# The Art of Linearization:
## From a KZG's Trick to a General Commitment Framework

Janno Siim[1][0000−0001−5824−7215]

University of Tartu, Tartu, Estonia
jannosiim@gmail.com

**Abstract.** A useful linearization technique (or a trick) was introduced in the Marlin and Plonk SNARKs, which significantly reduces the number of KZG polynomial commitment openings a SNARK prover has to send. Subsequently, many other KZG-based protocols have taken advantage of it. We revisit and formalize this technique:

- We define a Linearization Polynomial Commitment Scheme (LPCS) that abstracts their linearization technique.
- We formalize LinKZG, a LPCS version of the KZG commitment scheme, and show that it achieves a weak form of extractability under a target group version of the ARSDH assumption. We show that Plonk is secure under the same assumption in the ROM.
- We show how to construct LPCSs from any homomorphic polynomial commitment scheme. Thus, enabling the linearization technique also for those polynomial commitment schemes, and potentially improving the efficiency of many other SNARKs.

**Keywords:** Polynomial commitment scheme · linearization trick · KZG · Plonk · SNARK

## 1 Introduction

Zero-knowledge proofs have found widespread real-world adoption in the last decade. Of particular importance are (zero-knowledge) succinct non-interactive arguments of knowledge (SNARKs) for general $\mathsf{NP}$ relations, which allows us to prove the correctness of arbitrary computations with a short proof and fast verification. Analysing the security of SNARKs and improving their efficiency is an ongoing challenge.

A convenient three-step paradigm has become popular in recent years for designing SNARKs. First, one designs an interactive public-coin information-theoretically secure proof, such as the polynomial interactive oracle proof (PIOP) [BFS20, CHM+20]. In this model, the prover sends polynomial oracles, the verifier responds with random challenges to the prover, and the verifier can query the oracles. Typically, these proofs are non-succinct. Second, a polynomial commitment scheme (PCS) [KZG10] is used to compress the PIOP transcript. PCS allows to succinctly commit to a polynomial $f$ and later open $f(\alpha)$ at a point $\alpha$ and show with a short opening proof that the evaluation is correct. PCSs can be

based on various cryptographic assumptions [KZG10, BBB+18, BBHR18, NS24]. Thirdly, we can use the Fiat–Shamir heuristic [FS87] to make the proof non-interactive.

The security of the resulting SNARK relies on the security of the Fiat-Shamir heuristic (which can be proven secure in the random oracle model (ROM) [AFK23]) and on some form of extractability of the PCS. Currently, the SNARKs with the smallest communication complexity [CHM+20, GWC19, CFF+21, LSZ22] can be achieved by using the KZG polynomial commitments [KZG10] based on bilinear pairings. Earlier works [CHM+20] showed that the KZG commitment is extractable in the algebraic group model (AGM) [FKL18], which is a sufficient property for proving the security of the SNARK in the above framework. However, the use of AGM is not entirely satisfactory, as there are known contrived protocols that are secure in AGM but insecure in the real world [Zha22].

A recent work by Lipmaa, Parisella, and Siim [LPS24] showed that the KZG commitment is extractable with rewinding (more precisely, they prove computational special soundness) under a falsifiable assumption[1] called the Adaptive Rational Strong Diffie-Hellman (ARSDH) assumption. ARSDH can be seen as a generalization of the well-known Strong Diffie-Hellman (SDH) assumption [BB04]. Thus, their work and its extension to multivariate KZG [BDH+25] have put KZG-based SNARKs on a much firmer theoretical foundation.

However, typically KZG-based SNARKs slightly divert from the above three-step paradigm and employ various optimizations to reduce the proof size and verification time. Of particular importance is the linearization technique introduced in the Marlin [CHM+20] and Plonk [GWC19] SNARKs[2]. Consider the common scenario in SNARKs, where the prover has committed to two vectors of polynomials $\boldsymbol{g}$ and $\boldsymbol{f}$, and the goal is to verify that $\mathcal{V}(X) = \sum_{i=1}^{m} g_i(X)f_i(X) = 0$. The naive way to test this is for the verifier to send a random point $\alpha$ and the prover will send $2m$ evaluations $\bar{g}_i = g_i(\alpha)$ and $\bar{f}_i = f_i(\alpha)$ and additionally an opening proof for each of them. The verifier checks that the opening proofs are valid and that $\mathcal{V}(\alpha) = \sum_{i=1}^{m} \bar{g}_i\bar{f}_i = 0$. The probability that $\mathcal{V}$ is a non-zero polynomial, but $\mathcal{V}(\alpha) = 0$ is negligible.

The works of [CHM+20, GWC19] observe that one can significantly reduce the number of evaluations and evaluation proofs that must be sent with KZG. The prover can send $\bar{g}_i$ as before, but as the KZG is homomorphic, it is possible to compute the commitment $C$ to the polynomial $\mathcal{V}'(X) = \sum_{i=1}^{m} \bar{g}_i f_i(X)$. Then, it is sufficient to show that $C$ opens to 0 at the point $\alpha$. This results in $m$ field elements (the points $\bar{g}_i$) and $m + 1$ evaluation proofs. On top of this optimization, one can batch all the evaluation proofs and reduce the communication to $m$ field elements and 1 evaluation proof. This linearization technique (sometimes also called the linearization trick or Maller's trick) has found widespread adoption in KZG-based protocols, including Plonk [GWC19], Lunar [CFF+21], Vampire [LSZ22], Mercury [EG25], Samaritan [GPS25], and many more.

[1] Recall that an assumption is falsifiable if it can be expressed as a game between an efficient challenger and an efficient adversary.

[2] Plonk [GWC19] credits specifically Mary Maller for this technique.

However, the security of this technique has come under question in some recent works [LPS23, FFR24, LPS25]. Essentially, all of them observe that, in some cases, the polynomials $f_i$ are not extractable when using the linearization technique. Faonio, Fiore, and Russo [FFR24] show that the linearization technique preserves extractability in the AGM with Oblivious Sampling (AGMOS) [LPS23] (AGM where the adversary is able to sample group elements without knowing their discrete logarithms) when $\{g_i\}_i$ is a linearly independent set of polynomials. To prove simulation-extractability of Plonk [GWC19] and Marlin [CHM+20] in the AGM + ROM, they require an even stronger form of linear independence of $g_i$-polynomials.

On the other hand, Lipmaa, Parisella, and Siim [LPS25] study the security of the linearization technique and Plonk in the ROM under falsifiable assumptions. They propose a sanitized version of the linearization technique, where (in addition to the protocol above) the verifier sends an additional random challenge $\gamma$ and the prover opens $v = \sum_{i=1}^{m} \gamma^{i-1} f_i(\alpha)$. The verifier checks that the respective homomorphically computed commitment opens to $v$ at point $\alpha$. This they show is sufficient to prove the linearization technique's extractability under the ARSDH assumption. However, it adds one more field element to the proof size and $\mathcal{O}(m)$ exponentiations to the verification time. Although it is a relatively small overhead when adapted to, say, Plonk, it is still not the original version of Plonk used in practice. In fact, [LPS25] also proves the knowledge soundness of the original Plonk in the ROM under ARSDH and a new SplitRSDH assumption. The recent work by Lipmaa [Lip25] extends this result (in a modular way) to simulation-extractability [Sah99, DDO+01] in the ROM under the same assumptions. Simulation-extractability is a stronger security notion, which implies non-malleability and is often desired in practice. Both ARSDH and SplitRSDH are falsifiable assumptions, but SplitRSDH is more complicated than ARSDH and also more tailored to the precise scenario of Plonk. This motivates our first research question:

> $\mathcal{Q}_1$ : *Can we prove knowledge-soundness of Plonk (in the ROM) under only the ARSDH assumption?*

There are two main reasons why understanding the security of Plonk is of particular importance. There are two main reasons why understanding the security of Plonk is of particular importance. First, Plonk is frequently deployed in practice, particularly as a core component in zkEVM constructions, where the security of the system directly impacts the protection of enormous financial assets. Second, there is significant ongoing effort to formally verify the Plonk verifier [Rou25]. Thus, having a firm theoretical foundation for Plonk is of utmost importance.

However, we also want to go beyond the Plonk and understand the linearization technique more generally. In light of the above attacks to the extractability of KZG's linearization trick [LPS23, FFR24, LPS25], we ask:

> $\mathcal{Q}_2$ : *Is there a useful security property that we can prove about the KZG's linearization technique under falsifiable assumptions?*

The fact that [LPS25] proved the security of (interactive) Plonk under falsifiable assumptions indicates that the linearization technique must be, at least in some sense, secure under falsifiable assumptions.

Finally, we note that the KZG commitment has a few significant limitations: it does not have a linear-time prover, it requires a trusted setup, and it is not post-quantum secure. This motivates our last question:

> $\mathcal{Q}_3$ : *Can the linearization technique apply to other polynomial commitment schemes?*

A positive answer would significantly broaden the applicability of the linearization technique and improve efficiency of much wider class of SNARKs.

**Our Contribution.** We initiate a systematic study of the linearization technique used in KZG-based protocols. We define a *Linearization Polynomial Commitment Scheme* (LPCS) that abstracts the linearization technique. A linearization PCS allows to commit to a vector of low-degree polynomials $\boldsymbol{f} = (f_1, \ldots, f_m)$ and later open $\mathcal{V}(\alpha)$ at a point $\alpha$, where $\mathcal{V}(X) = \sum_{i=1}^{m} g_i(X) f_i(X)$ and $\boldsymbol{g} = (g_1, \ldots, g_m)$ are some (non-committed) low-degree polynomials. The verifier gets, as an input, the commitment $C$ to $\boldsymbol{f}$, honestly computed $\boldsymbol{g}(\alpha) = (g_1(\alpha), \ldots, g_m(\alpha))$, and the claimed evaluation $y$, which in the honest case is $y = \mathcal{V}(\alpha)$. The committer initiates (either a non-interactive or interactive) protocol with the verifier to prove that $y = \mathcal{V}(\alpha)$. We assume that $\boldsymbol{g}(\alpha)$ is honestly computed as it helps to focus on the core idea of the linearization technique. It would be simple to extend the definition to the case where $\boldsymbol{g}$ are also committed with a PCS and the committer sends $\boldsymbol{g}(\alpha)$ together with opening proofs.

We define two extractability notions for LPCS, *computational special soundness (CSS)* and *weak computational special soundness (wCSS)*. In both definitions, the adversary outputs a transcript tree for the same commitment $C$, but with $K$ different challenges $\alpha_i$, claimed evaluations $y_i$ (in the honest case $y_i = \mathcal{V}(\alpha_i)$), and respective evaluation proof trees. The difference is that in CSS, the extractor has to extract each polynomial $f_i$ consistent with the commitment, and in the weak version, the extractor has to extract only the linearized (low-degree) polynomial $\mathcal{V}$ such that $\mathcal{V}(\alpha_i) = y_i$ for all $i \in [1..K]$. Weak CSS is inspired by [CGKY25], which also argues that extracting each $f_i$ precisely consistent with each commitment is unnecessary for many applications.

Next, we cast the KZG PCS in this new framework, which gives rise to the LinKZG LPCS. We define a target group version of the ARSDH assumption. This assumption is identical to the original ARSDH assumption in [LPS24], except that one of the group elements that the adversary outputs is in the group $\mathbb{G}_T$ instead of $\mathbb{G}_1$, where pairing is defined as a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. It is easy to show that the target group ARSDH assumption implies the original ARSDH assumption. We show that LinKZG has weak CSS under the target group ARSDH assumption. Thus, we obtain the answer to the question $\mathcal{Q}_2$, the KZG's linearization technique is secure in the sense of weak CSS under a falsifiable assumption, despite the negative results of [LPS23, FFR24, LPS25].

In a special case when $m = 1$ and $g_1(X)$ is a non-zero polynomial, we show that LinKZG has (full) CSS under the $\mathbb{G}_1$ version of the ARSDH assumption. However, as noted earlier, one cannot hope to prove full CSS in the general case due to the impossibility results mentioned before.

The weak CSS of LinKZG is sufficient to prove the knowledge soundness of Plonk in the ROM under the target group ARSDH assumption. Thus, we obtain a partial positive answer to $\mathcal{Q}_1$. Plonk is secure under the ARSDH assumption, but one has to consider the target group version. In a recent work, Lipmaa [Lip25] shows that Plonk is simulation-extractable in the ROM under falsifiable assumptions. Since his proof is modular (it relies on the knowledge-soundness of Plonk as a black box), we can also easily obtain that Plonk is simulation-extractable in the ROM under the target group ARSDH assumption.

Finally, we focus on the linearization technique in other polynomial commitment schemes (question $\mathcal{Q}_3$). It is, in fact, straightforward to extend any homomorphic polynomial commitment scheme into a LPCS. Many known polynomial commitment schemes (besides KZG) are homomorphic, including Bulletproof's PCS [BBB$^+$18], Hyrax's PCS [WTs$^+$18], and Dory's PCS [Lee21], which use Pedersen-style commitments. Although the construction of LPCS is simple, proving that it has (even weak) CSS is non-trivial. We isolate a property that we call linearization friendliness, and we use it to prove the CSS of the LPCS. We show that if the homomorphic PCS is binding, linearization-friendly, and its evaluation protocol has CSS, then the resulting LPCS has CSS. It then remains to show that known homomorphic PCSs are linearization-friendly. We manage to show it for two important cases: (1) We show that if $\boldsymbol{g}$ are linearly independent polynomials (as was also required in [FFR24]), then linearization friendliness holds without any further assumptions, and thus the resulting LPCS has CSS; (2) We show that any Pedersen-style PCS has linearization friendliness in the algebraic group model, and thus, the corresponding LPCS also has CSS in the algebraic group model.

We hope that by enabling the linearization technique for other polynomial commitment schemes, our work will help to improve the efficiency of many other (besides KZG-based) SNARK constructions that are based on these polynomial commitment schemes.

## 2 Preliminaries

We denote the security parameter by $\lambda$. PPT stands for probabilistic polynomial time and DPT for deterministic polynomial time. We write $f(\lambda) \approx_\lambda 0$ if $f$ is a negligible function in $\lambda$. For integers $a \leq b$, we write $[a..b]$ to denote the set of integers $\{a, a+1, \ldots, b\}$. For a finite set $A$, $a \leftarrow\!\!\$\ A$ denotes that $a$ is chosen uniformly at random from $A$. Let $\mathbb{F}$ be a finite field of prime order $p$, $\mathbb{F}[X]$ be the set of univariate polynomials over $\mathbb{F}$, and $\mathbb{F}^{\leq n}[X]$ be the set of univariate polynomials over $\mathbb{F}$ of degree at most $n$. By $\mathbb{N}$ we denote the set of natural numbers (without 0). We use boldface letters such as $\boldsymbol{a}$ to denote vectors and capital boldface letters such as $\boldsymbol{A}$ to denote matrices. The $i$th row vector of a

matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{A}_i$. Given a vector of polynomials $\boldsymbol{f} \in (\mathbb{F}[X])^m$ and a point $\alpha \in \mathbb{F}$, we sometimes write $\boldsymbol{f}(\alpha) := (f_1(\alpha), \ldots, f_m(\alpha))$.

**Bilinear groups.** Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be additive cyclic groups of prime order $p$ with generators $[1]_1, [1]_2, [1]_T$, respectively. We write $[a]_1$ for $a \in \mathbb{F}$ to denote the group element $a \cdot [1]_1$, and similarly for $[a]_2$ and $[a]_T$. This notion extends to vectors, e.g., $[a_1, \ldots, a_n]_1 := ([a_1]_1, \ldots, [a_n]_1)$ for $a_1, \ldots, a_n \in \mathbb{F}$. Let $\bullet : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map such that for all $a, b \in \mathbb{F}$, $[a]_1 \bullet [b]_2 = [ab]_T$. We assume that the bilinear map is non-degenerate, i.e., $[1]_1 \bullet [1]_2 = [1]_T$, and efficiently computable. In definitions, we use $\mathsf{Pgen}$ to denote a PPT algorithm that on input $1^\lambda$ outputs public parameters $\mathsf{p} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [1]_1, [1]_2, \bullet)$.

**Interpolation.** Given a vector of points $(x_i, y_i)_{i=1}^K \in (\mathbb{F}^2)^K$, we denote by $L(X) := \mathsf{Inter}((x_i, y_i)_{i=1}^K)$ the interpolation polynomial in $\mathbb{F}[X]$ over these points. Namely, $L(X)$ is the unique polynomial of degree at most $K-1$ such that $L(x_i) = y_i$ for all $i \in [1..K]$. For a set $S = \{x_1, \ldots, x_K\} \subset \mathbb{F}$, we denote by

$$\mathsf{Z}_S(X) := \textstyle\prod_{s \in S}(X - s)$$

the vanishing polynomial over the set $S$ and by $\ell_i^S(X) \in \mathbb{F}^{\leq K-1}[X]$ the $i$th Lagrange polynomial over the set $S$ for $i \in [1..K]$. Namely, $\ell_i^S(X)$ is the unique polynomial of degree at most $K-1$ such that $\ell_i^S(x_j) = 1$ if $i = j$ and 0 if $i \neq j$. Furthermore, a Lagrange polynomial can be expressed as follows:

$$\ell_i^S(X) = \textstyle\prod_{j \in [1..K] \setminus \{i\}} \frac{X - x_j}{x_i - x_j} = d_i \frac{\mathsf{Z}_S(X)}{(X - x_i)} \ , \tag{1}$$

where $d_i = 1/\prod_{j \in [1..K] \setminus \{i\}}(x_i - x_j)$. Lagrange polynomials allow us to express the interpolation polynomial as follows:

$$L(X) = \mathsf{Inter}((x_i, y_i)_{i=1}^K) = \textstyle\sum_{i=1}^K y_i \ell_i^S(X) \ .$$

**Alternant matrix.** For polynomials $\boldsymbol{g} = (g_1, \ldots, g_m) \in (\mathbb{F}[X])^m$ and a set of points $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K) \in \mathbb{F}^K$, we define their alternant matrix to be

$$\boldsymbol{A}_{\boldsymbol{g},\boldsymbol{\alpha}} = \begin{pmatrix} g_1(\alpha_1) & g_2(\alpha_1) & \cdots & g_m(\alpha_1) \\ g_1(\alpha_2) & g_2(\alpha_2) & \cdots & g_m(\alpha_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(\alpha_K) & g_2(\alpha_K) & \cdots & g_m(\alpha_K) \end{pmatrix} \in \mathbb{F}^{K \times n} \ .$$

This is a generalization of a Vandermonde matrix, which is obtained when $g_i(X) = X^{i-1}$ for all $i \in [1..m]$.

**Lemma 1.** *Let $m, K, d \in \mathbb{N}$ such that $K \geq d + 1 > m$. If $\boldsymbol{g} \in (\mathbb{F}[X])^m$ are linearly independent polynomials of degree at most $d$, and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K) \in \mathbb{F}^K$ are pairwise distinct points, then the alternant matrix $\boldsymbol{A}_{\boldsymbol{g},\boldsymbol{\alpha}}$ has rank $m$.*

*Proof.* Suppose for the sake of contradiction that $\boldsymbol{A}_{\boldsymbol{g},\boldsymbol{\alpha}}$ has rank less than $m$. Then there exists a non-zero vector $\boldsymbol{c} = (c_1, \ldots, c_m) \in \mathbb{F}^m$ such that $\boldsymbol{A}_{\boldsymbol{g},\boldsymbol{\alpha}} \cdot \boldsymbol{c}^\top = \boldsymbol{0}$. Define a polynomial $G(X) = \sum_{i=1}^{m} c_i g_i(X)$. Then for all $j \in [1..K]$, we have $G(\alpha_j) = 0$. Since $G(X)$ is a non-zero polynomial of degree at most $d$, it can have at most $d$ roots. However, we have $K \geq d+1$ roots $\alpha_1, \ldots, \alpha_K$, which is a contradiction. Hence, the alternant matrix $\boldsymbol{A}_{\boldsymbol{g},\boldsymbol{\alpha}}$ must have rank $m$. □

### 2.1 Assumptions

We state a few pairing-based assumptions that will be relevant in our work. Recall, $\mathsf{Pgen}(1^\lambda)$ outputs public parameters $\mathsf{p} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [1]_1, [1]_2, \bullet)$ of a pairing.

**Definition 1.** *The* $n$-PDL (Power Discrete Logarithm) *assumption holds for* $\mathsf{Pgen}$ *in* $\mathbb{G}_1$ *if for any PPT* $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{pdl}}_{\mathsf{Pgen},n,\mathbb{G}_1,\mathcal{A}}(\lambda) :=$

$$\Pr\left[\, x = \mathcal{A}(\mathsf{ck}) \,\middle|\, \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); x \leftarrow\!\$\, \mathbb{F}; \mathsf{ck} \leftarrow ([(x^i)_{i=0}^n]_1, [1, x]_2) \,\right] \approx_\lambda 0 \ .$$

**Definition 2 ([BB04]).** *The* $n$-SDH (Strong Diffie-Hellman) *assumption holds for* $\mathsf{Pgen}$ *in* $\mathbb{G}_1$ *if for any PPT* $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{sdh}}_{\mathsf{Pgen},n,\mathbb{G}_1,\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l} c \in \mathbb{F} \wedge x - c \neq 0 \\ [\varphi]_1 = [1/(x-c)]_1 \end{array} \middle| \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); x \leftarrow\!\$\, \mathbb{F}; \\ \mathsf{ck} \leftarrow ([(x^i)_{i=0}^n]_1, [1, x]_2); (c, [\varphi]_1) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array} \right] \approx_\lambda 0 \ .$$

There is also a well-known target group SDH (TSDH) assumption [PHGR13], where the adversary outputs $[\varphi]_T$ instead of $[\varphi]_1$.

**Definition 3 ([LPS24]).** *The* $n$-ARSDH (Adaptive Rational Strong Diffie-Hellman) *assumption holds for* $\mathsf{Pgen}$ *in* $\mathbb{G}_1$ *if for any PPT* $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{arsdh}}_{\mathsf{Pgen},n,\mathbb{G}_1,\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l} S \subset \mathbb{F} \wedge |S| = n+1 \wedge \\ [g]_1 \neq [0]_1 \wedge [g]_1 = [\varphi]_1 \mathsf{Z}_S(x) \end{array} \middle| \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); x \leftarrow\!\$\, \mathbb{F}; \\ \mathsf{ck} \leftarrow ([(x^i)_{i=0}^n]_1, [1, x]_2); \\ (S, [g, \varphi]_1) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array} \right] \approx_\lambda 0 \ ,$$

*where* $\mathsf{Z}_S(X) := \prod_{s \in S}(X - s)$.

A version of the ARSDH assumption (RSDH), where the set $S$ is fixed was originally introduced in [GR19]. [LPS24] shows that the $n$-ARSDH assumption in $\mathbb{G}_1$ implies the $n$-SDH assumption in $\mathbb{G}_1$. ARSDH can be seen as a generalization of the SDH assumption, where $\varphi = {}^1\!/\!_{(x-c)}$ is replaced by $\varphi = {}^g\!/\!_{\mathsf{Z}_S(x)} = \prod_{c \in S} {}^g\!/\!_{(x-c)}$.

We also state the SplitRSDH assumption that, in addition to the ARSDH assumption, was used in [LPS25] to prove the security of Plonk. We will not use this assumption in our work, but it is useful to compare it to the target group ARSDH assumption that we will introduce in Section 4.2.

**Definition 4 (par-SplitRSDH).** *Let* $\mathsf{par} = (m, n_\psi, n_1, n_s, (\psi_k(X))_{k=1}^m)$ *be such that* $m, n_\psi, n_1, n_s \in \mathsf{poly}(\lambda)$ *are positive integers,* $n_s > n_1 + n_\psi + 1$*, and* $\psi_k(X) \in \mathbb{F}^{\leq n_\psi}[X]$*. For any PPT* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathrm{splitrsdh}}_{\mathsf{Pgen},\mathsf{par},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l|l} S \subset \mathbb{F} \wedge |S| = n_s \wedge L(X) \in \mathbb{F}[X] \wedge & x \leftarrow\!\$\, \mathbb{F}; \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \\ n_1 + n_\psi + 1 \leq \deg L(X) \leq n_s - 1 \wedge & \mathsf{ck} \leftarrow ([(x^i)_{i=0}^{n_1}]_1, [1, x]_2); \\ \sum_{k=1}^m [\tau_k \psi_k(x)]_1 = [L(x)]_1 + [\varphi \mathsf{Z}_S(x)]_1 & \begin{pmatrix} S, L(X), \\ [(\tau_k)_{k=1}^m, \varphi]_1 \end{pmatrix} \leftarrow \mathcal{A}(\mathsf{p}, \mathsf{ck}) \end{array}\right] \approx_\lambda 0.$$

**Algebraic Group Model.** Algebraic Group Model (AGM) [FKL18] is an idealized security model used in the discrete logarithm setting. In brief, we say that a PPT adversary $\mathcal{A}$ is algebraic if given some group element inputs $[x_1, \ldots, x_n]_1$, whenever $\mathcal{A}$ outputs a group element $[y]_1$, it must also provide a vector $\boldsymbol{f} \in \mathbb{F}^n$ such that $[y]_1 = \sum_{i=1}^n f_i [x_i]_1$. Intuitively, it means that the adversary can only output group elements that are linear combinations of the input group elements. For simplicity, we stick to the informal definition above. There are several approaches to formalize [Zha22, JM24] or generalize AGM [LPS23]. Like with other common idealized security models, there are some contrived protocols, which are secure in the AGM but insecure in the real world [Zha22]. Nevertheless, AGM is a useful tool to provide an initial security analysis of a cryptographic protocol or an assumption.

### 2.2 Special Sound Argument Systems

Let $\mathcal{R}$ be a binary relation. An interactive (public coin) argument system $\mathsf{ARG} = (\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ for a relation $\mathcal{R}$ consists of three PPT algorithms. The key generation algorithm $\mathsf{KGen}(1^\lambda)$ outputs a common reference string $\mathsf{crs}$ which will be given as an input to the prover and the verifier. The prover $\mathsf{P}$ and the verifier $\mathsf{V}$ are a pair of interactive PPT algorithms. The prover gets as an input a statement $x$ and a witness $w$ such that $(x, w) \in \mathcal{R}$, and the verifier gets as an input only $x$. In each round, the prover sends a message to the verifier, who responds with a uniformly randomly and independently chosen challenge. After a fixed number of rounds, the verifier outputs 1 (accept) or 0 (reject). We denote this interaction by $b \leftarrow \langle \mathsf{P}(w), \mathsf{V} \rangle(\mathsf{crs}, x)$, where $b \in \{0, 1\}$.

Two of the most standard notions that are essentially always required from an argument system are completeness and computational soundness.

We say that an argument system $\mathsf{ARG}$ is complete if for any $(x, w) \in \mathcal{R}$,

$$\Pr\left[\, \langle \mathsf{P}(w), \mathsf{V} \rangle(\mathsf{crs}, x) = 1 \,\middle|\, \mathsf{crs} \leftarrow \mathsf{KGen}(1^\lambda) \,\right] = 1 \ .$$

We say that an argument system $\mathsf{ARG}$ is computationally sound if for any pair of PPT adversaries $\mathcal{A}$, $\mathsf{P}^*$

$$\Pr\left[\begin{array}{l|l} \langle \mathsf{P}^*(st), \mathsf{V} \rangle(\mathsf{crs}, x) = 1 \wedge & \\ \forall w : (x, w) \notin \mathcal{R} & \mathsf{crs} \leftarrow \mathsf{KGen}(1^\lambda); (x, st) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] \approx_\lambda 0 \ .$$

We also require a stronger property called computational special soundness, which is a form of extractability. First, let us define a transcript tree.

**Definition 5 (Transcript Tree).** *Let* $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_\mu) \in \mathbb{N}^\mu$. *A* $\boldsymbol{\kappa}$-tree of transcripts $\mathcal{T}$ *for a* $(2\mu+1)$*-move public-coin interactive argument* $\mathsf{ARG} = (\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ *is a set of* $K = \prod_{i=1}^{\mu} \kappa_i$ *transcripts organized as a tree structure. The nodes in this tree store the prover's messages and the edges store the verifier's challenges. Every node at depth* $i$ *has* $\kappa_i$ *children corresponding to* $\kappa_i$ *pairwise distinct challenges. A path from a root node to a leaf node corresponds to an accepting transcript of* ARG. *We say that* $\mathcal{T}$ *is accepting if the argument's verifier accepts every such path in the tree.*

**Definition 6.** *Let* $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_\mu)$. *A* $(2\mu+1)$*-move public-coin interactive argument* $\mathsf{ARG} = (\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ *for a relation* $\mathcal{R}$ *has* $\boldsymbol{\kappa}$-computational special soundness *(CSS) if there exists a DPT extractor* Ext*, such that for any PPT* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen},\mathsf{ARG},\mathsf{Ext},\boldsymbol{\kappa},\mathcal{A}}(\lambda) :=$$

$$\Pr\left[\begin{array}{l|l} \mathcal{T} \text{ is an accepting } \boldsymbol{\kappa}\text{-tree of} & \mathsf{crs} \leftarrow \mathsf{KGen}(1^\lambda); \\ \mathsf{ARG} \wedge (x,w) \notin \mathcal{R} & (x, \mathcal{T}) \leftarrow \mathcal{A}(\mathsf{crs}); w \leftarrow \mathsf{Ext}(\mathsf{crs}, x, \mathcal{T}) \end{array}\right] \approx_\lambda 0 \ .$$

Attema, Fehr, and Klooß [AFK22] show that special soundness implies knowledge soundness of the non-interactive argument when applying the Fiat-Shamir heuristic.

### 2.3 Polynomial Commitments

**Non-interactive PCS.** A non-interactive polynomial commitment scheme (PCS) [KZG10] PC consists of five PPT algorithms:

$\mathsf{Pgen}(1^\lambda)$**:** outputs public parameters $\mathsf{p}$. We assume, by default, that all other algorithms get $\mathsf{p}$ as an implicit input.

$\mathsf{KGen}(n)$**:** outputs a commitment key $\mathsf{ck}$ for polynomials of degree at most $n$.

$\mathsf{Com}(\mathsf{ck}, f)$**:** takes as input $\mathsf{ck}$ and a polynomial $f \in \mathbb{F}^{\leq n}[X]$ and outputs a commitment $C$ to $f$.

$\mathsf{Open}(\mathsf{ck}, f, C, \alpha)$**:** takes as input $\mathsf{ck}$, a commitment $C$, and the respective polynomial $f$, and a point $\alpha \in \mathbb{F}$. It computes $y \leftarrow f(\alpha)$ and a proof $\pi$ and outputs $(y, \pi)$.

$\mathsf{Vf}(\mathsf{ck}, C, \alpha, y, \pi)$**:** [3] takes as input a commitment key $\mathsf{ck}$, a commitment $C$, a point $\alpha \in \mathbb{F}$, a proof $\pi$, and a claimed evaluation $y$. It outputs 1 (accept) or 0 (reject).

We say that a PCS PC is correct if for any $n \in \mathsf{poly}(\lambda)$, any polynomial $f \in \mathbb{F}^{\leq n}[X]$, and any point $\alpha \in \mathbb{F}$,

$$\Pr\left[\mathsf{Vf}(\mathsf{ck}, C, \alpha, y, \pi) = 1 \,\middle|\, \begin{array}{l} \mathsf{ck} \leftarrow \mathsf{KGen}(n); C \leftarrow \mathsf{Com}(\mathsf{ck}, f); \\ (y, \pi) \leftarrow \mathsf{Open}(\mathsf{ck}, f, C, \alpha) \end{array}\right] = 1 \ .$$

**Definition 7.** *We say that a PCS PC is evaluation binding [KZG10] if for any* $n \in \mathsf{poly}(\lambda)$*, and any PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathrm{eb}}_{PC,n,\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l} y \neq y' \wedge \mathsf{Vf}(\mathsf{ck}, C, \alpha, y, \pi) = 1 \wedge \\ \mathsf{Vf}(\mathsf{ck}, C, \alpha, y', \pi') = 1 \end{array} \,\middle|\, \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{KGen}(n); \\ (C, \alpha, y, \pi, y', \pi') \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \approx_\lambda 0 \ .$$

[3] V denotes the verifier in an argument system and Vf denotes the verifier in a PCS.

Lipmaa, Parisella, and Siim [LPS24] define computational special soundess as the main extractability property for polynomial commitment schemes. In brief, the adversary outputs a commitment $C$ and $n+1$ accepting evaluation proofs for $n+1$ distinct points $\alpha_i$ with claimed evaluations $y_i$. The adversary wins if the extractor fails to recover a polynomial $f$ such that $\mathsf{Com}(\mathsf{ck}, f) = C$ and $f(\alpha_i) = y_i$ for all $i \in [1..n+1]$.

**Definition 8.** *We say that a PCS PC is $(n+1)$-computationally special sound (CSS) [LPS24] if there exists a DPT extractor* $\mathsf{Ext}$ *such that for any PPT adversary $\mathcal{A}$,* $\mathsf{Adv}^{\mathrm{css}}_{PC,n,\mathsf{Ext},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l}\forall i \neq j : \alpha_i \neq \alpha_j \wedge \\ \forall i : \mathsf{Vf}(\mathsf{ck}, C, \alpha_i, y_i, \pi_i) = 1 \wedge \\ (\mathsf{Com}(\mathsf{ck}, f) \neq C \vee f \notin \mathbb{F}^{\leq n}[X] \\ \vee \exists i : f(\alpha_i) \neq y_i)\end{array} \middle| \begin{array}{l}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{KGen}(n); \\ \mathcal{T} = (C, (\alpha_i, y_i, \pi_i)_{i=1}^{n+1}) \leftarrow \mathcal{A}(\mathsf{ck}); \\ f \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathcal{T});\end{array}\right] \approx_\lambda 0.$$

**Interactive PCS.** An interactive polynomial commitment scheme (IPCS) $\mathsf{IPC}$ consists of the same algorithms $\mathsf{Pgen}, \mathsf{KGen}, \mathsf{Com}$. However, instead of $\mathsf{Open}$ and $\mathsf{Vf}$, it has an interactive evaluation protocol $(\mathsf{P}_E, \mathsf{Vf}_E)$. We require that a tuple $(\mathsf{KGen}, \mathsf{P}_E, \mathsf{V}_E)$ is a public-coin argument system for the following relation:

$$\mathcal{R}^E_{\mathsf{ck},n} = \{((C, \alpha, y), f) : f \in \mathbb{F}^{\leq n}[X] \wedge C = \mathsf{Com}(\mathsf{ck}, f) \wedge y = f(\alpha)\} \ .$$

Here, the prover gets as an input $\mathsf{ck}$, $x = (C, \alpha, y)$ and $w = f$, and the verifier gets as an input only $\mathsf{ck}$ and $x$. We denote the verifier's decision by $b \leftarrow \langle \mathsf{P}_E(w), \mathsf{V}_E\rangle(\mathsf{ck}, x)$, which outputs 1 (accept) or 0 (reject).

We say that an IPCS $\mathsf{IPC}$ is correct if for any $n \in \mathsf{poly}(\lambda)$, any polynomial $f \in \mathbb{F}^{\leq n}[X]$, and any point $\alpha \in \mathbb{F}$,

$$\Pr\left[\langle \mathsf{P}_E(w), \mathsf{V}_E\rangle(\mathsf{ck}, x) = 1 \middle| \begin{array}{l}\mathsf{ck} \leftarrow \mathsf{KGen}(n); C \leftarrow \mathsf{Com}(\mathsf{ck}, f); \\ y \leftarrow f(\alpha); x \leftarrow (C, \alpha, y); w \leftarrow f;\end{array}\right] = 1 \ .$$

**Definition 9.** *Let $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_\mu)$, and let $\mathsf{IPC} = (\mathsf{Pgen}, \mathsf{KGen}, \mathsf{Com}, \mathsf{P}_E, \mathsf{V}_E)$ be an IPCS such that $(\mathsf{KGen}, \mathsf{P}_E, \mathsf{V}_E)$ is a $(2\mu+1)$-move public-coin argument system for $\mathcal{R}^E_{\mathsf{ck},n}$. We say that an IPCS $\mathsf{IPC}$ has $\boldsymbol{\kappa}$-computational special soundness if $(\mathsf{KGen}, \mathsf{P}_E, \mathsf{V}_E)$ for $\mathcal{R}^E_{\mathsf{ck},n}$ has computational $\boldsymbol{\kappa}$-special soundness as in Definition 6.*

**KZG PCS.** We present the well-known KZG [KZG10] polynomial commitment scheme . Let $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$ be the pairing description from before. For $n \in \mathsf{poly}(\lambda)$, $\mathsf{KGen}(n)$ outputs a commitment key $\mathsf{ck} \leftarrow ([(x^i)_{i=0}^n]_1, [1, x]_2)$ for a random $x \leftarrow\!\!\$\ \mathbb{F}$. The commitment algorithm $\mathsf{Com}(\mathsf{ck}, f)$ for a polynomial $f(X) = \sum_{i=0}^n f_i X^i \in \mathbb{F}^{\leq n}[X]$ outputs a commitment $[c]_1 = [f(x)]_1 = \sum_{i=0}^n f_i [x^i]_1$. The opening algorithm $\mathsf{Open}(\mathsf{ck}, f, [c]_1, \alpha)$ for a point $\alpha \in \mathbb{F}$ computes $y = f(\alpha)$ and $h(X) = (f(X) - y)/(X - \alpha)$, and outputs $(y, [\pi]_1 \leftarrow [h(x)]_1)$. The verification algorithm $\mathsf{Vf}(\mathsf{ck}, [c]_1, \alpha, y, [\pi]_1)$ accepts if $([c]_1 - y[1]_1) \bullet [1]_2 = [\pi]_1 \bullet ([x]_2 - \alpha[1]_2)$.

The KZG PCS is correct, evaluation binding under the $n$-SDH assumption [KZG10], and $(n+1)$-computationally special sound under the $n$-ARSDH assumption [LPS24]. We also recall the KZG's batching lemma from [LPS24] that we will use extensively.

**Lemma 2 (Batching lemma [LPS24]).** *Let $S = \{\alpha_1, \ldots, \alpha_K\} \subseteq \mathbb{F}$ be a set of size $K$. Let $c, x \in \mathbb{F}$. If for all $j \in [1..K]$, $(c - y_j) = \pi_j \cdot (x - \alpha_j)$, for some $\pi_j, y_j \in \mathbb{F}$, then $c - L(x) = \pi \mathsf{Z}_S(x)$, where $L(X) = \mathsf{Inter}((\alpha_j, y_j)_{j=1}^K)$, $\mathsf{Z}_S(X)$ is the vanishing polynomial over $S$, and $\pi = \sum_{j=1}^K d_j \pi_j$, where $d_j$ is defined as in Eq. (1).*

## 3 Linearization PCS

We define a new type of polynomial commitment scheme that we call a *linearization polynomial commitment scheme (LPCS)*. This will formalize and abstract the linearization technique [GWC19, CHM+20] as an independent building block. In LPCS, the prover commits to a vector of polynomials $\boldsymbol{f} = (f_1, \ldots, f_m)$ of degree $\leq n$ and additionally fixes a public vector of polynomials $\boldsymbol{g} = (g_1, \ldots, g_m)$ of degree $\leq d$. The prover defines a polynomial $L(X) = \sum_{i=1}^m g_i(X) f_i(X)$ and sends $y \leftarrow L(\alpha)$ to the verifier, where $\alpha$ chosen by the verifier. The verifier gets as an input an honestly computed $\boldsymbol{g}(\alpha) = (g_1(\alpha), \ldots, g_m(\alpha))$ and possibly a maliciously computed $y$. The prover and the verifier run (either non-interactive or interactive) evaluation protocol to convince the verifier that $y = L(\alpha)$. We discuss later why it makes sense to assume that $\boldsymbol{g}(\alpha)$ is honestly computed.

We give two definitions of computational special soundness (CSS) for LPCSs. The (standard) CSS for LPCS is essentially a generalization of the definition for PCS from Definition 8, where the extractor must extract the entire vector of polynomials $\boldsymbol{f}$. However, in weak CSS, we only require the extractor to extract a polynomial $L^*(X)$ of degree at most $n + d$ such that $y_i = L^*(\alpha_i)$ for arbitrary (polynomial) number of evaluation points $\alpha_i$. Weak CSS follows the approach in [CGKY25], where they also do not require that the extracted value is consistent with the commitment. Such definition turns out to be useful for LinKZG that we present in Section 4.

We decided to define a non-interactive and interactive LPCSs separately because they have a slightly different syntax. Technically, a non-interactive LPCS is a special case of an interactive LPCS where the evaluation protocol contains a single message.

**Non-interactive LPCS.** Let $m, n, d \in \mathsf{poly}(\lambda)$ be some natural numbers. Let $\mathbb{F}$ be a finite field. A *non-interactive linearization polynomial commitment scheme (LPCS)* $\mathsf{LPC}$ for polynomials in $\mathbb{F}[X]$ consists of the following PPT algorithms:

$\mathsf{Pgen}(1^\lambda)$**:** outputs public parameters $\mathsf{p}$. We assume by default that all other algorithms get $\mathsf{p}$ as an implicit input.

$\mathsf{KGen}(n)$**:** outputs a commitment key $\mathsf{ck}$ for polynomials of degree at most $n$.

$\mathsf{Com}(\mathsf{ck}, \boldsymbol{f})$**:** takes as an input a vector of polynomials $\boldsymbol{f} = (f_1, \ldots, f_m)$, where $f_i \in \mathbb{F}^{\leq n}[X]$ for all $i \in [1..m]$ and outputs a commitment $C$ to $\boldsymbol{f}$.

$\mathsf{Open}(\mathsf{ck}, C, \boldsymbol{f}, \boldsymbol{g}, \alpha)$**:** besides $\mathsf{ck}$, $C$, $\boldsymbol{f}$, defined earlier, it takes as an input a vector of polynomials $\boldsymbol{g} = (g_1, \ldots, g_m)$, where $g_i \in \mathbb{F}^{\leq d}[X]$ for all $i \in [1..m]$, and a point $\alpha \in \mathbb{F}$. It computes $y \leftarrow \sum_{i=1}^{m} g_i(\alpha) \cdot f_i(\alpha)$ and a proof $\pi$ and outputs $(y, \pi)$.

$\mathsf{Vf}(\mathsf{ck}, C, \boldsymbol{g}(\alpha), \alpha, y, \pi)$**:** takes as an input a commitment key $\mathsf{ck}$, a commitment $C$, a vector $\boldsymbol{g}(\alpha)$, where $\boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m$, a point $\alpha \in \mathbb{F}$, a proof $\pi$, and a claimed evaluation $y$. It outputs 1 (accept) or 0 (reject).

The first two notions we require are quite standard.

**Definition 10.** *Let $m, n, d \in \mathsf{poly}(\lambda)$. We say that a non-interactive LPCS LPC is correct if for any $\boldsymbol{f} \in (\mathbb{F}^{\leq n}[X])^m$, $\boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m$, and any point $\alpha \in \mathbb{F}$,*

$$\Pr\left[\mathsf{Vf}(\mathsf{ck}, C, \boldsymbol{g}(\alpha), \alpha, y, \pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{KGen}(n); \\ C \leftarrow \mathsf{Com}(\mathsf{ck}, \boldsymbol{f}); \\ (y, \pi) \leftarrow \mathsf{Open}(\mathsf{ck}, \boldsymbol{f}, \boldsymbol{g}, C, \alpha) \end{array}\right] = 1 \ .$$

**Definition 11.** *Let $m, n, d \in \mathsf{poly}(\lambda)$. We say that a non-interactive LPCS LPC is binding if for any PPT $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l} \boldsymbol{f}_1, \boldsymbol{f}_2 \in (\mathbb{F}^{\leq n}[X])^m \wedge \\ \mathsf{Com}(\mathsf{ck}, \boldsymbol{f}_1) = \mathsf{Com}(\mathsf{ck}, \boldsymbol{f}_2) \wedge \\ \boldsymbol{f}_1 \neq \boldsymbol{f}_2 \end{array} \;\middle|\; \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{KGen}(n); \\ (\boldsymbol{f}_1, \boldsymbol{f}_2) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \approx_\lambda 0 \ .$$

**Definition 12.** *Let $m, n, d \in \mathsf{poly}(\lambda)$. We say that a non-interactive LPCS LPC is evaluation binding if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{eb}}_{\mathit{LPC},n,d,m,\mathcal{A}}(\lambda) :=$*

$$\Pr\left[\begin{array}{l} y_1 \neq y_2 \wedge \boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m \wedge \\ \mathsf{Vf}(\mathsf{ck}, C, \boldsymbol{g}(\alpha), \alpha, y_1, \pi_1) = 1 \wedge \\ \mathsf{Vf}(\mathsf{ck}, C, \boldsymbol{g}(\alpha), \alpha, y_2, \pi_2) = 1 \end{array} \;\middle|\; \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{KGen}(n); \\ (C, \boldsymbol{g}, \alpha, y_1, \pi_1, y_2, \pi_2) \leftarrow \mathcal{A}(\mathsf{ck}); \end{array}\right] \approx_\lambda 0 \ .$$

The most important property of LPCS is weak computational special soundness (wCSS). In essense, it states that there always exists a polynomial $L(X)$ of degree at most $n + d$ consistent with all the evaluations.

**Definition 13.** *Let $n, d, m, K \in \mathsf{poly}(\lambda)$. We say that a non-interactive LPCS LPC is $(n, d, m, K)$-Weakly Computationally Special Sound (*wCSS*) if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{wcss}}_{\mathit{LPC},n,d,m,K,\mathcal{A}}(\lambda) :=$*

$$\Pr\left[\begin{array}{l} \forall i \neq j : \alpha_i \neq \alpha_j \wedge \boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m \\ \forall i \in [1..K] : \\ \quad \mathsf{Vf}(\mathsf{ck}, C, \boldsymbol{g}(\alpha_i), \alpha_i, y_i, \pi_i) = 1 \wedge \\ \deg(L) > n + d \end{array} \;\middle|\; \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{KGen}(n); \\ \begin{pmatrix} \boldsymbol{g}, C \\ (\alpha_i, y_i, \pi_i)_{i=1}^K \end{pmatrix} \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \approx_\lambda 0,$$

*where $L(X) := \mathsf{Inter}((\alpha_i, y_i)_{i=1}^K)$.*

Clearly, wCSS is trivially satisfied if $K \leq n + d + 1$.

We also define the (standard) computational special soundness for LPCS, which is a generalization of Definition 8.

**Definition 14.** *Let $n, d, m, K \in \mathsf{poly}(\lambda)$. We say that a non-interactive LPCS $\mathsf{LPC}$ is $(n, d, m, K)$-Computationally Special Sound (CSS) if there exists a DPT extractor $\mathsf{Ext}$ such that for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{css}}_{n,d,m,K,\mathcal{A}}(\lambda) :=$*

$$\Pr\left[\begin{array}{l|l} (\forall i \neq j : \alpha_i \neq \alpha_j) \wedge & \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{KGen}(n); \\ \boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m & \mathcal{T} = \begin{pmatrix} \boldsymbol{g}, C \\ (\alpha_i, y_i, \pi_i)_{i=1}^K \end{pmatrix} \leftarrow \mathcal{A}(\mathsf{ck}); \\ \forall i : \mathsf{Vf}(\mathsf{ck}, C, \boldsymbol{g}(\alpha_i), \alpha_i, y_i, \pi_i) = 1 & \boldsymbol{f} \leftarrow \mathsf{Ext}(\mathsf{ck}, \boldsymbol{g}, \mathcal{T}); \\ \wedge(\mathsf{Com}(\mathsf{ck}, \boldsymbol{f}) \neq C \vee & \\ \boldsymbol{f} \notin (\mathbb{F}^{\leq n}[X])^m \vee \exists i : L(\alpha_i) \neq y_i) & \end{array}\right] \approx_\lambda 0,$$

*where $L(X) := \sum_{i=1}^m g_i(X) f_i(X)$.*

It is easy to see that $(n, d, m, K)$-CSS implies $(n, d, m, K)$-wCSS. If $L(X) := \mathsf{Inter}((\alpha_i, y_i)_{i=1}^K)$ has degree $> n + d$, then it is impossible for extractor to find $\boldsymbol{f} \in (\mathbb{F}^{\leq n}[X])^m$ such that $L(X) = \sum_{i=1}^m g_i(X) f_i(X)$ since it would mean that $L(X)$ has degree $\leq n + d$.

**Interactive LPCS.** We also define an interactive version of LPCS.

An interactive LPCS (ILPCS) $\mathsf{ILPC}$ for polynomials in $\mathbb{F}[X]$ consists of PPT algorithms $\mathsf{Pgen}$, $\mathsf{KGen}$, $\mathsf{Com}$, $\mathsf{P}_{\mathsf{Lin}}$, $\mathsf{V}_{\mathsf{Lin}}$. The first three algorithms have identical syntax to the non-interactive case. The additional algorithms $\mathsf{P}_{\mathsf{Lin}}, \mathsf{V}_{\mathsf{Lin}}$ are a pair of interactive prover and verifier PPT algorithms that run an evaluation protocol. Typically, we assume that $\mathsf{V}_{\mathsf{Lin}}$ is a public-coin algorithm. The joint input to $\mathsf{P}_{\mathsf{Lin}}$ and $\mathsf{V}_{\mathsf{Lin}}$ are $\mathsf{ck}$ and $x = (C, \boldsymbol{g}(\alpha), \alpha, y)$, where $C$ is a commitment to some $\boldsymbol{f} \in (\mathbb{F}^{\leq n}[X])^m$, $\boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m$, $\alpha \in \mathbb{F}$, and $y \in \mathbb{F}$. The $\mathsf{P}_{\mathsf{Lin}}$ additionally gets as an input $w = (\boldsymbol{f}, \boldsymbol{g})$. We denote the verifier's decision by $b \leftarrow \langle \mathsf{P}_{\mathsf{Lin}}(w), \mathsf{V}_{\mathsf{Lin}} \rangle(\mathsf{ck}, x)$, which outputs 1 (accept) or 0 (reject).

**Definition 15.** *Let $m, n, d \in \mathsf{poly}(\lambda)$. We say that an interactive LPCS $\mathsf{ILPC}$ is correct if for any $n \in \mathsf{poly}(\lambda)$, any polynomial $\boldsymbol{f} \in (\mathbb{F}^{\leq n}[X])^m$, $\boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m$, and any point $\alpha \in \mathbb{F}$,*

$$\Pr\left[\langle \mathsf{P}_{\mathsf{Lin}}(w), \mathsf{V}_{\mathsf{Lin}} \rangle(\mathsf{ck}, x) = 1 \left| \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{KGen}(n); \\ C \leftarrow \mathsf{Com}(\mathsf{ck}, \boldsymbol{f}); y \leftarrow \sum_{i=1}^m g_i(\alpha) f_i(\alpha); \\ x \leftarrow (\mathsf{ck}, C, \boldsymbol{g}(\alpha), \alpha, y); w \leftarrow (\boldsymbol{f}, \boldsymbol{g}); \end{array} \right.\right] = 1 \ .$$

Suppose that $(\mathsf{P}_{\mathsf{Lin}}, \mathsf{V}_{\mathsf{Lin}})$ is a $(2\mu + 1)$-move public-coin protocol and let $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_\mu) \in \mathbb{N}^\mu$. Fix some $x = (\mathsf{ck}, C, \boldsymbol{g}(\alpha), \alpha, y)$. For an input $x$, we define the $\boldsymbol{\kappa}$-tree of transcripts $\mathcal{T}$ for a $(2\mu + 1)$-move public-coin protocol $(\mathsf{P}_{\mathsf{Lin}}, \mathsf{V}_{\mathsf{Lin}})$ as in Definition 5. Namely, $\mathcal{T}$ is a set of $K = \prod_{i=1}^\mu \kappa_i$ transcripts structured as a tree, where prover's messages are in the nodes and the verifier's challenges are on the edges. Each node at depth $i$ has $\kappa_i$ children corresponding to $\kappa_i$ pairwise distinct challenges. We say that $\mathcal{T}$ is accepting $\boldsymbol{\kappa}$-tree for $x$, if every path from a root node to a leaf node forms a transcript for which $\mathsf{V}_{\mathsf{Lin}}(x)$ outputs 1.

We are now ready to define weak CSS for an interactive LPCS.

**Definition 16.** *Let $n, d, m, K \in \mathsf{poly}(\lambda)$ and let $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_\mu) \in \mathbb{N}^\mu$. We say that an interactive LPCS $\mathsf{ILPC}$ is $(n, d, m, K, \boldsymbol{\kappa})$-weak CSS (wCSS) if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{wcss}}_{\mathsf{ILPC},n,d,m,K,\boldsymbol{\kappa},\mathcal{A}}(\lambda) :=$*

$$\Pr\left[\begin{array}{l|l} \forall i \neq j : \alpha_i \neq \alpha_j \wedge \boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m & \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{KGen}(n); \\ \forall i \in [1..K] : & \begin{pmatrix} \boldsymbol{g}, C \\ (\alpha_i, y_i, \mathcal{T}_i)_{i=1}^K \end{pmatrix} \leftarrow \mathcal{A}(\mathsf{ck}); \\ \mathcal{T}_i \text{ is an accepting } \boldsymbol{\kappa}\text{-tree for } x_i \wedge & \forall i \in [1..K] : \\ \deg(L) > n + d & \quad x_i \leftarrow (\mathsf{ck}, C, \boldsymbol{g}(\alpha_i), \alpha_i, y_i) \end{array}\right] \approx_\lambda 0,$$

*where $L(X) := \mathsf{Inter}((\alpha_i, y_i)_{i=1}^K)$.*

Similarly as in the non-interactive case, we can also define the (standard) CSS for an interactive LPCS.

**Definition 17.** *Let $n, d, m, K \in \mathsf{poly}(\lambda)$ and let $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_\mu) \in \mathbb{N}^\mu$. We say that an interactive LPCS $\mathsf{ILPC}$ is $(n, d, m, K, \boldsymbol{\kappa})$-Computationally Special Sound (CSS) if there exists a DPT extractor $\mathsf{Ext}$ such that for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{css}}_{\mathsf{ILPC},\mathsf{Ext},n,d,m,K,\boldsymbol{\kappa},\mathcal{A}}(\lambda) :=$*

$$\Pr\left[\begin{array}{l|l} (\forall i \neq j : \alpha_i \neq \alpha_j) \wedge \boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m & \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{KGen}(n); \\ \forall i \in [1..K] : \mathcal{T}_i \text{ is an} & \mathcal{T} = \begin{pmatrix} \boldsymbol{g}, C \\ (\alpha_i, y_i, \mathcal{T}_i)_{i=1}^K \end{pmatrix} \leftarrow \mathcal{A}(\mathsf{ck}); \\ \quad \text{accepting } \boldsymbol{\kappa}\text{-tree for } x_i \wedge & \boldsymbol{f} \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathcal{T}); \\ (\mathsf{Com}(\mathsf{ck}, \boldsymbol{f}) \neq C \vee \boldsymbol{f} \notin (\mathbb{F}^{\leq n}[X])^m & \forall i \in [1..K] : \\ \vee \exists i \in [1..K] : L(\alpha_i) \neq y_i) & \quad x_i \leftarrow (\mathsf{ck}, C, \boldsymbol{g}(\alpha_i), \alpha_i, y_i) \end{array}\right] \approx_\lambda 0,$$

*where $L(X) := \sum_{i=1}^m g_i(X) f_i(X)$.*

**On honestly computed $\boldsymbol{g}(\boldsymbol{\alpha})$.** One might question why it makes sense to assume that $\boldsymbol{g}(\alpha)$ given as an input to the verifier is honestly computed. There are two main reasons for that. In some situations, $g_i(X)$ might be a publicly known polynomial. In such case, the verifier can compute $g_i(\alpha)$ by itself and is assured that it is correct. In other situations, $g_i(X)$ might be a committed polynomials and then the prover convinces the verifier that $\deg(g_i) \leq d$ and that $g_i(\alpha)$ is correct via an evaluation protocol. In either case, we assume that that there are other means which guarantee that $\boldsymbol{g}(\alpha)$ are computed honestly and that $\boldsymbol{g}$ are some bounded degree polynomials. One could consider a definition of LPCS, where $\boldsymbol{g}$ are committed as well, but this would complicate the definitions and constructions unnecessarily.

In Plonk, some $g_i(X)$ are public and some depend on the committed polynomials. As we will see in Section 5, the main challenge in the Plonk's security proof is to show that a commitment $[t_\mathsf{l} + x^n t_\mathsf{m} + x^{2n} t_\mathsf{h}]_1$ binds to some low-degree polynomial, where $[t_\mathsf{l}]_1$, $[t_\mathsf{m}]_1$, $[t_\mathsf{h}]_1$ are KZG commitments chosen by the adversary. Here, $g_1(X) = 1$, $g_2(X) = X^n$, and $g_3(X) = X^{2n}$ are public polynomials.

## 4 Linearization KZG Commitment

We now instantiate KZG polynomial commitment scheme as a non-interactive linearization PCS and study its properties.

### 4.1 Construction: LinKZG

$\mathsf{KGen}(1^\lambda)$**:** Sample $x \leftarrow\!\!\$\ \mathbb{F}$ and output $\mathsf{ck} = ([(x^i)_{i=0}^n]_1, [1, x]_2)$.
$\mathsf{Com}(\mathsf{ck}, \boldsymbol{f} \in (\mathbb{F}^{\leq n}[X])^m)$**:** Output the commitment $[\boldsymbol{c}]_1 \leftarrow [f_1(x), \ldots, f_m(x)]_1$.
$\mathsf{Open}(\mathsf{ck}, \boldsymbol{f} \in (\mathbb{F}^{\leq n}[X])^m), \boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m), \alpha \in \mathbb{F})$**:** Sets $H(X) \leftarrow \sum_{i=1}^m g_i(\alpha) f_i(X)$, $y \leftarrow H(\alpha)$, and $h(X) \leftarrow (H(X) - y)/(X - \alpha)$. Output $[\pi]_1 \leftarrow [h(x)]_1$ and $y$.
$\mathsf{Vf}(\mathsf{ck}, \boldsymbol{g}(\alpha) \in (\mathbb{F}^{\leq d}[X])^m), [\boldsymbol{c}]_1 \in \mathbb{G}_1^m, \alpha \in \mathbb{F}, [\pi]_1, y \in \mathbb{F})$**:** Accepts if

$$\left(\left(\sum_{j=1}^m g_j(\alpha)[c_j]_1\right) - y\right) \bullet [1]_2 \stackrel{?}{=} [\pi]_1 \bullet [x - \alpha]_2 \ .$$

**Fig. 1.** The LinKZG polynomial commitment scheme

We present LinKZG commitment scheme that has been implicitly used in Marlin [CHM+20], Plonk [GWC19] and many other KZG-based protocols to optimize the number of openings the prover must send. We prove that LinKZG is a non-interactive LPCS with correctness, evaluation binding, and weak computational special soundness. Note that LinKZG does not satisfy (non-weak) computational special soundness as was shown in [LPS23, FFR24, LPS25].

We now summarize the construction in Fig. 1. Let $d, n \in \mathsf{poly}(\lambda)$ be two degree parameters. The commitment key $\mathsf{ck} = ([(x^i)_{i=0}^n]_1, [1, x]_2)$ is the KZG commitment key for degree $\leq n$ polynomials. We have a vector of $m$ polynomial $g_1, \ldots, g_m \in \mathbb{F}^{\leq d}[X]$ of degree at most $d$. The prover gets as an input $f_1, \ldots, f_m \in \mathbb{F}^{\leq n}[X]$ of degree at most $n$. The prover sends KZG commitments $[c_1, \ldots, c_m]_1$, where $[c_i]_1 = [f_i(x)]_1$, and the verifier responds with a query point $\alpha \in \mathbb{F}$. Finally, the prover computes a polynomial $H(X) = \sum_{i=1}^m g_i(\alpha) f_i(X)$, and $h(X) = (H(X) - H(\alpha))/(X - \alpha)$. The prover sends the commitment $[\pi]_1 \leftarrow [h(x)]_1$ and the evaluation $y \leftarrow H(\alpha)$ to the verifier. The verifier checks that

$$\left(\left(\sum_{j=1}^m g_j(\alpha)[c_j]_1\right) - y\right) \bullet [1]_2 \stackrel{?}{=} [\pi]_1 \bullet [x - \alpha]_2 \ .$$

Often in applications, $y = 0$ and the real goal is to prove that

$$\mathcal{V}(X) := \sum_{j=1}^m g_j(X) f_j(X) = 0 \ .$$

Since $\mathcal{V}(\alpha) = H(\alpha)$, then if $H(\alpha) = 0$ for $\alpha \leftarrow\!\!\$\ \mathbb{F}$, it follows from the Schwartz-Zippel lemma that $\mathcal{V}(X) \neq 0$ only with negligible probability $(n + d)/|\mathbb{F}|$. Conversely, it can be concluded that $\mathcal{V}(X) = 0$ with an overwhelming probability.

### 4.2 ARSDH in $\mathbb{G}_T$

Lipmaa et al. [LPS24] define the Adaptive Rational Strong Diffie-Hellman (ARSDH) assumption in the source group $\mathbb{G}_1$. We propose a new assumption, which extends the ARSDH assumption to the target group $\mathbb{G}_T$.

**Definition 18.** *Let $n_1, n_2 \in \mathsf{poly}(\lambda)$ be non-negative integers. The $(n_1, n_2)$-ARSDH assumption holds for $\mathsf{Pgen}$ in $\mathbb{G}_T$ if for any PPT $\mathcal{A}$,*
$\mathsf{Adv}^{\mathrm{arsdh}}_{\mathsf{Pgen},n_1,n_2,\mathbb{G}_T,\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l} S \subset \mathbb{F} \wedge |S| = n_1+n_2+1 \wedge \\ [g]_T \neq [0]_T \wedge [g]_T = [\varphi]_1 \bullet [\mathsf{Z}_S(x)]_T \end{array} \middle| \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); x \leftarrow\!\!\$\ \mathbb{F}; \\ \mathsf{ck} \leftarrow ([(x^i)_{i=0}^{n_1}]_1, [(x^i)_{i=0}^{n_2}]_2); \\ (S, [g]_T, [\varphi]_1) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array} \right] \approx_\lambda 0 \ ,$$

*where $\mathsf{Z}_S(X) := \prod_{s\in S}(X-s)$.*

The algebraic group model intuition for this assumption is straightforward. Suppose the $[g]_T$ corresponds to some polynomial $g(X)$ of degree $n_1+n_2$ and $[\varphi]_1$ to $\varphi(X)$ of degree at most $n_1$. Then, we have that $g(X) = \varphi(X)\mathsf{Z}_S(X)$, but the left hand size has degree at most $n_1+n_2$ and the right hand side at least $n_1+n_2+1$ unless $\varphi(X) = 0$. The latter would imply that $g(X) = 0$, which is not allowed.

We show that the ARSDH assumption in $\mathbb{G}_T$ implies the original ARSDH assumption in $\mathbb{G}_1$.

**Lemma 3.** *Let $n_1$, $n_2$ be non-negative integers such that $|\mathbb{F}| \geq n_1+n_2+2$. If the $(n_1, n_2)$-*ARSDH *assumption holds in $\mathbb{G}_T$, then the $n_1$-*ARSDH *assumption holds in $\mathbb{G}_1$.*

*Proof.* Suppose there exists an adversary $\mathcal{A}$ that breaks the $n_1$-ARSDH assumption in $\mathbb{G}_1$ with non-negligible advantage. We construct an adversary $\mathcal{B}$ that breaks the $(n_1, n_2)$-ARSDH assumption in $\mathbb{G}_T$ with the same advantage.

$\mathcal{B}$ receives the commitment key $\mathsf{ck} = ([(x^i)_{i=0}^{n_1}]_1, [(x^i)_{i=0}^{n_1}]_2)$ from the challenger and forwards $\mathsf{ck}' = ([(x^i)_{i=0}^{n_1}]_1, [1, x]_2)$ to $\mathcal{A}$. Then, $\mathcal{A}$ returns $(S_1, [g']_1, [\varphi]_1)$. Next, $\mathcal{B}$ picks any set $S_2 \subset \mathbb{F} \setminus (S_1 \cup \{x\})$ of size $n_2$ and computes $[g]_T := [g']_1 \bullet [\mathsf{Z}_{S_2}(x)]_2$. Finally, $\mathcal{B}$ outputs $(S := S_1 \cup S_2, [g]_T, [\varphi]_1)$.

If $\mathcal{A}$ wins the $n_1$-ARSDH game, then we have that $g' = \varphi\mathsf{Z}_{S_1}(x)$. Therefore, $g = g'\mathsf{Z}_{S_2}(x) = \varphi\mathsf{Z}_{S_1}(x)\mathsf{Z}_{S_2}(x) = \varphi\mathsf{Z}_S(x)$. Additionally, since $g' \neq 0$ and $\mathsf{Z}_{S_2}(x) \neq 0$ (it would be 0 if $x \in S_2$, which we explicitly excluded), we have that $g \neq 0$. Thus, $\mathcal{B}$ wins the $(n_1, n_2)$-ARSDH game whenever $\mathcal{A}$ wins the $n_1$-ARSDH game. More precisely, for any PPT $\mathcal{A}$, there exists a PPT $\mathcal{B}$ such that $\mathsf{Adv}^{\mathrm{arsdh}}_{\mathsf{Pgen},n_1,\mathbb{G}_1,\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{arsdh}}_{\mathsf{Pgen},n_1,n_2,\mathbb{G}_T,\mathcal{B}}(\lambda)$. □

**Comparison to SplitRSDH.** One of our technical contributions (see Section 5) is to show that the Plonk is knowledge sound under the ARSDH assumption in $\mathbb{G}_T$ in the ROM. The prior work [LPS25] showed the same result under the ARSDH assumption in $\mathbb{G}_1$ (Definition 3) and the SplitRSDH assumption (Definition 4). We are unaware of any trivial reductions between SplitRSDH

and ARSDH in $\mathbb{G}_T$. However, at least visually, ARSDH in $\mathbb{G}_T$ seems less technical assumption and also less tailored for Plonk. We note that ARSDH in $\mathbb{G}_1$ compares to ARSDH in $\mathbb{G}_T$ similarly as SDH (in $\mathbb{G}_1$) compares to Target SDH (TSDH). The Target SDH is widely used [PHGR13, DFGK14, DGP+19, WZ25].

### 4.3 Security of LinKZG

We show that LinKZG has all the previously defined properties of a linearization PCS except CSS. We first start with three easier properties to prove.

**Theorem 1 (Correctness, Binding, Evaluation Binding).** LinKZG *is a non-interactive LPCS with following properties: (i) Correctness, (ii) If* KZG *is binding, then* LinKZG *is binding, and (iii) If* KZG *is evaluation binding, then* LinKZG *is evaluation binding.*

*Proof.* **Correctness.** This is straightforward to verify. Following the notation from Fig. 1, we have that

$$\begin{aligned}[\pi]_1 \bullet [x-\alpha]_2 =& [h(x)]_1 \bullet [x-\alpha]_2 = \left[\frac{H(x)-H(\alpha)}{x-\alpha}\right]_1 \bullet [x-\alpha]_2 \\ =& [H(x)-H(\alpha)]_1 \bullet [1]_2 = \left(\left(\textstyle\sum_{j=1}^m g_j(\alpha)[c_j]_1\right) - y\right) \bullet [1]_2 \ .\end{aligned}$$

**Binding.** Since LinKZG commitment is a just a vector of KZG commitments, the claim follows directly from the binding property of KZG.

**Evaluation binding.** Suppose a PPT adversary $\mathcal{A}$ breaks the evaluation binding of LinKZG. Then, it outputs $([c_j]_1)_{j=1}^m, \boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m, \alpha \in \mathbb{F}, y_1, y_2 \in \mathbb{F}, [\pi_1]_1, [\pi_2]_1 \in \mathbb{G}_1$ such that $y_1 \neq y_2$ and for both $b \in \{1,2\}$, it holds that

$$\left(\left(\textstyle\sum_{j=1}^m g_j(\alpha)[c_j]_1\right) - [y_b]_1\right) \bullet [1]_2 = [\pi_b]_1 \bullet [x-\alpha]_2 \ .$$

We now construct an adversary $\mathcal{B}$ that breaks the evaluation binding of KZG. The adversary $\mathcal{B}(\mathsf{ck})$ runs $\mathcal{A}(\mathsf{ck})$ to obtain the output above, computes a commitment $[c]_1 \leftarrow \sum_{j=1}^m g_j(\alpha)[c_j]_1$, and outputs $([c]_1, \alpha, y_1, [\pi_1]_1, y_2, [\pi_2]_1)$. Whenever $\mathcal{A}$ breaks the evaluation binding of LinKZG, $\mathcal{B}$ breaks the evaluation binding of KZG. Thus, $\mathsf{Adv}^{\mathrm{eb}}_{\mathsf{LinKZG},n,d,m,\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{eb}}_{\mathsf{KZG},n,\mathcal{B}}(\lambda)$. □

**Theorem 2 (Weak CSS).** *Let $n, d, m, K \in \mathsf{poly}(\lambda)$. If the $n$-*PDL *assumption holds in $\mathbb{G}_1$ and the $(n,d)$-*ARSDH *assumption holds in $\mathbb{G}_T$, then* LinKZG *has $(n,d,m,K)$-*wCSS.

*Proof.* Suppose a PPT wCSS adversary $\mathcal{A}$ outputs $(g_j(X), [c_j]_1)_{j=1}^m, (\alpha_i, y_i, [\pi_i]_1)_{i=1}^K$, such that it breaks the wCSS property. That is, interpolation polynomial $L(X)$ of $(\alpha_i, y_i)_{i=1}^K$ has degree greater than $n+d$, $\alpha_i$ are distinct, $\deg(g_j) \leq d$ for all $j \in [1..m]$, and the following equation holds for all $i \in [1..K]$:

$$\left(\left(\textstyle\sum_{j=1}^m g_j(\alpha_i)[c_j]_1\right) - y_i\right) \bullet [1]_2 = [\pi_i]_1 \bullet [x-\alpha_i]_2 \ . \tag{2}$$

Firstly, if $K \leq n+d+1$, then $\deg(L) \leq K-1 \leq n+d$, contradicting the claim that $\deg(L) > n+d$. Thus, we assume in the following that $K \geq n+d+2$.

Let $S_1 = \{\alpha_i\}_{i=1}^{n+d+1}$ and $L_1(X) := \mathsf{Inter}(\{\alpha_i, y_i\}_{i=1}^{n+d+1})$ be the interpolation polynomial over the respective points. Then, $L_1$ is a unique polynomial of degree at most $n+d$ such that $L_1(\alpha_i) = y_i$ for all $i \in [1..n+d+1]$. There exists $j^*$ such that $L(\alpha_{j^*}) \neq L_1(\alpha_{j^*})$ (otherwise $L = L_1$ and $\deg(L) \leq n+d$). Let $L_2(X) := \mathsf{Inter}(\{(\alpha_i, y_i)\}_{i=1}^{n+d} \cup \{(\alpha_{j^*}, y_{j^*})\})$. It also holds that $L_1(X) \neq L_2(X)$ because $L_1(\alpha_{j^*}) \neq L_2(\alpha_{j^*})$.

Let $\ell_i^{S_1}(X) \in \mathbb{F}^{\leq n+d}[X]$ be the $i$th Lagrange polynomial over the set $S_1$ (see Eq. (1)) for $i \in [1..n+d+1]$. Then, $L_1(X) = \sum_{i=1}^{n+d+1} y_i \ell_i^{S_1}(X)$. Recall also that

$$\ell_i^{S_1}(X) = d_i^{S_1} \mathsf{Z}_{S_1}(X)/(X-\alpha_i) \ , \tag{3}$$

where $d_i^{S_1} = 1/\prod_{j \in [1..n+d+1]\setminus\{i\}}(\alpha_i - \alpha_j)$. Now, observe that

$$\begin{aligned}
&\textstyle\sum_{j=1}^{m} g_j(x)c_j - L_1(x) = \\
&\textstyle\sum_{j=1}^{m} \left(\sum_{i=1}^{n+d+1} g_j(\alpha_i)\ell_i^{S_1}(x)\right) c_j - \sum_{i=1}^{n+d+1} y_i \ell_i^{S_1}(x) = \\
&\textstyle\sum_{i=1}^{n+d+1} \ell_i^{S_1}(x) \left(\sum_{j=1}^{m} g_j(\alpha_i)c_j - y_i\right) \overset{(2)}{=} \sum_{i=1}^{n+d+1} \ell_i^{S_1}(x)\pi_i(x-\alpha_i) \overset{(3)}{=} \\
&\left(\textstyle\sum_{i=1}^{n+d+1} d_i^{S_1}\pi_i\right) \mathsf{Z}_{S_1}(x) = \hat{\pi}_1 \mathsf{Z}_{S_1}(x) \ .
\end{aligned}$$

where $\hat{\pi}_1 = \sum_{i=1}^{n+d+1} d_i^{S_1}\pi_i$. On the first step, we used that $\deg(g_j) \leq d$ and thus $g_j(X) = \sum_{i=1}^{n+d+1} g_j(\alpha_i)\ell_i^{S_1}(X)$ and similarly $L_1(X) = \sum_{i=1}^{n+d+1} y_i \ell_i^{S_1}(X)$.

By defining $S_2 = \{\alpha_i\}_{i=1}^{n+d} \cup \{\alpha_{j^*}\}$, we can apply the same derivations to $S_2$ and $L_2(X)$. Thus, we obtain that for $b \in \{1,2\}$,

$$\textstyle\sum_{j=1}^{m} g_j(x)c_j - L_b(x) = \hat{\pi}_b \mathsf{Z}_{S_b}(x) \ . \tag{4}$$

Let us define the left hand side as $\hat{g}_b = \sum_{j=1}^{m} g_j(x)c_j - L_b(x)$ for $b \in \{1,2\}$. We look at two cases:

Case 1: If $\hat{g}_1 \neq 0$ or $\hat{g}_2 \neq 0$, we can construct an adversary $\mathcal{B}$ that breaks the $(n,d)$-ARSDH assumption. Namely, $\mathcal{B}$ outputs $(S_b, [\hat{g}_b]_T, [\hat{\pi}_b]_1)$, where $[\hat{g}_b]_T = \sum_{j=1}^{m} [c_j]_1 \bullet [g_j(x)]_2 - [L_b(x)]_T$ for $b$ such that $\hat{g}_b \neq 0$. We present the complete description of the adversary $\mathcal{B}$ in Fig. 2.

Case 2: Otherwise, $\hat{g}_1 = 0$ and $\hat{g}_2 = 0$. Then, we have that $\sum_{j=1}^{m} g_j(x)c_j = L_1(x)$ and $\sum_{j=1}^{m} g_j(x)c_j = L_2(x)$. However, $L_1(X) \neq L_2(X)$, which implies $\Delta(X) := L_1(X) - L_2(X)$ is a non-zero polynomial of degree at most $n+d$, which has $x$ as a root. Thus, we can find roots of $\Delta(X)$ and break the $n$-PDL assumption in $\mathbb{G}_1$. We present the PDL adversary $\mathcal{C}$ in Fig. 2.

Thus, we can conclude that for any PPT adversary $\mathcal{A}$ there exists PPT adversaries $\mathcal{B}$ and $\mathcal{C}$ such that $\mathsf{Adv}^{\mathrm{wcss}}_{\mathsf{LinKZG},n,d,m,K,\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{arsdh}}_{\mathsf{Pgen},n,d,\mathbb{G}_T,\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathrm{pdl}}_{\mathsf{Pgen},n,\mathbb{G}_1,\mathcal{C}}(\lambda)$. □

Clearly, the $(n,d)$-ARSDH assumption in $\mathbb{G}_T$ implies the $n$-PDL assumption in $\mathbb{G}_1$. Thus, we can simplify the theorem statement as follows.

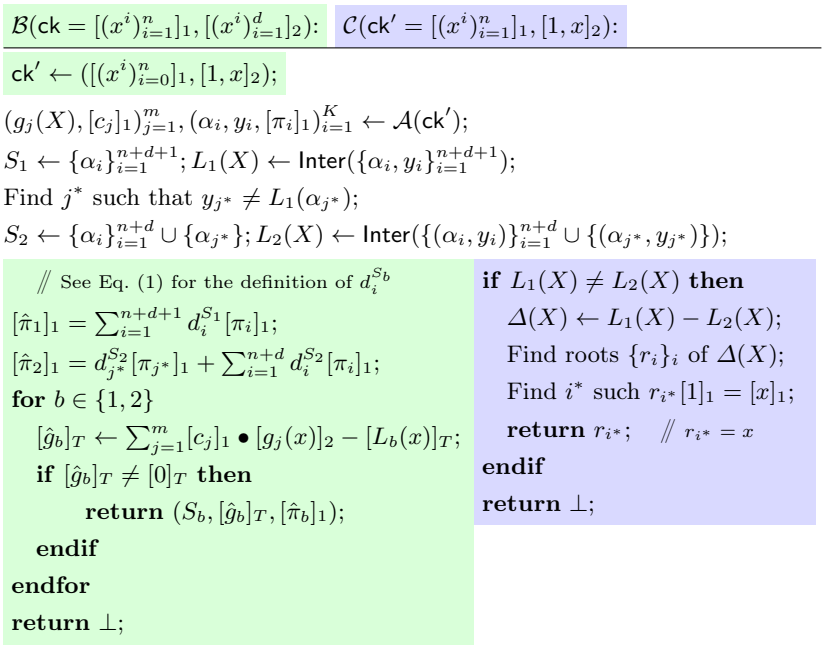



**Fig. 2.** Reductions $\mathcal{B}$ and $\mathcal{C}$ in Theorem 2 that respectively break the $(n, d)$-ARSDH assumption in $\mathbb{G}_T$ and the $n$-PDL assumption in $\mathbb{G}_1$ given access to wCSS adversary $\mathcal{A}$ for LinKZG. Lines with the green background are only present in $\mathcal{B}$ and lines with the blue background are only present in $\mathcal{C}$.

**Corollary 1.** *Let $n, d, m, K \in \mathsf{poly}(\lambda)$. If the $(n, d)$-ARSDH assumption holds in $\mathbb{G}_T$, then* LinKZG *has $(n, d, m, K)$-wCSS.*

Next, we show that in the special case of $m = 1$, wCSS holds under the potentially weaker $\mathbb{G}_1$ version of the ARSDH assumption.

**Theorem 3 (wCSS for $m = 1$).** *Let $n, d, K \in \mathsf{poly}(\lambda)$ and $m = 1$. If the $n$-ARSDH assumption holds in $\mathbb{G}_1$, then* LinKZG *has $(n, d, m, K)$-wCSS.*

*Proof.* Suppose a PPT adversary $\mathcal{A}$ outputs $(g(X), [c]_1), (\alpha_i, y_i, [\pi_i]_1)_{i=1}^K$, such that it breaks the wCSS property of LinKZG (see Definition 14). Namely, the interpolation polynomial $L(X)$ of $(\alpha_i, y_i)_{i=1}^K$ has degree greater than $n + d$, $\alpha_i$ are distinct, $\deg(g) \leq d$, and for $i \in [1..K]$,

$$\big(g(\alpha_i)[c]_1 - y_i\big) \bullet [1]_2 = [\pi_i]_1 \bullet [x - \alpha_i]_2 \ . \tag{5}$$

In the following, we assume that $K > n + d + 1$ since otherwise $\deg(L) \leq K - 1 \leq n + d$.

Case 1: Suppose $g(X) = 0$. Then, we have that $-y_i = [\pi_i]_1 \bullet [x - \alpha_i]_2$ for all $i \in [1..K]$. If there exists $k^*$ such that $y_{k^*} \neq 0$, we have that

$$[\varphi]_1 := {}^{-[\pi_{k^*}]_1}\!/_{y_{k^*}} = [{}^1\!/_{(x - \alpha_{k^*})}]_1 \ .$$

This already breaks the $n$-SDH assumption (Definition 2) in $\mathbb{G}_1$. The latter implies the $n$-ARSDH assumption in $\mathbb{G}_1$ [LPS24].

Alternatively, if $y_i = 0$ for all $i \in [1..K]$, then $L(X) = 0$. This contradicts the claim that $\deg(L) > n + d \geq 0$.

Case 2: Suppose $g(X) \neq 0$. Since $\deg(g) \leq d$, $g$ has at most $d$ roots. Therefore, in $\overline{\{\alpha_i\}_{i=1}^K}$, there are at least $n+1$ non-roots of $g$. Without loss of generality, let them be $\alpha_1, \ldots, \alpha_{n+1}$. We define $\beta_i := y_i/g(\alpha_i)$ and $\pi'_i := \pi_i/g(\alpha_i)$ for $i \in [1..n+1]$, which is well-defined since $g(\alpha_i) \neq 0$. Then, Eq. (2) can be rewritten as

$$\big([c]_1 - \beta_i\big) \bullet [1]_2 = [\pi'_i]_1 \bullet [x - \alpha_i]_2 \ .$$

By applying the batching lemma Lemma 2, we have that

$$[c - f(x)]_1 = [\pi]_1 \mathsf{Z}_S(x) \ ,$$

where $f(X) = \mathsf{Inter}((\alpha_j, \beta_j)_{j=1}^{n+1})$ and $[\pi]_1 = \sum_{j=1}^{n+1} d_j \pi'_j$ for $d_j$ defined in Eq. (1). If $[c]_1 \neq [f(x)]_1$, we can break the $n$-ARSDH assumption in $\mathbb{G}_1$ by outputting $(\{\alpha_i\}_{i=1}^{n+1}, [c - f(x)]_1, [\pi]_1)$.

Suppose now that $[c]_1 = [f(x)]_1$. We claim that $L(X) = g(X)f(X)$, which implies that $\deg(L) \leq \deg(g) + \deg(f) \leq d + n$, contradicting the claim that $\deg(L) > n + d$. Suppose that there exists $j^* \in [1..K]$ such that $L(\alpha_{j^*}) \neq g(\alpha_{j^*})f(\alpha_{j^*})$. Let $f(X) = f(\alpha_{j^*}) + \pi_{j^*}(X)(X - \alpha_{j^*})$, where $\pi_{j^*}(X) \in \mathbb{F}^{\leq n-1}[X]$. Then,

$$\begin{aligned} g(\alpha_{j^*})c - L(\alpha_{j^*}) &= g(\alpha_{j^*})f(x) - L(\alpha_{j^*}) \\ &= g(\alpha_{j^*})f(\alpha_{j^*}) - L(\alpha_{j^*}) + g(\alpha_{j^*})\pi_{j^*}(x)(x - \alpha_{j^*}) \ . \end{aligned}$$

We can now derive from Eq. (5) that

$$[g(\alpha_{j^*})f(\alpha_{j^*}) - L(\alpha_{j^*})]_1 \bullet [1]_2 = [\pi_{j^*} - g(\alpha_{j^*})\pi_{j^*}(x)]_1 \bullet [x - \alpha_{j^*}]_2 \ .$$

Let $v := g(\alpha_{j^*})f(\alpha_{j^*}) - L(\alpha_{j^*}) \neq 0$ and $[\varphi]_1 := {}^{[\pi_{j^*} - g(\alpha_{j^*})\pi_{j^*}(x)]_1}/_{v}$. Then, $[\varphi]_1 = [{}^1/_{(x - \alpha_{j^*})}]_1$ and $x \neq \alpha_{j^*}$ (that would break the $n$-PDL assumption). The reduction can thus break the $n$-SDH assumption in $\mathbb{G}_1$, which implies the $n$-ARSDH assumption in $\mathbb{G}_1$ [LPS24]. Therefore, $L(\alpha_i) = g(\alpha_i)f(\alpha_i)$ for all $i \in [1..K]$, which implies $L(X)$ has degree at most $d + n$. □

Finally, we show that if we additionally require in the $m = 1$ setting that $g(X) \neq 0$ and $K \geq n + d + 1$, then we can prove the stronger CSS property.

**Theorem 4 (CSS for $m = 1$).** *Let $n, d, K \in \mathsf{poly}(\lambda)$, $m = 1$, and $K \geq n+d+1$. If the $n$-ARSDH assumption holds in $\mathbb{G}_1$, then $\mathsf{LinKZG}$ has $(n, d, m, K)$-CSS under the restriction that $g_1(X) = g(X) \neq 0$.*

*Proof.* Suppose a PPT adversary $\mathcal{A}$ outputs $\mathcal{T} = \big((g(X), [c]_1), (\alpha_i, y_i, [\pi_i]_1)_{i=1}^K\big)$, such that it breaks the CSS property of $\mathsf{LinKZG}$ (see Definition 14) and additionally $g(X) \neq 0$. In particular, $\alpha_i$ are distinct, $\deg(g) \leq d$, and for $i \in [1..K]$,

$$\big(g(\alpha_i)[c]_1 - y_i\big) \bullet [1]_2 = [\pi_i]_1 \bullet [x - \alpha_i]_2 \ . \tag{6}$$

$\mathsf{Ext}_{\mathsf{LinKZG}}(\mathsf{ck}, (g(X), [c]_1), (\alpha_i, y_i, [\pi_i]_1)_{i=1}^K)$:

1 : Find $j_1, \ldots, j_{n+1} \in [1..K]$ such that $g(\alpha_{j_i}) \neq 0$ for all $i \in [1..n+1]$;
2 : **for** $i \in [1..n+1]$ **do**
3 : $\quad \beta_i \leftarrow y_i / g(\alpha_{j_i})$; $[\pi'_i]_1 \leftarrow [\pi_{j_i}]_1 / g(\alpha_{j_i})$;
4 : **endfor**
5 : $\mathcal{T}_{\mathsf{KZG}} \leftarrow \left([c]_1, (\alpha_{j_i}, \beta_i, [\pi'_i]_1)_{i=1}^{n+1}\right)$;
6 : **return** $\mathsf{Ext}_{\mathsf{KZG}}(\mathsf{ck}, \mathcal{T}_{\mathsf{KZG}})$;

**Fig. 3.** LinKZG CSS extractor $\mathsf{Ext}_{\mathsf{LinKZG}}$ for $m = 1$, $K \geq n+d+1$, and $g(X) \neq 0$, where $\mathsf{Ext}_{\mathsf{KZG}}$ is the $(n+1)$-CSS extractor for KZG.

Let $\mathsf{Ext}_{\mathsf{KZG}}$ be the $(n+1)$-CSS extractor for KZG. We start by constructing CSS extractor $\mathsf{Ext}_{\mathsf{LinKZG}}$ for LinKZG using $\mathsf{Ext}_{\mathsf{KZG}}$ as a subroutine. The extractor $\mathsf{Ext}_{\mathsf{LinKZG}}$ receives $\mathcal{T}$ from $\mathcal{A}$ and must compute $f(X) \in \mathbb{F}^{\leq n}[X]$ such that $[c]_1 = [f(x)]_1$.

Since $\deg(g) \leq d$, $g$ has at most $d$ roots. Therefore, in $\{\alpha_i\}_{i=1}^K$, there are at least $n+1$ non-roots of $g$. Let us denote them by $\alpha'_1, \ldots, \alpha'_{n+1}$. We define $\beta_i := y_i / g(\alpha'_i)$ and $\pi'_i := \pi_i / g(\alpha'_i)$ for $i \in [1..n+1]$, which are well-defined since $g(\alpha'_i) \neq 0$. Then, Eq. (2) can be rewritten as

$$\left([c]_1 - [\beta_i]_1\right) \bullet [1]_2 = [\pi'_i]_1 \bullet [x - \alpha'_i]_2 \ .$$

Thus, we have obtained an instance of the $(n+1)$-CSS extraction problem for KZG. Our extractor $\mathsf{Ext}_{\mathsf{LinKZG}}$ computes the above transcripts and invokes $\mathsf{Ext}_{\mathsf{KZG}}$ to recover some polynomial $f(X)$, which it then returns. We present the complete description of $\mathsf{Ext}_{\mathsf{LinKZG}}$ in Fig. 3.

Recall, $\mathcal{A}$ wins the CSS game for LinKZG if the extractor fails to output a polynomial $f(X)$ such that $\deg(f) \leq n$, $[c]_1 \neq [f(x)]_1$, and for $L(X) = g(X)f(X)$ it holds that $L(\alpha_i) = y_i$ for all $i \in [1..K]$.

We look at two cases:

Case 1: Suppose either $\deg(f) > n$ or $[c]_1 \neq [f(x)]_1$. In this case, we can construct an adversary $\mathcal{B}$ that breaks $(n+1)$-CSS of KZG. The construction of $\mathcal{B}$ is quite simple. $\mathcal{B}$ gets $\mathsf{ck}$ as an input and forwards it to $\mathcal{A}$. Then, $\mathcal{B}$ receives from $\mathcal{A}$ the transcript $\mathcal{T}$ of LinKZG and computes the transcripts $\mathcal{T}_{\mathsf{KZG}} = \left([c]_1, (\alpha_{j_i}, \beta_i, [\pi'_i]_1)_{i=1}^{n+1}\right)$ exactly as $\mathsf{Ext}_{\mathsf{LinKZG}}$ computes on the step 5 in Fig. 3. $\mathcal{B}$ outputs $\mathcal{T}_{\mathsf{KZG}}$ and if $\mathsf{Ext}_{\mathsf{KZG}}(\mathsf{ck}, \mathcal{T}_{\mathsf{KZG}})$ fails to output $f(X)$ such that $\deg(f) \leq n$ and $[c]_1 = [f(x)]_1$, then $\mathcal{B}$ wins the $(n+1)$-CSS game for KZG.

Case 2: Otherwise, $\deg(f) \leq n$ and $[c]_1 = [f(x)]_1$. Suppose that there exists $j^* \in [1..K]$ such that $L(\alpha_{j^*}) \neq y_{j^*}$. We construct and adversary $\mathcal{C}$ that breaks the $n$-SDH assumption in $\mathbb{G}_1$. $\mathcal{C}$ receives $\mathsf{ck}$ as an input and forwards it to $\mathcal{A}$. Then, it runs KZG extractor to recover $f(X)$ as described in Case 1. Based on the verification equation, we have that $g(\alpha_{j^*})f(x) - y_{j^*} = \pi_{j^*}(x - \alpha_{j^*})$. We can express $f(X) = f(\alpha_{j^*}) + (X - \alpha_{j^*})q(X)$ for some polynomial $q(X)$ of degree $\leq n-1$. Inserting this expression into the previous equation, we have

that $g(\alpha_{j^*})f(\alpha_{j^*}) - y_{j^*} = (\pi_{j^*} - g(\alpha_{j^*})q(x))(x - \alpha_{j^*})$. If $L(\alpha_{j^*}) \neq y_{j^*}$, then the left-hand side of the equation is non-zero. Thus, $\mathcal{C}$ can break the $n$-SDH assumption in $\mathbb{G}_1$ by outputting $[\varphi]_1 = [1/(x-\alpha_{j^*})]_1$, where $[\varphi]_1 = ([\pi_{j^*}]_1 - g(\alpha_{j^*})[q(x)]_1)/(L(\alpha_{j^*}) - y_{j^*})$.

Thus, we have shown that for any PPT adversary $\mathcal{A}$ there exists PPT adversaries $\mathcal{B}$ and $\mathcal{C}$ such that $\mathsf{Adv}^{\mathrm{css}}_{n,d,1,K,\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{css}}_{\mathsf{KZG},n+1,\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathrm{sdh}}_{\mathsf{Pgen},n,\mathbb{G}_1,\mathcal{C}}(\lambda)$ constrained that $g(X) \neq 0$. We recall that $\mathsf{KZG}$ has $(n+1)$-CSS also under the $n$-ARSDH assumption in $\mathbb{G}_1$. □

**Why LinKZG does not satisfy CSS?** As we saw above, $\mathsf{LinKZG}$ satisfies wCSS under the ARSDH assumption in $\mathbb{G}_T$, but we managed to prove CSS only when we have only one left polynomial $g(X)$ and furthermore $g$ must be a non-zero. The extractability of the right polynomials $f_i(X)$ was studied in [LPS23, FFR24, LPS25]. Most relevant to us, Lipmaa et al. [LPS25] studied the computational special soundness of linearization technique in the case when $y_i = 0$ in our $\mathsf{LinKZG}$ protocol. They show that CSS (without any additional restrictions) is impossible unless one breaks the discrete logarithm assumption.

For the sake of completeness, we sketch a special case of their attack. Suppose a CSS adversary $\mathcal{A}$ samples $y \leftarrow\!\$\ \mathbb{F}$, sets $[c_1]_1 = [y]_1$, $[c_2]_1 = -[y]_1$, and $g_1(X) = g_2(X) = 1$. In such case, for any $\alpha_i$, $[c]_1 = g_1(\alpha_i)[c_1]_1 + g_2(\alpha_i)[c_2]_1 = [y]_1 - [y]_1 = [0]_1$. Thus, $[c]_1$ is also a commitment to a zero polynomial. This makes it easy for $\mathcal{A}$ to produce a valid CSS transcript tree

$$\mathcal{T} = \left(g_1(X) = 1, g_2(X) = 1, [c_1]_1, [c_2]_1, (\alpha_i, y_i = 0, [\pi_i]_1 = [0]_1)_{i=1}^{K}\right) ,$$

for any $K \in \mathsf{poly}(\lambda)$ and any distinct points $\alpha_i$. Suppose there exists a CSS extractor $\mathsf{Ext}$ that given $\mathsf{ck}$ and $\mathcal{T}$ as an input, recovers polynomials $f_1(X)$ and $f_2(X)$ such that $[c_1]_1 = [f_1(x)]_1$ and $[c_2]_1 = [f_2(x)]_1$.

We can now construct a discrete logarithm adversary $\mathcal{B}$ that gets $[y]_1$ as an input and must output $y$. $\mathcal{B}$ does it by emulating $\mathcal{A}$ while sampling $x \leftarrow\!\$\ \mathbb{F}$ and its own public key $\mathsf{ck} = ([1, x, \ldots, x^n]_1, [1, x]_2)$. It sets $[c_1]_1 = [y]_1$ and $[c_2]_1 = -[y]_1$ and computes $\mathcal{T}$ exactly as $\mathcal{A}$. Then, $\mathcal{B}$ invokes the extractor $\mathsf{Ext}$ on $(\mathsf{ck}, \mathcal{T})$ to obtain $f_1(X)$ and $f_2(X)$. However, note that if $\mathsf{Ext}$ succeeds then $c_1 = y$. $\mathcal{B}$ knows the trapdoor $x$, thus it can output $f_1(x) = c_1 = y$ and break the discrete logarithm assumption. We refer to [LPS25, Theorem 3] for a more general attack.

## 5 Security Proof of Plonk

### 5.1 Summary of Plonk

We briefly summarize the Plonk SNARK [GWC19] and its knowledge soundness proof under falsifiable assumptions in the random oracle model [LPS25]. Plonk is a popular zero-knowledge SNARK in practice due to its efficiency and for its flexible "Plonkish" constraint system for describing the relations. Plonk also

supports integration with lookup arguments [GW20], which has further increased its appeal.

The original Plonk paper [GWC19] first describes an idealized low-degree protocol (similar to the popular polynomial interactive oracle proof model [BFS20]), which is then compiled to a SNARK using the KZG polynomial commitment scheme. They prove the security of the interactive Plonk in the algebraic group model (AGM) [FKL18] and then apply the Fiat-Shamir heuristic. Later, Lipmaa et al. [LPS25] proved CSS of Plonk under falsifiable assumptions by leveraging their earlier work [LPS24], which proved CSS of the KZG.

We roughly follow the notation in [GWC19]. The strategy in Plonk is that the prover and the verifier exchange several rounds of commitments and challenges. The verifier's goal is to check that a certain polynomial equation $\mathsf{F}(X)$, which depends on the prover's witness polynomials, satisfies $\mathsf{F}(\omega_i) = 0$ for all $\omega_i \in \mathbb{H}$, where $\mathbb{H}$ is public set of size $n$. This is equivalent to proving that $\mathsf{F}(X) = \mathsf{t}(X)\mathsf{Z}_{\mathbb{H}}(X)$ is satisfied, where $\mathsf{Z}_{\mathbb{H}}(X)$ is a vanishing polynomial over $\mathbb{H}$, and $\mathsf{t}(X)$ is some quotient polynomial. Eventually, the verifier sends an evaluation point $\mathfrak{z} \leftarrow\!\!\$\ \mathbb{F}$ to the prover and asks to open several polynomials at that point. The goal is to test that $\mathcal{V}(X) := \mathsf{F}(X) - \mathsf{t}(X)\mathsf{Z}_{\mathbb{H}}(X)$ opens to zero at $\mathfrak{z}$, which would imply that $\mathcal{V}(X) = 0$ polynomial with an overwhelming probability.

There are three types of commitments in Plonk:

- Commitments to public polynomials $\mathsf{q}_M(X), \mathsf{q}_L(X), \mathsf{q}_R(X), \ldots$ that describe the constraint system and which are computed in the preprocessing phase. Their main purpose is to outsource the polynomial evaluation to the prover.
- Commitments to witness polynomials $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{z}(X)$ that encode the prover's witness. The prover sends evaluations of all of these polynomials at the challenge point $\mathfrak{z}$ (or in the case of $\mathsf{z}$, the evaluation point is $\omega\mathfrak{z}$, where $\omega$ is the generator of $\mathbb{H}$).
- For efficiency reason, the quotient polynomial $\mathsf{t}(X)$ is split into three polynomials of lower degree such that $\mathsf{t}(X) = t_{\mathsf{l}}(X) + X^n t_{\mathsf{m}}(X) + X^{2n} t_{\mathsf{h}}(X)$. The prover sends commitments to $t_{\mathsf{l}}(X)$, $t_{\mathsf{m}}(X)$, and $t_{\mathsf{h}}(X)$, but never sends evaluations of these polynomials at $\mathfrak{z}$.

The CSS proof of Plonk in [LPS25] uses CSS of KZG PCS to extract the witness polynomials $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{z}(X)$. This is possible because the prover sends evaluations of these polynomials and thus it is possible to interpolate the polynomials. Importantly, only the ARSDH assumption in $\mathbb{G}_1$ is needed for this part of the proof. The witness polynomials and the public polynomials together define a unique polynomial $\mathsf{F}(X)$. However, the quotient polynomial $t_{\mathsf{l}}(X)$, $t_{\mathsf{m}}(X)$, and $t_{\mathsf{h}}(X)$ are never evaluated by the prover. Thus, they cannot be extracted straightforwardly using CSS of KZG. Eventually, their proof boils down to the following equations:

$$[\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) \cdot (t_{\mathsf{l}} + \mathfrak{z}_i^n t_{\mathsf{m}} + \mathfrak{z}_i^{2n} t_{\mathsf{h}}) - \mathsf{F}(\mathfrak{z}_i)]_1 \bullet [1]_2 = [\pi_i]_1 \bullet [x - \mathfrak{z}_i]_2 \ ,$$

for some number of points $\mathfrak{z}_i$ and KZG openings $[\pi_i]_1$ provided by the prover (we formalize it in the next section). Here, $[t_{\mathsf{l}}, t_{\mathsf{m}}, t_{\mathsf{h}}]_1$ are commitments to the

respective quotient polynomials. They proceed by using a new assumption called SplitRSDH (Definition 4), to show that $\mathsf{Z}_{\mathbb{H}}(X)$ divides $\mathsf{F}(X)$. This is sufficient to conclude that $\mathsf{F}(\omega_i) = 0$ for all $\omega_i \in \mathbb{H}$ holds. However, we make the observation that weak CSS of $\mathsf{LinKZG}$ is sufficient to prove the same property without relying on the SplitRSDH assumption. This allows us to reduce the security of Plonk to only the ARSDH assumption in $\mathbb{G}_T$.

### 5.2 Security of Plonk with LinKZG

There is only one lemma in the CSS proof of [LPS25] that uses the SplitRSDH assumption. The rest of the proof relies on the ARSDH assumption in $\mathbb{G}_1$ as we explained above. We first paraphrase that property as a technical assumption called Plonk Division (PDiv) assumption.

**Definition 19.** *Let $\mathbb{H}$ be a set of size $n := |\mathbb{H}| \in \mathsf{poly}(\lambda)$, $\kappa_{\mathsf{kzg}} := n + 5$ and $\kappa_{\mathfrak{z}} := 4\kappa_{\mathsf{kzg}} + 1$. We say that the $n$-Plonk Division (PDiv) assumption holds for $\mathsf{Pgen}$ if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{pdiv}}_{\mathsf{Pgen},n,\mathcal{A}}(\lambda) :=$*

$$\Pr\left[\begin{array}{l|l} \forall i \neq j : \mathfrak{z}_i \neq \mathfrak{z}_j \wedge & \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); x \leftarrow\!\!\$\, \mathbb{F}; \\ \forall i \in [1..\kappa_{\mathfrak{z}}] : \textit{Eq. (7) holds} \wedge & \mathsf{ck} \leftarrow ([(x^i)_{i=0}^{\kappa_{\mathsf{kzg}}}]_1, [1, x]_2); \\ \deg(\mathsf{F}) \leq \kappa_{\mathfrak{z}} - 1 \wedge \mathsf{Z}_{\mathbb{H}}(X) \nmid \mathsf{F}(X) & \begin{pmatrix} \mathsf{F}(X), [t_{\mathsf{l}}, t_{\mathsf{m}}, t_{\mathsf{h}}]_1, \\ (\mathfrak{z}_i, [\pi_i]_1)_{i=1}^{\kappa_{\mathfrak{z}}} \end{pmatrix} \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \approx_\lambda 0 \ ,$$

*where Eq. (7) is defined as follows:*

$$[\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) \cdot (t_{\mathsf{l}} + \mathfrak{z}_i^n t_{\mathsf{m}} + \mathfrak{z}_i^{2n} t_{\mathsf{h}}) - \mathsf{F}(\mathfrak{z}_i)]_1 \bullet [1]_2 = [\pi_i]_1 \bullet [x - \mathfrak{z}_i]_2 \ . \tag{7}$$

We now prove that the PDiv assumption follows from the evaluation binding of $\mathsf{KZG}$ and wCSS property of $\mathsf{LinKZG}$.

**Theorem 5.** *Let $\mathbb{H}$ be a set of size $n := |\mathbb{H}| \in \mathsf{poly}(\lambda)$, $\kappa_{\mathsf{kzg}} := n + 5$ and $\kappa_{\mathfrak{z}} := 4\kappa_{\mathsf{kzg}} + 1$. If $\mathsf{LinKZG}$ has $(\kappa_{\mathsf{kzg}}, 2n, 3, \kappa_{\mathfrak{z}})$-wCSS and $\mathsf{KZG}$ is evaluation binding, then the $n$-PDiv assumption holds.*

*Proof.* Let $\mathcal{A}$ be a PPT adversary that outputs $\mathsf{F}(X)$, $[t_{\mathsf{l}}]_1$, $[t_{\mathsf{m}}]_1$, $[t_{\mathsf{h}}]_1$, and $(\mathfrak{z}_i, [\pi_i]_1)_{i=1}^{\kappa_{\mathfrak{z}}}$ such that it breaks the PDiv assumption. Let $\mathcal{J}_0 := \{i \in [1..\kappa_{\mathfrak{z}}] : \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) = 0\}$ and $\mathcal{J}_1 := [1..\kappa_{\mathfrak{z}}] \setminus \mathcal{J}_0$. Notice that $|\mathcal{J}_0| \leq n$ since $\mathsf{Z}_{\mathbb{H}}(X)$ has degree $n$ and $|\mathcal{J}_1| \geq \kappa_{\mathfrak{z}} - n$. For $i \in \mathcal{J}_1$, we define $\beta_i := \mathsf{F}(\mathfrak{z}_i)/\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)$ and $[\pi_i']_1 := [\pi_i]_1/\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)$, and rewrite Eq. (7) as

$$[t_{\mathsf{l}} + \mathfrak{z}_i^n t_{\mathsf{m}} + \mathfrak{z}_i^{2n} t_{\mathsf{h}} - \beta_i]_1 = [\pi_i']_1 \bullet [x - \mathfrak{z}_i]_2 \ . \tag{8}$$

Notice that $\beta_i$ is well-defined since $\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) \neq 0$ for $i \in \mathcal{J}_1$. Furthermore, we define $L(X) := \mathsf{Inter}(\{(\mathfrak{z}_i, \beta_i)\}_{i \in \mathcal{J}_1})$.

We divide the proof into two cases.

Case 1: Suppose $\deg(L) > \kappa_{\mathsf{kzg}} + 2n$. We can use $\mathcal{A}$ to break the $(\kappa_{\mathsf{kzg}}, 2n, 3, K)$-wCSS property of $\mathsf{LinKZG}$, where $K := |\mathcal{J}_1|$. Namely, let PPT $\mathcal{B}(\mathsf{ck})$ be an adversary that receives $\mathsf{ck} = ([(x^i)_{i=1}^{\kappa_{\mathsf{kzg}}}]_1, [1, x]_2)$ as an input and forwards it to $\mathcal{A}$. Then,

$\mathcal{B}$ receives from $\mathcal{A}$ the polynomials and the KZG openings as described above. Now, $\mathcal{B}$ can simply output $g_1(X) = 1$, $g_2(X) = X^n$, $g_3(X) = X^{2n}$, $[c_1]_1 = [t_\mathsf{l}]_1$, $[c_2]_1 = [t_\mathsf{m}]_1$, $[c_3]_1 = [t_\mathsf{h}]_1$ and $(\mathfrak{z}_i, \beta_i, [\pi'_i]_1)_{i\in\mathcal{J}_1}$. Clearly, $\deg(g_j) \leq 2n$ for $j \in [1..3]$, $\mathfrak{z}_i$ are distinct, and Eq. (8) is satisfied for each $i$ as required in the wCSS property. Since additionally $\deg(L) > \kappa_\mathsf{kzg} + 2n$, the adversary $\mathcal{B}$ breaks the $(\kappa_\mathsf{kzg}, 2n, 3, K)$-wCSS property of $\mathsf{LinKZG}$.

Case 2: Suppose for some $i^* \in \mathcal{J}_0$, $\mathsf{F}(\mathfrak{z}_{i^*}) \neq 0$. Since $\mathsf{Z}_\mathbb{H}(\mathfrak{z}_{i^*}) = 0$, we have that $-\mathsf{F}(\mathfrak{z}_{i^*})[1]_1 = [\pi_{i^*}]_1 \bullet [x - \mathfrak{z}_{i^*}]_2$ .

The previous equation states that a commitment to a zero-polynomial (which is $[0]_1$) opens to a non-zero value $\mathsf{F}(\mathfrak{z}_{i^*})$ at point $\mathfrak{z}_{i^*}$. This is impossible based on the evaluation binding property of KZG. Formally, we can construct an adversary $\mathcal{C}$ that breaks the evaluation binding property of KZG by outputting $([0]_1, \mathfrak{z}_{i^*}, \mathsf{F}(\mathfrak{z}_{i^*}), [\pi_{i^*}]_1, 0, [0]_1)$. It follows that $\mathsf{F}(\mathfrak{z}_i) = 0$ for all $i \in \mathcal{J}_0$.

Now we are in the following situation. If $i \in \mathcal{J}_0$, then $\mathsf{F}(\mathfrak{z}_i) = 0 = \mathsf{Z}_\mathbb{H}(\mathfrak{z}_i) \cdot L(\mathfrak{z}_i)$. If $i \in \mathcal{J}_1$, then $\beta_i = L(\mathfrak{z}_i) = \mathsf{F}(\mathfrak{z}_i)/\mathsf{Z}_\mathbb{H}(\mathfrak{z}_i)$. Thus, also $\mathsf{F}(\mathfrak{z}_i) = \mathsf{Z}_\mathbb{H}(\mathfrak{z}_i) \cdot L(\mathfrak{z}_i)$.

Notice that $\deg(\mathsf{F}) \leq \kappa_\mathfrak{z} - 1 = 4(n+5) = 4n + 20$ and $\deg(\mathsf{Z}_\mathbb{H} \cdot L) \leq n + (n + 5 + 2n) = 4n + 5$. Since $\mathsf{F}(X)$ and $\mathsf{Z}_\mathbb{H}(X) \cdot L(X)$ agree on $\kappa_\mathfrak{z}$ distinct points, $\mathsf{F}(X) = \mathsf{Z}_\mathbb{H}(X) \cdot L(X)$, and therefore $\mathsf{Z}_\mathbb{H}(X) \mid \mathsf{F}(X)$.

Thus, we have shown that for any PPT adversary $\mathcal{A}$ there exists PPT adversaries $\mathcal{B}$ and $\mathcal{C}$ such that

$$\mathsf{Adv}^{\mathsf{pdiv}}_{\mathsf{Pgen},n,\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{wcss}}_{\mathsf{LinKZG},\kappa_\mathsf{kzg},2n,3,K,\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathrm{eb}}_{\mathsf{KZG},\kappa_\mathsf{kzg},\mathcal{C}}(\lambda).$$

□

By combining Lemma 3 and Corollary 2, we immediately obtain the following corollary.

**Corollary 2.** *Let $\mathbb{H}$ be a set of size $n := |\mathbb{H}| \in \mathsf{poly}(\lambda)$, $\kappa_\mathsf{kzg} := n + 5$ and $\kappa_\mathfrak{z} := 4\kappa_\mathsf{kzg} + 1$. If the $(\kappa_\mathsf{kzg}, 2n)$-ARSDH assumption holds in $\mathbb{G}_T$, then the $n$-PDiv assumption holds.*

The rest of the security proof in [LPS25] relies on the $(n + 5)$-ARSDH assumption in $\mathbb{G}_1$. Thus, we obtain the following result for Plonk.

**Theorem 6.** *If the $(n + 5, 2n)$-ARSDH assumption holds in $\mathbb{G}_T$, then the interactive Plonk has computational special soundness.*

If one applies the Fiat-Shamir heuristic, then by using the result of [AFK22], we obtain a knowledge sound SNARK in the ROM under the same assumption. The recent work by Lipmaa [Lip25] shows that Plonk is simulation-extractable (we refer to [Lip25] for the definition) in the ROM under falsifiable assumptions. His results uses the CSS result of [LPS25] as a black-box. Thus, by plugging in our CSS proof of Plonk, we obtain the following corollary.

**Corollary 3.** *If the $(n+5, 2n)$-ARSDH assumption holds in $\mathbb{G}_T$, then Plonk is simulation-extractable in the ROM.*

## 6 ILPCS From Homomorphic IPCS

We now show that one can construct an interactive LPCS from any interactive PCS that is homomorphic in the sense defined below.

**Definition 20 (Homomorphic PCS).** *An interactive (or non-interactive) PCS $\mathsf{PC}$ is called* homomorphic *if for any $n \in \mathsf{poly}(\lambda)$ and $\mathsf{ck} \leftarrow \mathsf{KGen}(n)$ the following properties holds:*

1. *For any $f, g \in \mathbb{F}^{\leq n}[X]$, $\mathsf{Com}(\mathsf{ck}, f+g) = \mathsf{Com}(\mathsf{ck}, f) + \mathsf{Com}(\mathsf{ck}, g)$.*
2. *For any $f \in \mathbb{F}^{\leq n}[X]$ and $c \in \mathbb{F}$, $c \cdot \mathsf{Com}(\mathsf{ck}, f) = \mathsf{Com}(\mathsf{ck}, c \cdot f)$.*

For instance, most PCSs that use either KZG or Pedersen style commitments are homomorphic [BBB+18, WTs+18, Lee21]. [4] More generally, suppose the commitment key is defined as $\mathsf{ck} = ([x_0, \ldots, x_n]_1, \mathsf{aux})$ for arbitrary trapdoors $x_i$ ($\mathsf{aux}$ models any additional information in $\mathsf{ck}$) and the commitment is defined as $\mathsf{Com}(\mathsf{ck}, f) := \sum_{i=0}^{n} f_i [x_i]_1$, where $f(X) = \sum_{i=0}^{n} f_i X^i$. In such case, it is easy to verify that the homomorphic properties hold. In the KZG-style commitments $x_i = x^i$ for some secret $x$ and in the Pedersen-style commitments $x_i$ are uniformly random field elements.

The construction of an ILPCS from a homomorphic IPCS is straightforward and follows roughly the same idea we used for $\mathsf{LinKZG}$. Briefly, the prover commits component-wise to $m$ polynomials $f_1, \ldots, f_m$ of degree $\leq n$ using the underlying IPCS. To prove that $H(X) := \sum_{i=1}^{m} g_i(\alpha) f_i(X)$ opens to $H(\alpha) = y$, the prover and verifier run the IPCS's evaluation protocol for the commitment $C' := \sum_{i=1}^{m} g_i(\alpha) C_i$, where $C_i$ commits to $f_i$. Since the underlying IPCS is homomorphic, $C'$ is a valid commitment to $H(X)$. Thus, it is is possible to execute the evaluation protocol. We present the detailed constuction in Fig. 4.

Although the construction is straightforward, proving that the resulting ILPCS has CSS is more challenging. We start by defining a slightly simpler property than CSS for ILPCS that we call Linearization Friendliness.

**Definition 21.** *Let $n, d, m, K \in \mathsf{poly}(\lambda)$. We say that an interactive (or non-interactive) homomorphic PCS $\mathsf{PC}$ is $(n, d, m, K)$-Linearization Friendly if there exists a DPT extractor $\mathsf{Ext}_{\mathsf{fr}}$ such that for any PPT adversary $\mathcal{A}$,*
$\mathsf{Adv}^{\mathrm{fr}}_{\mathsf{PC}, \mathsf{Ext}_{\mathsf{fr}}, n, d, m, K, \mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l|l} \forall i \neq j : \alpha_i \neq \alpha_j \wedge \boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m & \mathsf{ck} \leftarrow \mathsf{KGen}(n); \\ \big(\forall i \in [1..K] : \deg(H_i) \leq n \wedge & \mathcal{T}_{\mathsf{fr}} = \begin{pmatrix} \boldsymbol{g}, \boldsymbol{C} \\ (\alpha_i, H_i)_{i=1}^{K} \end{pmatrix} \leftarrow \mathcal{A}(\mathsf{ck}); \\ \mathsf{Com}(\mathsf{ck}, H_i) = \sum_{j=1}^{m} g_j(\alpha_i) C_j \big) \wedge & \boldsymbol{f} \leftarrow \mathsf{Ext}_{\mathsf{fr}}(\mathsf{ck}, \mathcal{T}_{\mathsf{fr}}) \\ \big(\exists j \in [1..m] : C_j \neq \mathsf{Com}(\mathsf{ck}, f_j) \vee & \\ \deg(f_j) > n\big) & \end{array}\right] \approx_\lambda 0.$$

We show that if the underlying IPCS has binding, computational special soundness and linearization friendliness, then the ILPCS construction in Fig. 4 has CSS.

[4] We leave it as a open question if any lattice-based PCSs have a suitable homomorphic property.

$\mathsf{ILPC.KGen}(1^\lambda)$: Output $\mathsf{ck} \leftarrow \mathsf{IPC.KGen}(1^\lambda)$.
$\mathsf{ILPC.Com}(\mathsf{ck}, \boldsymbol{f} \in (\mathbb{F}^{\leq n}[X])^m)$: Output the commitment $\boldsymbol{C}$, which is a length $m$ vector, where $C_i \leftarrow \mathsf{IPC.Com}(\mathsf{ck}, f_i)$ for $i = 1, \ldots, m$.
$\langle \mathsf{ILPC.P_{Lin}}(w), \mathsf{ILPC.V_{Lin}} \rangle(x)$:

- Is a pair of interactive algorithms with a joint input $x = (\mathsf{ck}, C, \boldsymbol{g}(\alpha), \alpha, y)$ and a private input $w = (\boldsymbol{f}, \boldsymbol{g})$ for the prover, where $\boldsymbol{C}$ is a commitment to $\boldsymbol{f}$; $\alpha, y \in \mathbb{F}$; and $\boldsymbol{g} \in (\mathbb{F}^{\leq d}[X])^m$.
- Both $\mathsf{P_{Lin}}$ and $\mathsf{V_{Lin}}$ define a commitment $C' := \sum_{i=1}^m g_i(\alpha) C_i$.
- The prover defines $H(X) \leftarrow \sum_{i=1}^m g_i(\alpha) f_i(X)$. In the honest case, $H(\alpha) = y$ and $C' = \mathsf{IPC.Com}(\mathsf{ck}, H)$.
- Both run the protocol $b \leftarrow \langle \mathsf{IPC.P}_E(H), \mathsf{IPC.V}_E \rangle(\mathsf{ck}, C', \alpha, y)$.
- The verifier outputs $b$.

**Fig. 4.** The ILPC polynomial commitment scheme constructed from an homomorphic interactive PCS $\mathsf{IPC} = (\mathsf{Pgen}, \mathsf{KGen}, \mathsf{Com}, \mathsf{P}_E, \mathsf{V}_E)$.

**Theorem 7.** *Let $n, d, m, K \in \mathsf{poly}(\lambda)$. Let $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_\mu) \in \mathbb{N}^\mu$. Let IPC be an interactive PCS that is binding, homomorphic (Definition 20), $\boldsymbol{\kappa}$-computational special sound (Definition 9), and $(n, d, m, K)$-linearization friendly (Definition 21). Then,* ILPC *constructed from IPC in Fig. 4 has $(n, d, m, K, \boldsymbol{\kappa})$-CSS (Definition 16).*

*Proof.* Let $\mathcal{A}$ be a PPT adversary that breaks the $(n, d, m, K, \boldsymbol{\kappa})$-CSS property of ILPC. Then, $\mathcal{A}(\mathsf{ck})$ outputs $\mathcal{T} = \big((\boldsymbol{g}, \boldsymbol{C}), (\alpha_i, y_i, \mathcal{T}_i)_{i=1}^K\big)$ such that $\deg(g_j) \leq d$ for $j \in [1..m]$, and $\mathcal{T}_i$ is an accepting $\boldsymbol{\kappa}$-tree for $x_i = (\mathsf{ck}, \boldsymbol{C}, \boldsymbol{g}(\alpha_i), \alpha_i, y_i)$.

Let $\mathsf{Ext_{fr}}$ be the extractor for the linearization friendliness property of IPC and $\mathsf{Ext_{ipc}}$ be the extractor for the $\boldsymbol{\kappa}$-computational special soundness of IPC. We construct an extractor $\mathsf{Ext_{ilpcs}}$ for the $(n, d, m, K, \boldsymbol{\kappa})$-CSS property of ILPC as follows. Our extractor $\mathsf{Ext_{ilpcs}}$ takes as an input $(\mathsf{ck}, \mathcal{T})$ outputted by $\mathcal{A}$. For each $i \in [1..K]$, it runs $\mathsf{Ext_{ipc}}$ on the homomorphically computed $C'_i = \sum_{j=1}^m g_j(\alpha_i) C_j$ to extract $H_i(X)$ such that $C'_i = \mathsf{IPC.Com}(\mathsf{ck}, H_i)$. If extraction of those polynomials is successful, it runs $\mathsf{Ext_{fr}}$ on $\mathcal{T}_{\mathsf{fr}} = (\boldsymbol{g}, \boldsymbol{C}, (\alpha_i, H_i)_{i=1}^K)$ to extract $f_j(X)$ for $j \in [1..m]$ such that $C_j = \mathsf{IPC.Com}(\mathsf{ck}, f_j)$. If extraction fails at any step, it outputs $\perp$. We present the details of $\mathsf{Ext_{ilpcs}}$ in Fig. 5.

There are three failure events on which $\mathsf{Ext_{ilpcs}}$ outputs $\perp$.

Case 1: On step 5 of $\mathsf{Ext_{ilpcs}}$, for some $i \in [1..K]$, it holds that $\mathsf{Com}(\mathsf{ck}, H_i) \neq C'_i$ or $\deg(H_i) > n$ or $H_i(\alpha_i) \neq y_i$. In such case, we can construct an adversary $\mathcal{B}$ that breaks the $\boldsymbol{\kappa}$-computational special soundness of IPC as follows. $\mathcal{B}$ receives $\mathsf{ck}$ as an input and runs $\mathcal{A}(\mathsf{ck})$ that output $\mathcal{T}$ as described above. Then, for all $i \in [1..K]$, $\mathcal{B}$ computes $\mathsf{Ext}(C'_i, \alpha_i, y_i, \mathcal{T}_i)$ to recover $H_i(X)$. If for some $i$ it holds that $\mathsf{Com}(\mathsf{ck}, H_i) \neq C'_i$ or $\deg(H_i) > n$ or $H_i(\alpha_i) \neq y_i$, then $\mathcal{B}$ outputs $(C'_i, \alpha_i, y_i, \mathcal{T}_i)$ and breaks the $\boldsymbol{\kappa}$-computational special soundness of IPC.

Case 2: Suppose the previous case did not happen, but on step 9 of $\mathsf{Ext_{ilpcs}}$, for some $j \in [1..m]$, it holds that $\mathsf{Com}(\mathsf{ck}, f_j) \neq C_j$. This event is bounded by the advantage of breaking linearization friendliness of IPC. More precisely, we

$\mathsf{Ext}_{\mathsf{ilpcs}}(\mathsf{ck}, \mathcal{T})$

1 : $(\boldsymbol{g}, \boldsymbol{C}), (\alpha_i, y_i, \mathcal{T}_i)_{i=1}^K \leftarrow \mathcal{T}$; // Parse the tree
2 : **for** $i \in [1..K]$ **do** // Extract $H_i(X)$ for $i \in [1..K]$
3 : $C_i' \leftarrow \sum_{j=1}^m g_j(\alpha_i) C_j$;
4 : $H_i(X) \leftarrow \mathsf{Ext}_{\mathsf{ipc}}(\mathsf{ck}, C_i', \alpha_i, y_i, \mathcal{T}_i)$;
5 : **if** $\mathsf{Com}(\mathsf{ck}, H_i) \neq C_i' \vee \deg(H_i) > n \vee H_i(\alpha_i) \neq y_i$ **then** // Breaks CSS of IPC
6 : **return** $\bot$; **endif**
7 : **endfor**
8 : $\boldsymbol{f} \leftarrow \mathsf{Ext}_{\mathsf{fr}}(\mathsf{ck}, (\boldsymbol{g}, \boldsymbol{C}, (\alpha_i, H_i)_{i=1}^K))$; // Extract $f_j(X)$ for $j \in [1..m]$
9 : **if** $\exists j \in [1..m] : \mathsf{Com}(\mathsf{ck}, f_j) \neq C_j$ **then** // Breaks linearization friendliness of IPC
10 : **return** $\bot$; **endif**
11 : **if** $\exists i \in [1..K] : H_i(X) \neq \sum_{j=1}^m g_j(\alpha_i) f_j(X)$ **then** // Breaks binding of IPC
12 : **return** $\bot$; **endif**
13 : **return** $\boldsymbol{f}$;

**Fig. 5.** Extractor $\mathsf{Ext}_{\mathsf{ilpcs}}$ for the $(n, d, m, K, \boldsymbol{\kappa})$-CSS property of ILPC. $\mathsf{Ext}_{\mathsf{ipc}}$ is IPC's CSS extractor and $\mathsf{Ext}_{\mathsf{fr}}$ is IPC's linearization friendliness extractor.

can construct an adversary $\mathcal{C}$ that breaks the linearization friendliness of IPC as follows. $\mathcal{C}$ receives $\mathsf{ck}$ as an input and runs $\mathcal{A}(\mathsf{ck})$ that outputs $\mathcal{T}$. Then, it extracts $H_i(X)$ for $i \in [1..K]$ as in $\mathsf{Ext}_{\mathsf{ilpcs}}$. Finally, it returns $\mathcal{T}_{\mathsf{fr}} = (\boldsymbol{g}, \boldsymbol{C}, (\alpha_i, H_i)_{i=1}^K)$. If for some $j \in [1..m]$, $\mathsf{Com}(\mathsf{ck}, f_j) \neq C_j$, then $\mathcal{C}$ has broken the linearization friendliness of IPC.

Case 3: Suppose neither of the previous events happened, but on step 11 of $\mathsf{Ext}_{\mathsf{ilpcs}}$, for some $i^* \in [1..K]$, it holds that $H_{i^*}(X) \neq \sum_{j=1}^m g_j(\alpha_{i^*}) f_j(X)$. This event breaks the binding property of IPC. We can construct an adversary $\mathcal{D}(\mathsf{ck})$ that runs $\mathcal{A}(\mathsf{ck})$ to obtain $\mathcal{T}$. Then, it extracts $H_i(X)$ for $i \in [1..K]$ and $\boldsymbol{f}$ as in $\mathsf{Ext}_{\mathsf{ilpcs}}$. Finally, it outputs $\left(C_{i^*}' = \sum_{j=1}^m g_j(\alpha_{i^*}) C_j, H_{i^*}(X), \sum_{i=1}^m g_j(\alpha_{i^*}) f_j(X)\right)$. Since $C_{i^*}'$ is a commitment to both $H_{i^*}(X)$ and $\sum_{j=1}^m g_j(\alpha_{i^*}) f_j(X)$, but those two polynomials are not equal, $\mathcal{D}$ breaks the binding property of IPC.

Suppose now that none of the above failure events happened. Then, we have that for all $i \in [1..m]$, $C_i = \mathsf{Com}(\mathsf{ck}, f_i)$ and $\deg(f_i) \leq n$. Furthermore, if we define $L(X) := \sum_{i=1}^m g_i(X) f_i(X)$, we have that $L(\alpha_i) = H_i(\alpha_i) = y_i$ for all $i \in [1..K]$. Thus, $\mathcal{A}$ has not broken the $(n, d, m, K, \boldsymbol{\kappa})$-CSS property of ILPC.

We have constructed an extractor $\mathsf{Ext}_{\mathsf{ilpcs}}$ such that for any PPT adversary $\mathcal{A}$, there exists PPT adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ such that

$$\mathsf{Adv}^{\mathrm{css}}_{\mathsf{ILPC},\mathsf{Ext}_{\mathsf{ilpcs}},n,d,m,K,\boldsymbol{\kappa},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{css}}_{\mathsf{IPC},\mathsf{Ext}_{\mathsf{ipc}},n,\boldsymbol{\kappa},\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathrm{fr}}_{\mathsf{IPC},\mathsf{Ext}_{\mathsf{fr}},n,d,m,K,\mathcal{C}}(\lambda) + \mathsf{Adv}^{\mathrm{bind}}_{\mathsf{IPC},\mathcal{D}}(\lambda) \ .$$

□

We have now moved from the $CSS$ property of ILPC to a slightly simpler linearization friendliness property. It still remains to show that existing homomorphic PCSs satisfy linearization friendliness. We show that linearization

friendliness holds under the extra constraint that $\boldsymbol{g}$ polynomials are linearly independent.

For the sake of the next theorem, we say that $\mathsf{PC}$ has linearization friendliness with linear independence if the winning condition of the definition of linearization friendliness (Definition 21) holds with the extra constraint that $\boldsymbol{g} = (g_1, \ldots, g_m)$ are linearly independent polynomials. Surprisingly, the following theorem is satisfied information-theoretically for any homomorphic PCS.

**Theorem 8.** *Let $n, d, m, K \in \mathsf{poly}(\lambda)$ such that $K > d + 1 > m$. If PCS $\mathsf{PC}$ is homomorphic, then $\mathsf{PC}$ satisfies linearization friendliness with linear independence.*

$\mathsf{Ext}_{\mathsf{fr}}(\mathsf{ck}, \mathcal{T}_{\mathsf{fr}})$

$(\boldsymbol{g}, \boldsymbol{C}), (\alpha_i, H_i)_{i=1}^K \leftarrow \mathcal{T}_{\mathsf{fr}}$; // Parse the tree

Define $\boldsymbol{A} = (g_j(\alpha_i))_{i,j} \in \mathbb{F}^{K \times m}$ as in Eq. (9)

Compute the left inverse $\boldsymbol{B} = (b_{i,j})_{i,j} \in \mathbb{F}^{m \times K}$ of $\boldsymbol{A}$;

**for** $i \in [1..m]$ **do** $f_i(X) \leftarrow \sum_{j=1}^K b_{i,j} H_j(X)$; **endfor**

**return** $(f_1(X), \ldots, f_m(X))$;

**Fig. 6.** Extractor $\mathsf{Ext}_{\mathsf{fr}}$ for the linearization friendliness property of $\mathsf{PC}$, when $\boldsymbol{g}$ are linearly independent polynomials.

*Proof.* Let $\mathcal{A}$ be a PPT adversary against the linearization friendliness with linear independence property of $\mathsf{PC}$. Then, $\mathcal{A}(\mathsf{ck})$ outputs $\mathcal{T}_{\mathsf{fr}} = ((\boldsymbol{g}, \boldsymbol{C}), (\alpha_i, H_i)_{i=1}^K)$ such that $\deg(g_j) \leq d$ for $j \in [1..m]$, $\deg(H_i) \leq n$ and $\mathsf{Com}(\mathsf{ck}, H_i) = \sum_{j=1}^m g_j(\alpha_i) C_j$ for $i \in [1..K]$, and $\boldsymbol{g}$ are linearly independent polynomials.

We describe an extractor $\mathsf{Ext}_{\mathsf{fr}}$, which is also presented in Fig. 6, that takes as an input $(\mathsf{ck}, \mathcal{T}_{\mathsf{fr}})$ outputted by $\mathcal{A}$ and outputs polynomials $f_j(X)$ for $j \in [1..m]$ such that $C_j = \mathsf{Com}(\mathsf{ck}, f_j)$ and $\deg(f_j) \leq n$.

Given the adversary's output, we can write the following matrix equation:

$$\underbrace{\begin{pmatrix} g_1(\alpha_1) & g_2(\alpha_1) & \cdots & g_m(\alpha_1) \\ g_1(\alpha_2) & g_2(\alpha_2) & \cdots & g_m(\alpha_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(\alpha_K) & g_2(\alpha_K) & \cdots & g_m(\alpha_K) \end{pmatrix}}_{:=\boldsymbol{A}} \boldsymbol{C}^\top = \underbrace{\begin{pmatrix} \mathsf{Com}(\mathsf{ck}, H_1) \\ \mathsf{Com}(\mathsf{ck}, H_2) \\ \vdots \\ \mathsf{Com}(\mathsf{ck}, H_K) \end{pmatrix}}_{:=\boldsymbol{C}_H}, \tag{9}$$

where $\boldsymbol{A}$ is an alternant matrix (see Section 2). Since $\boldsymbol{g}$ are linearly independent polynomials of degree $\leq d$, and $K > d+1 > m$, then $\boldsymbol{A}$ has rank $m$ by Lemma 1. Thus, we can find a left inverse $\boldsymbol{B} = (b_{i,j})_{i,j} \in \mathbb{F}^{m \times K}$ of $\boldsymbol{A}$ such that $\boldsymbol{C}^\top = \boldsymbol{B}\boldsymbol{C}_H$. Since we are considering a homomorphic PCSs, we have that

$$C_i = \boldsymbol{B}_i \boldsymbol{C}_H = \mathsf{Com}\big(\mathsf{ck}, \textstyle\sum_{j=1}^K b_{i,j} H_j\big) \ ,$$

where $\boldsymbol{B}_i$ is the $i$-th row of $\boldsymbol{B}$ and $i \in [1..m]$. We denote $f_i(X) := \sum_{j=1}^{K} b_{i,j} H_j(X)$ for all $i \in [1..m]$, thus $C_i = \mathsf{Com}\big(\mathsf{ck}, f_i(X)\big)$. Since $\deg(H_j) \leq n$, it follows that $\deg(f_i) \leq n$. □

Let us now consider Pedersen-style commitments, where $\mathsf{ck} = [x_1, \ldots, x_{n+1}]_1$ for some sampled $x_i$ (for instance, $x_i \leftarrow\!\!\$\ \mathbb{F}$) and a commitment to a polynomial $f(X) = \sum_{i=0}^{n} f_i X^i \in \mathbb{F}^{\leq n}[X]$ is defined as $\mathsf{Com}(\mathsf{ck}, f(X)) := \sum_{i=0}^{n} f_i [x_{i+1}]_1$. We show that Pedersen-style PCSs satisfy linearization friendliness (without the restriction to linear independence) in the algebraic group model.

**Theorem 9.** *If PCS $\mathsf{PC}$ is a binding Pedersen-style commitment scheme, then $\mathsf{PC}$ satisfies the linearization friendliness in the algebraic group model.*

*Proof.* Let us fix some parameters $n, d, m, K \in \mathsf{poly}(\lambda)$ as before. Suppose $\mathcal{A}$ is an arbitrary PPT algebraic adversary against the linearization friendliness property. The challenger samples a Pedersen-style $\mathsf{ck} = [x_1, \ldots, x_{n+1}]_1$ as defined above and gives it as an input to $\mathcal{A}$. Then, $\mathcal{A}$ outputs $(\boldsymbol{g}, \boldsymbol{C})$ and $(\alpha_i, H_i)_{i=1}^{K}$ such that $\deg(g_j) \leq d$ for $j \in [1..m]$, $\deg(H_i) \leq n$ for $i \in [1..K]$, and $\mathsf{Com}(\mathsf{ck}, H_i) = \sum_{j=1}^{m} g_j(\alpha_i) C_j$. Additionally, since $\mathcal{A}$ is algebraic, for each $C_j$ it outputs coefficients $f_{j,0}, \ldots, f_{j,n} \in \mathbb{F}$ such that $C_j = \sum_{i=0}^{n} f_{j,i} [x_{i+1}]_1$. Thus, by defining $f_j(X) := \sum_{i=0}^{n} f_{j,i} X^i$, we have trivially obtained a polynomial $f_j$ of degree $\leq n$ such that $C_j = \mathsf{Com}(\mathsf{ck}, f_j)$. □

Now we have established that a large class of homomorphic polynomial commitment schemes can use the same linearization technique that has been previously used in the KZG PCS. We hope this insight will be useful for obtaining more efficient SNARKs as it has been the case for the KZG-based SNARKs and other KZG-based protocols. We leave it as an open problem to prove linearization friendliness of homomorphic PCSs without requiring linear independence of $\boldsymbol{g}$ or the algebraic group model.

**Acknowledgment.** We have used AI tools to help with grammar and spell checking. The author was funded by the European Union and Estonian Research Council via grant PSG1167 and project TEM-TA119.

## References

AFK22. Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-Shamir transformation of multi-round interactive proofs. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 113–142. Springer, Cham, November 2022. doi:10.1007/978-3-031-22318-1_5. 2.2, 5.2

AFK23. Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-Shamir transformation of multi-round interactive proofs (extended version). *Journal of Cryptology*, 36(4):36, October 2023. doi:10.1007/s00145-023-09478-y. 1

BB04. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Berlin, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_4. 1, 2

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00020. 1, 1, 6

BBHR18. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018. doi:10.4230/LIPIcs.ICALP.2018.14. 1

BDH+25. Juraj Belohorec, Pavel Dvorák, Charlotte Hoffmann, Pavel Hubácek, Kristýna Masková, and Martin Pastyrík. On extractability of the KZG family of polynomial commitment schemes. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 584–616. Springer, Cham, August 2025. doi:10.1007/978-3-032-01887-8_19. 1

BFS20. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Cham, May 2020. doi:10.1007/978-3-030-45721-1_24. 1, 5.1

CFF+21. Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Cham, December 2021. doi:10.1007/978-3-030-92078-4_1. 1

CGKY25. Alessandro Chiesa, Ziyi Guan, Christian Knabenhans, and Zihan Yu. On the Fiat-Shamir security of succinct arguments from functional commitments. Cryptology ePrint Archive, Report 2025/902, 2025. URL: https://eprint.iacr.org/2025/902. 1, 3

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Cham, May 2020. doi:10.1007/978-3-030-45721-1_26. 1, 3, 4.1

DDO+01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Berlin, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_33. 1

DFGK14. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Berlin, Heidelberg, December 2014. doi:10.1007/978-3-662-45611-8_28. 4.2

DGP+19. Vanesa Daza, Alonso González, Zaira Pindado, Carla Ràfols, and Javier Silva. Shorter quadratic QA-NIZK proofs. In Dongdai Lin and Kazue Sako,

editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 314–343. Springer, Cham, April 2019. doi:10.1007/978-3-030-17253-4_11. 4.2

EG25. Liam Eagen and Ariel Gabizon. MERCURY: A multilinear polynomial commitment scheme with constant proof size and no prover FFTs. Cryptology ePrint Archive, Report 2025/385, 2025. URL: https://eprint.iacr.org/2025/385. 1

FFR24. Antonio Faonio, Dario Fiore, and Luigi Russo. Real-world universal zk-SNARKs 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. 1, 1, 4.1, 4.3

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. 1, 2.1, 5.1

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. 1

GPS25. Chaya Ganesh, Sikhar Patranabis, and Nitin Singh. Samaritan: Linear-time prover SNARK from new multilinear polynomial commitments. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part V*, volume 16249 of *LNCS*, pages 456–489. Springer, Singapore, December 2025. doi:10.1007/978-981-95-5116-3_15. 1

GR19. Alonso González and Carla Ràfols. Shorter pairing-based arguments under standard assumptions. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 728–757. Springer, Cham, December 2019. doi:10.1007/978-3-030-34618-8_25. 2.1

GW20. Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Report 2020/315, 2020. URL: https://eprint.iacr.org/2020/315. 5.1

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. 1, 2, 3, 4.1, 5.1

JM24. Joseph Jaeger and Deep Inder Mohan. Generic and algebraic computation models: When AGM proofs transfer to the GGM. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part V*, volume 14924 of *LNCS*, pages 14–45. Springer, Cham, August 2024. doi:10.1007/978-3-031-68388-6_2. 2.1

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. 1, 2.3, 7, 2.3

Lee21. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 1–34. Springer, Cham, November 2021. doi:10.1007/978-3-030-90453-1_1. 1, 6

Lip25. Helger Lipmaa. Plonk is simulation extractable in ROM under falsifiable assumptions. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025, Part IV*, volume 16271 of *LNCS*, pages 3–36. Springer, Cham, December 2025. doi:10.1007/978-3-032-12290-2_1. 1, 1, 5.2

LPS23. Helger Lipmaa, Roberto Parisella, and Janno Siim. Algebraic group model with oblivious sampling. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 363–392. Springer, Cham, November / December 2023. doi:10.1007/978-3-031-48624-1_14. 1, 1, 2.1, 4.1, 4.3

LPS24. Helger Lipmaa, Roberto Parisella, and Janno Siim. Constant-size zk-SNARKs in ROM from falsifiable assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 34–64. Springer, Cham, May 2024. doi:10.1007/978-3-031-58751-1_2. 1, 1, 3, 2.1, 2.3, 8, 2.3, 2, 4.2, 4.3, 5.1

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. 1, 1, 2.1, 4.1, 4.2, 4.3, 5.1, 5.2, 5.2, 5.2

LSZ22. Helger Lipmaa, Janno Siim, and Michal Zajac. Counting vampires: From univariate sumcheck to updatable ZK-SNARK. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 249–278. Springer, Cham, December 2022. doi:10.1007/978-3-031-22966-4_9. 1

NS24. Ngoc Khanh Nguyen and Gregor Seiler. Greyhound: Fast polynomial commitments from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 243–275. Springer, Cham, August 2024. doi:10.1007/978-3-031-68403-6_8. 1

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013. doi:10.1109/SP.2013.47. 2.1, 4.2

Rou25. Jonathan Rouach. Choosing plonk, the verified verifier. https://youtu.be/dNLa5B2ER74, 2025. YouTube video, accessed 20 Oct 2025. 1

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. doi:10.1109/SFFCS.1999.814628. 1

WTs+18. Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00060. 1, 6

WZ25. Benedikt Wagner and Arantxa Zapico. Proving the security of PeerDAS without the AGM. Cryptology ePrint Archive, Report 2025/1683, 2025. URL: https://eprint.iacr.org/2025/1683. 4.2

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. 1, 2.1