \in \mathbb{F}$, then $\mathsf{cm}_i \in \mathbb{F}$; and if $m_i \in \mathbb{F}^{n+3}$, then $\mathsf{cm}_i \in \mathbb{G}_1$.*
3. *m contains a prefix $(m_1, \ldots, m_{\ell+8}) \in (\mathbb{F}^{n+3})^8 \times \mathbb{F}^\ell$ corresponding to $(S_\mathsf{L}, S_\mathsf{R}, S_\mathsf{O}, S_\mathsf{M}, S_\mathsf{C}, S_{\sigma,1}, S_{\sigma,2}, S_{\sigma,3}, \mathbb{x})$ that represents a valid Plonk representation of a circuit with ℓ public inputs (cf. Section 4).*
4. *After this prefix, m contains components $A \coloneqq m_{\ell+9}, B \coloneqq m_{\ell+10}, C \coloneqq m_{\ell+11} \in \mathbb{F}^{n+3}$. Let $\mathbb{i}$ be the Plonk representation of the circuit defined from $m_1, \ldots, m_8$, $\mathbb{x} \coloneqq (m_{i+8})_{i \in [\ell]}$, and $\mathbb{w} \coloneqq (w_i)_{i \in [3n]}$ where*

$$w_i \coloneqq \begin{cases} A(\omega^i) & \text{if } i \in [n], \\ B(\omega^i) & \text{if } i \in (n, 2n], \\ C(\omega^i) & \text{if } i \in (2n, 3n]. \end{cases} \tag{17}$$

Then $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}_{\mathsf{Plonk}}$.

Theorem 5 (Knowledge Soundness of Plonk in the AGM from ZT). Plonk *(Constr. 7) is knowledge-sound in the algebraic group model if the used hash function H has the zero-testing property for the function $D_{\mathsf{ks}}\colon \mathcal{M} \to \mathbb{F}^{(\leq 4n+8)}[X][Y]$ defined in Constr. 10. Specifically, for every algebraic adversary $\mathcal{A}$, there exists an extractor $\mathcal{E}_{\mathcal{A}}$ and adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$, each running in time roughly the same as $\mathcal{A}$, such that*

$$\mathsf{Adv}^{\mathsf{ks},n}_{\mathcal{A},\mathcal{E}_{\mathcal{A}},\mathsf{Plonk}}(\lambda) \leq \mathsf{Adv}^{(n+2)\text{-}\mathsf{DL}}_{\mathcal{B}_1,\mathsf{GGen}}(\lambda) + \mathsf{Adv}^{(n+2,D_{\mathsf{ks}})\text{-}\mathsf{mZT}}_{\mathcal{B}_2,(H,\mathsf{extKZG})}(\lambda).$$

Proof. Consider an algebraic adversary $\mathcal{A}$ against knowledge soundness (Def. 6) of Plonk and its output:

$$\begin{aligned}
\mathbb{i} &\coloneqq \big((\mathbf{s_L}, \mathbf{s_R}, \mathbf{s_O}, \mathbf{s_M}, \mathbf{s_C}) \in \mathbb{F}^{5n},\ \sigma\colon [3n] \to [3n]\big)\\
\mathbb{x} &\coloneqq (x_1, \ldots, x_\ell) \in \mathbb{F}^{\ell}\\
\pi &\coloneqq \left(\begin{matrix}\mathsf{cm}_A, \mathsf{cm}_B, \mathsf{cm}_C, \mathsf{cm}_\Phi, \mathsf{cm}_{T_1}, \mathsf{cm}_{T_2}, \mathsf{cm}_{T_3}, \mathsf{cm}_{Q_1}, \mathsf{cm}_{Q_2},\\ a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega}\end{matrix}\right) \in \mathbb{G}_1^9 \times \mathbb{F}^6
\end{aligned}$$

Extractor. Let $(a_i, b_i, c_i, \phi_i, t_{1,i}, t_{2,i}, t_{3,i}, q_{1,i}, q_{2,i})_{i=0}^{n+2} \in \mathbb{F}^{9(n+3)}$ be the AGM representations in basis $\big([\tau^i]_1\big)_{i=0}^{n+2}$ of the group elements in the proof π, respectively. From these representations, define the following degree-$(n+2)$ polynomials:

$$\begin{aligned}
A(X) &\coloneqq \sum_{i=0}^{n+2} a_i X^i & B(X) &\coloneqq \sum_{i=0}^{n+2} b_i X^i & C(X) &\coloneqq \sum_{i=0}^{n+2} c_i X^i\\
\Phi(X) &\coloneqq \sum_{i=0}^{n+2} \phi_i X^i & T_1(X) &\coloneqq \sum_{i=0}^{n+2} t_{1,i} X^i & T_2(X) &\coloneqq \sum_{i=0}^{n+2} t_{2,i} X^i\\
T_3(X) &\coloneqq \sum_{i=0}^{n+2} t_{3,i} X^i & Q_1(X) &\coloneqq \sum_{i=0}^{n+2} q_{1,i} X^i & Q_2(X) &\coloneqq \sum_{i=0}^{n+2} q_{2,i} X^i
\end{aligned} \tag{18}$$

Define the extractor $\mathcal{E}_{\mathcal{A}}$ as specified in Constr. 9. Note that this is the same extractor as used in the original security proof by Gabizon et al. in the AGM+ROM [GWC19].

Construction 9: The extractor for the proof of Theorem 5.

$\underline{\mathcal{E}_{\mathcal{A}}(\mathsf{srs}, \rho)}$:

1. Run $(\mathbb{i}, \mathbb{x}, \pi) \leftarrow \mathcal{A}(\mathsf{srs}; \rho)$, implicitly obtaining $\mathcal{A}$'s AGM representations.
2. From these, define the polynomials $A, B, C \in \mathbb{F}^{(\leq n+2)}[X]$ as in the text.
3. For $i \in [n]$, compute $w_i \coloneqq A(\omega^i)$, $w_{n+i} \coloneqq B(\omega^i)$, $w_{2n+i} \coloneqq C(\omega^i)$.
4. Output $\mathbb{w} \coloneqq (w_i)_{i \in [3n]}$.

Definition of function D_{ks} for ZT. Let

$$\mathsf{trans}_0 \coloneqq (\mathsf{srs}, \mathsf{cm}_{S_{\mathsf{L}}}, \mathsf{cm}_{S_{\mathsf{R}}}, \mathsf{cm}_{S_{\mathsf{O}}}, \mathsf{cm}_{S_{\mathsf{M}}}, \mathsf{cm}_{S_{\mathsf{C}}}, \mathsf{cm}_{S_{\sigma,1}}, \mathsf{cm}_{S_{\sigma,2}}, \mathsf{cm}_{S_{\sigma,3}}, \mathbb{x})$$

and define the challenges as well as trans_i for $i \in [6]$ as in Plonk (Constr. 7):

$$\begin{aligned}
\beta &\coloneqq H(\mathsf{trans}_1), \text{ where } \mathsf{trans}_1 \coloneqq \mathsf{trans}_0 \| \mathsf{cm}_A \| \mathsf{cm}_B \| \mathsf{cm}_C\\
\gamma &\coloneqq H(\mathsf{trans}_2), \text{ where } \mathsf{trans}_2 \coloneqq \mathsf{trans}_1 \| \beta\\
\alpha &\coloneqq H(\mathsf{trans}_3), \text{ where } \mathsf{trans}_3 \coloneqq \mathsf{trans}_2 \| \gamma \| \mathsf{cm}_\Phi\\
\delta &\coloneqq H(\mathsf{trans}_4), \text{ where } \mathsf{trans}_4 \coloneqq \mathsf{trans}_3 \| \alpha \| \mathsf{cm}_{T_1} \| \mathsf{cm}_{T_2} \| \mathsf{cm}_{T_3}\\
\varepsilon &\coloneqq H(\mathsf{trans}_5), \text{ where } \mathsf{trans}_5 \coloneqq \mathsf{trans}_4 \| \delta \| a_\delta \| b_\delta \| c_\delta \| s_{\sigma_1,\delta} \| s_{\sigma_2,\delta} \| \phi_{\delta\omega}\\
\zeta &\coloneqq H(\mathsf{trans}_6), \text{ where } \mathsf{trans}_6 \coloneqq \mathsf{trans}_5 \| \varepsilon \| \mathsf{cm}_{Q_1} \| \mathsf{cm}_{Q_2}
\end{aligned} \tag{19}$$

We define several (multivariate) polynomials, from which we construct the function D_{ks} for ZT from the theorem. For these, from $\mathbb{i}$ we derive the associated polynomials $S_{\mathsf{L}}, S_{\mathsf{R}}, S_{\mathsf{O}}, S_{\mathsf{M}}, S_{\mathsf{C}}, S_{\sigma,1}, S_{\sigma,2}, S_{\sigma,3}$ as in Plonk.Index (Constr. 7), and from $\mathbb{x}$ we define $S_{\mathsf{PI}}(X) \coloneqq \sum_{i\in[\ell]} x_i \mathcal{L}_i(X)$.

Construction 10: The function $D_{\mathsf{ks}}\colon \mathcal{M} \to \mathbb{F}^{(\leq 4n+8)}[X][Y]$ for Plonk.

$\underline{D_{\mathsf{ks}}(m)}$:

1. If m is compatible (Def. 13) with
$$\begin{cases} \mathsf{trans}_1, & \text{return } P_\sigma(X,Y) \text{ from (20)},\\ \mathsf{trans}_2, & \text{return } P_\sigma(\beta,X) \text{ from (20)},\\ \mathsf{trans}_3, & \text{return } R_\alpha(X,Y) \text{ from (24)},\\ \mathsf{trans}_4, & \text{return } P_\delta(X) \text{ from (25)},\\ \mathsf{trans}_5, & \text{return } R_\varepsilon(X,Y) \text{ from (26)},\\ \mathsf{trans}_6, & \text{return } V(X,Y) \text{ from (28)}. \end{cases}$$
2. Else, return 0.

Consider an m that is compatible with trans_1, thus of the form:

$$m = (S_{\mathsf{L}}, S_{\mathsf{R}}, S_{\mathsf{O}}, S_{\mathsf{M}}, S_{\mathsf{C}}, S_{\sigma,1}, S_{\sigma,2}, S_{\sigma,3}, \mathbb{x}, A, B, C)$$

From these, define f and g (corresponding to step 3 of Plonk.Prove), which define P_σ:

$$\begin{aligned} &f(X_\beta, X_\gamma, X_\delta)\\ &\coloneqq \big(A(X_\delta) + X_\beta X_\delta + X_\gamma\big)\big(B(X_\delta) + X_\beta k_1 X_\delta + X_\gamma\big)\big(C(X_\delta) + X_\beta k_2 X_\delta + X_\gamma\big) \end{aligned}$$

$$\begin{aligned} &g(X_\beta, X_\gamma, X_\delta)\\ &\coloneqq \big(A(X_\delta) + X_\beta S_{\sigma,1}(X_\delta) + X_\gamma\big)\big(B(X_\delta) + X_\beta S_{\sigma,2}(X_\delta) + X_\gamma\big)\big(C(X_\delta) + X_\beta S_{\sigma,3}(X_\delta) + X_\gamma\big) \end{aligned}$$

$$P_\sigma(X_\beta, X_\gamma) \coloneqq \prod_{i\in[n]} f(X_\beta, X_\gamma, \omega^i) - \prod_{i\in[n]} g(X_\beta, X_\gamma, \omega^i) \tag{20}$$

This polynomial captures the copy constraints of Plonk and is zero if and only if $w_i = w_{\sigma(i)}$ for all $i \in [3n]$ (see [GWC19, Claim A.1] for a proof). Note that an m that is compatible with trans_2 additionally contains the challenge β, and so can be used to derive $P_\sigma(\beta, X_\gamma)$.

Next, consider an m that is compatible with trans_3, thus of the form:

$$m = (\ldots, \mathbb{x}, A, B, C, \beta, \gamma, \Phi)$$

From these elements we define the following polynomials:

$$\begin{aligned} \mathsf{Gates}(X_\delta) \coloneqq\ & S_{\mathsf{L}}(X_\delta)A(X_\delta) + S_{\mathsf{R}}(X_\delta)B(X_\delta) + S_{\mathsf{O}}(X_\delta)C(X_\delta)\\ &+ S_{\mathsf{M}}(X_\delta)A(X_\delta)B(X_\delta) + S_{\mathsf{C}}(X_\delta) - S_{\mathsf{PI}}(X_\delta) \end{aligned} \tag{21}$$

$$\mathsf{Copy}_1(X_\delta) \coloneqq \Phi(X_\delta) f(\beta,\gamma,X_\delta) - \Phi(X_\delta \cdot \omega) g(\beta,\gamma,X_\delta) \tag{22}$$

$$\mathsf{Copy}_2(X_\delta) \coloneqq \mathcal{L}_1(X_\delta)\big(\Phi(X_\delta) - 1\big) \tag{23}$$

The following bivariate polynomial captures the batched polynomial division performed in step 5 of Plonk.Prove and is zero if and only if the polynomials $\mathsf{Gates}, \mathsf{Copy}_1, \mathsf{Copy}_2$ are all divisible by $Z_{\mathbb{H}}$:

$$\begin{aligned} R_\alpha(X_\alpha, X_\delta) \coloneqq\ & [\mathsf{Gates}(X_\delta) \bmod Z_{\mathbb{H}}(X_\delta)] + X_\alpha \cdot [\mathsf{Copy}_1(X_\delta) \bmod Z_{\mathbb{H}}(X_\delta)]\\ &+ X_\alpha^2 \cdot [\mathsf{Copy}_2(X_\delta) \bmod Z_{\mathbb{H}}(X_\delta)] \end{aligned} \tag{24}$$

Next, consider an m that is compatible with trans_4, thus of the form:

$$m = (\ldots, \mathbb{x}, A, B, C, \beta, \gamma, \Phi, \alpha, T_1, T_2, T_3)$$

We define the polynomial $P_\delta(X_\delta)$, which corresponds to the main polynomial identity checked by Plonk at the evaluation challenge δ assuming that all provided evaluations $\delta, a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}$ are correct:

$$\begin{aligned}
\mathsf{div}(X_\delta) &\coloneqq Z_{\mathbb{H}}(X_\delta)\big(T_1(X_\delta) + X_\delta^{n+2} \cdot T_2(X_\delta) + X_\delta^{2n+4} \cdot T_3(X_\delta)\big) \\
P_\delta(X_\delta) &\coloneqq \mathsf{Gates}(X_\delta) + \alpha \cdot \mathsf{Copy}_1(X_\delta) + \alpha^2 \cdot \mathsf{Copy}_2(X_\delta) - \mathsf{div}(X_\delta)
\end{aligned} \tag{25}$$

Next, consider an m that is compatible with trans_5, thus of the form:

$$m = (\ldots, \mathbb{x}, A, B, C, \beta, \gamma, \Phi, \alpha, T_1, T_2, T_3, \delta, a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega})$$

From these we construct the following polynomials, which correspond to the commitments computed by $\mathsf{Plonk.Verify}$ (e.g., $[L_{\mathsf{Gates}}(\tau)]_1 = \mathsf{cm}_{L_{\mathsf{Gates}}}$, etc.):

$$\begin{aligned}
L_{\mathsf{Gates}}(X_\tau) &\coloneqq S_{\mathsf{L}}(X_\tau)a_\delta + S_{\mathsf{R}}(X_\tau)b_\delta + S_{\mathsf{O}}(X_\tau)c_\delta + S_{\mathsf{M}}(X_\tau)a_\delta b_\delta + S_{\mathsf{C}}(X_\tau) - S_{\mathsf{PI}}(\delta), \\
L_{\mathsf{Copy}_1}(X_\tau) &\coloneqq \Phi(X_\tau)(a_\delta + \beta\delta + \gamma)(b_\delta + \beta k_1\delta + \gamma)(c_\delta + \beta k_2\delta + \gamma) \\
&\quad - \phi_{\delta\omega}(a_\delta + \beta s_{\sigma_1,\delta} + \gamma)(b_\delta + \beta s_{\sigma_2,\delta} + \gamma)(c_\delta + \beta S_{\sigma,3}(X_\tau) + \gamma), \\
L_{\mathsf{Copy}_2}(X_\tau) &\coloneqq \mathcal{L}_1(\delta)(\Phi(X_\tau) - 1), \\
L_{\mathsf{div}}(X_\tau) &\coloneqq Z_{\mathbb{H}}(\delta)\big(T_1(X_\tau) + \delta^{n+2} \cdot T_2(X_\tau) + \delta^{2n+4} \cdot T_3(X_\tau)\big), \\
L(X_\tau) &\coloneqq L_{\mathsf{Gates}}(X_\tau) + \alpha \cdot L_{\mathsf{Copy}_1}(X_\tau) + \alpha^2 \cdot L_{\mathsf{Copy}_2}(X_\tau) - L_{\mathsf{div}}(X_\tau),
\end{aligned}$$

These define the following polynomial, which corresponds to the batched KZG opening of the polynomials $L, A, B, C, S_{\sigma,1}, S_{\sigma,2}$ at the evaluation challenge δ (cf. step 9 of $\mathsf{Plonk.Prove}$):

$$\begin{aligned}
R_\varepsilon(X_\varepsilon, X_\tau) \coloneqq\ & [L(X_\tau) \bmod (X_\tau - \delta)] + X_\varepsilon \cdot [A(X_\tau) - a_\delta \bmod (X_\tau - \delta)] \\
& + X_\varepsilon^2 \cdot [B(X_\tau) - b_\delta \bmod (X_\tau - \delta)] + X_\varepsilon^3 \cdot [C(X_\tau) - c_\delta \bmod (X_\tau - \delta)] \\
& + X_\varepsilon^4 \cdot [S_{\sigma,1}(X_\tau) - s_{\sigma_1,\delta} \bmod (X_\tau - \delta)] \\
& + X_\varepsilon^5 \cdot [S_{\sigma,2}(X_\tau) - s_{\sigma_2,\delta} \bmod (X_\tau - \delta)]
\end{aligned} \tag{26}$$

Finally, consider an m that is compatible with trans_6, thus of the form:

$$m = (\ldots, \mathbb{x}, A, B, C, \beta, \gamma, \Phi, \alpha, T_1, T_2, T_3, \delta, a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega}, \varepsilon, Q_1, Q_2) \tag{27}$$

From these we construct the following polynomials, of which C_1 and C_2 correspond to cm_1 and cm_2 in $\mathsf{Plonk.Verify}$ and V to the final verification equation. (Note that we have $R_\varepsilon(\varepsilon, X_\tau) = [C_1(X_\tau) \bmod (X_\tau - \delta)]$.)

$$\begin{aligned}
C_1(X_\tau) &\coloneqq L(X_\tau) + \varepsilon \cdot \big(A(X_\tau) - a_\delta\big) + \varepsilon^2 \cdot \big(B(X_\tau) - b_\delta\big) + \varepsilon^3 \cdot \big(C(X_\tau) - c_\delta\big) \\
&\quad + \varepsilon^4 \cdot \big(S_{\sigma,1}(X_\tau) - s_{\sigma_1,\delta}\big) + \varepsilon^5 \cdot \big(S_{\sigma,2}(X_\tau) - s_{\sigma_2,\delta}\big) \\
C_2(X_\tau) &\coloneqq \Phi(X_\tau) - \phi_{\delta\omega} \\
V_1(X_\tau) &\coloneqq C_1(X_\tau) - Q_1(X_\tau)\big(X_\tau - \delta\big) \\
V_2(X_\tau) &\coloneqq C_2(X_\tau) - Q_2(X_\tau)\big(X_\tau - \delta\omega\big) \\
V(X_\zeta, X_\tau) &\coloneqq V_1(X_\tau) + X_\zeta \cdot V_2(X_\tau)
\end{aligned} \tag{28}$$

$\mathbf{G}_0(\lambda, n)$, $\boxed{\mathbf{G}_1(\lambda, n)}$, $\boxed{\boxed{\mathbf{G}_2(\lambda, n)}}$:

1. $\mathsf{srs} \leftarrow \mathsf{Plonk.Setup}(1^\lambda, n)$.
2. $\big((\mathbb{i}, \mathbb{x}, \pi); \mathbb{w}\big) \leftarrow (\mathcal{A} \parallel \mathcal{E}_\mathcal{A})(\mathsf{srs})$.
3. $\boxed{\text{From } \mathcal{A}\text{'s AGM representations, define } V(X_\zeta, X_\tau) \text{ as in Eq. (28).}}$
4. $(\mathsf{pp}, \mathsf{vp}) := \mathsf{Plonk.Index}(\mathsf{srs}, \mathbb{i})$.
5. If $\mathsf{Plonk.Verify}(\mathsf{vp}, \mathbb{x}, \pi) \neq 1$, return 0.
6. $\boxed{\text{If } V(\zeta, X_\tau) \not\equiv 0, \text{ abort.}}$ (SolvePDL)
7. If $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{Plonk},n}$, return 0.
8. $\boxed{\boxed{\text{Abort.}}}$ (SolveZT)
9. Return 1.

Fig. 2. The games $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$ for the proof of Theorem 5. $\mathbf{G}_0$ ignores all boxed code, $\mathbf{G}_1$ ignores only double-boxed code, and $\mathbf{G}_2$ includes all code.

Indeed, since from (18) we have $\mathsf{cm}_{Q_i} = [Q_i(\tau)]_1$, if $\mathsf{Plonk.Verify}(\mathsf{vp}, \mathbb{x}, \pi) = 1$, we have

$$\begin{aligned}
& e\big(\mathsf{cm}_1 + \delta \cdot \mathsf{cm}_{Q_1} + \zeta \cdot (\mathsf{cm}_2 + \delta\omega \cdot \mathsf{cm}_{Q_2}), [1]_2\big) = e\big(\mathsf{cm}_{Q_1} + \zeta \cdot \mathsf{cm}_{Q_2}, [\tau]_2\big) \\
\iff \quad & C_1(\tau) + \delta Q_1(\tau) + \zeta \cdot \big(C_2(\tau) + \delta\omega Q_2(\tau)\big) = \tau Q_1(\tau) + \zeta \cdot \tau Q_2(\tau) \\
\iff \quad & C_1(\tau) - Q_1(\tau)(\tau - \delta) + \zeta \cdot \big(C_2(\tau) - Q_2(\tau)(\tau - \delta\omega)\big) = 0 \\
\iff \quad & V(\zeta, \tau) = 0.
\end{aligned}$$

Hybrid games. We define a sequence of games $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$ in Fig. 2, where $\mathbf{G}_0$ corresponds to the knowledge-soundness experiment in Def. 6, i.e.,

$$\Pr[\mathbf{G}_0] = \mathsf{Adv}^{\mathsf{ks},n}_{\mathcal{A},\mathcal{E}_\mathcal{A},\mathsf{Plonk}}(\lambda). \tag{29}$$

$\mathbf{G}_0 \to \mathbf{G}_1$. If the abort (SolvePDL) introduced in $\mathbf{G}_1$ happens, then verification passes and

$$V(\zeta, X_\tau) \not\equiv 0.$$

As derived after Eq. (28), we have

$$\mathsf{Plonk.Verify}(\mathsf{vp}, \mathbb{x}, \pi) = 1 \iff V(\zeta, \tau) = 0.$$

Thus, the trapdoor τ is a root of the non-zero degree-$(n+3)$ polynomial $V(\zeta, X)$. In this case, the adversary $\mathcal{B}_1$ given in Constr. 11 solves $(n+2)$-DL, showing that

$$\Pr[\mathbf{G}_0] \leq \Pr[\mathbf{G}_1] + \mathsf{Adv}^{(n+2)\text{-DL}}_{\mathcal{B}_1,\mathsf{GGen}}(\lambda). \tag{30}$$

Construction 11: Reduction to $(n+2)$-DL.

$\mathcal{B}_1(\mathcal{G}, [\tau]_1, \ldots, [\tau^{n+2}]_1, [\tau]_2)$:

1. $\mathsf{srs} := (\mathcal{G}, [\tau]_1, \ldots, [\tau^{n+2}]_1, [\tau]_2)$.
2. $(\mathbb{i}, \mathbb{x}, \pi) \leftarrow \mathcal{A}(\mathsf{srs})$.
3. From $\mathcal{A}$'s AGM representations derive $V(X_\zeta, X_\tau)$ as in Eq. (28).
4. From π derive ζ as in (19).
5. If $0 \not\equiv V(\zeta, X) \in \mathbb{F}^{(\leq n+3)}[X]$, factor it. Otherwise, abort.
6. If one of the roots $x_i \in \mathbb{F}$ satisfies $[x_i]_1 = [\tau]_1$, return x_i. Otherwise, abort.

$\mathbf{G}_1 \to \mathbf{G}_2$. Consider the adversary $\mathcal{B}_2$ against $(n+2, D_{\mathsf{ks}})$-ZT given in Constr. 12.

Construction 12: Reduction to $(n+2, D_{\mathsf{ks}})$-ZT.

$\mathcal{B}_2(\mathcal{G}, [\tau]_1, \ldots, [\tau^{n+2}]_1, [\tau]_2)$:

1. $\mathsf{srs} := (\mathcal{G}, [\tau]_1, \ldots, [\tau^{n+2}]_1, [\tau]_2)$.
2. $(\mathbb{i}, \mathbb{x}, \pi) \leftarrow \mathcal{A}(\mathsf{srs})$.
3. From $\mathcal{A}$'s AGM representations derive the polynomials $V, R_\varepsilon, \ldots$ used below, as well as the preimage of trans_6 from Eq. (27).
4. $(\mathsf{pp}, \mathsf{vp}) := \mathsf{Plonk.Index}(\mathsf{srs}, \mathbb{i})$.
5. If $\mathsf{Plonk.Verify}(\mathsf{vp}, \mathbb{x}, \pi) \neq 1$, abort.
6. If $V(\zeta, X_\tau) \not\equiv 0$, abort.
7. If $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{Plonk},n}$, abort.
8. Set $i := \begin{cases} 6 & \text{if } V(X_\zeta, X_\tau) \not\equiv 0, \\ 5 & \text{if } R_\varepsilon(X_\varepsilon, X_\tau) \not\equiv 0, \\ 4 & \text{if } P_\delta(X_\delta) \not\equiv 0, \\ 3 & \text{if } R_\alpha(X_\alpha, X_\delta) \not\equiv 0, \\ 2 & \text{if } P_\sigma(\beta, X_\gamma) \not\equiv 0, \\ 1 & \text{otherwise.} \end{cases}$
9. Return m such that $\mathsf{extKZG.Com}_{\mathsf{srs}}(m) = \mathsf{trans}_i$.

We claim that if the abort ($\mathsf{SolveZT}$) introduced in $\mathbf{G}_2$ happens, then $\mathcal{B}_2$ breaks the multivariate zero-testing assumption on (H, extKZG) w.r.t. D_{ks}. To show this, we make a case distinction based on step 8 of $\mathcal{B}_2$:

$i = 6$. In this case, we have $V(X_\zeta, X_\tau) \not\equiv 0$ (cf. Eq. (28)), but also, since the abort in step 6 did not occur, $V(\zeta, X_\tau) \equiv 0$. Thus, $\zeta := H(\mathsf{trans}_6)$ is a root of the non-zero polynomial $V(X, Y) \in \mathbb{F}[X][Y]$ of degree 1 in X with coefficients in $\mathbb{F}[X]$. The extKZG preimage m of trans_6 (cf. Eq. (27)), computed from $\mathcal{A}$'s AGM representations, is thus a solution to multivariate ZT (Def. 12) for D_{ks} from Constr. 10: $D_{\mathsf{ks}}(m) = V(X, Y) \not\equiv 0$ and $V(H(\mathsf{extKZG.Com}_{\mathsf{srs}}(m)), Y) = V(\zeta, Y) \equiv 0$.

$i = 5$. In this case, we have $R_\varepsilon(X_\varepsilon, X_\tau) \not\equiv 0$ (cf. Eq. (26)), but also

$$V(X_\zeta, X_\tau) \equiv 0,$$

which implies $V_1(X_\tau) \equiv V_2(X_\tau) \equiv 0$. Since $V_1(X_\tau) := C_1(X_\tau) - Q_1(X_\tau)(X_\tau - \delta)$, we get

$$(X_\tau - \delta) \mid C_1(X_\tau).$$

Using the observation after Eq. (27), this means that $R_\varepsilon(\varepsilon, X_\tau) = [C_1(X_\tau) \bmod (X_\tau - \delta)] \equiv 0$. Thus, $\varepsilon := H(\mathsf{trans}_5)$ is a root of the non-zero polynomial $R_\varepsilon(X, Y) \in \mathbb{F}[X][Y]$ of degree 5 in X with coefficients in $\mathbb{F}[X]$. The extKZG preimage m of trans_5 is thus a solution to multivariate ZT.

$i = 4$. In this case, we have $P_\delta(X_\delta) \not\equiv 0$ (cf. Eq. (25)), but also $R_\varepsilon(X_\varepsilon, X_\tau) \equiv 0$, which implies that all the claimed evaluations at δ output by $\mathcal{A}$ are correct, namely

$$L(\delta) = 0,\ A(\delta) = a_\delta,\ B(\delta) = b_\delta,\ C(\delta) = c_\delta,\ S_{\sigma,1}(\delta) = s_{\sigma_1,\delta},\ S_{\sigma,2}(\delta) = s_{\sigma_2,\delta}.$$

As in the previous case, we also have $V_2(X_\tau) \equiv 0$. Plugging in $\delta\omega$ for X_τ, we obtain $V_2(\delta\omega) = 0$, which implies $\Phi(\delta\omega) = \phi_{\delta\omega}$, i.e., the claimed evaluation of Φ at $\delta\omega$ is also correct. Thus, we get

$$P_\delta(\delta) = L(\delta) = 0,$$

and $\delta \coloneqq H(\mathsf{trans}_4)$ is a root of the non-zero degree-$(4n+8)$ polynomial $P_\delta(X)$.

$\underline{i = 3.}$ In this case, we have $R_\alpha(X_\alpha, X_\delta) \not\equiv 0$ (cf. Eq. (24)), but also $P_\delta(X_\delta) \equiv 0$, i.e.,

$$\mathsf{Gates}(X_\delta) + \alpha \cdot \mathsf{Copy}_1(X_\delta) + \alpha^2 \cdot \mathsf{Copy}_2(X_\delta) \equiv \mathsf{div}(X_\delta).$$

Since the right-hand side is a multiple of $Z_{\mathbb{H}}(X_\delta)$, we get:

$$Z_{\mathbb{H}}(X_\delta) \mid \mathsf{Gates}(X_\delta) + \alpha \cdot \mathsf{Copy}_1(X_\delta) + \alpha^2 \cdot \mathsf{Copy}_2(X_\delta).$$

The latter implies $R_\alpha(\alpha, X_\delta) \equiv 0$, and so $\alpha \coloneqq H(\mathsf{trans}_3)$ is a root of the non-zero polynomial $R_\alpha(X, Y) \in \mathbb{F}[X][Y]$ of degree 2 in X with coefficients in $\mathbb{F}[X]$.

$\underline{i = 2.}$ In this case, we have $P_\sigma(\beta, X_\gamma) \not\equiv 0$ (cf. Eq. (20)), but also $R_\alpha(X_\alpha, X_\delta) \equiv 0$, which implies:

$$Z_{\mathbb{H}}(X_\delta) \mid \mathsf{Gates}(X_\delta), \tag{31}$$
$$Z_{\mathbb{H}}(X_\delta) \mid \mathsf{Copy}_1(X_\delta), \tag{32}$$
$$Z_{\mathbb{H}}(X_\delta) \mid \mathsf{Copy}_2(X_\delta). \tag{33}$$

Moreover, recall from Eq. (15) that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{Plonk},n}$ if and only if

(a) $\forall i \in [n]\colon s_{\mathsf{L}i} w_i + s_{\mathsf{R}i} w_{n+i} + s_{\mathsf{O}i} w_{2n+i} + s_{\mathsf{M}i} w_i w_{n+i} + s_{\mathsf{C}i} - x_i = 0$,
(b) $\forall i \in [3n]\colon w_i = w_{\sigma(i)}$.

Since $Z_{\mathbb{H}}(X) \coloneqq \prod_{i\in[n]}(X - \omega^i)$, we can write (31) as

$$\forall i \in [n]\colon \mathsf{Gates}(\omega^i) = 0,$$

which is equivalent to condition (a) (cf. Eq. (21)). Moreover, according to [GWC19, Claim A.1], $P_\sigma(X, Y) \equiv 0$ is equivalent to condition (b). Since $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}_{\mathsf{Plonk},n}$ (due to line 7 in $\mathbf{G}_2$), we conclude that $P_\sigma(X, Y) \not\equiv 0$.

Next, we show that $P_\sigma(\beta, \gamma) = 0$. From (32), we get (cf. Eq. (22))

$$\forall i \in [n]\colon \Phi(\omega^i) f(\beta, \gamma, \omega^i) = \Phi(\omega^{i+1}) g(\beta, \gamma, \omega^i),$$

and from (33), we get that $\Phi(\omega) = 1$ (cf. Eq. (23)). Since $\omega^{n+1} = \omega$, we obtain the following n equalities:

$$\begin{aligned}
f(\beta, \gamma, \omega) &= g(\beta, \gamma, \omega)\Phi(\omega^2) \\
\Phi(\omega^2) f(\beta, \gamma, \omega^2) &= g(\beta, \gamma, \omega^2)\Phi(\omega^3) \\
&\vdots \\
\Phi(\omega^{n-1}) f(\beta, \gamma, \omega^{n-1}) &= g(\beta, \gamma, \omega^{n-1})\Phi(\omega^n) \\
\Phi(\omega^n) f(\beta, \gamma, \omega^n) &= g(\beta, \gamma, \omega^n)
\end{aligned}$$

Starting from $\prod_{i\in[n]} f(\beta,\gamma,\omega^i)$ and repeatedly substituting the relations above yields:

$$\begin{aligned}
\prod_{i\in[n]} f(\beta,\gamma,\omega^i) &= \underbrace{f(\beta,\gamma,\omega)}_{=g(\beta,\gamma,\omega)\Phi(\omega^2)} \cdot \prod_{i=2}^{n} f(\beta,\gamma,\omega^i) \\
&= g(\beta,\gamma,\omega)\cdot \underbrace{\Phi(\omega^2) f(\beta,\gamma,\omega^2)}_{=g(\beta,\gamma,\omega^2)\Phi(\omega^3)} \cdot \prod_{i=3}^{n} f(\beta,\gamma,\omega^i) \\
&\;\;\vdots \\
&= \prod_{i=1}^{n-1} g(\beta,\gamma,\omega^i)\cdot \underbrace{\Phi(\omega^n) f(\beta,\gamma,\omega^n)}_{=g(\beta,\gamma,\omega^n)} \\
&= \prod_{i\in[n]} g(\beta,\gamma,\omega^i)
\end{aligned}$$

This shows that $P_\sigma(\beta,\gamma) \coloneqq \prod_{i\in[n]} f(\beta,\gamma,\omega^i) - \prod_{i\in[n]} g(\beta,\gamma,\omega^i) = 0$, and so $\gamma \coloneqq H(\mathsf{trans}_2)$ is a root of the non-zero degree-$3n$ polynomial $P_\sigma(\beta, X)$.

$\underline{i=1.}$ In this case, we have $P_\sigma(\beta, X_\gamma) \equiv 0$, but also $P_\sigma(X_\beta, X_\gamma) \not\equiv 0$ (because otherwise $(\mathbb{i},\mathbb{x},\mathbb{w}) \in \mathcal{R}_{\mathsf{Plonk},n}$ as in the previous case). Thus, $\beta \coloneqq H(\mathsf{trans}_1)$ is a root of the non-zero polynomial $P_\sigma(X,Y) \in \mathbb{F}[X][Y]$ of degree $3n$ in X with coefficients in $\mathbb{F}[X]$.

Altogether, this shows that

$$\Pr[\mathbf{G}_1] \le \Pr[\mathbf{G}_2] + \mathsf{Adv}^{(n+2,D_{\mathsf{ks}})\text{-}\mathsf{mZT}}_{\mathcal{B}_2,(H,\mathsf{extKZG})}(\lambda). \tag{34}$$

$\mathbf{G}_2$. Since $\mathbf{G}_2$ never reaches the return statement in line 9 due to the abort ($\mathsf{SolveZT}$) in line 8, we have

$$\Pr[\mathbf{G}_2] = 0. \tag{35}$$

Finally, combining Eqs. (29), (30), (34) and (35) yields the bound in Theorem 5. □

4.3 Necessity of ZT for Knowledge Soundness of Plonk

We argue that our choice of D_{ks} for the zero-testing assumption is not merely an artifact of our knowledge soundness proof of Plonk. It is indeed necessary, in the sense that breaking ZT can be used to create a proof that passes verification but from which the knowledge extractor fails to extract a valid witness.

For example, consider a successful $(n+2, D_{\mathsf{ks}})$-mZT adversary $\mathcal{A}$ (Def. 12) with D_{ks} from Constr. 10 whose solution m^* is compatible (Def. 13) with trans_6, that is, it is of the form

$$m^* = \begin{pmatrix} S_{\mathsf{L}}, S_{\mathsf{R}}, S_{\mathsf{O}}, S_{\mathsf{M}}, S_{\mathsf{C}}, S_{\sigma,1}, S_{\sigma,2}, S_{\sigma,3}, \mathbb{x}, \\ A, B, C, \beta, \gamma, \Phi, \alpha, T_1, T_2, T_3, \\ \delta, a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega}, \varepsilon, Q_1, Q_2 \end{pmatrix} \in \begin{matrix} (\mathbb{F}^{n+3})^8 \times \mathbb{F}^\ell \\ \times (\mathbb{F}^{n+3})^3 \times \mathbb{F}^2 \times \mathbb{F}^{n+3} \times \mathbb{F} \times (\mathbb{F}^{n+3})^3 . \\ \times \mathbb{F}^8 \times (\mathbb{F}^{n+3})^2 \end{matrix}$$

To break ZT, m^* must satisfy

(a) $V(X,Y) \coloneqq D_{\mathsf{ks}}(m^*) \not\equiv 0$ (cf. Eq. (28)), but

(b) $V(\zeta, Y) \equiv 0$ for $\zeta \coloneqq H(\mathsf{extKZG.Com}_{\mathsf{srs}}(m^*))$.

We use $\mathcal{A}$ to build an (algebraic) adversary $\mathcal{B}$ against knowledge soundness (Def. 6) of Plonk for the canonical AGM extractor $\mathcal{E}_{\mathcal{B}}$ defined in Constr. 9:

1. Upon receiving $\mathsf{srs} \leftarrow \mathsf{Plonk.Setup}(1^\lambda, n)$, run $m^* \leftarrow \mathcal{A}(\mathsf{srs})$.
2. If m^* is a ZT solution of the form described above, set $\mathbf{s_L} := (S_{\mathsf{L}}(\omega^i))_{i\in[n]}$, $\mathbf{s_R} := (S_{\mathsf{R}}(\omega^i))_{i\in[n]}$, $\mathbf{s_O} := (S_{\mathsf{O}}(\omega^i))_{i\in[n]}$, $\mathbf{s_M} := (S_{\mathsf{M}}(\omega^i))_{i\in[n]}$, $\mathbf{s_C} := (S_{\mathsf{C}}(\omega^i))_{i\in[n]}$, and define $\sigma^*\colon [3n] \to \mathbb{H}'$ as
$$\sigma^*(i) := \begin{cases} S_{\sigma,1}(\omega^i) & \text{if } i \in [n], \\ S_{\sigma,2}(\omega^i) & \text{if } i \in (n, 2n], \\ S_{\sigma,3}(\omega^i) & \text{if } i \in (2n, 3n]. \end{cases}$$
Set $\mathbb{i} := ((\mathbf{s_L}, \mathbf{s_R}, \mathbf{s_O}, \mathbf{s_M}, \mathbf{s_C}), \sigma^*)$ and
$$\pi := \mathsf{extKZG}_{\mathsf{srs}}(A, B, C, \Phi, T_1, T_2, T_3, Q_1, Q_2, a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega}) \in \mathbb{G}_1^9 \times \mathbb{F}^6.$$
3. Return $(\mathbb{i}, \mathbb{x}, \pi)$ as well as the AGM representations $(A, B, C, \Phi, T_1, T_2, T_3, Q_1, Q_2)$ of the 9 group elements in π.

Consider the witness $\mathbb{w}$ extracted by $\mathcal{E}_{\mathcal{B}}$ from $\mathcal{B}$ (cf. Constr. 9). Observe that this is the same value $\mathbb{w}$ which D_{ks} computes from m^* when checking condition 4 of compatibility. Since m^* is compatible with trans_6, we have

(c) $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}_{\mathsf{Plonk}}$.

From (b) above, we have $V(\zeta, \tau) = 0$, which is equivalent to $\mathsf{Plonk.Verify}(\mathsf{vp}, \mathbb{x}, \pi) = 1$ (as argued after Eq. (28)). Together with (c), this means the adversary $\mathcal{B}$ successfully breaks knowledge soundness of Plonk for the canonical AGM extractor $\mathcal{E}_{\mathcal{B}}$.

Similar adversaries can be constructed if the ZT adversary $\mathcal{A}$ returns a solution m^* that is compatible with any other trans_i for $i \in [5]$, confirming the necessity of our assumption beyond the case of trans_6.

Furthermore, if the mZT solution m^* returned by $\mathcal{A}$ corresponds to an unsatisfiable circuit, i.e., there is no witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{Plonk}}$, then the constructed adversary $\mathcal{B}$ unconditionally breaks (knowledge) soundness of Plonk even in the *standard* model.

4.4 Zero Knowledge of Plonk in the Standard Model

After establishing knowledge soundness of Plonk without random oracles, we turn to zero knowledge and show that Plonk is computationally zero-knowledge in the standard model, without relying on any idealized models such as the AGM.

Sefranek [Sef24] points out and fixes an issue with the zero-knowledge property of an earlier version of Plonk and shows that its (subsequently patched) current version is *statistically* zero-knowledge in the ROM. We show how to adapt his proof avoiding random oracles, instead making the zero-testing assumption on the underlying hash function H. The trade-off is that this leads to the weaker notion of computational, rather than statistical, zero knowledge.

Theorem 6 (Computational ZK of Plonk from ZT). Plonk *(Constr. 7) is computationally zero-knowledge if the used hash function H has the zero-testing property for the function $D_{\mathsf{zk}}\colon \mathcal{M} \to \mathbb{F}^{(\leq 3n)}[X]$ defined in Constr. 15. Specifically, there exists a PPT simulator $\mathcal{S}$ such that for every adversary $\mathcal{A}$, there exist adversaries $\mathcal{B}_1, \mathcal{B}_2$, each running in time roughly the same as $\mathcal{A}$, such that*
$$\mathsf{Adv}^{\mathsf{czk},n}_{\mathcal{A},\mathcal{S},\mathsf{Plonk}}(\lambda) \leq \mathsf{Adv}^{(n+2)\text{-DL}}_{\mathcal{B}_1,\mathsf{GGen}}(\lambda) + \mathsf{Adv}^{(n+2,D_{\mathsf{zk}})\text{-ZT}}_{\mathcal{B}_2,(H,\mathsf{extKZG})}(\lambda).$$

We give a proof outline here, deferring the full proof to Appendix A.2.

Proof sketch. Sefranek [Sef24] defines a PPT simulator for Plonk that perfectly simulates proofs conditioned on none of the following "bad" events happening:

$$\begin{aligned} &(1)\ \ \tau \in \mathbb{H}, && (2)\ \ \delta = \tau \text{ or } \delta\omega = \tau, \\ &(3)\ \ \delta \in \mathbb{H}, && (4)\ \ \exists j \in [n] \colon g(\omega^j) = 0. \end{aligned}$$

We use the same simulator and bound the probability of the above events. Events (1) and (2) allow to recover the SRS trapdoor τ, breaking the $(n+2)$-DL assumption. Events (3) and (4) can be used to break the zero-testing assumption on (H, extKZG): event (3) is equivalent to $\delta \coloneqq H(\mathsf{trans}_4)$ being a root of the vanishing polynomial $Z_{\mathbb{H}}(X) = \prod_{a \in \mathbb{H}}(X - a)$, while event (4) implies that $\gamma \coloneqq H(\mathsf{trans}_2)$ is a root of a degree-$3n$ polynomial defined by the product of $g(\omega^j)$ over all $j \in [n]$ (cf. the abort in step 3 of Plonk.Prove). In the full proof we then use a sequence of game hops together with suitable reductions to establish the bound in the theorem.

Acknowledgments

The authors would like to thank Ariel Gabizon for preliminary discussions. This work was funded by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG18002] and by the Austrian Science Fund (FWF) [10.55776/F8515-N].

References

BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Berlin, Heidelberg, May 2005. doi:10.1007/11426639_26.

BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, January 2012. doi:10.1145/2090236.2090263.

BCMS20. Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Recursive proof composition from accumulation schemes. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 1–18. Springer, Cham, November 2020. doi:10.1007/978-3-030-64378-2_1.

BK04. Mihir Bellare and Tadayoshi Kohno. Hash function balance and its impact on birthday attacks. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 401–418. Springer, Berlin, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_24.

BMRS20. Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. Mina: Decentralized cryptocurrency at scale, 2020. URL: https://minaprotocol.com/wp-content/uploads/technicalWhitepaper.pdf.

BN06. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006. doi:10.1145/1180405.1180453.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. doi:10.1145/168588.168596.

BR06. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Berlin, Heidelberg, May / June 2006. doi:10.1007/11761679_25.

CGH98. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998. doi:10.1145/276698.276741.

CX23. Michele Ciampi and Yu Xia. Multi-theorem fiat-shamir transform from correlation-intractable hash functions. In Mehdi Tibouchi and Xiaofeng Wang, editors, *ACNS 2023, Part II*, volume 13906 of *LNCS*, pages 555–581. Springer, Cham, June 2023. doi:10.1007/978-3-031-33491-7_21.

DMWG23. Quang Dao, Jim Miller, Opal Wright, and Paul Grubbs. Weak fiat-shamir attacks on modern proof systems. In *2023 IEEE Symposium on Security and Privacy*, pages 199–216. IEEE Computer Society Press, May 2023. doi:10.1109/SP46215.2023.10179408.

EGS+24. Liam Eagen, Ariel Gabizon, Marek Sefranek, Patrick Towa, and Zachary J. Williamson. Stackproofs: Private proofs of stack and contract execution using protogalaxy. Cryptology ePrint Archive, Report 2024/1281, 2024. URL: https://eprint.iacr.org/2024/1281.

FFR24. Antonio Faonio, Dario Fiore, and Luigi Russo. Real-world universal zkSNARKs are non-malleable. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3138–3151. ACM Press, October 2024. doi:10.1145/3658644.3690351.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. doi: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 Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987. doi: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 Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Berlin, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_37.

GKR15. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27:1–27:64, 2015. doi:10.1145/2699436.

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985. doi:10.1145/22145.22178.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Berlin, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_19.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Berlin, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_11.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. doi:10.1145/1993636.1993651.

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. URL: https://eprint.iacr.org/2019/953.

KRS25. Dmitry Khovratovich, Ron D. Rothblum, and Lev Soukhanov. How to prove false statements: Practical attacks on fiat-shamir. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 3–26. Springer, Cham, August 2025. doi:10.1007/978-3-032-01887-8_1.

KST22. Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 359–388. Springer, Cham, August 2022. doi:10.1007/978-3-031-15985-5_13.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_11.

Lip25. Helger Lipmaa. Plonk is simulation extractable in ROM under falsifiable assumptions. Cryptology ePrint Archive, Report 2025/1759, 2025. URL: https://eprint.iacr.org/2025/1759.

LPS25. Helger Lipmaa, Roberto Parisella, and Janno Siim. On knowledge-soundness of plonk in ROM from falsifiable assumptions. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VII*, volume 16006 of *LNCS*, pages 362–395. Springer, Cham, August 2025. doi:10.1007/978-3-032-01907-3_12.

LS24. Hyeonbum Lee and Jae Hong Seo. On the security of nova recursive proof system. Cryptology ePrint Archive, Report 2024/232, 2024. URL: https://eprint.iacr.org/2024/232.

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

PS96. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Berlin, Heidelberg, May 1996. doi:10.1007/3-540-68339-9_33.

Sef24. Marek Sefranek. How (not) to simulate PLONK. In Clemente Galdi and Duong Hieu Phan, editors, *SCN 24, Part I*, volume 14973 of *LNCS*, pages 96–117. Springer, Cham, September 2024. doi:10.1007/978-3-031-71070-4_5.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18.

TSH22. Louis Tremblay Thibault, Tom Sarry, and Abdelhakim Senhaji Hafid. Blockchain scaling using rollups: A comprehensive survey. *IEEE Access*, 10:93039–93054, 2022. doi:10.1109/ACCESS.2022.3200051.

Val08. Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, March 2008. doi:10.1007/978-3-540-78524-8_1.

Zha22. Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Cham, August 2022. doi:10.1007/978-3-031-15982-4_3.

A Deferred Proofs

A.1 Proof of d-General Forking Lemma

We give a proof of our variant of the general forking lemma (Lemma 1) that follows the proof of Lemma 3.1 in [BN06] with minor modifications.

Proof (of Lemma 1). We first establish that

$$\mathsf{fork}(\mathcal{F}_{\mathcal{A}}) \geq \mathsf{acc}(\mathcal{A}) \cdot \left(\frac{\mathsf{acc}(\mathcal{A})}{q} - \frac{d}{|R|} \right), \tag{36}$$

and then show that it implies Eq. (7). For any input x let

$$\begin{aligned} \mathsf{acc}_x(\mathcal{A}) &:= \Pr\left[y_1, \ldots, y_q \leftarrow R;\ (I, \mathsf{out}) \leftarrow \mathcal{A}(x, y_1, \ldots, y_q) : I \neq \bot \right], \\ \mathsf{fork}_x(\mathcal{F}_{\mathcal{A}}) &:= \Pr\left[\mathsf{sol} \leftarrow \mathcal{F}_{\mathcal{A}}(x) : \mathsf{sol} \neq \bot \right]. \end{aligned}$$

We claim that for all x,

$$\mathsf{fork}_x(\mathcal{F}_{\mathcal{A}}) \geq \mathsf{acc}_x(\mathcal{A}) \cdot \left(\frac{\mathsf{acc}_x(\mathcal{A})}{q} - \frac{d}{|R|} \right). \tag{37}$$

Then, with expectation taken over $x \leftarrow \mathsf{IG}$, we have

$$\begin{aligned} \mathsf{fork}(\mathcal{F}_{\mathcal{A}}) &= \mathbb{E}\left[\mathsf{fork}_x(\mathcal{F}_{\mathcal{A}}) \right] \\ &\geq \mathbb{E}\left[\mathsf{acc}_x(\mathcal{A}) \cdot \left(\frac{\mathsf{acc}_x(\mathcal{A})}{q} - \frac{d}{|R|} \right) \right] = \frac{\mathbb{E}[\mathsf{acc}_x(\mathcal{A})^2]}{q} - \frac{d \cdot \mathbb{E}[\mathsf{acc}_x(\mathcal{A})]}{|R|} \\ &\geq \frac{\mathbb{E}[\mathsf{acc}_x(\mathcal{A})]^2}{q} - \frac{d \cdot \mathbb{E}[\mathsf{acc}_x(\mathcal{A})]}{|R|} = \mathsf{acc}(\mathcal{A}) \cdot \left(\frac{\mathsf{acc}(\mathcal{A})}{q} - \frac{d}{|R|} \right). \end{aligned}$$

This establishes Eq. (36). Above we used Eq. (37) and the fact that $\mathbb{E}[X^2] \geq \mathbb{E}[X]^2$ for any real-valued random variable X.

We proceed to the proof of Eq. (37). For any input x, with probabilities taken over the randomness used by $\mathcal{F}_{\mathcal{A}}$ (see Constr. 3), we have

$$\begin{aligned}
\mathsf{fork}_x(\mathcal{F}_{\mathcal{A}}) &= \Pr[I = I' \wedge I \neq \bot \wedge y'_I \notin S(\mathsf{out})] \\
&\geq \Pr[I = I' \wedge I \neq \bot] - \Pr[I \neq \bot \wedge y'_I \in S(\mathsf{out})] \\
&\geq \Pr[I = I' \wedge I \neq \bot] - \Pr[I \neq \bot] \cdot \frac{d}{|R|} \\
&= \Pr[I = I' \wedge I \neq \bot] - \mathsf{acc}_x(\mathcal{A}) \cdot \frac{d}{|R|},
\end{aligned}$$

where we used $\Pr[I \neq \bot \wedge y'_I \in S(\mathsf{out})] = \Pr[I \neq \bot] \cdot \Pr[y'_I \in S(\mathsf{out})]$ since y'_I is sampled independently of the first run of $\mathcal{A}$, and $\Pr[y'_I \in S(\mathsf{out})] \leq d/|R|$.

It remains to show that $\Pr[I = I' \wedge I \neq \bot] \geq \mathsf{acc}_x(\mathcal{A})^2/q$. Let Ω denote the randomness space of $\mathcal{A}$. For each $i \in [q]$, let $X_i \colon \Omega \times R^{i-1} \to [0,1]$ be defined as

$$X_i(\rho, y_1, \ldots, y_{i-1}) \coloneqq \Pr\big[y_i, \ldots, y_q \leftarrow R;\ (J, \mathsf{out}) \coloneqq \mathcal{A}(x, y_1, \ldots, y_q; \rho) : J = i\big]$$

for all $\rho \in \Omega$ and $y_1, \ldots, y_{i-1} \in R$. Regard X_i as a random variable over the uniform distribution on its domain. Then

$$\begin{aligned}
\Pr[I = I' \wedge I \neq \bot] &= \sum_{i=1}^{q} \Pr[I = i \wedge I' = i] = \sum_{i=1}^{q} \Pr[I = i] \cdot \Pr[I' = i \mid I = i] \\
&= \sum_{i=1}^{q} \sum_{\rho, y_1, \ldots, y_{i-1}} \frac{X_i(\rho, y_1, \ldots, y_{i-1})}{|\Omega| \cdot |R|^{i-1}} \cdot X_i(\rho, y_1, \ldots, y_{i-1}) \\
&= \sum_{i=1}^{q} \sum_{\rho, y_1, \ldots, y_{i-1}} \frac{X_i(\rho, y_1, \ldots, y_{i-1})^2}{|\Omega| \cdot |R|^{i-1}} \\
&= \sum_{i=1}^{q} \mathbb{E}[X_i^2] \geq \sum_{i=1}^{q} \mathbb{E}[X_i]^2.
\end{aligned}$$

Next, we will apply Lemma 3.3 of [BN06] which states that, for $q \geq 1$ real numbers $x_1, \ldots, x_q$, it holds that $\sum_{i=1}^{q} x_i^2 \geq \frac{1}{q} \cdot \left(\sum_{i=1}^{q} x_i\right)^2$ (a special case of Jensen's inequality). We get

$$\sum_{i=1}^{q} \mathbb{E}[X_i]^2 \geq \frac{1}{q} \cdot \left(\sum_{i=1}^{q} \mathbb{E}[X_i]\right)^2 = \frac{1}{q} \cdot \mathsf{acc}_x(\mathcal{A})^2.$$

This completes the proof of Eq. (37) and thus of Eq. (36).

We now show how to obtain Eq. (7). Using Eq. (36), we have

$$\left(\mathsf{acc}(\mathcal{A}) - \frac{qd}{2|R|}\right)^2 = \mathsf{acc}(\mathcal{A})^2 - \frac{qd}{|R|} \cdot \mathsf{acc}(\mathcal{A}) + \frac{q^2 d^2}{4|R|^2} \leq q \cdot \mathsf{fork}(\mathcal{F}_{\mathcal{A}}) + \frac{q^2 d^2}{4|R|^2}.$$

Taking the square root of both sides, and using the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for any real numbers $a, b \geq 0$, we get

$$\mathsf{acc}(\mathcal{A}) - \frac{qd}{2|R|} \leq \sqrt{q \cdot \mathsf{fork}(\mathcal{F}_{\mathcal{A}})} + \sqrt{\frac{q^2 d^2}{4|R|^2}} = \sqrt{q \cdot \mathsf{fork}(\mathcal{F}_{\mathcal{A}})} + \frac{qd}{2|R|}.$$

Rearranging terms yields Eq. (7). □

A.2 Proof of Computational ZK

To prove that Plonk is computationally zero-knowledge in the standard model (Theorem 6), we rely on the following lemma shown in [Sef24, Lemma 1], which explains why the prover's witness polynomials are randomized by adding the product of a random blinding polynomial and $Z_{\mathbb{H}}$. In particular, it shows that giving out a limited number of evaluations of a polynomial randomized in this way reveals no information about the original polynomial.

Lemma 2 ([Sef24]). *Let $S \subset \mathbb{F}$ and $Z_S(X) \coloneqq \prod_{a \in S}(X - a)$. Fix a polynomial $f \in \mathbb{F}[X]$ and any distinct values $x_1, \ldots, x_k \in \mathbb{F} \setminus S$. Then the following distribution is uniform in $\mathbb{F}^k$:*

1. *Choose a random polynomial $\rho \leftarrow \mathbb{F}^{(\leq k-1)}[X]$ of degree $k-1$ and define*
$$\tilde{f}(X) \coloneqq f(X) + \rho(X) Z_S(X).$$
2. *Output $\big(\tilde{f}(x_1), \ldots, \tilde{f}(x_k)\big) \in \mathbb{F}^k$.*

Furthermore, we employ the simulator defined by Sefranek for proving statistical zero knowledge of Plonk in the ROM [Sef24], given as Constr. 13 below.

Construction 13: Simulator for Plonk (from [Sef24]).

$\underline{\mathcal{S}_0(1^\lambda, n)}$:

1. Run $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$ and select $H\colon \{0,1\}^* \to \mathbb{F}$ as in Plonk.Setup (Constr. 7).
2. Sample $\tau \leftarrow \mathbb{F}$. If $\tau \in \mathbb{H}$, abort.
3. Return $\mathsf{srs} \coloneqq \big(\mathcal{G}, [\tau]_1, \big[\tau^2\big]_1, \ldots, \big[\tau^{n+2}\big]_1, [\tau]_2\big)$ and τ.

$\underline{\mathcal{S}_1\big(\tau, \mathsf{pp}, (x_i)_{i \in [\ell]}\big)}$:

1. Parse pp as $(\mathsf{srs}, S_{\mathsf{L}}, S_{\mathsf{R}}, S_{\mathsf{O}}, S_{\mathsf{M}}, S_{\mathsf{C}}, S_{\sigma,1}, S_{\sigma,2}, S_{\sigma,3})$. Sample $a_\tau, b_\tau, c_\tau \leftarrow \mathbb{F}$ and compute the commitments
$$\mathsf{cm}_A \coloneqq [a_\tau]_1, \ \mathsf{cm}_B \coloneqq [b_\tau]_1, \ \mathsf{cm}_C \coloneqq [c_\tau]_1.$$
2. Compute the permutation challenges $\beta, \gamma \in \mathbb{F}$ as in step 2 of Plonk.Prove.
3. Sample $\phi_\tau \leftarrow \mathbb{F}$ and compute the commitment $\mathsf{cm}_\Phi \coloneqq [\phi_\tau]_1$.
4. Compute the quotient challenge $\alpha \in \mathbb{F}$ as in step 4 of Plonk.Prove.
5. Sample $\phi_{\tau\omega} \leftarrow \mathbb{F}$ (if $\tau = 0$, set $\phi_{\tau\omega} \coloneqq \phi_\tau$ instead) and compute the evaluation of the quotient polynomial T at τ as
$$t_\tau \coloneqq \frac{1}{Z_{\mathbb{H}}(\tau)} \left(\begin{array}{c} S_{\mathsf{L}}(\tau)a_\tau + S_{\mathsf{R}}(\tau)b_\tau + S_{\mathsf{O}}(\tau)c_\tau + S_{\mathsf{M}}(\tau)a_\tau b_\tau + S_{\mathsf{C}}(\tau) + S_{\mathsf{PI}}(\tau) \\ +\alpha\Big(\begin{array}{c}\phi_\tau(a_\tau+\beta\tau+\gamma)(b_\tau+\beta k_1\tau+\gamma)(c_\tau+\beta k_2\tau+\gamma) \\ -\phi_{\tau\omega}(a_\tau+\beta S_{\sigma,1}(\tau)+\gamma)(b_\tau+\beta S_{\sigma,2}(\tau)+\gamma)(c_\tau+\beta S_{\sigma,3}(\tau)+\gamma)\end{array}\Big) \\ +\alpha^2\big(\mathcal{L}_1(\tau)(\phi_\tau - 1)\big) \end{array} \right).$$
Sample $t_2, t_3 \leftarrow \mathbb{F}$ and set $t_1 \coloneqq t_\tau - \tau^{n+2} \cdot t_2 - \tau^{2n+4} \cdot t_3$. Compute the commitments
$$\mathsf{cm}_{T_1} \coloneqq [t_1]_1, \ \mathsf{cm}_{T_2} \coloneqq [t_2]_1, \ \mathsf{cm}_{T_3} \coloneqq [t_3]_1.$$
6. Compute the evaluation challenge $\delta \in \mathbb{F}$ as in step 6 of Plonk.Prove; if $\delta = \tau$ or $\delta\omega = \tau$, abort.
7. If $\delta \in \mathbb{H}$, abort. Otherwise, sample $a_\delta, b_\delta, c_\delta, \phi_{\delta\omega} \leftarrow \mathbb{F}$ and compute the evaluations $s_{\sigma_1,\delta} \coloneqq S_{\sigma,1}(\delta)$, $s_{\sigma_2,\delta} \coloneqq S_{\sigma,2}(\delta)$.
8. Compute the opening challenge $\varepsilon \in \mathbb{F}$ as in step 8 of Plonk.Prove.

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

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

Then, compute the opening proofs as

$$\mathsf{cm}_{Q_1} := \left[\frac{1}{\tau-\delta}\begin{pmatrix} l_\tau+\varepsilon(a_\tau-a_\delta)+\varepsilon^2(b_\tau-b_\delta)+\varepsilon^3(c_\tau-c_\delta) \\ +\varepsilon^4(S_{\sigma,1}(\tau)-S_{\sigma,1}(\delta))+\varepsilon^5(S_{\sigma,2}(\tau)-S_{\sigma,2}(\delta)) \end{pmatrix}\right]_1, \quad \mathsf{cm}_{Q_2} := \left[\frac{\phi_\tau-\phi_{\delta\omega}}{\tau-\delta\omega}\right]_1.$$

Output the simulated proof

$$\tilde{\pi} := \begin{pmatrix} \mathsf{cm}_A, \mathsf{cm}_B, \mathsf{cm}_C, \mathsf{cm}_\Phi, \mathsf{cm}_{T_1}, \mathsf{cm}_{T_2}, \mathsf{cm}_{T_3}, \mathsf{cm}_{Q_1}, \mathsf{cm}_{Q_2}, \\ a_\delta, b_\delta, c_\delta, s_{\sigma_1,\delta}, s_{\sigma_2,\delta}, \phi_{\delta\omega} \end{pmatrix} \in \mathbb{G}_1^9 \times \mathbb{F}^6.$$

Proof (of Theorem 6). We consider the following sequence of games $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$, where $\mathbf{G}_0$ corresponds to the interaction of the adversary $\mathcal{A}$ with the real Plonk protocol (i.e., Plonk.Setup and Plonk.Prove) in the zero-knowledge experiment (cf. Def. 8), and $\mathbf{G}_2$ corresponds to its interaction with the simulator $\mathcal{S}$ given in Constr. 13.

$\mathbf{G}_1$ modifies $\mathbf{G}_0$ by introducing two aborts that the simulator can make: first, in Plonk.Setup if an SRS trapdoor $\tau \in \mathbb{H}$ is sampled (equivalent to running $\mathcal{S}_0$); and second, in Plonk.Prove if the evaluation challenge δ satisfies $\delta = \tau$ or $\delta\omega = \tau$ (corresponding to step 6 in $\mathcal{S}_1$). The adversary $\mathcal{B}_1$ against $(n+2)$-DL given in Constr. 14 allows us to bound this difference as

$$\big|\Pr[\mathbf{G}_0] - \Pr[\mathbf{G}_1]\big| \leq \mathsf{Adv}^{(n+2)\text{-DL}}_{\mathcal{B}_1,\mathsf{GGen}}(\lambda). \tag{38}$$

Construction 14: Reduction to $(n+2)$-DL.

$\mathcal{B}_1(\mathcal{G}, [\tau]_1, \ldots, [\tau^{n+2}]_1, [\tau]_2)$:

1. If $[\tau^n]_1 = [1]_1$ (i.e., $\tau \in \mathbb{H}$), return $\omega^i \in \mathbb{H}$ such that $[\omega^i]_1 = [\tau]_1$.
2. $\mathsf{srs} := (\mathcal{G}, [\tau]_1, \ldots, [\tau^{n+2}]_1, [\tau]_2)$.
3. $(\mathbb{i}, \mathbb{x}, \mathbb{w}, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{srs})$.
4. $(\mathsf{pp}, \mathsf{vp}) := \mathsf{Plonk.Index}(\mathsf{srs}, \mathbb{i})$.
5. $\pi \leftarrow \mathsf{Plonk.Prove}(\mathsf{pp}, \mathbb{x}, \mathbb{w})$.
6. Compute $\delta := H(\mathsf{trans}_4)$ (as in Constr. 7).
7. If $[\delta]_1 = [\tau]_1$, return δ. If $[\delta\omega]_1 = [\tau]_1$, return $\delta\omega$. Else, abort.

$\mathbf{G}_2$ corresponds to $\mathcal{A}$'s interaction with the simulator $\mathcal{S}$. Following the core argument in the proof by Sefranek [Sef24, Theorem 1], which leverages Lemma 2, the only differences between $\mathbf{G}_1$ and $\mathbf{G}_2$ are the following two aborts:

(a) In $\mathbf{G}_1$, Plonk.Prove aborts in step 3 if $g(\omega^j) = 0$ for some $j \in [n]$, i.e., $\gamma := H(\mathsf{trans}_2)$ is a root of the non-zero degree-$3n$ polynomial

$$G(X) := \prod_{i\in[n]} \big(w_i+\beta S_{\sigma,1}(\omega^i)+X\big)\big(w_{n+i}+\beta S_{\sigma,2}(\omega^i)+X\big)\big(w_{2n+i}+\beta S_{\sigma,3}(\omega^i)+X\big). \tag{39}$$

(b) In $\mathbf{G}_2$, $\mathcal{S}_1$ aborts in step 7 if $\delta \in \mathbb{H}$, i.e., $\delta := H(\mathsf{trans}_4)$ is a root of the non-zero degree-n polynomial

$$Z_{\mathbb{H}}(X) := \prod_{i\in[n]} (X-\omega^i) = X^n - 1. \tag{40}$$

Either of these aborts can be used to break the zero-testing property of (H, extKZG) for the following function $D_{\mathsf{zk}} \colon \mathcal{M} \to \mathbb{F}^{(\leq 3n)}[X]$. (From an $m = (S_{\mathsf{L}}, S_{\mathsf{R}}, S_{\mathsf{O}}, S_{\mathsf{M}}, S_{\mathsf{C}}, S_{\sigma,1}, S_{\sigma,2}, S_{\sigma,3}, \mathbb{x}, A, B, C, \beta)$ compatible with trans_2, D_{zk} derives the values w_i used to define $G(X)$ in Eq. (39) in the obvious way, as in Eq. (17).)

Construction 15: The function $D_{\mathsf{zk}} \colon \mathcal{M} \to \mathbb{F}^{(\leq 3n)}[X]$ for Plonk.

$\underline{D_{\mathsf{zk}}(m)}$:

1. If m is compatible (Def. 13) with $\begin{cases} \mathsf{trans}_2, & \text{return } G(X) \text{ from (39)}, \\ \mathsf{trans}_4, & \text{return } Z_{\mathbb{H}}(X) \text{ from (40)}. \end{cases}$
2. Else, return 0.

Consider the adversary $\mathcal{B}_2$ against $(n+2, D_{\mathsf{zk}})$-ZT given in Constr. 16. Since $\mathbf{G}_1$ and $\mathbf{G}_2$ are equivalent up to the event "abort (a) or abort (b)" occurring, in which case $\mathcal{B}_2$ is successful in breaking ZT, we get

$$\left|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_2]\right| \leq \mathsf{Adv}^{(n+2,D_{\mathsf{zk}})\text{-}\mathsf{ZT}}_{\mathcal{B}_2,(H,\mathsf{extKZG})}(\lambda). \tag{41}$$

Construction 16: Reduction to $(n+2, D_{\mathsf{zk}})$-ZT.

$\underline{\mathcal{B}_2(\mathcal{G}, [\tau]_1, \ldots, [\tau^{n+2}]_1, [\tau]_2)}$:

1. If $[\tau^n]_1 = [1]_1$ (i.e., $\tau \in \mathbb{H}$), abort.
2. $\mathsf{srs} := (\mathcal{G}, [\tau]_1, \ldots, [\tau^{n+2}]_1, [\tau]_2)$.
3. $(\mathbb{i}, \mathbb{x}, \mathbb{w}, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{srs})$.
4. $(\mathsf{pp}, \mathsf{vp}) := \mathsf{Plonk.Index}(\mathsf{srs}, \mathbb{i})$.
5. $\pi \leftarrow \mathsf{Plonk.Prove}(\mathsf{pp}, \mathbb{x}, \mathbb{w})$.
6. Compute $\gamma := H(\mathsf{trans}_2)$ and $\delta := H(\mathsf{trans}_4)$ (as in Constr. 7).
7. If $[\delta]_1 = [\tau]_1$ or $[\delta\omega]_1 = [\tau]_1$, abort.
8. If $G(\gamma) = 0$, return m such that $\mathsf{extKZG.Com}_{\mathsf{srs}}(m) = \mathsf{trans}_2$.
9. If $Z_{\mathbb{H}}(\delta) = 0$, return m such that $\mathsf{extKZG.Com}_{\mathsf{srs}}(m) = \mathsf{trans}_4$.
10. Abort.

Finally, combining Eqs. (38) and (41) yields the bound in Theorem 6. □