On Knowledge-Soundness of Plonk in ROM from Falsifiable Assumptions*

October 10, 2024

Helger Lipmaa [1], Roberto Parisella [2], and Janno Siim [2]

[1] University of Tartu, Tartu, Estonia
[2] Simula UiB, Bergen, Norway

Abstract. Lipmaa, Parisella, and Siim [Eurocrypt, 2024] proved the extractability of the KZG polynomial commitment scheme under the falsifiable assumption ARSDH. They also showed that variants of real-world zk-SNARKs like Plonk can be made knowledge-sound in the random oracle model (ROM) under the ARSDH assumption. However, their approach did not consider various batching optimizations, resulting in their variant of Plonk having approximately 3.5 times longer argument. Our contributions are: (1) We prove that several batch-opening protocols for KZG, used in modern zk-SNARKs, have computational special-soundness under the ARSDH assumption. (2) We prove that interactive Plonk has computational special-soundness under the ARSDH assumption and a new falsifiable assumption SplitRSDH. We also prove that two minor modifications of the interactive Plonk have computational special-soundness under only the ARSDH and a simpler variant of SplitRSDH. We define a new type-safe oracle framework of the AGMOS (AGM with oblivious sampling) and prove SplitRSDH is secure in it. The Fiat-Shamir transform can be applied to obtain non-interactive versions, which are secure in the ROM under the same assumptions.

Keywords: Batching · KZG · Plonk · special-soundness · zk-SNARKs

1 Introduction

As shown by Gentry and Wichs [GW11], the soundness of SNARGs necessarily relies on non-falsifiable assumptions. However, their result applies only in the standard model, where one does not use random oracles. In the current landscape of universal zk-SNARKs, where the trusted setup does not depend on the computation, we can observe three ways (a trichotomy) to consider Gentry-Wichs. First, the GKMMM zk-SNARK of Groth et al. [GKM+18] uses non-falsifiable knowledge assumptions but does not use random oracles. However, GKMMM has a quadratic-length Structured Reference String (SRS) and is thus only of theoretical interest. Second, many well-known zk-SNARKs like Bulletproofs [BBB+18],

* Second eprint. Substantial rewrite of the previous eprint version, with several new results, including remodeling AGMOS, a more standard security assumption, a new analysis of Lin (including the impossibility result), and the analysis of SmallPlonk.

Brakedown [GLS+23], and FRI-based [BBHR18] STARKs work in the random oracle model (ROM) and only use falsifiable assumptions. While zk-SNARKs of this category excel in other aspects (e.g., they offer excellent prover time [DP23] or a transparent SRS), they have a longer argument and higher verification costs.

Until recently, constant-argument and linear-SRS length zk-SNARKs relied on both the ROM and non-falsifiable assumptions (including an idealized group model like the Generic Group Model (GGM) [Sho97,Mau05] or the Algebraic Group Model (AGM) [FKL18]). This includes well-known updatable zk-SNARKs [GWC19,CHM+20,RZ21,CFF+21,LSZ22]. In particular, Plonk [GWC19] is widely deployed in practice due to its general efficiency and the use of the Plonkish arithmetization that allows for custom gates [GW19] and lookup arguments [GW20,EFG22,STW24,CFF+24]. Thus, this paper focuses on Plonk, though our techniques are broadly applicable.

Both the ROM [CGH98] and idealized group models [Den02] are known to be non-instantiable. Using only a single idealized model in a security proof is preferable, whether it be the ROM, the AGM, or the GGM. The ROM is more established and better understood, making it desirable to avoid idealized group models as much as possible. In addition, the understanding of the AGM is lacking, with several recent papers finding bugs and proposing fixes [Zha22,ZZK22,LPS23,BFHK23,JM24]. Using the terminology of Jaeger and Mohan [JM24], most of the found bugs were illusory or ambiguous: they relied on one possible interpretation of AGM and disappeared in a different interpretation of AGM. Following Zhandry [Zha22], Jaeger and Mohan proposed a coherent interpretation of the AGM, in particular, proving that under reasonable assumptions, AGM security implies GGM security, [JM24].

Separately, Lipmaa et al. [LPS23] proposed AGM with Oblivious Sampling (AGMOS), a more realistic variant of the AGM where the adversary has an additional capacity to sample group elements obliviously.[3] They noted that a common use of the seminal KZG polynomial commitment scheme [KZG10], secure in the AGM, is not secure in the AGMOS and thus not in the standard model. While [JM24]'s framework for AGM is more rigorous, they also do not consider oblivious sampling, which can significantly affect the security of KZG-based zk-SNARKs. One also faces the tedious but necessary job of rewriting many existing AGM proofs. Only a few AGMOS proofs exist (e.g., for Plonk [FFR24] and Polymath [Lip24]); the security of other KZG-based zk-SNARKs in the AGMOS is still unproven. Because idealized group models frequently change, using them as minimally as possible is desirable. For example, such models are a valuable tool for sanity-checking new standard-looking falsifiable assumptions, which has been their primary (and significantly less controversial) use case in the past.

The trichotomic situation changed in 2024 when Lipmaa et al. (LPS, [LPS24]) proved that the KZG polynomial commitment scheme [KZG10] is special-sound under a new falsifiable assumption ARSDH (Adaptive Rational

[3] Oblivious sampling means sampling a group element without knowing the respective discrete logarithm. In elliptic curve groups, it is possible with hash-and-increment or admissible encodings [Ica09,LPS23].

Table 1. Comparison of different versions of Plonk, secure under falsifiable assumptions. The number of bits are given for the BLS381-12 pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Additionally, n denotes the number of gates, and ℓ is the number of elements in the public input of the circuit whose satisfiability is being proven.

Argument	Proof size (bits)	Prover	Verifier	Assumptions
Plonk's polynomial IOP compiled with [LPS24]	$23\|\mathbb{F}\| + 30\|\mathbb{G}_1\|$ (17408)	$30n$ $\mathbb{G}_1$ Exp., $\mathcal{O}(n \log n)$ $\mathbb{F}$ Ops.	46 Pair., 24 $\mathbb{G}_1$ Exp., $\mathcal{O}(\ell + \log n)$ $\mathbb{F}$ Ops.	ARSDH
SanPlonk (current work)	$7\|\mathbb{F}\| + 9\|\mathbb{G}_1\|$ (5248)	$9n$ $\mathbb{G}_1$ Exp., $\mathcal{O}(n \log n)$ $\mathbb{F}$ Ops.	2 Pair., 19 $\mathbb{G}_1$ Exp., $\mathcal{O}(\ell + \log n)$ $\mathbb{F}$ Ops.	ARSDH
Plonk (current work)	$6\|\mathbb{F}\| + 9\|\mathbb{G}_1\|$ (4992)	$9n$ $\mathbb{G}_1$ Exp., $\mathcal{O}(n \log n)$ $\mathbb{F}$ Ops.	2 Pair., 18 $\mathbb{G}_1$ Exp., $\mathcal{O}(\ell + \log n)$ $\mathbb{F}$ Ops.	ARSDH, SplitRSDH
SmallPlonk (current work)	$6\|\mathbb{F}\| + 7\|\mathbb{G}_1\|$ (4224)	$11n$ $\mathbb{G}_1$ Exp., $\mathcal{O}(n \log n)$ $\mathbb{F}$ Ops.	2 Pair., 16 $\mathbb{G}_1$ Exp., $\mathcal{O}(\ell + \log n)$ $\mathbb{F}$ Ops.	ARSDH, SplitRSDH

Strong Diffie-Hellman assumption). Importantly, they only use AGM(OS) to build confidence that ARSDH is secure. LPS proposed a compiler that uses KZG to convert an efficient polynomial IOP (PIOP, [BFS20]) like Plonk's to a constant-communication zk-SNARK. Since Plonk and the zk-SNARKs [GWC19,CHM+20,RZ21,CFF+21,LSZ22] combine polynomial IOPs and KZG, the LPS compiler results in zk-SNARKs that have both constant-length argument and linear-length SRS. Importantly, the security of the resulting zk-SNARK relies only on the ROM and the ARSDH assumption.

Unfortunately, the LPS compiler has serious drawbacks. First, it works only with abstract PIOPs, meaning one cannot operate on polynomial commitments without opening them. Thus, one cannot use batching techniques like the classical "linearization trick" [GWC19,CHM+20]. (We will explain this trick in Sections 1.1 and 4.2.) Plonk, being highly optimized, uses batching techniques very aggressively. One reason why [LPS24] does not handle batching techniques is that [LPS23] observed that the linearization trick is sometimes not secure in the AGMOS. Very recently, Faonio et al. [FFR24] defined a precise condition under which the linearization trick is secure in the AGMOS; they used this to prove that Plonk is knowledge-sound in the AGMOS. Their proof techniques heavily rely on the properties of the AGMOS and seem not to work in the standard model. It is possible that the interactive Plonk is insecure in the standard model.

The LPS compiler introduces even more overhead: compared to the original PIOP, the prover has to open each polynomial commitment at a new random point. Put together, [LPS24] proves that a *variant* of (interactive) Plonk is knowledge-sound under the ARSDH assumption. However, the variant is much less efficient than the "real-world" Plonk. The most striking is the verifier's slowdown, who needs to implement 46 pairings instead of 2. See Table 1 for a com-

parison. We believe practitioners are unwilling to sacrifice efficiency by such a factor or change their battle-tested codebases.

An additional weakness of [LPS24] is that similarly to [GWC19], it proves *knowledge-soundness* of the interactive Plonk. As well-known [AFK22], knowledge-sound interactive protocols can suffer a significant security loss after Fiat-Shamir compilation to zk-SNARKs. Practitioners often assume tight security for Fiat-Shamir zk-SNARKs in their implementations. Given that [GWC19,LPS24] prove that the interactive Plonk is knowledge-sound, this holds only heuristically. Achieving tight security with Fiat-Shamir requires more robust soundness notions, such as round-by-round soundness, state-restoration soundness, or special soundness.

To sum it up, Plonk is known to be knowledge-sound, assuming both the ROM and the AGMOS [FFR24]. Lipmaa et al. [LPS24] proved that a variant of Plonk is knowledge-sound in the ROM under falsifiable assumptions. However, their variant of Plonk is too inefficient to be used in practice. Given the importance of Plonk in industry, this situation is highly non-satisfactory. A proof that works in the ROM and the AGM(OS) but not in the ROM under falsifiable assumptions leaves it open to unknown problems of the AGM(OS).

Finally, Plonk's paper [GWC19] has been modified repeatedly to correct bugs and improve efficiency. A knowledge-soundness proof of Plonk under reasonable falsifiable assumptions in the ROM can be seen as an independent security audit, showing that the current variant of Plonk is secure and does not have to be changed anymore (except to improve on efficiency). It also ascertains Plonk will stay secure even if more problems are found in the AGM(OS) or GGM, as long as one does not break the concrete falsifiable assumptions.

Our Contributions. We prove that interactive Plonk has computational special-soundness under falsifiable assumptions ARSDH [LPS24] and SplitRSDH. While SplitRSDH is novel, it is not too different from ARSDH. After applying the Fiat-Shamir transformation, Plonk is tightly [AFK22,DG23,AFKR23] knowledge-sound in the ROM under the same assumptions. Towards this end, we first study KZG batching protocols that are not covered by the LPS compiler. We prove the well-known KZG batching protocol (Batch) has computational special-soundness under the ARSDH assumption. We prove that the linearization trick protocol (Lin) never has computational special-soundness in the standard model. ([FFR24] showed that Lin is secure in the AGMOS in many situations, relevant in practice.) Since Plonk uses the linearization trick, this seems to contradict our main positive result about Plonk. To overcome this, we introduce two new techniques, sanitization and Rhino (reduction to a hard assumption if not polynomial). By using sanitization, we show that slightly less efficient "sanitized" versions SanLin and SanPlonk of the linearization trick and Plonk are special-sound under the ARSDH assumption alone. By using Rhino, we show that the "real-world" Plonk is special-sound under ARSDH and SplitRSDH, and its well-known variant SmallPlonk (also described in [GWC19]) is special-sound under ARSDH alone.

As a contribution of independent interest, we propose a rigorous framework to define AGMOS so that the reductions in the AGMOS can be lifted to reductions in the GGM. This framework relies heavily on [JM24]. We show that SplitRSDH is secure in the resulting framework of AGMOS.

1.1 Our Techniques And Results

We use the bracket notation for additive groups, with say $[f]_1 = f \cdot [1]_1 \in \mathbb{G}_1$, see Section 2. KZG's [KZG10] commitment to a polynomial $\mathsf{f}(X)$ is $[f]_1 = [\mathsf{f}(x)]_1$, and its opening proof of $[f]_1$ at point $\mathfrak{z}$ is $[t]_1 = [\mathsf{t}(x)]_1$, where x is a trapdoor. Crucially, $\mathsf{t}(X) = (\mathsf{f}(X) - \mathsf{f}(\mathfrak{z}))/(X - \mathfrak{z})$ is a polynomial iff he prover is honest.

Differently from previous work on Plonk and zk-SNARKs in general ([LPS24] proved computational special-soundness only for KZG, but not for their interactive argument), we will prove computational $\boldsymbol{\kappa}$-special-soundness for the arguments we consider. (See Section 2.2 for the definition of special-soundness and related notions like transcript trees.)

Case 1: Batch. To demonstrate related proof techniques, we start by proving that the standard KZG batching protocol Batch (see Section 4.1) is computational $(n+1, m)$-special-sound under the assumption that KZG is computational $(n+1)$-special-sound. Here, m is the batching factor. Like all subsequent proofs, Batch's proof consists of an information-theoretical tree extraction lemma (Lemma 1) and the main theorem (Theorem 2). In the lemma, we construct a tree extractor that, given as an input an accepting transcript tree $\mathcal{T}$, outputs a tuple $\mathbf{k.tr}$ of m accepting KZG transcripts $([f_t]_1, \mathfrak{z}_i, \ldots)$ for $t \in [1, m]$ and $i \in [1, n+1]$, where $[f_t]_1$ are polynomial commitments to be batch-opened and $\mathfrak{z}_i$ are mutually different opening points. The lemma uses standard linear algebraic techniques and relies on the fact that KZG is homomorphic. In Theorem 2, we call Lemma 1 to obtain the accepting tuple $\mathbf{k.tr}$. Given a promised $(n+1)$-special-soundness extractor of KZG and an $(n+1, m)$-special-soundness adversary for Batch, we construct an $(n+1, m)$-special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}$ for Batch and an $(n+1)$-special-soundness adversary $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ for KZG, such that either $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}$ succeeds in extracting the valid witness for Batch (thus Batch is special-sound) or $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ succeeds in breaking the $(n+1)$-special-soundness of KZG.

The rest of the proofs use a similar structure and will only describe their additional complications compared to Batch.

Case 2: Lin. We next consider the linearization trick Lin [GWC19,CHM+20] as formalized b [FFR24]. Assume that $\mathbf{a}(X) = (\mathsf{a}_\jmath(X))_{\jmath=1}^{n_a}$ and $\mathbf{d}(X) = (\mathsf{d}_t(X))_{t=1}^{n_b}$ are committed polynomials and $(\mathsf{g}_t(\boldsymbol{X}))_{t=1}^{n_b}$ are publicly known ("core") polynomials. The goal of the Lin prover is to prove that $\sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}(X))\mathsf{d}_t(X) = 0$; for example, $\mathsf{a}_1(X)\mathsf{d}_1(X) + (-1) \cdot \mathsf{d}_2(X)$ in the case of R1CS. In Lin, the verifier sends a random point $\mathfrak{z}$. The prover batch-opens $n_a + 1$ polynomials at $\mathfrak{z}$: (1) each $\mathsf{a}_\jmath(X)$ to $\bar{a}_\jmath := \mathsf{a}_\jmath(\mathfrak{z})$, and (2) $\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}})\mathsf{d}_t(X)$ to 0. Lin's communication is much better than that of Batch (see Table 2), and thus it and its variants are widely used in contemporary zk-SNARKS like Plonk.

Lin is well-known to be knowledge-sound in the AGM, independently of the choice of the core polynomials $\mathsf{g}_t(\boldsymbol{X})$. Lipmaa et al. [LPS23] described a simple attack, showing that Lin is not always knowledge-sound in the AGMOS. Their attack was based on the fact that the AGMOS adversary can do oblivious sampling. This result raised the question of the security of all zk-SNARKs that use Lin and have security proof in the AGM (or GGM) without oblivious sampling. Faonio et al. [FFR24] solved this issue by showing that Lin is computationally special-sound iff the public core polynomials are linearly independent. They then showed that the corresponding polynomials in Plonk are linearly independent and thus proved that Plonk is knowledge-sound in the AGMOS.

Adapting an idea of [GKP22], we prove that if DL is hard, Lin does not have computational special-soundness. Thus, the result of [FFR24] does not hold in the standard model. While our proof is for a simple choice of linearly independent core polynomials $\mathsf{g}_t(\boldsymbol{X})$, it can be generalized to *any* core polynomials. Hence, [FFR24]'s knowledge-soundness proof of Plonk cannot be used in the standard model since it relies on knowledge-soundness of some instance of Lin.

We introduce a new proof technique *sanitization* to solve this issue. We note that the AGMOS attacks of [LPS23,FFR24] are possible since one can obliviously sample the polynomials $\mathsf{d}_t(X)$ from some joint distribution. To protect against this, we ask the prover of SanLin (sanitized Lin, see Section 4.3) to batch-open all $\mathsf{d}_t(X)$ at a single location. SanLin is a minimal modification of Lin, with the prover's computation having just a single new field element. We then prove SanLin is computational special-sound (see Theorem 4) assuming that KZG has computational special-soundness and evaluation-binding, that is, under ARSDH.

Case 3: Plonk. Next, we refer always to *interactive* Plonk and its variants. Plonk is knowledge-sound in the AGM [GWC19]. Using Lin's special-soundness in the AGMOS, [FFR24] recently proved Plonk is knowledge-sound in the AGMOS. However, since (under the DL assumption) Lin is not special-sound in the standard model, their proof for Plonk does also not work in the standard model. To solve this issue, we propose three different solutions.

Case 3a: SanPlonk. When we replace Lin's use in Plonk with SanLin, we obtain SanPlonk, a variant of Plonk (see Table 1) with a communication of one additional field element. Since SanLin is computationally special-sound, assuming KZG is special-sound and evaluation-binding, so is SanPlonk. (The combined full computational special-soundness proof of our three Plonk variants is ten pages; we refer to Section 6 for details.) Since we use SanLin, this proof relies on our proof technique of sanitization.

Case 3b: Plonk. Since it is unlikely practitioners want to change their implementations, we also prove the actual real-world Plonk (as in the current version of [GWC19]) has computational special-soundness. This proof relies on Rhino (reduction to a hard assumption if not polynomial), a different proof technique. We use the same notation as in Plonk's paper [GWC19] to simplify reading. To explain this proof, we have to go into more detail about how Plonk works (see Section 6.2) and also explain some philosophy of KZG versus AGM.

In the security proof, we reach the following point. The Plonk's prover has sent a few polynomial commitments like $[a, b, c]_1$ and a few other group elements like $[t_{lo}, t_{mid}, t_{hi}]_1$. For the random opening point $\mathfrak{z}$ chosen by the verifier, the latter group elements satisfy $[t]_1 = [t_{lo} + \mathfrak{z}^n t_{mid} + \mathfrak{z}^{2n} t_{hi}]_1$ for some n. Here, $[t]_1$ is a KZG's opening proof. Now, since $[t_{lo}, t_{mid}, t_{hi}]_1$ are group elements, in the AGM proof, one can extract polynomials $\mathsf{t}_{\mathsf{lo}}(X)$, $\mathsf{t}_{\mathsf{mid}}(X)$, and $\mathsf{t}_{\mathsf{hi}}(X)$ from $[t_{lo}, t_{mid}, t_{hi}]_1$. One can then compute the polynomial $\mathsf{t}(X) = \mathsf{t}_{\mathsf{lo}}(X) + \mathfrak{z}^n \mathsf{t}_{\mathsf{mid}}(X) + \mathfrak{z}^{2n} \mathsf{t}_{\mathsf{hi}}(X)$ corresponding to the group element $[t]_1$.

Unfortunately, there is insufficient information in the standard model proof to extract $\mathsf{t}(X)$. In our security proof of Plonk (see Theorem 6), we thus do not extract $\mathsf{t}(X)$ from $[t_{lo}, t_{mid}, t_{hi}]_1$. Instead, we compute a candidate *rational function* $\mathsf{t}(X)$ from other polynomials extracted in the security proof thus far. If $\mathsf{t}(X)$ is a polynomial, then it is a valid opening of $[t]_1$, and then we can use it to finish the special-soundness proof of Plonk. If $\mathsf{t}(X)$ is *not* a polynomial, we break a novel but standard-looking assumption SplitRSDH (Split RSDH, see Definition 4). (Thus, the name *Rhino* for the proof technique.) This completes the computational special-soundness proof of Plonk under KZG's computational special-soundness and evaluation-binding and SplitRSDH.

Case 3c: SmallPlonk. We also consider SmallPlonk, a well-known variant of Plonk where $[t]_1$ is not split into $[t_{lo}, t_{mid}, t_{hi}]_1$. Thus, SmallPlonk has a shorter argument than Plonk (see Table 1). Using Rhino again, we prove that SmallPlonk has computational special-soundness under KZG's computational special-soundness and evaluation-binding and SplitRSDH. However, we have a simpler variant of SplitRSDH, and the proof itself is slightly simpler.

On the New Assumption. SplitRSDH is a novel but falsifiable and standard-looking assumption (see Definition 4). We prove that SplitRSDH is secure in the AGMOS [LPS23], probably the most realistic known variant of AGM. This proof can be lifted to the GGM, [JM24]. To show that the latter result holds, we describe AGMOS by using the oracle framework of AGM from [JM24] but adding extra oracles for oblivious sampling as was done in [LPS23]. (See Section 3.) Thus, the lifting result of [JM24] holds for the AGMOS. The new AGMOS framework is simpler to use and understand than the one in [LPS23]. This forms an independent contribution of the current paper.

We emphasize that Plonk does not rely on AGMOS or other ideal models for groups. The use of AGMOS to prove the security of SplitRSDH should be seen as an independent element of cryptanalysis for the new assumption.

Generalizations. Our impossibility proof for Lin shows that the special-soundness of zk-SNARKs cannot be solely based on the special-soundness of Lin. Our two techniques, sanitization, and Rhino, can also be used in other zk-SNARKs to overcome this issue. We leave it to the future work.

Fiat-Shamir. We proved that *interactive* Plonk, SanPlonk, and SmallPlonk have computational $\boldsymbol{\kappa}$-special-soundness (for a slightly different $\boldsymbol{\kappa}$) under falsifiable assumptions. Under the same assumptions, the Fiat-Shamir transformation can be applied to obtain a zk-SNARK that is knowledge-sound in the ROM. The tight-

ness of Fiat-Shamir when applied to computationally $\boldsymbol{\kappa}$-special-sound arguments was analyzed in [DG23,AFKR23]. The tightness of Fiat-Shamir is significantly better for special-sound interactive arguments than for knowledge-sound interactive arguments. Notably, this results in tighter security in the ROM, compared to previous proofs, which had less tight security and relied on both ROM and AGM/AGMOS. See Appendix D.6 for a brief discussion.

Zero-Knowledge. Up to now, we have ignored the issue of zero knowledge. Since sanitization implies batch-opening certain polynomials at one extra point, one sometimes has to add another randomizer to one of these polynomials to obtain zero knowledge. Since the needed change is usually straightforward (instead, the major innovation of the current work is in the analysis of special-soundness), we ignore the issue everywhere except for SanPlonk. SanPlonk's description includes a new randomizer. In Appendix E, we prove that SanPlonk has zero knowledge. Zero-knowledge of Plonk (and SmallPlonk) was proven in [Sef24].

2 Preliminaries

Let λ denote the security parameter. By $f(\lambda) \approx_\lambda 0$, we mean that f is a negligible function. PPT (resp. DPT) stands for probabilistic (resp. deterministic) polynomial time. We denote the concatenation of vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ as $\boldsymbol{u}\|\boldsymbol{v}$. $\mathbb{F}$ is a finite field of prime order p. $\mathbb{F}[X]$ is the polynomial ring in variable X over the field $\mathbb{F}$ and $\mathbb{F}_{\le n}[X] \subset \mathbb{F}[X]$ is the set of polynomials of at most degree n. We denote $[a,b] := \{a, a+1, \ldots, b\}$, where $a \le b$ are integers. Our notation is inspired by Plonk [GWC19] (for example, we denote polynomials by using SansSerif), but we do not follow it universally. For any set $\mathcal{S}$, $\mathsf{Z}_{\mathcal{S}}(X) = \prod_{s\in\mathcal{S}}(X-s)$ denotes the vanishing polynomial over a set $\mathcal{S}$.

Bilinear Groups. A bilinear group generator $\mathsf{Pgen}(1^\lambda)$ returns $\mathsf{p} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are additive cyclic (thus, abelian) groups of prime order p, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate efficiently computable bilinear pairing, and $[1]_\iota$ is a fixed generator of $\mathbb{G}_\iota$. While $[1]_\iota$ is part of p, we often give it as an explicit input to different algorithms for clarity. The bilinear pairing is of Type-3, that is, there is no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. We use the common bracket notation, that is, for $\iota \in \{1,2,T\}$ and $a \in \mathbb{Z}_p$, we write $[a]_\iota$ to denote $a[1]_\iota$. We denote $\hat{e}([a]_1, [b]_2)$ by $[a]_1 \bullet [b]_2$ and assume $[1]_T = [1]_1 \bullet [1]_2$. Thus, $[a]_1 \bullet [b]_2 = [ab]_T$ for any $a, b \in \mathbb{F}$, where $\mathbb{F} = \mathbb{Z}_p$.

We recall the following falsifiable assumption ARSDH [LPS24].

Definition 1. *The* $(n+1)$-ARSDH (Adaptive Rational Strong Diffie-Hellman) *assumption holds for* Pgen *in* $\mathbb{G}_1$ *if for any PPT* $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{arsdh}}_{\mathsf{Pgen},n,\mathbb{G}_1,\mathcal{A}}(\lambda) :=$

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

where $\mathsf{Z}_{\mathcal{S}}(X) := \prod_{s\in\mathcal{S}}(X-s)$. *(The condition* $[\varphi]_1 = [\frac{g}{\mathsf{Z}_{\mathcal{S}}(x)}]_1$ *is equivalent to* $[g]_1 \bullet [1]_2 = [\varphi]_1 \bullet [\mathsf{Z}_{\mathcal{S}}(x)]_2$*.)*

2.1 Polynomial Commitment Schemes

In a (univariate) polynomial commitment scheme (PCS, [KZG10]), the prover commits to a polynomial $\mathsf{f} \in \mathbb{F}_{\leq n}[X]$ and later opens it to $\mathsf{f}(\mathfrak{z})$ for $\mathfrak{z} \in \mathbb{F}$ chosen by the verifier. A *non-interactive* polynomial commitment scheme [KZG10] consists of the following algorithms:

Setup $\mathsf{Pgen}(1^\lambda) \mapsto \mathsf{p}$**:** Given 1^λ, return system parameters p.

Commitment key generation $\mathsf{KGen}(\mathsf{p}, n) \mapsto (\mathsf{ck}, \mathsf{tk})$**:** Given a system parameter p and an upperbound n on the polynomial degree, return $(\mathsf{ck}, \mathsf{tk})$, where ck is the commitment key and tk is the trapdoor. We assume ck implicitly contains p. In the current paper, we do not use the trapdoor.

Commitment $\mathsf{Com}(\mathsf{ck}, \mathsf{f}) \mapsto C$**:** Given a commitment key ck and a polynomial $\mathsf{f} \in \mathbb{F}_{\leq n}[X]$, return a commitment C to f.

Opening $\mathsf{Open}(\mathsf{ck}, C, \mathfrak{z}, \mathsf{f}) \mapsto (\bar{f}, \pi)$**:** Given a commitment key ck, a commitment C, an evaluation point $\mathfrak{z} \in \mathbb{F}$, and a polynomial $\mathsf{f} \in \mathbb{F}_{\leq n}[X]$, return $(\bar{f}, \pi)$, where $\bar{f} \leftarrow \mathsf{f}(\mathfrak{z})$ and π is an opening proof.

Verification $\mathsf{V}(\mathsf{ck}, C, \mathfrak{z}, \bar{f}, \pi) \mapsto \{0, 1\}$**:** Given a commitment key ck, a commitment C, an evaluation point $\mathfrak{z}$, a purported evaluation $\bar{f} \stackrel{?}{=} \mathsf{f}(\mathfrak{z})$, and an opening proof π, return 1 (accept) or 0 (reject).

The KZG commitment scheme is a well-known non-interactive PCS [KZG10]; another such scheme is PST (multilinear KZG) [PST13]. Many PCSs have either an interactive opening or verification phase [BBHR18,BBB+18,BFS20].

A non-interactive PCS PC is *complete*, if for any λ, $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $n \in \mathsf{poly}(\lambda)$, $\mathfrak{z} \in \mathbb{F}$, $\mathsf{f} \in \mathbb{F}_{\leq n}[X]$,

$$\Pr\left[\mathsf{V}(\mathsf{ck}, C, \mathfrak{z}, \bar{f}, \pi) = 1 \;\middle|\; \begin{array}{l}(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); C \leftarrow \mathsf{Com}(\mathsf{ck}, \mathsf{f}); \\ (\bar{f}, \pi) \leftarrow \mathsf{Open}(\mathsf{ck}, C, \mathfrak{z}, \mathsf{f})\end{array}\right] = 1.$$

A non-interactive PCS PC is *binding*, if for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{bind}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l}C = \mathsf{Com}(\mathsf{ck}, \mathsf{f}) = \mathsf{Com}(\mathsf{ck}, \mathsf{g}) \wedge \\ \mathsf{f} \neq \mathsf{g} \wedge \deg(\mathsf{f}) \leq n, \deg(\mathsf{g}) \leq n\end{array}\;\middle|\; \begin{array}{l}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); (\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}, n); \\ (C, \mathsf{f}, \mathsf{g}) \leftarrow \mathcal{A}(\mathsf{ck})\end{array}\right] \approx_\lambda 0 \ .$$

A PCS is evaluation-binding [KZG10] if it is hard to open the same evaluation point to different evaluations: PC is *evaluation-binding* for Pgen, if for any $n \in \mathsf{poly}(\lambda)$, and PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{evb}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A}}(\lambda) :=$

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

Evaluation-binding implies binding. Really, suppose $\mathcal{A}_{\mathsf{bind}}$ succeeded in breaking binding, outputting $([c]_1, f(X), f'(X))$ such that $c = f(x) = f'(x)$ and $\mathsf{f}(X) \neq f'(X)$. Then, we can find a point $\mathfrak{z}$, such that $f(\mathfrak{z}) \neq f'(\mathfrak{z})$, open $[c]_1$ at $f(\alpha)$, and $f'(\alpha)$, and break evaluation-binding.

We rely on the following terminology from [LPS24]. We call $\mathsf{tr} = (C, \mathfrak{z}, \bar{f}, \pi)$ a *transcript* of the PCS. We say that a commitment key ck and a transcript

tr is *accepting* when $\mathsf{V}(\mathsf{ck},\mathsf{tr})=1$. For any $n\geq 1$ and any commitment key ck outputted by $\mathsf{KGen}(\mathsf{p},n)$, we define the following relations.

$$\begin{aligned}\mathcal{R}_{\mathsf{ck}} &:= \{(C,\mathsf{f}) : C=\mathsf{PC.Com}(\mathsf{ck},\mathsf{f}) \wedge \deg(\mathsf{f})\leq n\}\ ,\\ \mathcal{R}_{\mathsf{ck},\mathbf{tr}} &:= \{(C,\mathsf{f}) : (C,\mathsf{f})\in\mathcal{R}_{\mathsf{ck}} \wedge \forall j\in[0,n].\mathsf{f}(\mathfrak{z}_j)=\bar{f}_j\}\ ,\end{aligned} \tag{1}$$

where $\mathbf{tr}=(\mathsf{tr}_0,\ldots,\mathsf{tr}_n)$ contains $n+1$ accepting transcripts $\mathsf{tr}_j=(C,\mathfrak{z}_j,\bar{f}_j,\pi_j)$ such that C is the same in all transcripts, but $\mathfrak{z}_j$-s are pairwise distinct.

Let $n\in\mathsf{poly}(\lambda)$ with $n\geq 1$. A non-interactive polynomial commitment scheme PC is *computationally* $(n+1)$*-special-sound* for Pgen, if there exists a DPT extractor $\mathsf{Ext}_{\mathsf{ss}}$, such that for any PPT adversary $\mathcal{A}_{\mathsf{ss}}$, $\mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext}_{\mathsf{ss}},n,\mathcal{A}_{\mathsf{ss}}}(\lambda) :=$

$$\Pr\left[\begin{array}{l|l}\mathbf{tr}=(\mathsf{tr}_j)_{j=0}^n \wedge & \mathsf{p}\leftarrow\mathsf{Pgen}(1^\lambda);\\ \forall j\in[0,n].\begin{pmatrix}\mathsf{tr}_j=(C,\mathfrak{z}_j,\bar{f}_j,\pi_j)\\ \wedge\,\mathsf{V}(\mathsf{ck},\mathsf{tr}_j)=1\end{pmatrix} & (\mathsf{ck},\mathsf{tk})\leftarrow\mathsf{KGen}(\mathsf{p},n);\\ \wedge\ (\forall i\neq j.\mathfrak{z}_i\neq\mathfrak{z}_j)\ \wedge\ (C,\mathsf{f})\notin\mathcal{R}_{\mathsf{ck},\mathbf{tr}} & \mathbf{tr}\leftarrow\mathcal{A}_{\mathsf{ss}}(\mathsf{ck});\\ & \mathsf{f}\leftarrow\mathsf{Ext}_{\mathsf{ss}}(\mathsf{ck},\mathbf{tr})\end{array}\right]\approx_\lambda 0\ .$$

The KZG [KZG10] polynomial commitment scheme is defined as follows:

$\mathsf{KZG.Pgen}(\lambda)$**:** return $\mathsf{p}\leftarrow\mathsf{Pgen}(1^\lambda)$.

$\mathsf{KZG.KGen}(\mathsf{p},n)$**:** $\mathsf{tk}=x\leftarrow_{\$}\mathbb{Z}_p^*$; $\mathsf{ck}\leftarrow(\mathsf{p},[(x^i)_{i=0}^n]_1,[1,x]_2)$; return $(\mathsf{ck},\mathsf{tk})$.

$\mathsf{KZG.Com}(\mathsf{ck},\mathsf{f})$**:** return $C\leftarrow[\mathsf{f}(x)]_1=\sum_{j=0}^n\mathsf{f}_j[x^j]_1$.

$\mathsf{KZG.Open}(\mathsf{ck},C,\mathfrak{z},\mathsf{f})$**:** $\bar{f}\leftarrow\mathsf{f}(\mathfrak{z})$; $\varphi(X)\leftarrow(\mathsf{f}(X)-\bar{f})/(X-\mathfrak{z})$; $\pi\leftarrow[\varphi(x)]_1$; return $(\bar{f},\pi)$.

$\mathsf{KZG.V}(\mathsf{ck},C,\mathfrak{z},\bar{f},\pi)$**:** Return 1 iff $(C-\bar{f}[1]_1)\bullet[1]_2=\pi\bullet[x-\mathfrak{z}]_2$.

KZG is evaluation-binding under the n-SDH assumption [KZG10] and non-black-box extractable in the AGM [FKL18] under the PDL assumption [Lip12,CHM+20] and in AGMOS [LPS23] under the PDL and TOFR assumptions. All of these are falsifiable assumptions. We refer to the respective papers for the definition of the assumptions. Lipmaa et al. [LPS24] proved the following result.

Theorem 1. *If the* $(n+1)$*-ARSDH assumption holds, then KZG for degree* $\leq n$ *polynomials is computationally* $(n+1)$*-special-sound: There exists a DPT KZG special-soundness extractor* $\mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}$*, such that for any PPT* $\mathcal{A}_{\mathsf{ss}}$*, there exists a PPT* $\mathcal{B}$*, such that* $\mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen},\mathsf{PC},\mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}},n,\mathcal{A}_{\mathsf{ss}}}(\lambda)\leq\mathsf{Adv}^{\mathrm{arsdh}}_{\mathsf{Pgen},n,\mathbb{G}_1,\mathcal{B}}(\lambda)$.

We say that a non-interactive polynomial commitment scheme is *triply homomorphic*, if: if $(C_j,\mathfrak{z},\bar{f}_j,\pi_j)$ is an accepting transcript for every j, then so is $(\sum s_jC_j,\mathfrak{z},\sum s_j\bar{f}_j,\sum s_j\pi_j)$ for any s_j. Clearly, KZG is triply homomorphic.

2.2 Interactive Arguments

Let $\mathcal{R}\subseteq\{0,1\}^*\times\{0,1\}^*\times\{0,1\}^*$ be a ternary relation. $\mathcal{R}$ contains triples $(\mathtt{srs},\mathbb{x},\mathbb{w})\in\mathcal{R}$ where $\mathtt{srs}$ is a public common reference string, $\mathbb{x}$ is a public statement, and $\mathbb{w}$ is a private witness. We only consider NP*-relations* relations $\mathcal{R}$, where the validity of a witness $\mathbb{w}$ can be verified in time polynomial in $|\mathbb{x}|+|\mathtt{srs}|$.

Let $\mathsf{Pgen}(1^\lambda)$ generate system parameters p that are available to all algorithms. We do not always explicitly write p as an input.

An *interactive argument* $\Pi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ for a relation $\mathcal{R}$ is an interactive protocol between a probabilistic polynomial time prover P and verifier V. The key generator KGen generates a common reference string $\mathtt{srs}$ at the beginning of the protocol. Both P and V take as public input $\mathtt{srs}$ and a statement $\mathbb{x}$ and, additionally, P takes as private input a witness $\mathbb{w}$. The verifier V either accepts or rejects the transcript (all messages exchanged in the protocol execution). Accordingly, we say the transcript is accepting or rejecting.

Let $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_\mu) \in \mathbb{N}^\mu$. A *$\boldsymbol{\kappa}$-tree of transcripts* for a $(2\mu+1)$-move public-coin interactive argument $\Pi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ is a set of $K = \prod_{i=1}^{\mu} \kappa_i$ transcripts arranged in the following tree structure. The nodes in this tree correspond to the prover's messages and the edges to the verifier's challenges. Every node at depth i has precisely κ_i children corresponding to κ_i pairwise distinct challenges. Every transcript corresponds to exactly one path from the root node to a leaf.

Definition 2. *Let $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_\mu)$. A $(2\mu+1)$-move public-coin interactive argument $\Pi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ for relation $\mathcal{R}$ is* computational $\boldsymbol{\kappa}$-special-sound *if there exists a DPT extractor $\mathsf{Ext}_{\mathsf{ss}}$ such that for any PPT $\mathcal{A}_{\mathsf{ss}}$,* $\mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen},\Pi,\mathsf{Ext}_{\mathsf{ss}},\boldsymbol{\kappa},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l|l} \mathcal{T} \textit{ is a } \boldsymbol{\kappa}\textit{-tree of} & \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); (\mathtt{srs}, \mathsf{tk}) \leftarrow \mathsf{KGen}(\mathsf{p}); \\ \textit{accepting transcripts} & (\mathbb{x}, \mathcal{T}) \leftarrow \mathcal{A}_{\mathsf{ss}}(\mathtt{srs}); \mathbb{w} \leftarrow \mathsf{Ext}_{\mathsf{ss}}(\mathtt{srs}, \mathbb{x}, \mathcal{T}) \\ \wedge\, (\mathtt{srs}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R} & \end{array}\right] \approx_\lambda 0 \ .$$

3 An Oracle Framework for AGMOS

Later, we propose a new assumption (SplitRSDH, Definition 4) and prove it is secure in the AGMOS (AGM with oblivious sampling, [LPS23]). We also compare our work to the work of Faonio et al. [FFR24]. We emphasize that we only use AGMOS as a sanity check that the new standard-looking assumption is likely to be secure. This is a standard and least contraversial use of the ideal group models. However, to have this part of paper in a more solid foundation, we recast AGMOS in the type-safe oracle framework of Jaeger and Mohan (Crypto 2024, [JM24]). Importantly, [JM24] proved an AGM to GGM lifting result that also carries over to our framework for AGMOS.

The resulting variant of AGMOS is more natural than AGMOS's original description in [LPS23]. Namely [LPS23] added sampling oracles on top of the AGM framework of [FKL18] which restricted the adversaries (by requiring them to output explanations) but did not have any other oracles. On the other hand, the oracle AGM framework of [JM24] restricts only the group access by providing oracles to group operations and other interesting functionalities. To this framework, our AGMOS framework just adds two more sampling oracles. We hope this makes AGMOS simpler to understand and use.

Algebraic group model (AGM, [FKL18]) is a very popular idealized model to prove the security of various assumptions and protocols. Compared to GGM,

AGM offers two major advantages. First, it is more realistic, allowing the adversaries to see the bit-representations of group elements. (Shoup's GGM [Sho97] assumes the bit-representations are randomized and Maurer's GGM [Mau05] does not reveal bit-representations.) Second, AGM proofs are shorter than GGM proofs: GGM proofs usually start and end with somewhat boilerplate steps, common to all GGM proofs. AGM allows to abstract these steps away and thus directly concentrate on the meat of the proof.

Fuchsbauer et al. [FKL18] left certain aspects of the AGM unformalized. This gave rise to a number of criticisms [Zha22,ZZK22] and fixes [Zha22,JM24]. In Crypto 2024, Jaeger and Mohan proposed [JM24] a rigorous interpretation of the AGM that answers to the criticisms. In particular, they prove that under mild assumptions, an AGM proof of security (in their interpretation) implies a GGM proof of security; see [JM24, Theorem 2] for the concrete statement. We will follow [JM24] when describing AGM and constructing AGM proofs. More precisely, we will use their oracle framework to AGM that is similar to Maurer's type-safe oracle GGM framework [Mau05]. (In addition, [JM24] considered a pseudocode and a circuit framework.)

Fix a pairing description $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$ and let σ be a fixed injective encoding of elements of any $\mathbb{G}_\iota$ as a bit-representation. In the oracle framework of bilinear AGM, there are two tables, **Val** (values) and **Expl** (explanations). The adversary is given handles to **Val** and access to a fixed family $\Pi_{\mathcal{O}}$ of oracles. We will denote the adversary's initial input (e.g., input from the challenger) in $\mathbb{G}_\iota$ by $[\mathbb{x}_\iota]_\iota$. We assume $[\mathbb{x}_\iota]_\iota$ always includes $[1]_\iota$. Let $\mathbb{x} = ([\mathbb{x}_1]_1, [\mathbb{x}_2]_2)$. **Val** is initialized with the elements of $\mathbb{x}$ and the corresponding entries of **Expl** are initialized with unit vectors $\boldsymbol{e}_i$ (corresponding to the explanations of type $\mathbb{x}_\iota[i] = \boldsymbol{e}_i^\mathsf{T}\mathbb{x}_\iota$). Most oracles only operate on group elements based on handles. All oracles can output $\perp$ (for example, when one is asked to add together elements from different groups).

In the bilinear setting, the two most important oracles are $\mathcal{O}p$ (takes two elements of the same group, referenced by handles i and j and stores their sum to a location k) and $\mathcal{P}air$ (takes elements of groups $\mathbb{G}_1$ and $\mathbb{G}_2$, referenced by handles i and j, and stores their sum to a location k). The oracles also compute explanations of the output. The explanation of the output of $\mathcal{O}p$ is the sum of explanations of its inputs. The output explanation of $\mathcal{P}air$ is a tensor product of the input explanations. Two oracles that output non-group elements are $\mathcal{E}q$ (takes two elements of the same group, referenced by handles i and j and returns 1 if they are equal) and $\mathcal{E}ncode$ (takes a group element, referenced by a handle i and returns its bit representation). For the sake of reference, we depict all mentioned oracles in Fig. 1.

Additional oracles are possible as long as they work on handles and are efficient to implement. For example, [JM24] considers a scalar multiplication oracle; however, it can be implemented by using $\mathcal{O}p$. Similarly, $\mathcal{E}q$ can be implemented by using $\mathcal{E}ncode$. If not being careful, adding or removing oracles can change the model drastically. The presence of $\mathcal{E}ncode$ is the only difference between the oracle framework of AGM and Maurer's GGM; in the latter, only $\mathcal{E}q$ can be used to obtain any information about the bit-representation of group elements. Accord-

$\mathcal{O}p(i,j,k)$

if $\neg\exists\iota \in \{1,2,T\}.(\mathbf{Val}_i \in \mathbb{G}_\iota \wedge \mathbf{Val}_i \in \mathbb{G}_\iota)$
then $\mathbf{Val}_k \leftarrow \bot$;
else $\mathbf{Val}[k] \leftarrow \mathbf{Val}[i] + \mathbf{Val}[j]$;
$\mathbf{Expl}[k] \leftarrow \mathbf{Expl}[i] + \mathbf{Expl}[j]$;

$\mathcal{P}air(i,j,k)$

if $\neg(\mathbf{Val}[i] \in \mathbb{G}_1 \wedge \mathbf{Val}[j] \in \mathbb{G}_2)$
then $\mathbf{Val}[k] \leftarrow \bot$;
else $\mathbf{Val}[k] \leftarrow \mathbf{Val}[i] \bullet \mathbf{Val}[j]$;
$\mathbf{Expl}[k] \leftarrow \mathbf{Expl}[i] \otimes \mathbf{Expl}[j]$;

$\mathcal{E}q(i,j)$

if $\neg\exists\iota \in \{1,2,T\}.(\mathbf{Val}[i] \in \mathbb{G}_\iota \wedge \mathbf{Val}[j] \in \mathbb{G}_\iota)$
then return $\bot$;
else return $\mathbf{Val}[i] =^? \mathbf{Val}[j]$

$\mathcal{E}ncode(i)$

if $\neg\exists\iota \in \{i,j\}.(\mathbf{Val}[i] \in \mathbb{G}_\iota)$
then return $\bot$;
else return $\sigma(\mathbf{Val}[i])$

$\mathcal{S}amp_\iota(i,E,D)$

if $E \notin \mathcal{EF}_{\mathsf{p},\iota} \vee D \notin \mathcal{DF}_{\mathsf{p}}$ **then return** $\bot$; **fi**
$s \leftarrow\!\!\$\ D$; $[\mathfrak{q}]_\iota \leftarrow E(s)$; $\mathsf{il}_\iota \leftarrow \mathsf{il}_\iota + 1$; $i \leftarrow |\mathbb{x}_\iota| + \mathsf{il}_\iota$;
$\mathbf{Val}[i] \leftarrow [\mathfrak{q}]_\iota$; $\mathbf{Expl}[i] \leftarrow (\mathbf{0}_{i-1}, 1)$; **return** s;

Fig. 1. The description of all considered oracles.

ing to [JM24], as the only difference with the AGM, in the *standard model*, the adversary has also access to a $\mathcal{D}ecode$ oracle that takes as an input a bit-string (a bit-representation of a group element) and outputs the group element.

Lipmaa et al. [LPS23] recently defined AGMOS (AGM with oblivious sampling). AGMOS is more realistic than AGM since AGMOS adversaries are given an additional power of sampling group elements without knowing their discrete logarithms. Importantly, as shown in [LPS23], certain uses of KZG [KZG10] are secure in AGM but not in AGMOS.

Let $\mathcal{EF}_{\mathsf{p},\iota}$ be a set of (polynomially many) functions $\mathbb{F} \to \mathbb{G}_\iota$. This can include elliptic-curve hashing [Ica09]. Let $\mathcal{DF}_{\mathsf{p}}$ be a family of distributions over $\mathbb{F}$. Lipmaa et al. [LPS23] introduced two oracles $\mathcal{S}amp_1$ and $\mathcal{S}amp_2$, one for each $\mathbb{G}_1$ and $\mathbb{G}_2$. The ith query (E,D) to $\mathcal{S}amp_\iota$ consists of a function $E \in \mathcal{EF}_{\mathsf{p},\iota}$ and a distribution $D \in \mathcal{DF}_{\mathsf{p}}$. The $\mathcal{S}amp_\iota$ oracle samples a random field element $s_i \leftarrow\!\!\$\ D$, computes $[\mathfrak{q}_{\iota_i}]_\iota \leftarrow E(s_i)$, stores $[\mathfrak{q}_{\iota_i}]_\iota$ in the table (with a self-referential explanation $(0,\ldots,0,1)$), and returns a field element s_i. One also keeps track on the number of $\mathcal{S}amp_\iota$ queries il_ι at any moment.[4]

Let $\Pi_{\mathcal{O}} = \{\mathcal{O}p, \mathcal{P}air, \mathcal{E}q, \mathcal{E}ncode, \mathcal{S}amp_1, \mathcal{S}amp_2\}$ be the collection of available oracles. We require that for any PPT oracle adversary $\mathcal{A}^{\Pi_{\mathcal{O}}}$, there exists a (non-uniform) PPT extractor $\mathsf{Ext}_{\mathcal{A}}^{\Pi_{\mathcal{O}}}$, such that: if $\mathcal{A}^{\Pi_{\mathcal{O}}}(\mathbb{x})$ outputs a vector of group elements $[\mathbb{y}]_\iota$, on input $\mathbb{x} = ([\mathbb{x}_1]_1, [\mathbb{x}_2]_2)$, then with an overwhelming probability, $\mathsf{Ext}_{\mathcal{A}}^{\Pi_{\mathcal{O}}}$ outputs field-element matrices $\boldsymbol{\gamma}$, $\boldsymbol{\delta}$, and $[\mathfrak{q}_\iota]_\iota$ ($\mathcal{S}amp_\iota$'s

[4] AGMOS suits better to the oracle framework of the AGM than to the original FKL [FKL18] framework of the AGM. In particular, [LPS23] defined sampling by using oracles $\mathcal{S}amp_\iota$. Since in the FKL framework, one does not define any other oracles (instead, one restricts the adveraries), $\mathcal{S}amp_\iota$ were the only oracles in [LPS23]. In the current paper, $\mathcal{S}amp_\iota$ are part of a larger family of oracles.

answer vector), such that

$$\mathbb{y} = \boldsymbol{\gamma}^{\mathsf{T}}\mathbb{x}_\iota + \boldsymbol{\delta}^{\mathsf{T}}\mathbb{q}_\iota \ . \tag{2}$$

Definition 3 (AGMOS). *Let $\mathcal{EF} = \{\mathcal{EF}_{\mathsf{p},\iota}\}$ be a collection of functions. Let $\mathcal{DF} = \{\mathcal{DF}_{\mathsf{p}}\}$ be a family of distributions. A PPT algorithm $\mathcal{A}$ is an $(\mathcal{EF}, \mathcal{DF})$-*AGMOS adversary *for* Pgen *if there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{A}}$*, such that for any* $\mathbb{x} = (\mathbb{x}_1, \mathbb{x}_2)$*,* $\mathsf{Adv}^{\mathrm{agmos}}_{\mathsf{Pgen},\mathcal{EF},\mathcal{DF},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) :=$

$$\Pr\left[\begin{array}{l} \mathbb{y}_1 \neq \boldsymbol{\gamma}_1^{\mathsf{T}}\mathbb{x}_1 + \boldsymbol{\delta}_1^{\mathsf{T}}\mathbb{q}_1 \vee \\ \mathbb{y}_2 \neq \boldsymbol{\gamma}_2^{\mathsf{T}}\mathbb{x}_2 + \boldsymbol{\delta}_2^{\mathsf{T}}\mathbb{q}_2 \end{array} \middle| \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); r \leftarrow \mathrm{RND}_\lambda(\mathcal{A}); \\ ([\mathbb{y}_1]_1, [\mathbb{y}_2]_2) \leftarrow\!\!\$\ \mathcal{A}^{\Pi_{\mathcal{O}}}(\mathsf{p}, \mathbb{x}; r); \\ (\boldsymbol{\gamma}_\iota, \boldsymbol{\delta}_\iota, [\mathbb{q}_\iota]_\iota)_{\iota=1}^2 \leftarrow \mathsf{Ext}_{\mathcal{A}}^{\Pi_{\mathcal{O}}}(\mathsf{p}, \mathbb{x}; r) \end{array} \right] = 0 \ .$$

$\mathcal{S}amp_\iota$ is the non-programmable oracle depicted in Fig. 1. Here, $[\mathbb{q}_\iota]_\iota$ is required to be the tuple of elements output by $\mathcal{S}amp_\iota$. We denote by il_ι the number of $\mathcal{S}amp_\iota$ calls.

In the Jaeger-Mohan oracle framework of AGM(OS), such extractor can be constructed efficiently since one is given access to all oracle queries and answers and by keeping track of the explanations of all already computed group elements. (Jaeger and Mohan [JM24] describe it in their Figure 8.) The extractor $\mathsf{Ext}_{\mathcal{A}}$ just runs $\mathcal{A}$ and then returns the outputs of $\mathcal{A}$ together with the corresponding entries of the table **Expl**. Notably, $\mathsf{Ext}_{\mathcal{A}}$ depends only on $\mathcal{A}$ being algebraic (i.e., having to access the oracles to perform group operations) and no other aspects of $\mathcal{A}$'s code. Both in this case and the original AGM framework of [FKL18], $\mathsf{Adv}^{\mathrm{agmos}}_{\mathsf{Pgen},\mathcal{EF},\mathcal{DF},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) = 0$.

Typically, AGMOS proofs rely on the following two assumptions, also used in the current work. (Their use does not depend on the details of [JM24].)

Let $d_1(\lambda), d_2(\lambda) \in \mathsf{poly}(\lambda)$. Pgen is $(d_1(\lambda), d_2(\lambda))$-PDL *(Power Discrete Logarithm, [Lip12]) secure* if for any non-uniform PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\mathcal{A}(\mathsf{p}, [(x^i)_{i=0}^{d_1}]_1, [(x^i)_{i=0}^{d_2}]_2) = x \mid \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda), x \leftarrow\!\!\$\ \mathbb{F}\right] = \mathsf{negl}(\lambda) \ .$$

Let $\mathcal{EF}$ be some family of function and $\mathcal{DF}$ a family of distributions. We say that Pgen is $(\mathcal{EF}, \mathcal{DF})$-TOFR (*Tensor Oracle FindRep*, [LPS23]) secure if for any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{tofr}}_{\mathsf{Pgen},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\boldsymbol{v} \neq \mathbf{0} \wedge \boldsymbol{v}^{\mathsf{T}} \cdot \begin{pmatrix} 1 \\ \mathbb{q}_1 \\ \mathbb{q}_2 \\ \mathbb{q}_1 \otimes \mathbb{q}_2 \end{pmatrix} = 0 \middle| \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \boldsymbol{v} \leftarrow \mathcal{A}^{\mathcal{S}amp_1, \mathcal{S}amp_2}(\mathsf{p}) \right] \approx_\lambda 0 \ .$$

Here, $\mathcal{S}amp_\iota$, $\mathbb{q}_1$, and $\mathbb{q}_2$ are as in Definition 3. $\mathcal{A}$ is a standard model adversary, it just does not see the discrete logarithms of elements created by $\mathcal{S}amp_\iota$.

Depending on the application, one might want to restrict the family of distributions to say uniform distributions. The smaller $\mathcal{DF}$ is, the weaker TOFR will become. See [LPS23] for discussions.

A *reduction* $\mathcal{B}$ from an assumption A to an assumption B is a map that maps an A-adversary $\mathcal{A}$ to a B-adversary $\mathcal{B}^{\mathcal{A}}$. $\mathcal{B}$ is *model-preserving* [JM24,

$\mathsf{KGen}(\mathsf{p})$: $x \leftarrow\!\!\$\, \mathbb{F}^*$; return $\mathtt{srs} \leftarrow ([(x^s)_{s=0}^n]_1, [1, x]_2)$ and $\mathtt{td}_{\mathtt{srs}} \leftarrow x$;

$\mathsf{P}(\mathtt{srs}, \mathbb{w} = (\mathsf{f}_t(X))_{t=1}^m)$: for $t \in [1, m]$, $[f_t]_1 \leftarrow [\mathsf{f}_t(x)]_1$; return $[(f_t)_{t=1}^m]_1$;
V: return $\mathfrak{z} \leftarrow\!\!\$\, \mathbb{F}$; // Evaluation point
P: for $t \in [1, m]$, $\bar{f}_t \leftarrow \mathsf{f}_t(\mathfrak{z})$; return $(\bar{f}_t)_{t=1}^m$;
V: return $v \leftarrow\!\!\$\, \mathbb{F}$; // Batch coefficient
P: $h(X) \leftarrow \left(\sum_{t=1}^m v^{t-1}(\mathsf{f}_t(X) - \bar{f}_t)\right)/(X - \mathfrak{z})$; return $[h]_1 \leftarrow [h(X)]_1$;
V: check $[\sum_{t=1}^m v^{t-1}(f_t - \bar{f}_t)]_1 \bullet [1]_2 = [h]_1 \bullet [x - \mathfrak{z}]_2$;

Fig. 2. The protocol Batch.

Definition 9], if $\mathcal{B}^{\mathcal{A}}$ is a type-safe (i.e. Maurer) generic adversary whenever $\mathcal{A}$ is one. That is, $\mathcal{B}$ does not use $\mathcal{E}ncode$ (or $\mathcal{D}ecode$). $\mathcal{B}$ is *efficient* if (1) it runs in time polynomial in the time of $\mathcal{A}$, and (2) $\mathcal{B}^{\mathcal{A}}$'s success in breaking B is at most a constant-time different from the success of $\mathcal{A}$ in breaking A. Jaeger and Mohan [JM24, Theorem 9] proved the following result that we state informally.

Fact 1 *If there exists a model-preserving and efficient algebraic reduction from assumption A to assumption B and B is generically hard, then A is generically hard.*

4 Special Soundness of KZG Batching Protocols

It is a common practice to prove the knowledge-soundness of KZG-based interactive arguments in idealized group models (GGM, AGM, AGMOS). Stretching the techniques of [LPS24], we prove that some of these arguments are computationally special-sound under falsifiable assumptions. We describe an argument Batch for KZG-batching, and the linearization trick Lin. The former, we prove to be computationally special-sound as it is. However, Lin we show is not special-sound. We apply a novel technique of sanitization to obtain a computationally special-sound version of Lin.

4.1 Special-Soundness of Batch-KZG

Assume the usual setting of KZG with trapdoor x and degree bound n. Consider the well-known interactive protocol Batch from Fig. 2, where the prover has committed to m polynomials and then engages in a single batch proof that opens all m polynomials simultaneously at the same point $\mathfrak{z}$. Here, $[f_t]_1$ are commitments to some polynomials, $\mathfrak{z}$ is the common evaluation point, $\bar{f}_t$ are purported evaluations of $[f_t]_1$ at $\mathfrak{z}$, and $[h]_1$ is the batched opening of all m commitments. Note that Batch's verifier essentially checks that $\mathsf{k.tr} = (\sum_{t=1}^m v^{t-1}[f_t]_1, \mathfrak{z}, \sum_{t=1}^m v^{t-1}\bar{f}_t, [h]_1)$ is an accepting KZG transcript for a randomly chosen v.

In Lemma 1, we show how to extract from a transcript tree a tuple of admissible transcripts. In Theorem 2, we use the constructed extractor to establish

$\mathsf{TE}_{\mathsf{batch}}(\mathsf{ck}, \mathcal{T})$

Parse $\mathcal{T} = (\mathsf{tr}_{ij})_{i\in[1,n+1],j\in[1,m]}$; // $\mathsf{tr}_{ij} = ([(f_t)_{t=1}^m]_1, \mathfrak{z}_i, (\bar{f}_{ti})_{t=1}^m, v_{ij}, [h_{ij}]_1)$
$[h'_{i1}, \ldots, h'_{im}]_1^\intercal \leftarrow \boldsymbol{V}_i^{-1}[h_{i1}, \ldots, h_{im}]_1^\intercal$; (*) // See $\boldsymbol{V}_i$ in Eq. (3)
for $t \in [1, m]$ **do**
  for $i \in [1, n+1]$ **do** $\mathsf{k.tr}'_{it} \leftarrow ([f_t]_1, \mathfrak{z}_i, \bar{f}_{ti}, [h'_{ti}]_1)$; **endfor** (**)
  $\mathbf{k.tr}'_t \leftarrow (\mathsf{k.tr}'_{1t}, \ldots, \mathsf{k.tr}'_{n+1,t})$; **endfor**
return $(\mathbf{k.tr}'_t)_{t=1}^m$;

Fig. 3. The subroutine $\mathsf{TE}_{\mathsf{batch}}$.

Batch's special-soundness. Thus, Batch can be used without modification when one moves away from the proofs in idealized group models. This is important since Batch and its variants are ubiquitous in modern updatable zk-SNARKs like Plonk [GWC19], Marlin [CHM+20], and others [CFF+21,RZ21,LSZ22].

Lemma 1. *Let $\mathcal{T} = (\mathsf{tr}_{ij})$ be an $(n+1, m)$-tree of* Batch*'s accepting transcripts, where tr_{ij} are as in Fig. 3. The DPT algorithm $\mathsf{TE}_{\mathsf{batch}}(\mathsf{ck}, \mathcal{T})$ in Fig. 3 computes a tuple of accepting KZG transcripts $(\mathbf{k.tr}'_t)_{t=1}^m$, such that $\mathsf{k.tr}'_{it} = ([f_t]_1, \mathfrak{z}_i, \ldots)$, with mutually different $\mathfrak{z}_i$ for $i \in [1, n+1]$.*

Proof. Let $\mathcal{T}$ be the given accepting tree of transcripts and

$$\boldsymbol{V}_i = \begin{pmatrix} 1 & v_{i1} & v_{i1}^2 & \cdots & v_{i1}^{m-1} \\ 1 & v_{i2} & v_{i2}^2 & \cdots & v_{i2}^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & v_{im} & v_{im}^2 & \cdots & v_{im}^{m-1} \end{pmatrix} \tag{3}$$

be a Vandermonde matrix. Given $\mathcal{T}$'s structure, $\mathfrak{z}_i$-s are distinct and v_{ij}-s are distinct for each $\mathfrak{z}_i$, rendering $\boldsymbol{V}_i$ non-singular for every $i \in [1, n+1]$.

Define $[h'_{i1}, \ldots, h'_{im}]_1^\intercal$ as in (*) in Fig. 3. As noted above, by the definition of Batch (see Fig. 2), for any i and j, since tr_{ij} is accepted by the Batch verifier, $\mathsf{k.tr}_{ij} := ([\varphi_{ij}]_1, \mathfrak{z}_i, \Phi_{ij}, [h_{ij}]_1)$ is accepted by the KZG verifier, where $\varphi_{ij} := \sum_{t=1}^m v_{ij}^{t-1} f_t$, $\Phi_{ij} := \sum_{t=1}^m v_{ij}^{t-1} \bar{f}_{ti}$, and $h_{ij} = \sum_{t=1}^m v_{ij}^{t-1} h'_{it}$. Namely,

$$\begin{bmatrix} \varphi_{i1} - \Phi_{i1} \\ \vdots \\ \varphi_{im} - \Phi_{im} \end{bmatrix}_1 \bullet [1]_2 = \boldsymbol{V}_i \begin{bmatrix} f_1 - \bar{f}_{i1} \\ \vdots \\ f_m - \bar{f}_{im} \end{bmatrix}_1 \bullet [1]_2 = \begin{bmatrix} h_{i1} \\ \vdots \\ h_{im} \end{bmatrix}_1 \bullet [x - \mathfrak{z}_i]_2 \ ,$$

for any $i \in [1, n+1]$. But then

$$\begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}_1 = \boldsymbol{V}_i^{-1} \begin{bmatrix} \varphi_{i1} \\ \vdots \\ \varphi_{im} \end{bmatrix}_1, \quad \begin{pmatrix} \bar{f}_{i1} \\ \vdots \\ \bar{f}_{im} \end{pmatrix} = \boldsymbol{V}_i^{-1} \begin{pmatrix} \Phi_{i1} \\ \vdots \\ \Phi_{im} \end{pmatrix}, \quad \begin{bmatrix} h'_{i1} \\ \vdots \\ h'_{im} \end{bmatrix}_1 := \boldsymbol{V}_i^{-1} \begin{bmatrix} h_{i1} \\ \vdots \\ h_{im} \end{bmatrix}_1 .$$

KZG's triple homomorphism ensures $\mathsf{k.tr}'_{it}$ (see (**) in Fig. 3) is an accepting KZG transcript. Thus, $\mathsf{TE}_{\mathsf{batch}}$ returns accepting KZG transcripts $\mathbf{k.tr}'_t = (\mathsf{k.tr}'_{1t}, \ldots, \mathsf{k.tr}'_{n+1,t})$ for $t \in [1, m]$, of the claimed form. □

$\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}(\mathsf{ck}, \mathcal{T})$	$\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck})$

$(\mathbf{k.tr}'_t)_{t=1}^m \leftarrow \mathsf{TE}_{\mathsf{batch}}(\mathsf{ck}, \mathcal{T})$;
for $t \in [1,m]$ **do**
  $\mathsf{f}_t^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}'_t)$; **endfor**
return $(\mathsf{f}_t^*(X))_{t=1}^m$;
	$\mathcal{T} \leftarrow \mathcal{A}_{\mathsf{ss}}^{\mathsf{batch}}(\mathsf{ck})$;
	$(\mathbf{k.tr}'_t)_{t=1}^m \leftarrow \mathsf{TE}_{\mathsf{batch}}(\mathsf{ck}, \mathcal{T})$;
	for $t \in [1,m]$ **do**
	  $\mathsf{f}_t^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}'_t)$;
	  if $([f_t]_1, \mathsf{f}_t^*(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}'_t}$ **then**
	  return $\mathbf{k.tr}'_t$; **fi endfor**
	return $\bot$;

Fig. 4. The $\boldsymbol{\kappa}$-special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}$ and the KZG $(n+1)$-special-soundness adversary $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ in Theorem 2.

Theorem 2. *Let $n, m \in \mathsf{poly}(\lambda)$ and $\boldsymbol{\kappa} = (n+1, m)$. If KZG is computational $(n+1)$-special-sound, then* Batch *is computational $\boldsymbol{\kappa}$-special-sound.*

Proof. Let $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$ be the promised $(n+1)$-special-soundness extractor of KZG and let $\mathcal{A}_{\mathsf{ss}}^{\mathsf{batch}}$ be any Batch $\boldsymbol{\kappa}$-special-soundness adversary. In Fig. 4, we depict a $\boldsymbol{\kappa}$-special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}$ for Batch and an $(n+1)$-special-soundness adversary $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ for KZG. Here, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}$ has an oracle access to $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$ and $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ has an oracle access to $\mathcal{A}_{\mathsf{ss}}^{\mathsf{batch}}$ and $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$.

$\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}$ inputs ck and a $\boldsymbol{\kappa}$-tree $\mathcal{T} = (\mathsf{tr}_{ij})_{i \in [1,n+1], j \in [1,m]}$ of accepting Batch transcripts, where tr_{ij} is defined as in Fig. 3. $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}$ calls the (deterministic) algorithm $\mathsf{TE}_{\mathsf{batch}}$ (see Fig. 3) to compute $n+1$ valid transcripts $\mathsf{k.tr}'_{it} = ([f_t]_1, \mathfrak{z}_i, \ldots)$ for each polynomial that has to be extracted. Then, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}$ calls m times the $(n+1)$-special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$ to compute the witness.

Let us bound the advantage of Batch's adversary $\mathcal{A}_{\mathsf{ss}}^{\mathsf{batch}}$ against the extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}$. Assume that $\mathcal{T}$ is an $(n+1, m)$-tree of Batch transcripts, each accepted by the Batch verifier. Let bad be the event that for at least one t, $([f_t]_1, \mathsf{f}_t^*(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}'_t}$. If bad does not occur, then $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}$ has computed a valid witness for Batch. Since $\mathfrak{z}_i$ are all distinct, $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ wins the $(n+1)$-special-soundness game if and only if bad happened. Thus, $\mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen},\mathsf{KZG},\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}},n+1,\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}}(\lambda) = \Pr[\mathsf{bad}] = \mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen},\mathsf{Batch},\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}},n+1,m,\mathcal{A}_{\mathsf{ss}}^{\mathsf{batch}}}(\lambda)$. This concludes the proof. □

4.2 Lin: Linearization Trick

Definition of Lin. In many zk-SNARKs, a natural subtask[5] is for the prover to prove that $\sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}(X))\mathsf{d}_t(X) = 0$, where $\mathbf{g}(X) = (\mathsf{g}_1(X), \ldots, \mathsf{g}_{n_b}(X))$ are publicly known polynomials, $\mathbf{a}(X) = (\mathsf{a}_1(X), \ldots, \mathsf{a}_{n_a}(X))$, and $\mathsf{a}_\mathfrak{z}(X)$ (for $\mathfrak{z} \in [1, n_a]$) and $\mathsf{d}_t(X)$ (for $t \in [1, n_b]$) are committed polynomials.

[5] Here, we follow closely the formalism of [FFR24]. One can easily generalize this framework to allow for more general expressions.

Table 2. Comparison of different protocols for the relation $\mathcal{R}^{\mathsf{Lin}}_{\mathsf{ck},\mathbf{g}}$. The communication does not include the commitment costs. KS stands for knowledge-soundness, and SS stands for special-soundness. The number of bits are given for the BLS381-12 pairing.

Method	$\lvert\pi\rvert$ (bits)	KS in the AGM	KS and SS in the standard model
Opening $\mathsf{a}_\imath, \mathsf{d}_t$ separately	$(n_a+n_b)(\lvert\mathbb{F}\rvert+\lvert\mathbb{G}_t\rvert)$ $(640(n_a+n_b))$	✓	✓
Batch-opening $a_\imath, d_t$	$(n_a+n_b)\lvert\mathbb{F}\rvert+\lvert\mathbb{G}_t\rvert$ $(256(n_a+n_b)+384)$	✓	✓
Lin	$n_a\lvert\mathbb{F}\rvert+\lvert\mathbb{G}_t\rvert$ $(256n_a+384)$	✓	✗
SanLin (our work)	$(n_a+1)\lvert\mathbb{F}\rvert+\lvert\mathbb{G}_t\rvert$ $(256n_a+640)$	✓	✓

Example 1. In the simple quadratic check (e.g., in R1CS), one has $n_b = 2$, $n_a = 1$, $\mathsf{g}_1(\mathbf{a}(X)) = \mathsf{a}_1(X)$, and $\mathsf{g}_2(\mathbf{a}(X)) = -1$. One checks that $\mathsf{a}_1(X)\mathsf{d}_1(X) - \mathsf{d}_2(X) = 0$.

We define the underlying relation as

$$\mathcal{R}^{\mathsf{Lin}}_{\mathsf{ck},\mathbf{g}} = \left(\begin{array}{c} \left([(a_\imath)_{\imath=1}^{n_a}, (d_t)_{t=1}^{n_b}]_1\right), \\ ((\mathsf{a}_\imath(X))_{\imath=1}^{n_a}, (\mathsf{d}_t(X))_{t=1}^{n_b}) \end{array} \middle| \begin{array}{l} \forall \imath.([a_\imath]_1, \mathsf{a}_\imath(X)) \in \mathcal{R}_{\mathsf{ck}} \wedge \\ \forall t.([d_t]_1, \mathsf{d}_t(X)) \in \mathcal{R}_{\mathsf{ck}} \wedge \\ \sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}(X))\mathsf{d}_t(X) = 0 \end{array} \right) .$$

We define $\mathcal{R}^{\mathsf{Lin}}_{\mathsf{ck},\mathbf{g},\mathbf{tr}}$ as usual, requiring the committed polynomials to be consistent with the provided transcripts. Following [FFR24], we say $\mathsf{a}_\imath(X)$ are *left* polynomials, $\mathsf{g}_t(\mathbf{a}(X))$ are *core* polynomials, and $\mathsf{d}_t(X)$ are *right* polynomials.

For this task, several natural solutions exist. First, one can open all $n_a + n_b$ polynomials separately at a random point $\mathfrak{z}$ and test that $\sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}(\mathfrak{z}))\mathsf{d}_t(\mathfrak{z}) = 0$. Second, one can do the same, but batch the opening proofs. Third, one can use the linearization trick, known at least from [GWC19,CHM+20]. Using the linearization trick, one performs a single batch-opening at a random point $\mathfrak{z}$, opening (1) all $\mathsf{a}_\imath(X)$, $\imath \in [1, n_a]$, to $\bar{a}_\imath := \mathsf{a}_\imath(\mathfrak{z})$, and (2) the *linearization polynomial* $\Lambda(X) := \sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}})\mathsf{d}_t(X)$ to 0. (See Fig. 5 for a detailed description.)

Insecurity of Lin. Lin is knowledge-sound in the AGM, [CHM+20] since one can extract all polynomials $\mathsf{a}_\imath(X)$ and $\mathsf{d}_t(X)$ from their commitments. However, Lipmaa et al. [LPS23] showed that (a simple case of) Lin is not knowledge-sound in the AGMOS. Faonio et al. [FFR24] generalized the attack of [LPS23], which we recall in Appendix A.1.

Faonio et al. [FFR24] also proved that Lin is knowledge-sound in the AGMOS iff $\mathsf{g}_t(\mathbf{a}(X))$ are linearly independent. Furthermore, they pointed out that in the case of Plonk, the core polynomials are linearly independent, thus justifying the use of Lin in Plonk. However, this still leaves it open whether Lin is special-sound in the standard model, assuming linear independence of core polynomials. We give a proof of the following impossibility result in Appendix A.2.

Theorem 3. *Let $\kappa_1, \kappa_2 \in \mathsf{poly}(\lambda)$. Assuming DL holds in $\mathbb{G}_1$, Lin is not computational (κ_1, κ_2)-special-sound even when the core polynomials $\mathsf{g}_t(\mathbf{a}(X))$ are linearly independent.*

KGen: $x \leftarrow_\$ \mathbb{F}^*$; return $\mathtt{srs} \leftarrow ([(x^t)_{t=0}^n]_1, [1, x]_2)$ and $\mathtt{td}_{\mathtt{srs}} \leftarrow x$;

$\mathsf{P}(\mathtt{srs}, \mathbb{w} = ((\mathsf{a}_\imath(X))_{\imath=1}^{n_a}, (\mathsf{d}_t(X))_{t=1}^{n_b})$:
for $\imath \in [1, n_a]$: $[a_\imath]_1 \leftarrow [\mathsf{a}_\imath(x)]_1$; for $t \in [1, n_b]$: $[d_t]_1 \leftarrow [\mathsf{d}_t(x)]_1$;
return $[(a_\imath)_{\imath=1}^{n_a}, (d_t)_{t=1}^{n_b}]_1$;

V: return $\mathfrak{z} \leftarrow_\$ \mathbb{F}$;

P: for $\imath \in [1, n_a]$: $\bar{a}_\imath \leftarrow \mathsf{a}_\imath(\mathfrak{z})$; return $(\bar{a}_\imath)_{\imath=1}^{n_a}$;

V: return $\gamma \leftarrow_\$ \mathbb{F}$;

P: return $\bar{d} \leftarrow \sum_{t=1}^{n_b} \gamma^{t-1} \mathsf{d}_t(\mathfrak{z})$;

V: return $\beta \leftarrow_\$ \mathbb{F}$;

P: $\mathsf{H}(X) \leftarrow \sum_{\imath=1}^{n_a} \beta^{\imath-1}(\mathsf{a}_\imath(X) - \bar{a}_\imath) + \beta^{n_a} \cdot \sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}})\mathsf{d}_t(X) + \beta^{n_a+1} \cdot \left(\sum_{t=1}^{n_b} \gamma^{t-1} \mathsf{d}_t(X) - \bar{d}\right)$;
$h(X) \leftarrow \mathsf{H}(X)/(X - \mathfrak{z})$; return $[h]_1 \leftarrow [h(x)]_1$;

V: check
$\left[\sum_{\imath=1}^{n_a} \beta^{\imath-1}(a_\imath - \bar{a}_\imath) + \beta^{n_a} \sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}) d_t + \beta^{n_a+1} \left(\sum_{t=1}^{n_b} \gamma^{t-1} d_t - \bar{d}\right)\right]_1 \bullet [1]_2 = [h]_1 \bullet [x - \mathfrak{z}]_2$;

Fig. 5. Lin (without highlighted parts) and SanLin (with highlighted parts).

To give some intuition of the above result, consider the simple case when one uses Lin to show that $1 \cdot \mathsf{d}_1(X) + X \cdot \mathsf{d}_2(X) = 0$. In this example, 1 and X are linearly independent polynomials. We can construct the following DL adversary that gets a challenge $[y]_1$ as an input. The adversary can pick a KZG public key ck and store the trapdoor x. In such case, the adversary can commit to polynomials $\mathsf{d}_1(X) = yX$ as $[d_1]_1 = x[y]_1$ and to $\mathsf{d}_2(X) = -y$ as $[d_1]_1 = -[y]_1$, which satisfy the requirement $1 \cdot \mathsf{d}_1(X) + X \cdot \mathsf{d}_2(X) = 0$. We show in the proof of Theorem 3 that one can also simulate the rest of (κ_1, κ_2)-tree of accepting transcripts (for any $\kappa_1, \kappa_2 \in \mathsf{poly}(\lambda)$) of Lin. If an efficient computational special-soundness extractor exists, the reduction can run it with an input ck and the transcript tree to recover the polynomial $\mathsf{d}_2(X) = -y$. Thus, the adversary can return y and break the DL assumption in $\mathbb{G}_1$.

4.3 SanLin: Sanitized Lin

Next, we will modify Lin so that it will become secure for any choice of $\mathsf{a}_\imath(X)$. The attack of [LPS23,FFR24] is possible since Lin's prover never opens any polynomials $\mathsf{d}_t(X)$, allowing it to "obliviously sample" $[d_t]_1$. However, $[a_\imath]_1$ (that are already opened) cannot be sampled obliviously. We counter this attack introducing a technique called *sanitization*. Sanitization involves batch-opening all polynomial commitments (e.g., that were not batch-opened otherwise). In SanLin (*sanitized* Lin), this means batch-opening all polynomials $\mathsf{d}_t(X)$ at a random point. Crucially, we do not need the actual evaluations $\mathsf{d}_t(\mathfrak{z})$. Thus, it suffices for the prover to send $\bar{d} \leftarrow \sum_{t=1}^{n_b} \gamma^{t-1} \mathsf{d}_t(\mathfrak{z})$, for a new batching coefficient γ, adding a single field element $\bar{d}$ (and one round) to Lin's communication. We depict SanLin in Fig. 5 and compare it to more simplistic protocols in Table 2.

Theorem 4. *Let* $n, n_a, n_b \in \mathsf{poly}(\lambda)$, $n_{\mathsf{g}} := \max_t \deg \mathsf{g}_t(\mathbf{a}_t(X)) \in \mathsf{poly}(\lambda)$, $\kappa_1 := \max(n_{\mathsf{g}}, n)+1$, *and* $\boldsymbol{\kappa} = (\kappa_1, n_b, n_a+2)$. *If KZG for degree* $\leq n$ *polynomials has computational* $(n+1)$*-special-soundness and evaluation-binding, then* SanLin *has computational* $\boldsymbol{\kappa}$*-special-soundness.*

We postpone SanLin's computational special-soundness proof to Appendix B.

5 New Assumption: SplitRSDH

Our proofs for Plonk and SmallPlonk rely on the following novel falsifiable assumption SplitRSDH.

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

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

is negligible. Here, $\mathsf{Z}_{\mathcal{S}}(X) := \prod_i (X - \mathfrak{z}_i)$ *is the vanishing polynomial. (The condition* $\sum_{k=1}^m [\tau_k \psi_k(x)]_1 = [L(x)]_1 + [\varphi \mathsf{Z}_{\mathcal{S}}(x)]_1$ *is equivalent to* $\sum_{k=1}^m ([\tau_k]_1 \bullet [\psi_k(x)]_2) = [1]_1 \bullet [L(x)]_2 + [\varphi]_1 \bullet [\mathsf{Z}_{\mathcal{S}}(x)]_2$.*)*

SplitRSDH is falsifiable since the challenger, knowing x, can verify the adversary's success. In Appendix C, we prove SplitRSDH's security in the new type-safe framework of AGMOS. The intuition behind SplitRSDH and its AGMOS proof is simple. Let $\tau_k(X)$ be the explanations of $[\tau_k]_1$ provided by the AGMOS adversary. Let $T(X) := \sum_{k=1}^m \tau_k(X)\psi_k(X) \in \mathbb{F}_{\leq n_1+n_\psi}[X]$. If the challenger accepts, then $T(X) - L(X)$ has a degree smaller than $n_{\mathcal{S}}$ and thus $\varphi(X) = 0$. But then $T(X) = L(X)$, which means that $\deg L(X) \leq n_1 + n_\psi$, a contradiction.

We define the parameters

$$\mathsf{par}_n^{\mathsf{plonk}} = (m, n_\psi, n_1, n_{\mathcal{S}}, (\psi_k(X))_{k=1}^m) \ ,$$

with $m = 3$, $n_\psi = 2n$, $n_1 = \kappa_{\mathsf{kzg}} = n+5$, $n_{\mathcal{S}} = \kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}} + 1$, and $\psi_k(X) = X^{(k-1)n}$, and

$$\mathsf{par}_n^{\mathsf{smallplonk}} = (m, n_\psi, n_1, n_{\mathcal{S}}, (\psi_k(X))_{k=1}^m) \ ,$$

with $m = 1$, $n_\psi = 0$, $n_1 = \kappa_{\mathsf{kzg}} = 3n+5$, $n_{\mathcal{S}} = \kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}} + 1$, and $\psi_1(X) = 1$. Thus, $n_{\mathcal{S}} > n_1 + n_\psi + n$ (we need the latter in Theorem 8). Here, $\mathsf{par}_n^{\mathsf{plonk}}$ is only useful for Plonk and $\mathsf{par}_n^{\mathsf{smallplonk}}$ is only useful for SmallPlonk.

For $m \geq 1$, we can reduce SplitRSDH to the following nicer-looking but purportedly stronger assumption TIDRSDH (target intermediate degree RSDH), where we allow the adversary to output $[g]_T \leftarrow \sum_{k=1}^m ([\tau_k]_1 \bullet [\psi_k(x)]_2)$. Then, it suffices for the following probability to be negligible (under the conditions of Definition 4):

$$\Pr\left[\begin{array}{l|l} \mathcal{S} \subset \mathbb{F} \wedge |\mathcal{S}| = n_{\mathcal{S}} \wedge L(X) \in \mathbb{F}[X] \wedge & x \leftarrow_{\$} \mathbb{F}; \\ \deg L(X) \in [n_1+n_\psi+1, n_{\mathcal{S}}-1] \wedge & \mathsf{ck} \leftarrow ([1, x, \ldots, x^{n_1}]_1, [1, x, \ldots, x^{n_\psi}]_2); \\ [g - L(x)]_T = [\varphi \mathsf{Z}_{\mathcal{S}}(x)]_T & (\mathcal{S}, L(X), [\varphi]_1, [g]_T) \leftarrow \mathcal{A}(\mathsf{p}, \mathsf{ck}); \end{array}\right] .$$

The AGM(OS) proof of TIDRSDH follows from noticing that if the adversary succeeds, then $g(X) - L(X)$ has degree $< n_{\mathcal{S}}$ and thus $\varphi(X) = 0$. But then $g(X) = L(X)$, which means $\deg L(X) \leq n_1 + n_\psi$ and the adversary failed. TIDRSDH is not weaker and (purportedly) stronger than SplitRSDH for the same par. First, it allows the adversary to output $\mathbb{G}_T$ elements[6]. Second, a SplitRSDH adversary has less freedom to choose $[g]_T$. Third, ck has more elements so the AGMOS reduction is to a stronger PDL. We will explicitly use SplitRSDH within the current paper but one can instead use TIDRSDH.

SplitRSDH can also be reduced to two somewhat more standard assumptions. Assume a par-SplitRSDH adversary succeeds. Consider two cases:

- $\varphi \neq 0$. Then we construct the following reduction to a "target ARSDH" assumption. Given $([1, \ldots, x^{n_{\mathcal{S}}-1-n_\psi}]_1, [1, \ldots, x^{n_\psi}]_2)$, it outputs $(\mathcal{S}, [g]_T, [\varphi]_1)$, where $[g]_T = \sum_{k=1}^{m}([\tau_k]_1 \bullet [\psi_k(x)]_2) - [L(x)]_T$, and checks that $[g]_T = [\varphi]_1 \bullet [\mathsf{Z}_{\mathcal{S}}(x)]_2$ and $\varphi \neq 0$.
- $\varphi = 0$. Then we construct the following reduction to a "target BDE assumption". Given $([1, \ldots, x^{n_1}]_1, [1, \ldots, x^{n_\psi}]_2)$, it outputs $(L(X), [g]_T)$, where $[g]_T = \sum_{k=1}^{m}([\tau_k]_1 \bullet [\psi_k(x)]_2)$, and once checks that $\deg L(X) > n_1 + n_\psi$ and $[g]_T = [L(x)]_T$.

6 Special-Soundness of Plonk And Variants

In this section, we prove that interactive Plonk [GWC19] has special-soundness, assuming that KZG is evaluation-binding and specially sound and a new, falsifiable assumption SplitRSDH holds. Recall that KZG is evaluation-binding and specially sound under the ARSDH assumption. In addition, we prove that both SanPlonk (a sanitized variant of Plonk with one field element of extra communication) and SmallPlonk (a well-known variant of Plonk, where one does not split $[t]_1$ to three group elements) have special-soundness under ARSDH alone. By applying the Fiat-Shamir transform to any of the three constructions, one can obtain a zk-SNARKs secure in the ROM under the same assumptions.

6.1 Preliminaries For Plonk

We recall Plonk [GWC19], a popular zk-SNARK for proving satisfiability of arbitrary arithmetic circuits. We closely follow the notation of [GWC19].

Let $\mathbb{H}$ be a multiplicative subgroup of $\mathbb{F}$ containing the nth roots of unity. Let ω be a primitive nth root of unity and a generator of $\mathbb{H}$, $\mathbb{H} = \{1, \omega, \ldots, \omega^{n-1}\}$. Let $\mathsf{Z}_{\mathbb{H}}(X) := X^n - 1$ be the vanishing polynomial on $\mathbb{H}$. For $i \in [1, n]$, $L_i(X)$ denotes the ith Lagrange polynomial on $\mathbb{H}$. Namely, $L_i(X)$ is the unique polynomial of at most degree $n-1$ such that $L_i(\omega^i) = 1$ and $L_i(\omega^j) = 0$ for all $j \in [1, n] \setminus \{i\}$. We assume that the number of constraints is upper bounded by n.

[6] Since $\mathbb{G}_T$ (a subgroup of the multiplicative group of a finite field) is not a generic group [JR10], we prefer not to handle adversaries who output $\mathbb{G}_T$ elements. See [JR10] for a discussion.

The SNARK proof relation. Let $\mathcal{P} = \{\mathsf{q_M}(X), \mathsf{q_L}(X), \mathsf{q_R}(X), \mathsf{q_O}(X), \mathsf{q_C}(X), \mathsf{S}_{\sigma 1}(X), \mathsf{S}_{\sigma 2}(X), \mathsf{S}_{\sigma 3}(X)\}$ be a set of polynomials that defines a given circuit. See Appendix D.1 for the exact conditions they satisfy. They are exactly the same as in [GWC19]. We use the notation $\mathsf{q_X}_i := \mathsf{q_X}(\omega^i)$.

Given $\ell \leq n$ and $\mathcal{P}$, we wish to prove statements of knowledge for the relation $\mathcal{R}_{\mathcal{P}} \subset \mathbb{F}^{\ell} \times \mathbb{F}^{3n-\ell}$ containing all pairs $\mathbb{x} = (w_i)_{i=1}^{\ell}, \mathbb{w} = (w_i)_{i=\ell+1}^{3n}$ such that

1. For $i \in [1, \ell]$: $\mathsf{q_M}_i = \mathsf{q_R}_i = \mathsf{q_O}_i = \mathsf{q}_{Ci} = 0$ and $\mathsf{q_L}_i = -1$, which guarantees

$$\mathsf{q_M}_i w_i w_{n+i} + \mathsf{q_L}_i w_i + \mathsf{q_R}_i w_{n+i} + \mathsf{q_O}_i w_{2n+i} + \mathsf{q}_{Ci} = -w_i \ . \tag{4}$$

   We see later that this is needed to force the prover to use the correct $\mathbb{x}$.
2. For all $i \in [\ell + 1, n]$:

$$\mathsf{q_M}_i w_i w_{n+i} + \mathsf{q_L}_i w_i + \mathsf{q_R}_i w_{n+i} + \mathsf{q_O}_i w_{2n+i} + \mathsf{q}_{Ci} = 0 \ , \tag{5}$$

3. For all $i \in [1, 3n]$:

$$w_i = w_{\sigma(i)} \ . \tag{6}$$

We refer to [GWC19] for the explanation how these constraints are related to arithmetic circuits.

6.2 Plonk And Variants

We present Plonk, SmallPlonk, and Plonk's sanitized variant SanPlonk. While we describe their interactive versions, to save space, we will omit the adjective "interactive". We describe them in parallel, highlighting changes in SanPlonk and SmallPlonk compared to Plonk. Compared to Plonk, SanPlonk batch opens $[t_{lo}]_1$, $[t_{mid}]_1$, and $[t_{hi}]_1$ (three group elements sent in Plonk), applying the sanitization technique from Section 4.2. This results in an interactive argument with two additional rounds and one more field element sent by the prover, compared to Plonk. On the other hand, SmallPlonk sends just a single group element $[t]_1$ instead of three elements $[t_{lo}, t_{mid}, t_{hi}]_1$. We prove the computational special-soundness of Plonk and SmallPlonk assuming that (1) KZG is evaluation-binding and special sound, and (2) a new falsifiable assumption SplitRSDH holds. We prove the computational special-soundness of SanPlonk solely under (1).

Common preprocessed input: n, $[x, \ldots, x^{n+5}]_1$ plus additional elements $[x^{n+6}, \ldots, x^{3n+5}]_1$ in SmallPlonk), $(q_{\mathsf{M}i}, q_{\mathsf{L}i}, q_{\mathsf{R}i}, q_{\mathsf{O}i}, q_{\mathsf{C}i})_{i=1}^{n}$, σ^*, $\mathsf{q_M}(X) = \sum_{i=1}^{n} q_{\mathsf{M}i} L_i(X)$, $\mathsf{q_L}(X) = \sum_{i=1}^{n} q_{\mathsf{L}i} L_i(X)$, $\mathsf{q_R}(X) = \sum_{i=1}^{n} q_{\mathsf{R}i} L_i(X)$, $\mathsf{q_O}(X) = \sum_{i=1}^{n} q_{\mathsf{O}i} L_i(X)$, $\mathsf{q_C}(X) = \sum_{i=1}^{n} q_{\mathsf{C}i} L_i(X)$, $\mathsf{S}_{\sigma 1}(X) = \sum_{i=1}^{n} \sigma^*(i) L_i(X)$, $\mathsf{S}_{\sigma 2}(X) = \sum_{i=1}^{n} \sigma^*(n+i) L_i(X)$, $\mathsf{S}_{\sigma 3}(X) = \sum_{i=1}^{n} \sigma^*(2n+i) L_i(X)$.

Verifier preprocessed input: $[\mathsf{q_M}]_1 := \mathsf{q_M}(x) \cdot [1]_1$, $[\mathsf{q_L}]_1 := \mathsf{q_L}(x) \cdot [1]_1$, $[\mathsf{q_R}]_1 := \mathsf{q_R}(x) \cdot [1]_1$, $[\mathsf{q_O}]_1 := \mathsf{q_O}(x) \cdot [1]_1$, $[\mathsf{q_C}]_1 := \mathsf{q_C}(x) \cdot [1]_1$, $[s_{\sigma 1}]_1 := \mathsf{S}_{\sigma 1}(x) \cdot [1]_1$, $[s_{\sigma 2}]_1 := \mathsf{S}_{\sigma 2}(x) \cdot [1]_1$, $[s_{\sigma 3}]_1 := \mathsf{S}_{\sigma 3}(x) \cdot [1]_1$, $x \cdot [1]_2$,

Public input: $(\ell, (w_i)_{i=1}^{\ell})$.

First round. On input $(w_i)_{i=1}^{3n}$, the prover does the following. Sample $(b_1, \ldots, b_9) \leftarrow_{\$} \mathbb{F}$. Compute wire polynomials $\mathsf{a}(X) \leftarrow \sum_{i=1}^{n} w_i L_i(X) + (b_1 X + b_2) \mathsf{Z}_{\mathbb{H}}(X) \in$

$\mathbb{F}_{\leq n+1}[X]$, $\mathsf{b}(X) \leftarrow \sum_{i=1}^{n} w_{n+i} L_i(X) + (b_3 X + b_4)\mathsf{Z}_{\mathbb{H}}(X) \in \mathbb{F}_{\leq n+1}[X]$, $\mathsf{c}(X) \leftarrow \sum_{i=1}^{n} w_{2n+i} L_i(X) + (b_5 X + b_6)\mathsf{Z}_{\mathbb{H}}(X) \in \mathbb{F}_{\leq n+1}[X]$. Send $[\mathsf{a}(x), \mathsf{b}(x), \mathsf{c}(x)]_1$ to V. V replies with $\beta, \gamma \leftarrow_{\$} \mathbb{F}$.

Second round. The prover computes polynomial $\mathsf{z}(X) \leftarrow L_1(X) + \sum_{i=1}^{n-1} \left(\prod_{j=1}^{i} \frac{(w_j+\beta\omega^j+\gamma)(w_{n+j}+\beta k_1\omega^j+\gamma)(w_{2n+j}+\beta k_2\omega^j+\gamma)}{(w_j+\sigma^*(j)\omega^j+\gamma)(w_{n+j}+\sigma^*(n+j)\omega^j+\gamma)(w_{2n+j}+\sigma^*(2n+j)\omega^j+\gamma)} \right) L_{i+1}(X) + (b_7 X^2 + b_8 X + b_9)\mathsf{Z}_{\mathbb{H}}(X) \in \mathbb{F}_{\leq n+2}[X]$ and sends $[\mathsf{z}(x)]_1$ to V. V replies with $\alpha \leftarrow_{\$} \mathbb{F}$.

Third round. The prover does the following. Compute

$$
\begin{aligned}
\mathsf{F}_0(X) &:= \mathsf{a}(X)\mathsf{b}(X)\mathsf{q}_{\mathsf{M}}(X) + \mathsf{a}(X)\mathsf{q}_{\mathsf{L}}(X) + \mathsf{b}(X)\mathsf{q}_{\mathsf{R}}(X) + \mathsf{c}(X)\mathsf{q}_{\mathsf{O}}(X) + \mathsf{PI}(X) + \mathsf{q}_{\mathsf{C}}(X) \\
\mathsf{F}_1(X) &:= (\mathsf{a}(X) + \beta X + \gamma)\,(\mathsf{b}(X) + \beta k_1 X + \gamma)\,(\mathsf{c}(X) + \beta k_2 X + \gamma)\,\mathsf{z}(X) \\
&\quad - (\mathsf{a}(X) + \beta \mathsf{S}_{\sigma 1}(X) + \gamma)(\mathsf{b}(X) + \beta \mathsf{S}_{\sigma 2}(X) + \gamma)(\mathsf{c}(X) + \beta \mathsf{S}_{\sigma 3}(X) + \gamma)\mathsf{z}(X\omega) \\
\mathsf{F}_2(X) &:= (\mathsf{z}(X) - 1)L_1(X) \\
\mathsf{F}(X) &:= \mathsf{F}_0(X) + \alpha \mathsf{F}_1(X) + \alpha^2 \mathsf{F}_2(X)\ , \\
\mathsf{t}(X) &:= \tfrac{\mathsf{F}(X)}{\mathsf{Z}_{\mathbb{H}}(X)} \in \mathbb{F}_{\leq 3n+5}[X]\ .
\end{aligned}
\tag{7}
$$

In SmallPlonk, the prover just sends $[\mathsf{t}(x)]_1$ to the verifier. In Plonk and SanPlonk, we split $\mathsf{t}(X)$ into polynomials $\mathsf{t}_{\mathsf{lo}}'(X), \mathsf{t}_{\mathsf{mid}}'(X)$ (both of degree less than n) and $\mathsf{t}_{\mathsf{hi}}'(X)$ (of degree at most $n+5$), such that $\mathsf{t}(X) = \mathsf{t}_{\mathsf{lo}}'(X) + X^n \mathsf{t}_{\mathsf{mid}}'(X) + X^{2n}\mathsf{t}_{\mathsf{hi}}'(X)$. Sample $b_{10}, b_{11}, b_{12} \leftarrow_{\$} \mathbb{F}$ and define $\mathsf{t}_{\mathsf{lo}}(X) := \mathsf{t}_{\mathsf{lo}}'(X) + b_{10}X^n + b_{12}X^{n+1}$, $\mathsf{t}_{\mathsf{mid}}(X) := \mathsf{t}_{\mathsf{mid}}'(X) - b_{10} - b_{12}X + b_{11}X^n$, and $\mathsf{t}_{\mathsf{hi}}(X) := \mathsf{t}_{\mathsf{hi}}'(X) - b_{11}$. ($b_{12}$ is required for the zero-knowledge proof of SanPlonk.) Note that $\mathsf{t}(X) = \mathsf{t}_{\mathsf{lo}}(X) + X^n \mathsf{t}_{\mathsf{mid}}(X) + X^{2n}\mathsf{t}_{\mathsf{hi}}(X)$. Send $[\mathsf{t}_{\mathsf{lo}}(x), \mathsf{t}_{\mathsf{mid}}(x), \mathsf{t}_{\mathsf{hi}}(x)]_1$ to the verifier. The verifier replies with the evaluation randomness $\mathfrak{z} \leftarrow_{\$} \mathbb{F}$.

Fourth round. The prover does the following. Set $\bar{a} \leftarrow \mathsf{a}(\mathfrak{z})$, $\bar{b} \leftarrow \mathsf{b}(\mathfrak{z})$, $\bar{c} \leftarrow \mathsf{c}(\mathfrak{z})$, $\bar{s}_{\sigma 1} \leftarrow \mathsf{S}_{\sigma 1}(\mathfrak{z})$, $\bar{s}_{\sigma 2} \leftarrow \mathsf{S}_{\sigma 2}(\mathfrak{z})$, $\bar{z}_{\omega} \leftarrow \mathsf{z}(\omega\mathfrak{z})$. Send $(\bar{a}, \bar{b}, \bar{c}, \bar{s}_{\sigma 1}, \bar{s}_{\sigma 2}, \bar{z}_{\omega})$ to the verifier. The verifier replies with the sanitization randomness $\delta \leftarrow_{\$} \mathbb{F}$.

Fourth (Plonk/SmallPlonk) or fifth (SanPlonk) round. The prover sends $\bar{t}_{\mathfrak{z}} \leftarrow \mathsf{t}_{\mathsf{lo}}(\mathfrak{z}) + \delta \mathsf{t}_{\mathsf{mid}}(\mathfrak{z}) + \delta^2 \mathsf{t}_{\mathsf{hi}}(\mathfrak{z})$. The verifier replies with $v \leftarrow_{\$} \mathbb{F}$.

Fifth (Plonk/SmallPlonk) or sixth (SanPlonk) round. Prover does the following. Compute the linearization polynomial $\mathsf{r}(X)$:

$$
\begin{aligned}
\Lambda_0(X) &= \bar{a}\bar{b} \cdot \mathsf{q}_{\mathsf{M}}(X) + \bar{a} \cdot \mathsf{q}_{\mathsf{L}}(X) + \bar{b} \cdot \mathsf{q}_{\mathsf{R}}(X) + \bar{c} \cdot \mathsf{q}_{\mathsf{O}}(X) + \mathsf{PI}(\mathfrak{z}) + \mathsf{q}_{\mathsf{C}}(X)\ , \\
\Lambda_1(X) &= (\bar{a} + \beta\mathfrak{z} + \gamma)(\bar{b} + \beta k_1 \mathfrak{z} + \gamma)(\bar{c} + \beta k_2 \mathfrak{z} + \gamma) \cdot \mathsf{z}(X) \\
&\quad - (\bar{a} + \beta \bar{s}_{\sigma 1} + \gamma)(\bar{b} + \beta \bar{s}_{\sigma 2} + \gamma)(\bar{c} + \beta \cdot \mathsf{S}_{\sigma 3}(X) + \gamma)\bar{z}_{\omega}\ , \\
\Lambda_2(X) &= (\mathsf{z}(X) - 1)L_1(\mathfrak{z})\ , \\
\mathsf{r}(X) &= \Lambda_0(X) + \alpha\Lambda_1(X) + \alpha^2 \Lambda_2(X) \\
&\quad - \begin{cases} \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}) \cdot (\mathsf{t}_{\mathsf{lo}}(X) + \mathfrak{z}^n \mathsf{t}_{\mathsf{mid}}(X) + \mathfrak{z}^{2n}\mathsf{t}_{\mathsf{hi}}(X)) & \text{(in Plonk and SanPlonk)} \\ \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}) \cdot \mathsf{t}(X) & \text{(in SmallPlonk)} \end{cases}\ .
\end{aligned}
\tag{8}
$$

Let

$$\begin{aligned}\mathsf{H}(X) &:= \mathsf{r}(X) + v\mathsf{a}(X) + v^2\mathsf{b}(X) + v^3\mathsf{c}(X) + v^4\mathsf{S}_{\sigma 1}(X) + v^5\mathsf{S}_{\sigma 2}(X)\\ &\quad + v^6(\mathsf{t}_{\mathsf{lo}}(X) + \delta\mathsf{t}_{\mathsf{mid}}(X) + \delta^2\mathsf{t}_{\mathsf{hi}}(X)) \ , \\ \bar{H} &:= v\bar{a} + v^2\bar{b} + v^3\bar{c} + v^4\bar{s}_{\sigma 1} + v^5\bar{s}_{\sigma 2} + v^6\bar{t}_{\mathfrak{z}} \ .\end{aligned} \tag{9}$$

Compute the opening proof polynomials $\mathsf{W}_{\mathfrak{z}}(X) := (\mathsf{H}(X) - \bar{H})/(X - \mathfrak{z})$ and

$$\mathsf{W}_{\mathfrak{z}\omega}(X) = \tfrac{\mathsf{z}(X) - \bar{z}_\omega}{X - \mathfrak{z}\omega} \ .$$

$[\mathsf{W}_{\mathfrak{z}}]_1 := [\mathsf{W}_{\mathfrak{z}}(x)]_1$; $[\mathsf{W}_{\mathfrak{z}\omega}]_1 := [\mathsf{W}_{\mathfrak{z}\omega}(x)]_1$; Send $[\mathsf{W}_{\mathfrak{z}}, \mathsf{W}_{\mathfrak{z}\omega}]_1$;

Verification algorithm.

1. Validate $(w_i)_{i=1}^{\ell} \in \mathbb{F}^\ell$ and $\left(\bar{a}, \bar{b}, \bar{c}, \bar{s}_{\sigma 1}, \bar{s}_{\sigma 2}, \bar{z}_\omega, \bar{t}_{\mathfrak{z}}\right) \in \mathbb{F}^7$.
2. Validate that $[a, b, c, z]_1$, $[t_{lo}, t_{mid}, t_{hi}]_1$ ($[t]_1$ in SmallPlonk), $[\mathsf{W}_{\mathfrak{z}}, \mathsf{W}_{\mathfrak{z}\omega}]_1 \in \mathbb{G}_1$.
3. Compute $\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}) = \mathfrak{z}^n - 1$, $L_1(\mathfrak{z}) = \frac{\omega(\mathfrak{z}^n - 1)}{n(\mathfrak{z} - \omega)}$, and $\mathsf{PI}(\mathfrak{z}) = \sum_{i \in [\ell]} w_i L_i(\mathfrak{z})$.
4. Split r into its constant and non-constant terms. Compute r's constant term: $r_0 := \mathsf{PI}(\mathfrak{z}) - L_1(\mathfrak{z})\alpha^2 - \alpha(\bar{a} + \beta\bar{s}_{\sigma 1} + \gamma)(\bar{b} + \beta\bar{s}_{\sigma 2} + \gamma)(\bar{c} + \gamma)\bar{z}_\omega$, and let $\mathsf{r}'(X) := \mathsf{r}(X) - r_0$.
5. Sample $u \leftarrow_{\$} \mathbb{F}$ and compute the first part of the batched polynomial commitment $[D]_1 := [\mathsf{r}'(x)]_1 + u \cdot [z]_1$:

$$\begin{aligned}[D]_1 &:= \bar{a}\bar{b} \cdot [\mathsf{q}_{\mathsf{M}}]_1 + \bar{a} \cdot [\mathsf{q}_{\mathsf{L}}]_1 + \bar{b} \cdot [\mathsf{q}_{\mathsf{R}}]_1 + \bar{c} \cdot [\mathsf{q}_{\mathsf{O}}]_1 + [\mathsf{q}_{\mathsf{C}}]_1 \\ &\quad + \left((\bar{a} + \beta\mathfrak{z} + \gamma)(\bar{b} + \beta k_1\mathfrak{z} + \gamma)(\bar{c} + \beta k_2\mathfrak{z} + \gamma)\alpha + L_1(\mathfrak{z})\alpha^2 + u\right) \cdot [z]_1 \\ &\quad - (\bar{a} + \beta\bar{s}_{\sigma 1} + \gamma)\ (\bar{b} + \beta\bar{s}_{\sigma 2} + \gamma)\alpha\beta\bar{z}_\omega \cdot [s_{\sigma 3}]_1 \\ &\quad - \begin{cases} \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}) \cdot ([t_{lo}]_1 + \mathfrak{z}^n \cdot [t_{mid}]_1 + \mathfrak{z}^{2n} \cdot [t_{hi}]_1) & \text{(in Plonk and SanPlonk)} \\ \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}) \cdot [t]_1 & \text{(in SmallPlonk)} \end{cases} .\end{aligned}$$

6. Compute full batched polynomial commitment $[F]_1 := [D]_1 + v \cdot [a]_1 + v^2 \cdot [b]_1 + v^3 \cdot [c]_1 + v^4 \cdot [s_{\sigma 1}]_1 + v^5 \cdot [s_{\sigma 2}]_1 + v^6[t_{lo} + \delta t_{mid} + \delta^2 t_{hi}]_1$.
7. Compute the batch evaluation $E_0 := -r_0 + v\bar{a} + v^2\bar{b} + v^3\bar{c} + v^4\bar{s}_{\sigma 1} + v^5\bar{s}_{\sigma 2} + v^6\bar{t}_{\mathfrak{z}}$, $E_1 := \bar{z}_\omega$, and $E := E_0 + uE_1$.
8. Batch validate all evaluations:

$$([\mathsf{W}_{\mathfrak{z}}]_1 + u \cdot [\mathsf{W}_{\mathfrak{z}\omega}]_1) \bullet [x]_2 \stackrel{?}{=} (\mathfrak{z} \cdot [\mathsf{W}_{\mathfrak{z}}]_1 + u\mathfrak{z}\omega \cdot [\mathsf{W}_{\mathfrak{z}\omega}]_1 + [F]_1 - [E]_1) \bullet [1]_2 \ . \tag{10}$$

Clearly, SanPlonk remains complete after the highlighted changes. In Appendix E, we prove that SanPlonk has zero knowledge.

6.3 Special-Soundness Proof of (San)Plonk's IP

We prove that Plonk, SanPlonk, and SmallPlonk (all described in the previous section) have computational special-soundness.

Before going on, we note that the Plonk verifier performs batch verification, using a batching coefficient u created after the prover's last message. That is,

after unrolling all optimizations, Eq. (10) can be replaced by two checks,

$$\begin{bmatrix} r+v(a-\bar{a})+v^2(b-\bar{b})+v^3(c-\bar{c})+v^4(s_{\sigma 1}-\bar{s}_{\sigma 1}) \\ +v^5(s_{\sigma 2}-\bar{s}_{\sigma 2})+v^6(t_{lo}+\delta t_{mid}+\delta^2 t_{hi}-\bar{t}_{\mathfrak{z}}) \end{bmatrix}_1 \bullet [1]_2 = [\mathsf{W}_{\mathfrak{z}}]_1 \bullet [x-\mathfrak{z}]_2 \ , \\ [z-\bar{z}_\omega]_1 \bullet [1]_2 = [\mathsf{W}_{\mathfrak{z}\omega}]_1 \bullet [x-\mathfrak{z}\omega]_2 \ , \tag{11}$$

where $[r]_1 = [D]_1 - u[z]_1 + r_0[1]_1$. Batching with u just introduces a soundness error $1/|\mathbb{F}|$. We say that a Plonk transcript is accepting when Eq. (11) are satisfied.

We divide the proof into several smaller lemmas. In Section 6.4, we will analyze a subtree of Plonk's, SanPlonk's, and SmallPlonk's accepting transcripts for fixed β, γ, and α, showing that from it, one can extract certain polynomials. In Section 6.5, we use that result to prove the special-soundness of Plonk, SanPlonk, and SmallPlonk.

In the following, $n \in \mathsf{poly}(\lambda)$ and,

$$\begin{aligned} \kappa_{\mathsf{kzg}} &= \begin{cases} n+5 & \text{(in Plonk and SanPlonk)} \ , \\ 3n+5 & \text{(in SmallPlonk)} \ , \end{cases} \\ \boldsymbol{\kappa}_{\mathsf{Plonk}} &= (\kappa_\beta = 3n+1, \kappa_\gamma = 3n+1, \kappa_\alpha = 3, \kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}}+1, \kappa_v^{\mathsf{Plonk}} = 6) \ , \\ \boldsymbol{\kappa}_{\mathsf{san}} &= (\kappa_\beta = 3n+1, \kappa_\gamma = 3n+1, \kappa_\alpha = 3, \kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}}+1, \kappa_\delta = 3, \kappa_v^{\mathsf{san}} = 7) \end{aligned} \tag{12}$$

corresponding to the branching factors of KZG, Plonk, SanPlonk, and SmallPlonk. Moreover, SmallPlonk uses $\boldsymbol{\kappa}_{\mathsf{SmallPlonk}} = \boldsymbol{\kappa}_{\mathsf{Plonk}}$, except for a different value of $\kappa_{\mathsf{kzg}} = 3n+5$. Moreover, define $\kappa_v = \kappa_v^{\mathsf{Plonk}}$ in the case of Plonk and $\kappa_v = \kappa_v^{\mathsf{san}}$ in the case of SanPlonk.

6.4 Subtree Analysis

In this subsection, we analyze a subtree of Plonk's transcripts that results from fixing β, γ, and α. As usual, we start with a tree extractor lemma that gets a tree of accepting Plonk transcripts as input and outputs many accepting KZG transcripts that open relevant polynomial commitments at many different locations.

Lemma 2 (Plonk/SanPlonk/SmallPlonk transcript tree to KZG transcripts). *Let $\mathcal{T}$ be a $\boldsymbol{\kappa}_{\mathsf{Plonk}}$-tree of Plonk's (resp., $\boldsymbol{\kappa}_{\mathsf{san}}$-tree of SanPlonk's and $\boldsymbol{\kappa}_{\mathsf{SmallPlonk}}$-tree of SmallPlonk's) accepting transcripts. Let $\hat{\mathcal{T}}_{\beta\gamma\alpha}$ be a $(1,1,1,\kappa_{\mathfrak{z}},\kappa_\delta,\kappa_v)$-subtree of $\mathcal{T}$ for any fixed β, γ, and α, with the transcripts in this subtree denoted as*

$$\mathsf{tr}_{ijk} = \begin{pmatrix} [a,b,c]_1, \beta, \gamma, [z]_1, \alpha, [t_{lo}, t_{mid}, t_{hi}]_1 \ \text{(}[t]_1 \text{ in SmallPlonk)}, \\ \mathfrak{z}_i, \bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{s}_{\sigma 1 i}, \bar{s}_{\sigma 2 i}, \bar{z}_{\omega i}, \delta_{ij}, \bar{t}_{\mathfrak{z} ij}, v_{ijk}, [\mathsf{W}_{\mathfrak{z} ijk}, \mathsf{W}_{\mathfrak{z}\omega ijk}]_1 \end{pmatrix} \ . \tag{13}$$

The DPT algorithm $\mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{\mathcal{T}}_{\beta\gamma\alpha})$ in Fig. 6 computes a tuple $((\mathbf{k.tr}_k)_k, \mathbf{k.tr}^{\boldsymbol{\omega}})$, where $k \in [1, \kappa_v^{\mathsf{Plonk}}] = [1,6]$ in Plonk and

$\mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{\mathcal{T}}_{\beta\gamma\alpha})$

1 : Parse $\hat{\mathcal{T}}_{\beta\gamma\alpha} = (\mathsf{tr}_{ijk})_{i\in[1,\kappa_{\mathfrak{z}}],j\in[1,\kappa_\delta],k\in[1,\kappa_v]}$; // tr_{ijk} as in Eq. (13); $\kappa_\delta = 1$ in Plonk
2 : **for** $i \in [1, \kappa_{\mathfrak{z}}]$ **do**
3 : $[\Lambda_{0i}]_1 \leftarrow \bar{a}_i\bar{b}_i[\mathsf{q}_\mathsf{M}]_1 + \bar{a}_i[\mathsf{q}_\mathsf{L}]_1 + \bar{b}_i[\mathsf{q}_\mathsf{R}]_1 + \bar{c}_i[\mathsf{q}_\mathsf{O}]_1 + \mathsf{PI}(\mathfrak{z})[1]_1 + [\mathsf{q}_\mathsf{C}]_1$;
4 : $[\Lambda_{1i}]_1 \leftarrow (\bar{a}_i + \beta\mathfrak{z}_i + \gamma)(\bar{b}_i + \beta k_1\mathfrak{z}_i + \gamma)(\bar{c}_i + \beta k_2\mathfrak{z}_i + \gamma)[z]_1$
5 : $\quad - (\bar{a}_i + \beta\bar{s}_{\sigma 1} + \gamma)(\bar{b}_i + \beta\bar{s}_{\sigma 2} + \gamma)(\bar{c}_i[1]_1 + \beta[\mathsf{S}_{\sigma 3}(x)]_1 + \gamma[1]_1)\bar{z}_{\omega,i}$;
6 : $[\Lambda_{2i}]_1 \leftarrow [z-1]_1 L_1(\mathfrak{z}_i)$;
7 : $[r_i]_1 \leftarrow [\Lambda_{0i}]_1 + \alpha[\Lambda_{1i}]_1 + \alpha^2[\Lambda_{2i}]_1 - \begin{cases} \mathsf{Z}_\mathbb{H}(\mathfrak{z}_i)\cdot([t_{lo}]_1 + \mathfrak{z}_i^n[t_{mid}]_1 + \mathfrak{z}_i^{2n}[t_{hi}]_1) \\ \mathsf{Z}_\mathbb{H}(\mathfrak{z}_i)\cdot[t]_1 \ \text{(SmallPlonk)} \end{cases}$;
8 : $[\mathsf{W}'_{\mathfrak{z}i\mathbf{1}1}, \dots, \mathsf{W}'_{\mathfrak{z}i\mathbf{1}\kappa_v}]_1^\intercal \leftarrow \boldsymbol{V}_{i\mathbf{1}}^{-1}[\mathsf{W}_{\mathfrak{z}i\mathbf{1}1}, \dots, \mathsf{W}_{\mathfrak{z}i\mathbf{1}\kappa_v}]_1^\intercal$; // $\boldsymbol{V}_{i\mathbf{1}}$ as in 15, 16
9 : $\mathsf{k.tr}_{1i} \leftarrow ([r_i]_1, \mathfrak{z}_i, 0, [\mathsf{W}'_{\mathfrak{z}i\mathbf{1}1}]_1)$; $\mathsf{k.tr}_{2i} \leftarrow ([a]_1, \mathfrak{z}_i, \bar{a}_i, [\mathsf{W}'_{\mathfrak{z}i\mathbf{1}2}]_1)$;
10 : $\mathsf{k.tr}_{3i} \leftarrow ([b]_1, \mathfrak{z}_i, \bar{b}_i, [\mathsf{W}'_{\mathfrak{z}i\mathbf{1}3}]_1)$; $\mathsf{k.tr}_{4i} \leftarrow ([c]_1, \mathfrak{z}_i, \bar{c}_i, [\mathsf{W}'_{\mathfrak{z}i\mathbf{1}4}]_1)$;
11 : $\mathsf{k.tr}_{5i} \leftarrow ([s_{\sigma 1}]_1, \mathfrak{z}_i, \bar{s}_{\sigma 1i}, [\mathsf{W}'_{\mathfrak{z}i\mathbf{1}5}]_1)$; $\mathsf{k.tr}_{6i} \leftarrow ([s_{\sigma 2}]_1, \mathfrak{z}_i, \bar{s}_{\sigma 2i}, [\mathsf{W}'_{\mathfrak{z}i\mathbf{1}6}]_1)$;
12 : **for** $j \in \{1,2,3\}$ **do** $\bar{t}_{\mathfrak{z}ij} \leftarrow (\boldsymbol{V}_{ij}^{-1})_7(\bar{H}_{ij1}, \dots, \bar{H}_{ij7})^\intercal$; **endfor**
13 : $(\bar{t}_{\mathfrak{z}lo,i}, \bar{t}_{\mathfrak{z}mid,i}, \bar{t}_{\mathfrak{z}hi,i})^\intercal := \boldsymbol{C}_i^{-1}(\bar{t}_{\mathfrak{z}i1}, \bar{t}_{\mathfrak{z}i2}, \bar{t}_{\mathfrak{z}i3})^\intercal$;
14 : $[W_{lo,i}, W_{mid,i}, W_{hi,i}]_1 \leftarrow \boldsymbol{C}_i^{-1}[\mathsf{W}'_{\mathfrak{z}i17}, \mathsf{W}'_{\mathfrak{z}i27}, \mathsf{W}'_{\mathfrak{z}i37}]_1^\intercal$;
15 : $\mathsf{k.tr}_{7i} \leftarrow ([t_{lo}]_1, \mathfrak{z}_i, \bar{t}_{\mathfrak{z}lo,i}, [\mathsf{W}_{lo,i}]_1)$; $\mathsf{k.tr}_{8i} \leftarrow ([t_{mid}]_1, \mathfrak{z}_i, \bar{t}_{\mathfrak{z}mid,i}, [\mathsf{W}_{mid,i}]_1)$;
16 : $\mathsf{k.tr}_{9i} \leftarrow ([t_{hi}]_1, \mathfrak{z}_i, \bar{t}_{\mathfrak{z}hi,i}, [\mathsf{W}_{hi,i}]_1)$;
17 : $\mathsf{k.tr}_i^\omega \leftarrow ([z]_1, \mathfrak{z}_i\omega, \bar{z}_{\omega i}, [\mathsf{W}_{\mathfrak{z}\omega i\mathbf{1}1}]_1)$; **endfor**
18 : $\mathbf{k.tr}^{\boldsymbol{\omega}} \leftarrow (\mathsf{k.tr}_i^\omega)_{i\in[1,\kappa_{\mathfrak{z}}]}$; **for** $k \in [1, \kappa_v + 2]$ **do** $\mathbf{k.tr}_k \leftarrow (\mathsf{k.tr}_{ki})_{i\in[1,\kappa_{\mathfrak{z}}]}$; **endfor**
19 : **return** $((\mathbf{k.tr}_k)_{k\in[1,\kappa_v+2]}, \mathbf{k.tr}^{\boldsymbol{\omega}})$;

Fig. 6. The subroutine $\mathsf{TE}_{*\mathsf{plonk}}$.

$k \in [1, \kappa_v^{\mathsf{san}} + 2] = [1, 9]$ *in* SanPlonk, *of KZG accepting transcripts, such that (1)* $\mathsf{k.tr}_{1i}$, $\mathsf{k.tr}_{2i}$, $\mathsf{k.tr}_{3i}$, $\mathsf{k.tr}_{4i}$, $\mathsf{k.tr}_{5i}$, *and* $\mathsf{k.tr}_{6i}$ *open (respectively)* $[r_i]_1$, $[a]_1$, $[b]_1$, $[c]_1$, $[s_{\sigma 1}]_1$ *and* $[s_{\sigma 2}]_1$ *to* 0, $\bar{a}_i$, $\bar{b}_i$, $\bar{c}_i$, $\bar{s}_{\sigma 1i}$, *and* $\bar{s}_{\sigma 2i}$ *at* $\mathfrak{z}_i$, *and (2)* $\mathsf{k.tr}_i^\omega$ *opens* $[z]_1$ *to* $\bar{z}_{\omega i}$ *at* $\mathfrak{z}_i\omega$. *In addition, in* SanPlonk, $\mathsf{k.tr}_{7i}$, $\mathsf{k.tr}_{8i}$, *and* $\mathsf{k.tr}_{9i}$ *open (respectively)* $[t_{lo}, t_{mid}, t_{hi}]_1$ *to some values* $\bar{t}_{\mathfrak{z}lo,i}$, $\bar{t}_{\mathfrak{z}mid,i}$, *and* $\bar{t}_{\mathfrak{z}hi,i}$ *at* $\mathfrak{z}_i$. *Moreover,* $\mathfrak{z}_i$ *are mutually different.*

Proof. Let $\mathcal{T}$ be a $(\kappa_\beta, \kappa_\gamma, \kappa_\alpha, \kappa_{\mathfrak{z}}, \dots)$-tree of accepting transcripts. We fix a $(1, 1, 1, \kappa_{\mathfrak{z}}, \kappa_\delta, \kappa_v)$-subtree $\hat{\mathcal{T}}_{\beta\gamma\alpha}$ of $\mathcal{T}$ for some β, γ, and α. Then, $\hat{\mathcal{T}}_{\beta\gamma\alpha} = \{\mathsf{tr}_{ijk}\}$ contains accepting transcripts given in Eq. (13) with mutually different $\mathfrak{z}_i$.

Let us unload some of the formulas in Fig. 6. First, $[\Lambda_{0i}]_1$, $[\Lambda_{1i}]_1$, $[\Lambda_{2i}]_1$, $[r_i]_1$ (lines 3, 4, 6, and 7 in Fig. 6) are commitments to ($\mathfrak{z}_i$-dependent) polynomials Λ_{0i}, Λ_{1i}, Λ_{2i}, and r, defined as in Eq. (8).

Recall that we analyze in the case the verifier individually tests the two verification equations in Eq. (11), ignoring the optimization induced by using the batching variable u. Let

$$\begin{aligned} \bar{H}_{ijk} &\leftarrow v_{ijk}^0 \cdot 0 + v_{ijk}^1\bar{a}_i + v_{ijk}^2\bar{b}_i + v_{ijk}^3\bar{c}_i + v_{ijk}^4\bar{s}_{\sigma 1i} + v_{ijk}^5\bar{s}_{\sigma 2i} + v_{ijk}^6\bar{t}_{\mathfrak{z}ij}\ , \\ [\mathsf{t}_{\delta_{ij}}]_1 &\leftarrow [t_{lo} + \delta_{ij}t_{mid} + \delta_{ij}^2 t_{hi}]_1\ , \\ [\mathsf{H}_{ijk}]_1 &\leftarrow v_{ijk}^0[r_i]_1 + v_{ijk}^1[a]_1 + v_{ijk}^2[b]_1 + v_{ijk}^3[c]_1 + v_{ijk}^4[s_{\sigma 1}]_1 + v_{ijk}^5[s_{\sigma 2}]_1 + v_{ijk}^6[\mathsf{t}_{\delta_{ij}}]_1. \end{aligned} \tag{14}$$

Since KZG is triply homomorphic, tr_{ijk} is an accepting Plonk transcript iff $([\mathsf{H}_{ijk}]_1, \mathfrak{z}_i, \bar{H}_{ijk}, [\mathsf{W}_{\mathfrak{z}ijk}]_1)$ and $\mathsf{k.tr}_i^{\omega}$ (line 17 in Fig. 6) are accepting KZG transcripts. We get this by slight rewriting of Plonk's verification equation. In particular, $\mathsf{k.tr}_i^{\omega}$ are accepting transcripts, with different values of $\mathfrak{z}_i$.

We will separate the rest of the proof to the case of (1) Plonk and SmallPlonk, and (2) SanPlonk. However, the subroutine on Fig. 6 corresponds to both.

Plonk And SmallPlonk. Here,

$$\boldsymbol{V}_i = \begin{pmatrix} 1 & v_{i1} & \dots & v_{i1}^5 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & v_{i6} & \dots & v_{i6}^5 \end{pmatrix} , \tag{15}$$

is an invertible Vandermonde matrix. According to Eq. (14), $(\bar{H}_{i1}, \dots, \bar{H}_{i6})^\intercal = \boldsymbol{V}_i \cdot (0, \bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{s}_{\sigma 1 i}, \bar{s}_{\sigma 2 i})^\intercal$ and $[\mathsf{H}_{i1}, \dots, \mathsf{H}_{i6}]_1^\intercal = \boldsymbol{V}_i \cdot [r_i, a, b, c, s_{\sigma 1}, s_{\sigma 2}]_1^\intercal$. Let $[\mathsf{W}'_{\mathfrak{z}i1}, \dots, \mathsf{W}'_{\mathfrak{z}i6}]_1^\intercal$ be as on line 8 of Fig. 6. By the triple homomorphism of KZG, $\mathsf{k.tr}_{ki}$ are accepting KZG transcripts for every i and k.

SanPlonk. Here,

$$\boldsymbol{V}_{ij} = \begin{pmatrix} 1 & v_{ij1} & \dots & v_{ij1}^6 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & v_{ij7} & \dots & v_{ij7}^6 \end{pmatrix} \quad \text{and} \quad \boldsymbol{C}_i := \begin{pmatrix} 1 & \delta_{i1} & \delta_{i1}^2 \\ 1 & \delta_{i2} & \delta_{i2}^2 \\ 1 & \delta_{i3} & \delta_{i3}^2 \end{pmatrix} \tag{16}$$

are invertible Vandermonde matrices. According to Eq. (14), $[\mathsf{H}_{ij1}, \dots, \mathsf{H}_{ij7}]_1 = \boldsymbol{V}_{ij} \cdot [r_i, a, b, c, s_{\sigma 1}, s_{\sigma 2}, \mathsf{t}_{\delta_{ij}}]_1^\intercal$ and $(\bar{H}_{ij1}, \dots, \bar{H}_{ij7})^\intercal = \boldsymbol{V}_{ij} \cdot (0, \bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{s}_{\sigma 1 i}, \bar{s}_{\sigma 2 i}, \bar{t}_{\mathfrak{z}ij})^\intercal$. Let $[\mathsf{W}'_{\mathfrak{z}ij1}, \dots, \mathsf{W}'_{\mathfrak{z}ij7}]_1^\intercal$ be defined as on line 8 of Fig. 6. By triple homomorphism, $\mathsf{k.tr}_{1i}, \dots, \mathsf{k.tr}_{6i}$ and $\mathsf{k.tr}_{7i}^*$ are accepting KZG transcripts, where $\mathsf{k.tr}_{1i}$ to $\mathsf{k.tr}_{6i}$ are as in Fig. 6 and $\mathsf{k.tr}_{7i}^* := ([\mathsf{t}_{\delta_{ij}}]_1, \mathfrak{z}_i, \bar{t}_{\mathfrak{z}ij}, [\mathsf{W}'_{\mathfrak{z}ij7}]_1)$.

Next, $\boldsymbol{C}_i \cdot [t_{lo}, t_{mid}, t_{hi}]_1^\intercal = [\mathsf{t}_{\delta_{i1}}, \mathsf{t}_{\delta_{i2}}, \mathsf{t}_{\delta_{i3}}]_1^\intercal$. Define $(\bar{t}_{\mathfrak{z}lo,i}, \bar{t}_{\mathfrak{z}mid,i}, \bar{t}_{\mathfrak{z}hi,i})^\intercal := \boldsymbol{C}_i^{-1} \cdot (\bar{t}_{\mathfrak{z}i1}, \bar{t}_{\mathfrak{z}i2}, \bar{t}_{\mathfrak{z}i3})^\intercal$ and $[W_{lo,i}, W_{mid,i}, W_{hi,i}]_1 := \boldsymbol{C}_i^{-1} [\mathsf{W}'_{\mathfrak{z}i17}, \mathsf{W}'_{\mathfrak{z}i27}, \mathsf{W}'_{\mathfrak{z}i37}]_1^\intercal$. Thus, $\mathsf{k.tr}_{7i}$, $\mathsf{k.tr}_{8i}$, and $\mathsf{k.tr}_{9i}$ are accepting KZG transcripts for every i. □

Constructing the Subtree Extractor.

Theorem 5 (Subtree extractor). *Let $\mathcal{T}$ be a $\boldsymbol{\kappa}_{\mathsf{Plonk}}$-tree of Plonk's (resp., $\boldsymbol{\kappa}_{\mathsf{san}}$-tree of SanPlonk's or $\boldsymbol{\kappa}_{\mathsf{SmallPlonk}}$-tree of SmallPlonk's) accepting transcripts. Let $\hat{\mathcal{T}}_{\beta\gamma\alpha} = (\mathsf{tr}_{ijk})$ be a subtree of $\mathcal{T}$ for any fixed β, γ, and α, where tr_{ijk} are as in Eq. (13). Assume that KZG is evaluation-binding and computational $(\kappa_{\mathsf{kzg}} + 1)$-special-sound. In the case of Plonk (resp., SmallPlonk), assume the $\mathsf{par}_n^{\mathsf{plonk}}$-SplitRSDH (resp., $\mathsf{par}_n^{\mathsf{smallplonk}}$-SplitRSDH) assumption holds. There exists a DPT extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$ that, given $\hat{\mathcal{T}}_{\beta\gamma\alpha}$, outputs $(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X))$, where $\mathsf{z}(X)$, $\mathsf{a}(X)$, $\mathsf{b}(X)$, and $\mathsf{c}(X)$ are consistent with the commitments and all $\kappa_{\mathfrak{z}}$ openings of $[z, a, b, c]_1$. Moreover, $\mathsf{t}(X)$ (defined as in Eq. (7)) is a polynomial.*

We postpone the proof of this result to Appendix D.2.

6.5 Full Special-Soundness Proof

Finally, we are ready to prove the special-soundness of Plonk, SanPlonk, and SmallPlonk. For this, we combine the subtree analysis of Section 6.4, KZG's

binding (required to guarantee that extracted polynomials like $\mathsf{a}(X)$ are the same in all subtrees), and an analysis of the permutation argument.

Theorem 6. *Let $n \in \mathsf{poly}(\lambda)$ and κ_{kzg}, $\boldsymbol{\kappa}_{\mathsf{Plonk}}$, and $\boldsymbol{\kappa}_{\mathsf{san}}$ be as in Eq.* (12).

1. *If KZG is computational $(\kappa_{\mathsf{kzg}}+1)$-special-sound and evaluation-binding, and $\mathsf{par}_n^{\mathsf{smallplonk}}$-SplitRSDH holds, then* SmallPlonk *is computational $\boldsymbol{\kappa}_{\mathsf{Plonk}}$-special-sound.*
2. *If KZG is computational $(\kappa_{\mathsf{kzg}}+1)$-special-sound and evaluation-binding, and $\mathsf{par}_n^{\mathsf{plonk}}$-SplitRSDH holds, then* Plonk *is computational $\boldsymbol{\kappa}_{\mathsf{Plonk}}$-special-sound.*
3. *If KZG is computational $(\kappa_{\mathsf{kzg}}+1)$-special-sound and evaluation-binding, then* SanPlonk *is computational $\boldsymbol{\kappa}_{\mathsf{san}}$-special-sound.*

Due to the page limit, we postpone the proof to Appendix D.5. Note that we prove special-soundness for the information-theoretic argument underlying Plonk as an implicit sub-claim in the proof of the previous theorem. To our knowledge, this is the first time that such a strong security property has been proven for Plonk's information-theoretic component.

Acknowledgments. We thank Antonio Faonio for useful discussions about [FFR24]. The first author was co-funded by the European Union and Estonian Research Council via 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. 1, D.6, 10

AFKR23. Thomas Attema, Serge Fehr, Michael Klooß, and Nicolas Resch. The fiat–shamir transformation of $(\gamma_1, \ldots, \gamma_\mu)$-special-sound interactive proofs. Technical Report 2023/1945, IACR, December 22, 2023. URL: https://eprint.iacr.org/2023/1945. 1, 1.1, D.6

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, 2.1

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, 2.1

BFHK23. Balthazar Bauer, Pooya Farshim, Patrick Harasser, and Markulf Kohlweiss. The Uber-Knowledge Assumption: A Bridge to the AGM. Technical Report 2023/1601, IACR, November 9, 2023. URL: https://eprint.iacr.org/2023/1601. 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, 2.1

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, 4.1

CFF+24. Matteo Campanelli, Antonio Faonio, Dario Fiore, Tianyu Li, and Helger Lipmaa. Lookup arguments: Improvements, extensions and applications to zero-knowledge decision trees. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14602 of *LNCS*, pages 337–369. Springer, Cham, April 2024. doi:10.1007/978-3-031-57722-2_11. 1

CGH98. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998. doi:10.1145/276698.276741. 1

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, 1.1, 2.1, 4.1, 4.2

Den02. Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 100–109. Springer, Berlin, Heidelberg, December 2002. doi:10.1007/3-540-36178-2_6. 1

DG23. Quang Dao and Paul Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 531–562. Springer, Cham, April 2023. doi:10.1007/978-3-031-30617-4_18. 1, 1.1

DP23. Benjamin E. Diamond and Jim Posen. Succinct arguments over towers of binary fields. Technical Report 2023/1784, IACR, November 17, 2023. URL: https://eprint.iacr.org/2023/1784. 1

EFG22. Liam Eagen, Dario Fiore, and Ariel Gabizon. cq: Cached quotients for fast lookups. Cryptology ePrint Archive, Report 2022/1763, 2022. URL: https://eprint.iacr.org/2022/1763. 1

FFR24. Antonio Faonio, Dario Fiore, and Luigi Russo. Real-world Universal zk-SNARKs Are Non-Malleable. In *ACM CCS 2024*, pages ?–?, Salt Lake City, USA, October 14–18, 2024. ACM. Accepted. 1, 1, 1.1, 3, 5, 4.2, 4.2, 4.2, 4.3, 6.5, A.1, A.2

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, 3, 4, 3

GKM+18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Cham, August 2018. doi:10.1007/978-3-319-96878-0_24. 1

GKP22. Chaya Ganesh, Hamidreza Khoshakhlagh, and Roberto Parisella. Niwi and new notions of extraction for algebraic languages. In Clemente Galdi and Stanislaw Jarecki, editors, *Security and Cryptography for Networks*, pages 687–710, Cham, 2022. Springer International Publishing. 1.1

GLS+23. Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 193–226. Springer, Cham, August 2023. doi:10.1007/978-3-031-38545-2_7. 1

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. doi:10.1145/1993636.1993651. 1

GW19. Ariel Gabizon and Zachary J. Williamson. The turbo-plonk program syntax for specifying snark programs, 2019. URL: https://docs.zkproof.org/pages/standards/accepted-workshop3/proposal-turbo_plonk.pdf. 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. 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, 1, 1.1, 2, 4.1, 4.2, 6, 6.1, 6.1, D.1, E

Ica09. Thomas Icart. How to hash into elliptic curves. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 303–316. Springer, Berlin, Heidelberg, August 2009. doi:10.1007/978-3-642-03356-8_18. 3, 3

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. 1, 1, 1.1, 3, 3, 3, C

JR10. Tibor Jager and Andy Rupp. The semi-generic group model and applications to pairing-based cryptography. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 539–556. Springer, Berlin, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_31. 6

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, 1.1, 2.1, 2.1, 3

Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Berlin, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_10. 2.1, 3, C

Lip24. Helger Lipmaa. Polymath: Groth16 is not the limit. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 170–206. Springer, Cham, August 2024. doi:10.1007/978-3-031-68403-6_6. 1

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, 3, 1.1, 2.1, 3, 3, 4, 3, 4.2, 4.3, A.1, C, C

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, 1.1, 2, 2.1, 2.1, 4

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, 4.1

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, December 2005. doi:10.1007/11586821_1. 1, 3

PST13. Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 222–242. Springer, Berlin, Heidelberg, March 2013. doi:10.1007/978-3-642-36594-2_13. 2.1

RZ21. Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, August 2021. Springer, Cham. doi:10.1007/978-3-030-84242-0_27. 1, 4.1

Sef24. Marek Sefranek. How (not) to simulate PLONK. Technical Report 2024/848, IACR, May 31, 2024. URL: https://eprint.iacr.org/2024/848. 1.1, E, 7, E

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18. 1, 3

STW24. Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Unlocking the lookup singularity with Lasso. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 180–209. Springer, Cham, May 2024. doi:10.1007/978-3-031-58751-1_7. 1

Zha22. Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Cham, August 2022. doi:10.1007/978-3-031-15982-4_3. 1, 3

ZZK22. Cong Zhang, Hong-Sheng Zhou, and Jonathan Katz. An analysis of the algebraic group model. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 310–322. Springer, Cham, December 2022. doi:10.1007/978-3-031-22972-5_11. 1, 3

A Insecurity of Lin

A.1 Attack Against Knowledge-Soundness of Lin.

For the sake of completeness, we reproduce the attack of [LPS23,FFR24] against knowledge-soundness of Lin.

The adversary $\mathcal{A}$ chooses some tuple $\{\mathsf{a}_\jmath(X)\}$ of polynomials, such that $V := \{\mathsf{g}_t(\mathbf{a}(X))\}_{t=1}^{n_b}$ is linearly dependent. $\mathcal{A}$ chooses a non-zero vector $\bar{\boldsymbol{d}}^*$ that is orthogonal to V, i.e., $\sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}(X))\bar{d}_t^* = 0$. $\mathcal{A}$ samples $[D]_1 \leftarrow_\$ \mathbb{G}_1$ obliviously, and sets $[d_t]_1 \leftarrow \bar{d}_t^*[D]_1$. $\mathcal{A}$ sets $\bar{a}_\jmath \leftarrow \mathsf{a}_\jmath(\mathfrak{z})$ and $[h]_1 \leftarrow [(\sum_{\jmath=1}^{n_a} \beta^{\jmath-1}(\mathsf{a}_\jmath(x) - \bar{a}_\jmath))/(x-\mathfrak{z})]_1$. Clearly,

$$\textstyle\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\mathbf{a}})d_t = \sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}(\mathfrak{z}))\bar{d}_t^* D = (\sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}(\mathfrak{z}))\bar{d}_t^*)D = 0 \ ,$$

and thus

$$\begin{aligned}[\textstyle\sum_{\jmath=1}^{n_a} \beta^{\jmath-1}(\mathsf{a}_\jmath(x) - \bar{a}_\jmath) + \beta^{n_a} \sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\mathbf{a}})d_t]_1 \bullet [1]_2 &= [\textstyle\sum_{\jmath=1}^{n_a} \beta^{\jmath-1}(\mathsf{a}_\jmath(x) - \bar{a}_\jmath)]_1 \\ &= [h]_1 \bullet [x-\mathfrak{z}]_2 \ .\end{aligned}$$

Thus, the Lin verifier accepts, but it is impossible to extract from obliviously sampled $[D]_1$ and thus from any of $[d_t]_1$.

A.2 Proof of Theorem 3.

Finally, we give the proof of Theorem 3, which shows that linear independence is insufficient for proving computational special-soundness of Lin.

Proof. We prove impossibility for the next special case. The general case follows. For the rest of the proof, let us fix some arbitrary polynomials $\mathsf{a}_1, \mathsf{a}_2$, $\mathsf{d}_1', \mathsf{d}_2' \in \mathbb{F}_{\le n}[X]$, and $\mathsf{g}_1, \mathsf{g}_2 \in \mathbb{F}[X]$, which satisfy the equation $\mathsf{g}_1(\mathsf{a}_1(X))\mathsf{d}_1'(X) + \mathsf{g}_2(\mathsf{a}_2(X))\mathsf{d}_2'(X) = 0$. In particular, $\mathsf{g}_1(X)$ and $\mathsf{g}_2(X)$ can be linearly independent, as was required in [FFR24]. For instance, one can choose $\mathsf{g}_1(X) = X$, and $\mathsf{g}_2 = 1$.

We also define $\mathsf{d}_1(X) = y\mathsf{d}_1'(X)$ and $\mathsf{d}_2(X) = y\mathsf{d}_2'(X)$ for a randomly chosen $y \leftarrow_\$ \mathbb{F}$ (y will later act as a DL challenge). Observe that $\mathsf{g}_1(\mathsf{a}_1(X))\mathsf{d}_1(X) + \mathsf{g}_2(\mathsf{a}_2(X))\mathsf{d}_2(X) = 0$. We show that if Lin has computational (κ_1, κ_2)-special-soundness (for any $\kappa_1, \kappa_2 \in \mathsf{poly}(\lambda)$) for committed polynomials $\mathsf{a}_1, \mathsf{a}_2, \mathsf{d}_1, \mathsf{d}_2$ and public polynomials $\mathsf{g}_1, \mathsf{g}_2$ as defined above, then it is possible to break the discrete logarithm assumption in $\mathbb{G}_1$.

Let $\mathsf{Ext}_{\mathsf{ss}}$ be a PPT extractor that gets as an input $\mathsf{ck} = ([(x^i)_{i=1}^n]_1, [x]_2)$ and $\mathsf{tr}_{ij} = ([a_1, d_1, a_2, d_2]_1, \mathfrak{z}_i, A_{1i}, A_{2i}, \beta_{ij}, [h_{ij}]_1)$, for $i, j \in [1, \kappa_1] \times [1, \kappa_2]$, such that

$$[(a_1 - A_{1i}) + \beta_{ij}(a_2 - A_{2i}) + \beta_{ij}^2(\mathsf{g}_1(A_{1i})d_1 + \mathsf{g}_1(A_{2i})d_2)]_1 \bullet [1]_2 = [h_{ij}]_1 \bullet [x - \mathfrak{z}_i]_2 \ .$$

Assume that for any computational special-soundness adversary $\mathcal{B}$, $\mathsf{Ext}_{\mathsf{ss}}$ outputs a correct tuple $(\mathsf{a}_1(X), \mathsf{d}_1(X), \mathsf{a}_2(X), \mathsf{d}_2(X)) \in \mathcal{R}_{\mathsf{ck},\mathbf{g},\mathsf{tr}}^{\mathsf{Lin}}$ with probability at least $1 - \varepsilon_{\mathcal{A}}$.

$\mathcal{A}(\mathsf{p}, [y]_1)$:

$x \leftarrow_\$ \mathbb{F}$; $\mathsf{ck} \leftarrow ([(x^i)_{i=1}^n]_1, [x]_2)$;
$[a_1]_1 \leftarrow \mathsf{a}_1(x)[1]_1$; $[a_2]_1 \leftarrow \mathsf{a}_2(x)[1]_1$; $[d_1]_1 \leftarrow \mathsf{d}'_1(x)[y]_1$; $[d_2]_1 \leftarrow \mathsf{d}'_2(x)[y]_1$;
for $i \in [1, \kappa_1]$ **do**
  $\mathfrak{z}_i \leftarrow i$; $A_{1i} \leftarrow a_1(\mathfrak{z}_i)$; $A_{2i} \leftarrow a_2(\mathfrak{z}_i)$;
  for $j \in [1, \kappa_2]$ **do**
    $\beta_{ij} \leftarrow j$;
    $[h_{ij}]_1 \leftarrow \frac{[(a_1 - A_{1i}) + \beta_{ij}(a_2 - A_{2i})]_1}{x - \mathfrak{z}_i} + \frac{\beta_{ij}^2(\mathsf{g}_1(A_{1i})[d_1]_1 + \mathsf{g}_2(A_{2i})[d_2]_1)}{x - \mathfrak{z}_i}$;
    $\mathsf{tr}_{ij} \leftarrow ([a_1, d_1, a_2, d_2]_1, \mathfrak{z}_i, A_{1i}, A_{2i}, \beta_{ij}, [h_{ij}]_1)$;
$(\tilde{a}_1(X), \tilde{d}_1(X), \tilde{a}_2(X), \tilde{d}_2(X)) \leftarrow \mathsf{Ext}_{\mathsf{ss}}(\mathsf{p}, \mathsf{ck}, \mathbf{tr})$;
return $\tilde{d}_1(x)/d'_1(x)$;

$\mathcal{B}(\mathsf{p}, \mathsf{ck})$:

$y \leftarrow_\$ \mathbb{F}$;
$[a_1]_1 \leftarrow [a_1(x)]_1$; $[a_2]_1 \leftarrow [a_2(x)]_1$; $[d_1]_1 \leftarrow y[d'_1(x)]_1$; $[d_2]_1 \leftarrow y[d'_2(x)]_1$;
for $i \in [1, \kappa_1]$ **do**
  $\mathfrak{z}_i \leftarrow i$; $A_{1i} \leftarrow \mathsf{a}_1(\mathfrak{z}_i)$; $A_{2i} \leftarrow \mathsf{a}_2(\mathfrak{z}_i)$;
  for $j \in [1, \kappa_2]$ **do**
    $\beta_{ij} \leftarrow j$;
    $h_{ij}(X) \leftarrow \frac{a_1(X) - A_{1i} + \beta_{ij}(a_2(X) - A_{2i})}{X - \mathfrak{z}_i} + \frac{\beta_{ij}^2(\mathsf{g}_1(A_{1i})yd'_1(X) - \mathsf{g}_2(A_{2i})yd'_2(X))}{X - \mathfrak{z}_i}$;
    $[h_{ij}]_1 = [h_{ij}(x)]_1$;
    $\mathsf{tr}_{ij} \leftarrow ([a_1, d_1, a_2, d_2]_1, \mathfrak{z}_i, A_{1i}, A_{2i}, \beta_{ij}, [h_{ij}]_1)$;
return $\mathbf{tr}$;

Fig. 7. Discrete logarithm adversary $\mathcal{A}$ and computational special-soundness adversary $\mathcal{B}$ for any fixed polynomials $\mathsf{a}_1, \mathsf{a}_2, \mathsf{d}'_1, \mathsf{d}'_2 \in \mathbb{F}_{\leq n}[X]$, and $\mathsf{g}_1, \mathsf{g}_2 \in \mathbb{F}[X]$ that satisfy $\mathsf{g}_1(\mathsf{a}_1(X))\mathsf{d}'_1(X) + \mathsf{g}_2(\mathsf{a}_2(X))\mathsf{d}'_2(X) = 0$.

In Fig. 7, we depict a PPT adversary $\mathcal{A}$ that breaks DL with probability $\geq 1 - \varepsilon$. $\mathcal{A}$ gets as an input a DL challenge $[y]_1$ for $y \leftarrow_\$ \mathbb{F}$. $\mathcal{A}$ samples a new trapdoor x and generates ck from x. $\mathcal{A}$ computes commitments $[a_1]_1 \leftarrow [\mathsf{a}_1(x)]_1, [a_2]_1 \leftarrow [\mathsf{a}_2(x)]_1$, $[d_1]_1 \leftarrow \mathsf{d}'_1(x)[y]_1$, and $[d_2]_1 \leftarrow \mathsf{d}'_2(x)[y]_1$. ($\mathcal{A}$ can compute $[d_1, d_2]_1$ only since it knows x.) $\mathcal{A}$ then generates transcripts tr_{ij} for $\mathfrak{z}_i = i$ and $\beta_{ij} = j$, with the openiing proofs

$$[h_{ij}]_1 = \frac{[(a_1 - A_{1i}) + \beta_{ij}(a_2 - A_{2i}) + \beta_{ij}^2(\mathsf{g}_1(A_{1i})\tilde{d}_1(X) - \mathsf{g}_2(A_{2i})\tilde{d}_2(X))]_1}{x - \mathfrak{z}_i} .$$

Finally, it runs $\mathsf{Ext}_{\mathsf{ss}}(\mathsf{p}, \mathsf{ck}, \mathbf{tr})$ to obtain polynomials $(\tilde{\mathsf{a}}_1(X), \tilde{\mathsf{d}}_1(X), \tilde{\mathsf{a}}_2(X), \tilde{\mathsf{d}}_2(X))$ and outputs $\tilde{\mathsf{d}}_1(x)/\mathsf{d}'_1(x)$ as a candidate for the DL y.

Suppose $\mathcal{A}$ succeeds in breaking DL with probability $\varepsilon_{\mathcal{A}}$. We present a sequence of games to show that $\varepsilon_{\mathcal{A}} \geq 1 - \varepsilon$, which implies that ε cannot be negligible if the discrete logarithm assumption holds.

Game_1: The first game in Fig. 8 samples ck in the first step and generates a discrete logarithm challenge $[y]_1 \leftarrow_\$ \mathbb{G}_1$. The rest of the steps are identical to $\mathcal{A}$. The game outputs 1 when $(\tilde{\mathsf{d}}_1(x)/\mathsf{d}'_1(x))[1]_1 = [y]_1$. It is easy to see that the

Game_1 :

1 : $x \leftarrow_\$ \mathbb{F}; \mathsf{ck} \leftarrow ([(x^i)_{i=1}^n]_1, [x]_2)$;
2 : $[y]_1 \leftarrow_\$ \mathbb{G}_1$;
3 : $[a_1]_1 \leftarrow \mathsf{a}_1(x)[1]_1; [a_2]_1 \leftarrow \mathsf{a}_2(x)[1]_1; [d_1]_1 \leftarrow \mathsf{d}_1'(x)[y]_1; [d_2]_1 \leftarrow \mathsf{d}_2'(x)[y]_1$;
4 : **for** $i \in [1, \kappa_1]$ **do**
5 : $\mathfrak{z}_i \leftarrow i; \quad A_{1i} \leftarrow a_1(\mathfrak{z}_i); A_{2i} \leftarrow a_2(\mathfrak{z}_i)$;
6 : **for** $j \in [1, \kappa_2]$ **do**
7 : $\beta_{ij} \leftarrow j$;
8 : $[h_{ij}]_1 \leftarrow \frac{[(a_x - A_{1i}) + \beta_{ij}(a_2(x) - A_{2i})]_1}{x - \mathfrak{z}_i} + \frac{\beta_{ij}^2(\mathsf{g}_1(A_{1i})[d_1]_1 + \mathsf{g}_2(A_{2i})[d_2]_1)}{x - \mathfrak{z}_i}$;
9 :
10 : $\mathsf{tr}_{ij} \leftarrow ([a_1, d_1, a_2, d_2]_1, \mathfrak{z}_i, A_{1i}, A_{2i}, \beta_{ij}, [h_{ij}]_1)$;
11 : $(\tilde{\mathsf{a}}_1(X), \tilde{\mathsf{d}}_1(X), \tilde{\mathsf{a}}_2(X)\tilde{\mathsf{d}}_2(X)) \leftarrow \mathsf{Ext}_{\mathsf{ss}}(\mathsf{p}, \mathsf{ck}, \mathbf{tr})$;
12 : **return** $(\tilde{\mathsf{d}}_1(x)/\mathsf{d}_1(x))[1]_1 = [y]_1$;

Game_2 :

1 : $x \leftarrow_\$ \mathbb{F}; \mathsf{ck} \leftarrow ([(x^i)_{i=1}^n]_1, [x]_2)$;
2 : $y \leftarrow_\$ \mathbb{F}$;
3 : $[a_1]_1 \leftarrow [a_1(x)]_1; [a_2]_1 \leftarrow [a_2(x)]_1$;
4 : $[d_1]_1 \leftarrow y[d_1'(x)]_1; [d_2]_1 \leftarrow y[d_2'(x)]_1$;
5 : **for** $i \in [1, \kappa_1]$ **do**
6 : $\mathfrak{z}_i \leftarrow i$;
7 : $A_{1i} \leftarrow a_1(\mathfrak{z}_i); A_{2i} \leftarrow a_2(\mathfrak{z}_i)$;
8 : **for** $j \in [1, \kappa_2]$ **do**
9 : $\beta_{ij} \leftarrow j$;
10 : $h_{ij}(X) \leftarrow \frac{a_1(X) - A_{1i} + \beta_{ij}(a_2(X) - A_{2i})}{X - \mathfrak{z}_i} + \frac{\beta_{ij}^2(\mathsf{g}_1(A_{1i})yd_1'(X) - \mathsf{g}_2(A_{2i})yd_2'(X))}{X - \mathfrak{z}_i}$;
11 : $[h_{ij}]_1 = [h_{ij}(x)]_1$;
12 : $\mathsf{tr}_{ij} \leftarrow ([a_1, d_1, a_2, d_2]_1, \mathfrak{z}_i, A_{1i}, A_{2i}, \beta_{ij}, [h_{ij}]_1)$;
13 : $(\tilde{a}_1(X), \tilde{d}_1(X), \tilde{a}_2(X), \tilde{d}_2(X)) \leftarrow \mathsf{Ext}_{\mathsf{ss}}(\mathsf{p}, \mathsf{ck}, \mathbf{tr})$;
14 : **return** $(\tilde{d}_1(x)/d_1'(x))[1]_1 = [y]_1$;

Fig. 8. Games in the proof of Theorem 3. Here, $\mathsf{a}_1, \mathsf{a}_2, \mathsf{d}_1', \mathsf{d}_2' \in \mathbb{F}_{\le n}[X]$, and $\mathsf{g}_1, \mathsf{g}_2 \in \mathbb{F}[X]$ are arbitrary fixed polynomials that satisfy $\mathsf{g}_1(\mathsf{a}_1(X))\mathsf{d}_1'(X) + \mathsf{g}_2(\mathsf{a}_2(X))\mathsf{d}_2'(X) = 0$

probability that this game outputs 1 is $\varepsilon_{\mathcal{A}}$, the probability that $\mathcal{A}$ breaks the DL assumption in the DL game.

Game_2: The second game in Fig. 8 is a very slight modification of Game_1. The game samples $y \leftarrow_\$ \mathbb{F}$ instead of directly sampling the group element $[y]_1$. This game uses the knowledge of y (instead of knowledge of x) to simulate identically distributed transcripts. Namely, on step 4 it computes now $[d_1]_1 \leftarrow y[\mathsf{d}_1'(x)]_1$, $[d_2]_1 \leftarrow y[\mathsf{d}_2'(x)]_1$, and it uses the fact that $h_{ij}(X)$ on step 10 is a polynomial to compute $[h_{ij}]_1 = [h_{ij}(x)]_1$. The latter is true since, $\mathsf{g}_1(\mathsf{a}_1(\mathfrak{z}_i))\mathsf{d}_1(\mathfrak{z}_i) + \mathsf{g}_2(\mathsf{a}_2(\mathfrak{z}_i))\mathsf{d}_2(\mathfrak{z}_i) = 0$ for any $\mathfrak{z}_i = i$, which means that $\mathsf{g}_1(\mathsf{a}_1(\mathfrak{z}_i))\mathsf{d}_1(X) + \mathsf{g}_2(\mathsf{a}_2(\mathfrak{z}_i))\mathsf{d}_2(X)$ is divisible by $X - \mathfrak{z}_i$. Thus, we can efficiently compute $[h_{ij}(x)]_1$ by only using ck (and not x). In fact, Game_2 uses the discrete logarithm of $[x]_1$ only on the very first step when it computes ck. Since inputs to $\mathsf{Ext}_{\mathsf{ss}}$ are identically distributed to Game_1, the probability that Game_2 outputs 1 is also $\varepsilon_{\mathcal{A}}$.

Now we can take the steps 2 to 12 in Game_2 and write a computational special-soundness adversary $\mathcal{B}$ based on it as shown in Fig. 7. This is possible because these steps do not require knowledge of x. By inlining $\mathcal{B}$ to the special-soundness game, it is clear that $\Pr[\mathsf{Game}_2 \neq 1]$ is bounded by the advantage of $\mathcal{B}$ in the special-soundness game. Namely, $\mathsf{Game}_2 \neq 1$ happens only if $\tilde{\mathsf{d}}_1(x) \neq d_1$. If this is the case, then $(\tilde{\mathsf{a}}_1(X), \tilde{\mathsf{d}}_1(X), \tilde{\mathsf{a}}_2(X), \tilde{\mathsf{d}}_2(X)) \notin \mathcal{R}^{\mathsf{Lin}}_{\mathsf{ck},\mathbf{g}}$, which means that $\mathcal{B}$ wins the special-soundness game. This happens at most with probability ε as we assumed $\mathsf{Ext}_{\mathsf{ss}}$ to fail at most with probability ε in the beginning of the proof. Therefore, the probability that $\tilde{\mathsf{d}}_1(x)/d_{\mathsf{d}}(x) = y$ in Game_2 must be greater than $1 - \varepsilon$.

We conclude that $\varepsilon_{\mathcal{A}} \geq 1 - \varepsilon$, which implies that if ε is negligible (and thus special soundness holds) then we break the DL assumption. □

B Proof of SanLin

Before proving Theorem 4, we state the following lemma.

Lemma 3. *Assume that $\mathcal{T} = (\mathsf{tr}_{ijk})$ is a $\boldsymbol{\kappa}$-tree of SanLin's accepting transcripts, where tr_{ijk} are as in step 2 in Fig. 9. The PPT algorithm $\mathsf{TE}_{\mathsf{lin}}(\mathsf{ck}, \mathcal{T})$ in Fig. 9 computes accepting KZG transcripts $(\mathbf{tr}_{a_\jmath})_{\jmath=1}^{n_a}$ and $(\mathbf{tr}_{d_t})_{t=1}^{n_b}$, s.t. $\mathsf{tr}_{a_\jmath,i} = ([a_\jmath]_1, \boldsymbol{\jmath}_i, \ldots)$ and $\mathsf{tr}_{d_t,i} = ([d_t]_1, \boldsymbol{\jmath}_i, \ldots)$ for $i \in [1, n+1]$, and $\boldsymbol{\jmath}_i$ are mutually distinct.*

Proof. Let $\mathcal{T}$ be the given accepting tree of transcripts. For $i \in [1, \kappa_1]$ and $j \in [1, n_b]$, let

$$\boldsymbol{C}_i = \begin{pmatrix} 1 & \gamma_{i1} & \gamma_{i1}^2 & \vdots & \gamma_{i1}^{n_b-1} \\ 1 & \gamma_{i2} & \gamma_{i2}^2 & \vdots & \gamma_{i2}^{n_b-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma_{in_b} & \gamma_{in_b}^2 & \vdots & \gamma_{in_b}^{n_b-1} \end{pmatrix} \quad \text{and} \quad \boldsymbol{B}_{ij} = \begin{pmatrix} 1 & \beta_{ij1} & \cdots & \beta_{ij1}^{n_a+1} \\ 1 & \beta_{ij2} & \cdots & \beta_{ij2}^{n_a+1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \beta_{ij,n_a+2} & \cdots & \beta_{ij,n_a+2}^{n_a+1} \end{pmatrix} \tag{17}$$

be Vandemonde matrices. Since $\mathcal{T}$ is a tree of transcripts, $\boldsymbol{\jmath}_i$ are all distinct, and for each $\boldsymbol{\jmath}_i$, all γ_{ij} are disctinct, and for each γ_{ij}, all β_{ijk} are distinct. Thus, for each $i \in [1, \kappa_1]$ and $j \in [1, n_b]$, $\boldsymbol{C}_i$ and $\boldsymbol{B}_{ij}$ are non-singular.

By the construction of SanLin, tr_{ijk} is an accepting SanLin transcript iff

$$\mathsf{k.tr}_{ijk} = ([\varphi_{ijk}]_1, \boldsymbol{\jmath}_i, \boldsymbol{\Phi}_{ijk}, [h_{ijk}]_1)$$

is an accepting KZG transcript, where

$$\varphi_{ijk} := \textstyle\sum_{\jmath=1}^{n_a} \beta_{ijk}^{\jmath-1} a_\jmath + \beta_{ijk}^{n_a} \sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i) d_t + \beta_{ijk}^{n_a+1} \sum_{t=1}^{n_b} \gamma_{ij}^{t-1} d_t$$

and

$$\boldsymbol{\Phi}_{ijk} := \textstyle\sum_{\jmath=1}^{n_a} \beta_{ijk}^{\jmath-1} \bar{a}_{i\jmath} + \beta_{ijk}^{n_a+1} \bar{d}_{ij} \ .$$

Thus,

$$\begin{aligned} (a_1, \ldots, a_{n_a}, \textstyle\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i) d_t, \sum_{t=1}^{n_b} \gamma_{ij}^{t-1} d_t)^\intercal &= \boldsymbol{B}_{ij}^{-1} \cdot (\varphi_{ij1}, \ldots, \varphi_{ij,n_a+2})^\intercal \ , \\ (\bar{a}_{i1}, \ldots, \bar{a}_{in_a}, 0, \bar{d}_{ij})^\intercal &= \boldsymbol{B}_{ij}^{-1} \cdot (\boldsymbol{\Phi}_{ij1}, \ldots, \boldsymbol{\Phi}_{ij,n_a+2})^\intercal \ . \end{aligned}$$

$\mathsf{TE}_{\mathsf{lin}}(\mathsf{ck}, \mathcal{T})$

1 : Parse $\mathcal{T} = (\mathsf{tr}_{ijk})_{i\in[1,\kappa_1],j\in[1,n_b],k\in[1,n_a+2]}$;
2 : Parse $\mathsf{tr}_{ijk} = ([(a_\imath)_{\imath=1}^{n_a}, (d_t)_{t=1}^{n_b}]_1, \mathfrak{z}_i, (\bar{a}_{it})_{t=1}^{n_b}, \gamma_{ij}, \bar{d}_{ij}, \beta_{ijk}, [h_{ijk}]_1)$;
3 : **for** $i \in [1, \kappa_1]$ **do for** $j \in [1, n_b]$ **do**
4 : $\quad [h^*_{ij1}, \ldots, h^*_{ij,n_a+2}]_1^\intercal \leftarrow \boldsymbol{B}_{ij}^{-1}[h_{ij1}, \ldots, h_{ij,n_a+2}]_1^\intercal$;
5 : $\quad$ **for** $\imath \in [1, n_a]$ **do** $\mathsf{k.tr}^*_{ij\imath} \leftarrow ([a_\imath]_1, \mathfrak{z}_i, \bar{a}_{i\imath}, [h^*_{ij\imath}]_1)$;
6 : $\quad \mathsf{k.tr}^*_{ij,n_a+1} \leftarrow ([\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i)d_t]_1, \mathfrak{z}_i, 0, [h^*_{ij,n_a+1}]_1)$;
7 : $\quad \mathsf{k.tr}^*_{ij,n_a+2} \leftarrow ([\sum_{t=1}^{n_b} \gamma_{ij}^{t-1} d_t]_1, \mathfrak{z}_i, \bar{d}_{ij}, [h^*_{ij,n_a+2}]_1)$;
8 : **for** $\imath \in [1, n_a]$ **do**
9 : $\quad$ **for** $i \in [1, \kappa_1]$ **do** $\mathsf{k.tr}_{a_\imath,i} \leftarrow \mathsf{k.tr}^*_{i1\imath}$;
10 : $\quad \mathbf{tr}_{a_\imath} \leftarrow (\mathsf{k.tr}_{a_\imath,1}, \ldots, \mathsf{k.tr}_{a_\imath,n+1})$;
11 : **for** $i \in [1, \kappa_1]$ **do**
12 : $\quad \begin{pmatrix} \bar{d}'_{i1} \\ \cdots \\ \bar{d}'_{in_b} \end{pmatrix} \leftarrow \boldsymbol{C}_i^{-1} \begin{pmatrix} \bar{d}_{i1} \\ \cdots \\ \bar{d}_{in_b} \end{pmatrix}; \begin{bmatrix} h'_{i1} \\ \cdots \\ h'_{in_b} \end{bmatrix}_1 \leftarrow \boldsymbol{C}_i^{-1} \begin{bmatrix} h^*_{i1,n_a+2} \\ \cdots \\ h^*_{in_b,n_a+2} \end{bmatrix}_1$;
13 : $\quad$ **for** $t \in [1, n_b]$ **do**
14 : $\quad\quad \mathsf{k.tr}_{d_t,i} \leftarrow ([d_t]_1, \mathfrak{z}_i, \bar{d}'_{it}, [h'_{it}]_1)$;
15 : $\quad \mathbf{tr}_{d_t} \leftarrow (\mathsf{k.tr}_{d_t,1}, \ldots, \mathsf{k.tr}_{d_t,n+1})$;
16 : **return** $((\mathbf{tr}_{a_\imath})_{\imath=1}^{n_a}, (\mathbf{tr}_{d_t})_{t=1}^{n_b})$;

Fig. 9. The $\mathsf{TE}_{\mathsf{lin}}$ subroutine.

Since KZG is triply homomorphic, for all $i, j, \imath$, $\mathsf{k.tr}^*_{ij\imath}$ (see Steps 5, 6, 7 in Fig. 9) are accepting KZG transcripts. Thus, one can define $\mathsf{tr}_{a_\imath,i} = \mathsf{k.tr}^*_{ij\imath}$ for $j = 1$ (one can choose any value of j).

Let $h^*_{i1,n_a+2}, \ldots, h^*_{in_b,n_a+2}$ be as in Step 4 of Fig. 9. We define $(\bar{d}'_{i1}, \ldots, \bar{d}'_{in_b})^\intercal \leftarrow \boldsymbol{C}_i^{-1} \cdot (\bar{d}_{i1}, \ldots, \bar{d}_{in_b})^\intercal$ and $[h'_{i1}, \ldots, h'_{in_b}]_1^\intercal \leftarrow \boldsymbol{C}_i^{-1} \cdot [h^*_{i1,n_a+2}, \ldots, h^*_{in_b,n_a+2}]_1^\intercal$. Moreover, the vector of first elements $[\sum_{t=1}^{n_b} \gamma_{ij}^{t-1} d_t]_1$ of $\mathsf{k.tr}^*_{ij,n_a+2}$ is equal to $\boldsymbol{C}_i \cdot [d_1, \ldots, d_{n_b}]_1^\intercal$. Since KZG is triple homomorphic and $\mathsf{k.tr}^*_{ij,n_a+2}$ are accepting, $\mathsf{k.tr}_{d_t i} \leftarrow ([d_t]_1, \mathfrak{z}_i, \bar{d}'_{it}, [h'_{it}]_1)$ is an accepting KZG transcript. □

Next, we prove Theorem 4.

Proof (Theorem 4). Recall that a valid witness contains $n_a + n_b$ polynomials $(\mathsf{a}^*_\imath(X))_{\imath=1}^{n_a}$ and $(\mathsf{d}^*_t(X))_{t=1}^{n_b}$ of degree at most n, such that $\sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}^*(X))\mathsf{d}^*_t(X) = 0$ and $\mathsf{a}^*_\imath(x) = a_\imath$, $\mathsf{d}^*_t(x) = d_t$ for $\imath \in [1, n_a]$, $t \in [1, n_b]$.

Recall $\boldsymbol{\kappa} = (\kappa_1, n_b, n_a + 2)$. Let $\mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}$ be an $(n+1)$-special-soundness extractor $\mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}$ of KZG. We depict the $\boldsymbol{\kappa}$-special-soundness extractor $\mathsf{Ext}^{\mathsf{sanlin}}_{\mathsf{ss}}$ for SanLin in Fig. 10. It has blackbox access to $\mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}$. On input a $\boldsymbol{\kappa}$-tree of SanLin accepting transcripts, $\mathsf{Ext}^{\mathsf{sanlin}}_{\mathsf{ss}}$ calls $\mathsf{TE}_{\mathsf{lin}}$ from Fig. 9 that returns accepting KZG transcripts $\mathbf{tr}_{a_\imath}, \mathbf{tr}_{d_t}$ for $\imath \in [1, n_a]$, $t \in [1, n_b]$. As stated in Lemma 3, each $\mathsf{tr}_{a_\imath i}$ (resp., $\mathsf{tr}_{d_t i}$) contains a transcript for the commitment $[a_\imath]_1$ (resp., $[d_t]_1$) with a distinct evaluation point $\mathfrak{z}_i$. We feed them separately into KZG's $(n+1)$-special-soundness extractor $\mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}$ to extract all $\mathsf{a}^*_\imath(X)$ and $\mathsf{d}^*_t(X)$.

$\mathsf{Ext}^{\mathsf{sanlin}}_{\mathsf{ss}}(\mathsf{ck}, \mathcal{T})$

$((\mathbf{tr}_{a_\imath})_{\imath=1}^{n_a}, (\mathbf{tr}_{d_t})_{t=1}^{n_b}) \leftarrow \mathsf{TE}_{\mathsf{lin}}(\mathsf{ck}, \mathcal{T})$;
for $\imath \in [1, n_a]$ **do**
  $\mathsf{a}^*_\imath(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{tr}_{a_\imath})$;
for $t \in [1, n_b]$ **do**
  $\mathsf{d}^*_t(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{tr}_{d_t})$;
return $((\mathsf{a}^*_\imath(X))_{\imath=1}^{n_a}, (\mathsf{d}^*_t(X))_{t=1}^{n_b})$;

$\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck})$

$\mathcal{T} \leftarrow \mathcal{A}(\mathsf{ck})$;
$((\mathbf{tr}_{a_\imath})_{\imath=1}^{n_a}, (\mathbf{tr}_{d_t})_{t=1}^{n_b}) \leftarrow \mathsf{TE}_{\mathsf{lin}}(\mathsf{ck}, \mathcal{T})$;
for $\imath \in [1, n_a]$ **do**
  $\mathsf{a}^*_\imath(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{tr}_{a_\imath})$;
  if $([a_\imath]_1, \mathsf{a}^*_\imath(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{tr}_{a_\imath}}$ **then return** $\mathbf{tr}_{a_\imath}$;
for $t \in [1, n_b]$ **do**
  $\mathsf{d}^*_t(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{tr}_{d_t})$;
  if $([d_t]_1, \mathsf{d}^*_t(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{tr}_{d_t}}$ **then return** $\mathbf{tr}_{d_t}$;
return $\bot$;

$\mathcal{C}_{\mathsf{evb}}(\mathsf{ck})$

$\mathcal{T} \leftarrow \mathcal{A}(\mathsf{ck})$; $((\mathbf{tr}_{a_\imath})_{\imath=1}^{n_a}, (\mathbf{tr}_{d_t})_{t=1}^{n_b}) \leftarrow \mathsf{TE}_{\mathsf{lin}}(\mathsf{ck}, \mathcal{T})$;
for $\imath \in [1, n_a]$ **do** $\mathsf{a}^*_\imath(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{tr}_{a_\imath})$;
for $t \in [1, n_b]$ **do** $\mathsf{d}^*_t(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{tr}_{d_t})$;
Parse $((\mathbf{tr}_{a_\imath})_{\imath=1}^{n_a}, (\mathbf{tr}_{d_t})_{t=1}^{n_b})$ as in Fig. 9;
for $i \in [n+2, \kappa_1]$ **do**
  for $\imath \in [1, n_a]$ **do**
    if $\bar{a}_{i\imath} \neq \mathsf{a}^*_\imath(\mathfrak{z}_i)$ **then return** $\left([a_\imath]_1, \mathfrak{z}_i, \bar{a}_{i\imath}, [h^*_{i1\imath}]_1, \mathsf{a}^*_\imath(\mathfrak{z}_i), \left[\frac{\mathsf{a}^*_\imath(x) - \mathsf{a}^*_\imath(\mathfrak{z}_i)}{x - \mathfrak{z}_i}\right]_1\right)$;
  for $t \in [1, n_b]$ **do**
    if $\bar{d}'_{it} \neq \mathsf{d}^*_t(\mathfrak{z}_i)$ **then return** $\left([d_t]_1, \mathfrak{z}_i, \bar{d}'_{it}, [h'_{it}]_1, \mathsf{d}^*_t(\mathfrak{z}_i), \left[\frac{\mathsf{d}^*_t(x) - \mathsf{d}^*_t(\mathfrak{z}_i)}{x - \mathfrak{z}_i}\right]_1\right)$;
for $i \in [1, \kappa_1]$ **do**
  if $\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i)\bar{d}'_{it} \neq 0$ **then**
    return $([\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i) d_t]_1, \mathfrak{z}_i, \sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i)\bar{d}'_{it}, [\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i) h'_{it}]_1, 0, [h^*_{i1,n_a+1}]_1)$;
return $\bot$;

Fig. 10. The extractor $\mathsf{Ext}^{\mathsf{sanlin}}_{\mathsf{ss}}$, the KZG special soundness-adversary $\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}$, and the KZG evaluation-binding adversary $\mathcal{C}_{\mathsf{evb}}$.

Next, we show that $\varepsilon = \mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen}, \mathsf{SanLin}, \mathsf{Ext}^{\mathsf{sanlin}}_{\mathsf{ss}}, \boldsymbol{\kappa}, \mathcal{A}}(\lambda)$ (see Definition 2) is negligible for any PPT adversary $\mathcal{A}$. That is, if $\mathcal{A}$ outputs an accepting tree of transcripts, then $\mathsf{Ext}^{\mathsf{sanlin}}_{\mathsf{ss}}$ fails to extract $(\mathsf{a}^*_\imath(X))_{\imath=1}^{n_a}, (\mathsf{d}^*_t(X))_{t=1}^{n_b}$, such that

$$\left(\left(([a_\imath]_1, \bar{a}_\imath)_{\imath=1}^{n_a}, ([d_t]_1, \bar{d}_t)_{t=1}^{n_b}\right), ((\mathsf{a}^*_\imath(X))_{\imath=1}^{n_a}, (\mathsf{d}^*_t(X))_{t=1}^{n_b})\right) \in \mathcal{R}^{\mathsf{Lin}}_{\mathsf{ck}, \mathbf{g}, \mathbf{tr}} \ ,$$

(for naturally defined $\mathbf{tr}$) with probability $\varepsilon = \mathsf{negl}(\lambda)$.

We consider the following two failure events for $\mathsf{Ext}^{\mathsf{sanlin}}_{\mathsf{ss}}$:

1. The event $\mathsf{bad}_{\mathsf{ext}}$ happens when $([a_\imath]_1, \mathsf{a}^*_\imath(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{tr}_{a_\imath}}$ or $([d_t]_1, \mathsf{d}^*_t(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{tr}_{d_t}}$ for some $\imath \in [1, n_a]$, $t \in [1, n_b]$.
2. The event $\mathsf{bad}_{\mathsf{evb}}$ happens when $\mathsf{bad}_{\mathsf{ext}}$ did not happen, but one of the following conditions hold:
   (a) For any $\imath \in [1, n_a]$, $t \in [1, n_b]$, either $\mathsf{a}^*_\imath(\mathfrak{z}_i) \neq \bar{a}_{i\imath}$ or $\mathsf{d}^*_t(\mathfrak{z}_i) \neq \bar{d}'_{it}$ for some $i \in [n+2, \kappa_1]$.

(b) $\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i)\bar{d}'_{it} \neq 0$ for any $i \in [1, \kappa_1]$.

Consider the scenario where neither $\mathsf{bad}_{\mathsf{ext}}$ nor $\mathsf{bad}_{\mathsf{evb}}$ occurs. Then, we have extracted polynomials $\mathsf{a}^*_\imath(X)$ and $\mathsf{d}^*_t(X)$ for $\imath \in [1, n_a]$, $t \in [1, n_b]$ of degree at most n which are consistent with the commitments. Denote $\mathsf{g}^*(X) := \sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}^*(X))\mathsf{d}^*_t(X)$. For any $i \in [1, \kappa_1]$,

$$\mathsf{g}^*(\mathfrak{z}_i) = \textstyle\sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}^*(\mathfrak{z}_i))\mathsf{d}^*_t(\mathfrak{z}_i) = \sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i)\bar{d}'_{it} = 0 \ .$$

Since $\mathsf{g}^*(X)$ is at most of degree $n_{\mathsf{g}} < \kappa_1$, it follows that $\mathsf{g}^*(X) = 0$ and consequently $\sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}^*(X))\mathsf{d}^*_t(X) = 0$. Hence, $\mathsf{Ext}^{\mathsf{sanlin}}_{\mathsf{ss}}$ extracts a valid witness.

Next, we bound the probabilities $\Pr[\mathsf{bad}_{\mathsf{ext}}]$ and $\Pr[\mathsf{bad}_{\mathsf{evb}}]$. To bound $\Pr[\mathsf{bad}_{\mathsf{ext}}]$, we construct a PPT adversary $\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}$ (see Fig. 10) that breaks the special-soundness of KZG when the event $\mathsf{bad}_{\mathsf{ext}}$ happens. $\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}$ runs $\mathcal{A}(\mathsf{ck})$ to recover the tree $\mathcal{T}$. Then, it obtains transcript vectors $((\mathbf{tr}_{a_\imath})_{\imath=1}^{n_a}, (\mathbf{tr}_{d_t})_{t=1}^{n_b}) \leftarrow \mathsf{TE}_{\mathsf{lin}}(\mathsf{ck}, \mathcal{T})$. For each transcript vector $\mathbf{tr}_f$, where $f \in \{a_\imath\}_{\imath=1}^{n_a} \cup \{d_t\}_{t=1}^{n_b}$, $\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}$ runs the deterministic extractor $\mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{tr}_f)$ to extract the polynomial $\mathsf{f}^*(X) \in \{\mathsf{a}^*_\imath\}_{\imath=1}^{n_a} \cup \{\mathsf{d}^*_t\}_{t=1}^{n_b}$. $\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}$ returns one transcript vector $\mathbf{tr}_f$, which satisfies $([f]_1, \mathsf{f}^*(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{tr}_f}$. When the event $\mathsf{bad}_{\mathsf{ext}}$ happens, $([f]_1, \mathsf{f}^*(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{tr}_f}$ for some f, and hence $\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}$ breaks the special soundness of KZG. Thus,

$$\Pr[\mathsf{bad}_{\mathsf{ext}}] = \mathsf{Adv}^{\mathsf{ss}}_{\mathsf{Pgen},\mathsf{KZG},\mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}},n+1,\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}}(\lambda) \ .$$

Second, we bind the probability $\Pr[\mathsf{bad}_{\mathsf{evb}}]$. We construct a PPT KZG's evaluation-binding adversary $\mathcal{C}_{\mathsf{evb}}$, depicted in Fig. 10. $\mathcal{C}_{\mathsf{evb}}$ runs $\mathcal{A}$, $\mathsf{TE}_{\mathsf{lin}}$ and $\mathsf{Ext}^{\mathsf{sanlin}}_{\mathsf{ss}}$ to obtain $\mathcal{T}$ and corresponding $((\mathsf{a}^*_\imath(X))_{\imath=1}^{n_a}, (\mathsf{d}^*_t(X))_{t=1}^{n_b})$. If the event $\mathsf{bad}_{\mathsf{evb}}$ happens, then for all $\imath$ and t, $([a_\imath]_1, \mathsf{a}^*_\imath(X)) \in \mathcal{R}_{\mathsf{ck},\mathbf{tr}_{a_\imath}}$ and $([d_t]_1, \mathsf{d}^*_t(X)) \in \mathcal{R}_{\mathsf{ck},\mathbf{tr}_{d_t}}$. Thus, $\mathsf{a}^*_\imath(x) = a_\imath$, and $\mathsf{d}^*_t(x) = d_t$. Note that

$$h_{a_\imath}(X) := (\mathsf{a}^*_\imath(X) - \mathsf{a}^*_\imath(\mathfrak{z}_i))/(X - \mathfrak{z}_i)$$

is always a polynomial. Hence, C_{evb} can compute $[h_{a_\imath}(x)]_1$ that satisfies

$$[h_{a_\imath}(x)]_1 \bullet [x - \mathfrak{z}_i]_2 = [\mathsf{a}^*_\imath(x) - \mathsf{a}^*_\imath(\mathfrak{z}_i)]_T = [a_\imath - \mathsf{a}^*_\imath(\mathfrak{z}_i)]_T \ .$$

Thus,

$$([a_\imath]_1, \mathfrak{z}_i, \mathsf{a}^*_\imath(\mathfrak{z}_i), [h_{a_\imath}(x)]_1)$$

is an accepting transcript. However, $\mathsf{k.tr}^*_{ij\imath} = ([a_\imath]_1, \mathfrak{z}_i, \bar{a}_{i\imath}, [h^*_{i1\imath}]_1)$ is also an accepting transcript. If $\mathsf{a}^*_\imath(\mathfrak{z}_i) \neq \bar{a}_{i\imath}$, $\mathcal{C}_{\mathsf{evb}}$ has broken KZG's evaluation-binding by finding a collision. Analogously, one can show that $\mathsf{d}^*_t(\mathfrak{z}_i) \neq \bar{d}'_{it}$ implies that $\mathcal{C}_{\mathsf{evb}}$ broke KZG's evaluation-binding.

We also need that $\mathsf{g}^*(\mathfrak{z}_i) = \sum_{t=1}^{n_b} \mathsf{g}_t(\mathbf{a}^*(\mathfrak{z}_i))\mathsf{d}^*_t(\mathfrak{z}_i) = 0$. From Lemma 3,

$$\mathsf{k.tr}_{d_t,i} = ([d_t]_1, \mathfrak{z}_i, \bar{d}'_{it}, [h'_{it}]_1)$$

and

$$\left([\textstyle\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i)d_t]_1 \, , \mathfrak{z}_i, 0, \left[h^*_{ij,n_a+1}\right]_1\right)$$

are accepting transcripts for all $i \in [1, \kappa_1]$. Since KZG is triply homomorphic, from all $\mathsf{k.tr}_{d_t,i}$ being accepting, we get

$$([\textstyle\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i) d_t]_1, \mathfrak{z}_i, \textstyle\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i) \bar{d}'_{it}, [\textstyle\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i) h'_{it}]_1)$$

is an accepting transcript. If $\sum_{t=1}^{n_b} \mathsf{g}_t(\bar{\boldsymbol{a}}_i) \bar{d}'_{it} \neq 0$, $\mathcal{C}_{\mathsf{evb}}$ has found a collision and thus broken KZG's evaluation-binding. Therefore,

$$\Pr[\mathsf{bad}_{\mathsf{evb}}] = \mathsf{Adv}^{\mathrm{evb}}_{\mathsf{Pgen},\mathsf{KZG},n,\mathcal{C}_{\mathsf{evb}}}(\lambda)$$

and we can conclude that

$$\mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen},\mathsf{SanLin},\mathsf{Ext}^{\mathsf{sanlin}}_{\mathsf{ss}},\boldsymbol{\kappa},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen},\mathsf{KZG},\mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}},n+1,\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}}(\lambda) + \mathsf{Adv}^{\mathrm{evb}}_{\mathsf{Pgen},\mathsf{KZG},n,\mathcal{C}_{\mathsf{evb}}}(\lambda) \ .$$

□

C Security of SplitRSDH in the AGMOS

Next, we prove that par-SplitRSDH is secure in the AGMOS (AGM with oblivious sampling, [LPS23]). We give the proof in the new oracle framework of Section 3. Because of Fact 1 in Section 3, the reduction implies that SplitRSDH is generically hard.

Theorem 7. *As in Definition 4, let* $\mathsf{par} = (m, n_\psi, n_1, n_\mathcal{S}, (\psi_k(X))_{k=1}^m)$. *Assume* $m, n_\psi, n_1, n_\mathcal{S} \in \mathsf{poly}(\lambda)$ *are positive integers so that* $n_\mathcal{S} > n_1 + n_\psi + 1$. *Assume* $\psi_k(X) \in \mathbb{F}_{\leq n_\psi}[X]$. *Fix* $\mathcal{EF}$ *and* $\mathcal{DF}$. *If the* $(n_1, 1)$-PDL *and* $(\mathcal{EF}, \mathcal{DF})$-TOFR *assumptions hold, then* par-SplitRSDH *holds in the AGMOS. More precisely, for every PPT* $\mathcal{A}$, *there exist PPT* $\mathcal{B}_{\mathrm{PDL}}$ *and PPT* $\mathcal{C}_{\mathrm{TOFR}}$, *such that*

$$\mathsf{Adv}^{\mathrm{splitrsdh}}_{\mathsf{Pgen},\mathsf{par},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{pdl}}_{n_1,1,\mathsf{Pgen},\mathcal{B}_{\mathrm{PDL}}}(\lambda) + \mathsf{Adv}^{\mathrm{tofr}}_{\mathsf{Pgen},\mathcal{C}_{\mathrm{TOFR}}}(\lambda) \ .$$

As usually, AGM and AGMOS proofs have a large number of boilerplate steps, needed to construct formal reductions. For the ease of parsing, we have marked one paragraph of the following proof as "meat of the analysis": this is essentially the only part that deeply depends on the concrete assumption. (Alternatively, it is a more formal rewording of the intuition we gave just after Definition 4.) Readers familiar with AGM(OS) proofs may skip the rest of the proof.

Proof. Let $\mathcal{A}$ be an $(\mathcal{EF}, \mathcal{DF})$-AGMOS adversary against the SplitRSDH assumption that succeeds with some non-negligible probability ε. In Fig. 11, we depict a PDL adversary $\mathcal{B}_{\mathrm{PDL}}$ and a TOFR adversary $\mathcal{C}_{\mathrm{TOFR}}$, such that $\Pr[\mathcal{A} \text{ is successful}] \leq \Pr[\mathcal{B}_{\mathrm{PDL}} \text{ is successful} \mid E] + \Pr[\mathcal{C}_{\mathrm{TOFR}} \text{ is successful} \mid \neg E]$ for some event E.

$\mathcal{B}_{\mathrm{PDL}}$ is given an input ck while $\mathcal{C}_{\mathrm{TOFR}}$ samples ck itself. (In the oracle framework of [JM24], it means $\mathcal{B}_{\mathrm{PDL}}$ can only use oracles to access ck while $\mathcal{C}_{\mathrm{TOFR}}$ knows x and can use it in computation.) After that, both reductions call $\mathcal{A}$ with ck, obtaining the output of a SplitRSDH adversary. According to Definition 3, there exists an extractor $\mathsf{Ext}_\mathcal{A}$ who (with a high probability) outputs the

$\mathcal{B}_{\mathrm{PDL}}(\mathsf{p}, \mathsf{ck} = ([1, x, \ldots, x^{n_1}]_1, [1, x]_2))$: $\mathcal{C}_{\mathrm{TOFR}}(\mathsf{p})$:

1 : $x \leftarrow\!\!\$\ \mathbb{F}; \mathsf{ck} \leftarrow ([1, x, \ldots, x^{n_1}]_1, [1, x]_2);$ // Only in the TOFR reduction:
2 : $(\mathcal{S} = \{\mathfrak{z}_i\}_{i=1}^{n_{\mathcal{S}}}, L(X), [(\tau_k)_{k=1}^m, \varphi]_1) \leftarrow \mathcal{A}^{\Pi_{\mathcal{O}}}(\mathsf{p}, \mathsf{ck});$ // Calling $\mathcal{A}$
3 : $((\{\tau'_k(X), \hat{\boldsymbol{\tau}}_k\}_{k=1}^m, (\varphi'(X), \hat{\boldsymbol{\varphi}}))) \leftarrow \mathsf{Ext}_{\mathcal{A}}^{\Pi_{\mathcal{O}}}(\mathsf{p}, \mathsf{ck});$ // Calling $\mathsf{Ext}_{\mathcal{A}}$
4 : // Defining verification polynomial based on extractor's output
5 : **for** $k \in [1, m]$ **do** $\tau_k(X, \mathbb{Q}) := \tau'_k(X) + \hat{\boldsymbol{\tau}}_k^{\intercal}\mathbb{Q}$;
6 : $\varphi(X, \mathbb{Q}) := \varphi'(X) + \hat{\boldsymbol{\varphi}}^{\intercal}\mathbb{Q}$;
7 : $V_0(X) := \sum_{k=1}^m \tau'_k(X)\psi_k(X) - L(X) - \varphi'(X)\mathsf{Z}_{\mathcal{S}}(X)$;
8 : **for** $j \in [1, \mathsf{il}]$ **do** $V_j(X) := \sum_{k=1}^m \hat{\tau}_{kj}\psi_k(X) - \hat{\varphi}_j(X)\mathsf{Z}_{\mathcal{S}}(X)$;
9 : $V(X, \mathbb{Q}) := V_0(X) + \sum_{j=1}^{\mathsf{il}} V_j(X)\mathbb{Q}_j$;
10 : **if** $V(X, \mathbb{Q}) = 0$ **then return** $\perp$;
11 : // Finishing off either PDL or TOFR reduction
12 : **if** $\forall j \in [0, \mathsf{il}].[V_j(x)]_1 = [0]_1$ **then** // PDL Reduction
13 : Let $j_0 \in [0, \mathsf{il}]$ be s.t. $(V_{j_0}(X) \neq 0)$;
14 : $(x_1, \ldots, x_{n_1}) \leftarrow \mathbf{Solve}(V_{j_0}(X) = 0)$;
15 : **for** $i \in [1, n_1]$ **do if** $[x]_1 = x_i[1]_1$ **then return** x_i;
16 : **else** // TOFR Reduction
17 : **for** $j \in [1, \mathsf{il}]$ **do** $v_j \leftarrow V_j(x)$;
18 : **return** $\boldsymbol{v}$;

Fig. 11. The PDL and TOFR reductions in the proof of Theorem 7. We have boxed the lines where the reduction handles group elements.

explanations behind $\mathcal{A}$'s output group elements. We use the explanations to define a verification polynomial $V(X, \mathbb{Q})$, where $\mathbb{Q}$ is the vector of indeterminates corresponding to sampling oracle queries. This polynomial is defined so that $V(x, \mathfrak{q}) = 0$ iff the SplitRSDH adversary succeeds in satisfying the last verification in Definition 4, i.e., $\sum_{k=1}^m ([\tau_k]_1 \bullet [\psi_k(x)]_2) = [1]_1 \bullet [L(x)]_2 + [\varphi]_1 \bullet [\mathsf{Z}_{\mathcal{S}}(x)]_2$. After that, as usually in AGMOS proofs [LPS23], we define an event E, such that if $E = \mathsf{true}$, we can and will construct a successful PDL adversary and otherwise a successful TOFR adversary.

Let us now give a detailed description of the reductions and their success probability. Consider the reduction $\mathcal{B}_{\mathrm{PDL}}$. (The analysis of $\mathcal{C}_{\mathrm{TOFR}}$ differs minimally.) $\mathcal{B}_{\mathrm{PDL}}$ (line 2) first calls $\mathcal{A}$ with correct input. Recall that $\psi_k(X) \in \mathbb{F}_{\leq n_\psi}[X]$, and $n_{\mathcal{S}} > n_1 + n_\psi + 1$. The AGMOS adversary $\mathcal{A}$ has access to $\Pi_{\mathcal{O}}$ that includes the sampling oracles. Let $\mathsf{bad}_{\mathcal{A}}$ be the event that $\mathcal{A}$ succeeds. Thus, $\Pr[\mathsf{bad}_{\mathcal{A}}] = \varepsilon \neq \mathsf{negl}(\lambda)$. That is, according to Definition 4, the following holds with probability ε:

$$(\mathcal{S} \subset \mathbb{F}) \wedge (|\mathcal{S}| = n_{\mathcal{S}}) \wedge (L(X) \in \mathbb{F}[X]) \wedge (L(X) \in [n_1 + n_\psi + 1, n_{\mathcal{S}} - 1]) \wedge$$
$$(\textstyle\sum_{k=1}^m [\tau_k \psi_k(x)]_1 = [L(x)]_1 + [\varphi \mathsf{Z}_{\mathcal{S}}(x)]_1)$$

We note that since SplitRSDH is not publicly-verifiable, the (model-preserving) reduction cannot test whether $\mathsf{bad}_{\mathcal{A}}$ holds (more precisely, it cannot test that

the boxed equality holds). This is however not important: we only show that either $\mathcal{B}_{\mathrm{PDL}}$ or $\mathcal{C}_{\mathrm{TOFR}}$ succeeds with high probability whenever $\mathsf{bad}_{\mathcal{A}} = \mathsf{true}$.

Assume now $\mathsf{bad}_{\mathcal{A}} = \mathsf{true}$. (This holds with probability ε.) According to Definition 3, there exists an extractor $\mathsf{Ext}_{\mathcal{A}}^{\Pi_{\mathcal{O}}}$ that, on input $(\mathsf{p}, \mathbb{x})$ and access to $\mathcal{A}$'s oracle queries, outputs the values on line 3, such that $\tau'_k(X), \varphi'(X) \in \mathbb{F}_{\le n_1}[X]$ and $\hat{\tau}_{kj}, \hat{\varphi}_j \in \mathbb{F}$, and $\mathfrak{q}$ is a vector of discrete logarithms of the answers of the oblivious sampling oracle $\mathcal{S}amp_1$. (Since $\mathcal{A}$ does not output $\mathbb{G}_2$ elements, we ignore the oracle $\mathcal{S}amp_2$.) According to Definition 3,if $\mathsf{bad}_{\mathcal{A}} = \mathsf{true}$, the reduction always aucceeds, that is, for all k, $[\tau_k]_1 = [\tau'_k(x)]_1 + \hat{\boldsymbol{\tau}}_k^{\intercal}[\mathfrak{q}]_1$ and $[\varphi]_1 = [\varphi'(x)]_1 + \hat{\boldsymbol{\varphi}}^{\intercal}[\mathfrak{q}]_1$. As explained earlier, this extractor can be constructed by observing the inputs and outputs of all oracle queries. Moreover, if $\mathcal{A}$ is model-preserving, then $\mathsf{Ext}_{\mathcal{A}}$ is model-preserving: $\mathsf{Ext}_{\mathcal{A}}$ never applies $\mathcal{E}ncode$ or $\mathcal{D}ecode$ to any group elements unless $\mathcal{A}$ does.

Next, given the extractor's outputs, $\mathcal{B}_{\mathrm{PDL}}$ defines verification polynomials $V(X, \mathbb{Q})$ and $V_j(X)$, such that $V(X, \mathbb{Q}) := V_0(X) + \sum_{j=1}^{\mathsf{il}} V_j(X)\mathbb{Q}_j$. Since the verifier accepts, $V(x, \mathfrak{q}) = 0$.

"Meat of the analysis." Let us show that if $\mathsf{bad}_{\mathcal{A}} = \mathsf{true}$, then $\mathcal{B}_{\mathrm{PDL}}$ aborts on line 10 (i.e., $V(X, \mathbb{Q}) = 0$) with probability 0. Really, assume $V(X, \mathbb{Q}) = 0$ as a polynomial. Since $\deg L(X) < n_{\mathcal{S}}$, $\varphi'(X) = 0$. Let $T(X) := \sum_{k=1}^{m} \tau'_k(X)\psi_k(X) \in \mathbb{F}_{\le n_1+n_\psi}[X]$. Then, $T(X) = L(X)$. Since $\deg T(X) \le n_1 + n_\psi$, we have $\deg L(X) = n_1 + n_\psi$, and thus this case is impossible. Thus, the reduction never aborts on line 10.

Now, know that $V(x, \mathfrak{q}) = 0$ but $V(X, \mathbb{Q}) \neq 0$. Let E be the event that the check on line 12 succeeds, i.e., $V_j(x) = 0$ for all $j \in [0, \mathsf{il}]$. If $E = \mathsf{true}$, we finish the PDL reduction. Otherwise, we finish the TOFR reduction.

$\underline{E = \mathsf{true}.}$ Since $E = \mathsf{true}$, $V_j(x) = 0$ for all $j \in [0, \mathsf{il}]$. Since $V(X, \mathbb{Q}) \neq 0$, $V_{j_0}(X) \neq 0$ for some $j_0 \in [0, \mathsf{il}]$. Thus, $V_{j_0}(X) \neq 0$, but $V_{j_0}(x) = 0$. Clearly, $V_{j_0}(X)$ is univariate and of degree at most n_1. Thus, it has at most n_1 roots. The PDL adversary computes x as one of the roots of $V_{j_0}(X)$.

$\underline{E = \mathsf{false}.}$ Since $E = \mathsf{false}$, $\exists j.(V_j(x) \neq 0)$ but the verifier accepts $(V(x, \mathfrak{q}) = 0)$. In this case, we construct an adversary $\mathcal{C}_{\mathrm{TOFR}}$ that breaks the TOFR assumption. $\mathcal{C}_{\mathrm{TOFR}}$ starts exactly as $\mathcal{B}_{\mathrm{PDL}}$, only sampling ck itself. Knowing the trapdoor x, $\mathcal{C}_{\mathrm{TOFR}}$ outputs the vector $\boldsymbol{v}$ defined by setting $v_j = V_j(x)$ for $j \ge 0$. Thus, this adversary is successful in breaking the TOFR assumption.

To sum it up,

$$\Pr[\mathcal{A} \text{ is successful}] \le \Pr[\mathcal{B}_{\mathrm{PDL}} \text{ fails} \mid E] + \Pr[\mathcal{C}_{\mathrm{TOFR}} \text{ fails} \mid \neg E] \ .$$

□

AGM to GGM Lifting. To be able to prove the lifting result of [JM24], we need to show that both $\mathcal{B}_{\mathrm{PDL}}$ and $\mathcal{C}_{\mathrm{TOFR}}$ are model-preserving and efficient. (We already know that PDL [Lip12] and TOFR [LPS23] are generically hard.) Really, assume $\mathcal{A}$ is an generic adversary. Thus, $\mathsf{Ext}_{\mathcal{A}}$ is also generic. The rest of the reductions touches the adverary's group element outputs (and, in the case of $\mathcal{B}_{\mathrm{PDL}}$, the input ck) in the following steps:

– Line 12 uses $\mathcal{O}p$ in $\mathbb{G}_2$, $\mathcal{P}ait$, and $\mathcal{E}q$ in $\mathbb{G}_T$ to check if $E = \mathsf{true}$.
– Line 15 uses $\mathcal{E}q$ in $\mathbb{G}_1$ to find correct root.

Thus, both reductions are model-preserving. Since PDL and TOFR are generically hard, we obtain that SplitRSDH is secure in Maurer's GGM.

We remark that first use of oracles is actually not needed. Really, there is no need to check if $E = \mathsf{true}$ or not (without this check, the success probability of the reduction can only increase). The second use is needed to solve PDL with a high probability.

D Postponed Material from Section 6

D.1 Plonk's Polynomials That Define A Specific Circuit

The following polynomials, along with the integer n, uniquely define our circuit:

– $\mathsf{q}_\mathsf{M}(X), \mathsf{q}_\mathsf{L}(X), \mathsf{q}_\mathsf{R}(X), \mathsf{q}_\mathsf{O}(X), \mathsf{q}_\mathsf{C}(X)$ are selector polynomials that define the circuit's arithmetization. We refer to [GWC19] for the explanation how they are defined based on an arithmetic circuit.
– $\mathsf{S}_{\mathsf{ID}_1}(X) = X$, $\mathsf{S}_{\mathsf{ID}_2}(X) = k_1 X$, $\mathsf{S}_{\mathsf{ID}_3}(X) = k_2 X$ encode an identity permutation respectively on groups $k_0 \cdot \mathbb{H}$, $k_1 \cdot \mathbb{H}$, and $k_2 \cdot \mathbb{H}$. Here, $k_0 := 1$ and $k_1, k_2 \in \mathbb{F}$ are chosen such that $\mathbb{H}, k_1 \cdot \mathbb{H}, k_2 \cdot \mathbb{H}$ are distinct cosets of $\mathbb{H}$ in $\mathbb{F}^*$, and thus consist of $3n$ distinct elements. For example, one can take ω to be a quadratic residue in $\mathbb{F}$, k_1 to be any quadratic non-residue, and k_2 to be a quadratic non-residue not contained in $k_1 \cdot \mathbb{H}$.
– Let us denote $\mathbb{H}' := \mathbb{H} \cup (k_1 \cdot \mathbb{H}) \cup (k_2 \cdot \mathbb{H})$. Let $\sigma : [1, 3n] \to [1, 3n]$ be a permutation. We encode an element $i \in [1, 3n]$ in $\mathbb{H}'$ such that if we express $i = \vartheta n + j$ for the unique $\vartheta \in [0, 2]$ and $0 \le j < n$, then $\mathbb{H}'[i] = k_\vartheta \omega^j$. Finally, define $\sigma^*(i) := \mathbb{H}'[\sigma(i)]$, which is an injective map on $\mathbb{H}'$. We encode σ^* by the three permutation polynomials $\mathsf{S}_{\sigma 1}(X) := \sum_{i=1}^{n} \sigma^*(i) L_i(X)$, $\mathsf{S}_{\sigma 2}(X) := \sum_{i=1}^{n} \sigma^*(n+i) L_i(X)$, and $\mathsf{S}_{\sigma 3}(X) := \sum_{i=1}^{n} \sigma^*(2n+i) L_i(X)$.

D.2 Proof of Theorem 5

Proof. Let $\mathcal{T}$ be a $(\kappa_\beta, \kappa_\gamma, \kappa_\alpha, \kappa_\mathfrak{z}, \kappa_\delta, \kappa_v)$-tree of accepting transcripts and let $\hat{\mathcal{T}}_{\beta\gamma\alpha}$ be its $(1, 1, 1, \kappa_\mathfrak{z}, \kappa_\delta, \kappa_v)$-subtree for any fixed α, β, γ. By Lemma 2, $\mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{\mathcal{T}}_{\beta\gamma\alpha})$ extracts accepting KZG transcripts $\mathsf{k.tr}_{ki}$ and $\mathsf{k.tr}_i^\omega$ for every i and k. We will consider separately the cases of (1) Plonk and SmallPlonk and (2) SanPlonk. However, the extractors and adversaries on most figures (say, Fig. 12) correspond to both.

Case of Plonk And SmallPlonk. In Fig. 12, we depict an extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$. $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$ invokes $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}^{\boldsymbol{\omega}})$ and $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}_k)$ for $k \in [2, 4]$, extracting polynomials $\mathsf{z}(X)$, $\mathsf{a}(X)$, $\mathsf{b}(X)$, and $\mathsf{c}(X)$ of at most degree κ_{kzg} (see Eq. (12)). After executing $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$, we use the following procedure to possibly set one of the "bad" flags:

$\mathsf{Ext}^{\mathsf{sub}}_{\mathsf{ss}}(\mathsf{ck}, \hat{\mathfrak{T}}_{\beta\gamma\alpha})$

$((\mathbf{k.tr}_k)_{k\in[1,\kappa_v+2]}, \mathbf{k.tr}^{\boldsymbol{\omega}}) \leftarrow \mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{\mathfrak{T}}_{\beta\gamma\alpha})$;
$\mathsf{z}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}^{\boldsymbol{\omega}})$; $\mathsf{a}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_2)$;
$\mathsf{b}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_3)$; $\mathsf{c}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_4)$;
$\mathsf{t}_{\mathsf{lo}}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_7)$; $\mathsf{t}_{\mathsf{mid}}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_8)$; $\mathsf{t}_{\mathsf{hi}}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_9)$;
return $(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{t}_{\mathsf{lo}}(X), \mathsf{t}_{\mathsf{mid}}(X), \mathsf{t}_{\mathsf{hi}}(X))$;

Fig. 12. Plonk's/SanPlonk's/SmallPlonk's special-soundness extractor $\mathsf{Ext}^{\mathsf{sub}}_{\mathsf{ss}}$.

$\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck})$

$\hat{\mathfrak{T}}_{\beta\gamma\alpha} \leftarrow \mathcal{A}^{\mathsf{plonk}}_{\mathsf{ss}}(\mathsf{ck})$;
$((\mathbf{k.tr}_k)_{k\in[1,\kappa_v+2]}, \mathbf{k.tr}^{\boldsymbol{\omega}}) \leftarrow \mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{\mathfrak{T}}_{\beta\gamma\alpha})$;
$(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{t}_{\mathsf{lo}}(X), \mathsf{t}_{\mathsf{mid}}(X), \mathsf{t}_{\mathsf{hi}}(X)) \leftarrow \mathsf{Ext}^{\mathsf{sub}}_{\mathsf{ss}}(\mathsf{ck}, \hat{\mathfrak{T}}_{\beta\gamma\alpha})$;
if $([z]_1, \mathsf{z}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}^{\boldsymbol{\omega}}}$ **then return** $\mathbf{k.tr}^{\boldsymbol{\omega}}$; **fi**
if $([a]_1, \mathsf{a}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_2}$ **then return** $\mathbf{k.tr}_2$; **fi**
if $([b]_1, \mathsf{b}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_3}$ **then return** $\mathbf{k.tr}_3$; **fi**
if $([c]_1, \mathsf{c}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_4}$ **then return** $\mathbf{k.tr}_4$; **fi**
if $([t_{lo}]_1, \mathsf{t}_{\mathsf{lo}}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_7}$ **then return** $\mathbf{k.tr}_7$; **fi**
if $([t_{mid}]_1, \mathsf{t}_{\mathsf{mid}}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_8}$ **then return** $\mathbf{k.tr}_8$; **fi**
if $([t_{hi}]_1, \mathsf{t}_{\mathsf{hi}}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_9}$ **then return** $\mathbf{k.tr}_9$; **fi**
return $\bot$;

Fig. 13. Plonk's/SanPlonk's KZG's special-soundness adversary $\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}$.

(i) $\mathsf{bad}_{\mathsf{ext}} \leftarrow \mathsf{false}$; $\mathsf{bad}_{\mathsf{evb}} \leftarrow \mathsf{false}$; $\mathsf{bad}_{\mathsf{rhino}} \leftarrow \mathsf{false}$;
(ii) if $([z]_1, \mathsf{z}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_1} \vee ([a]_1, \mathsf{a}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_2} \vee ([b]_1, \mathsf{b}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_3} \vee ([c]_1, \mathsf{c}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_4}$ (see Eq. (1)) then $\mathsf{bad}_{\mathsf{ext}} \leftarrow \mathsf{true}$; abort;
(iii) for $i \in [\kappa_{\mathsf{kzg}} + 2, \kappa_{\mathfrak{z}}]$: if $\mathsf{z}(\mathfrak{z}_i\omega) \neq \bar{z}_{\omega i} \vee \mathsf{a}(\mathfrak{z}_i) \neq \bar{a}_i \vee \mathsf{b}(\mathfrak{z}_i) \neq \bar{b}_i \vee \mathsf{c}(\mathfrak{z}_i) \neq \bar{c}_i$ then $\mathsf{bad}_{\mathsf{evb}} \leftarrow \mathsf{true}$; abort;
(iv) for $i \in [1, \kappa_{\mathfrak{z}}]$: if $\mathsf{S}_{\sigma 1}(\mathfrak{z}_i) \neq \bar{s}_{\sigma 1 i} \vee \mathsf{S}_{\sigma 2}(\mathfrak{z}_i) \neq \bar{s}_{\sigma 2 i}$ then $\mathsf{bad}_{\mathsf{evb}} \leftarrow \mathsf{true}$; abort;
(v) if $\mathsf{Z}_{\mathbb{H}}(X) \nmid \mathsf{F}(X)$, where $\mathsf{F}(X) = \mathsf{F}_0(X) + \alpha\mathsf{F}_1(X) + \alpha^2\mathsf{F}_2(X)$, then $\mathsf{bad}_{\mathsf{rhino}} \leftarrow \mathsf{true}$;

Importantly, only one of the "bad" flags is set at a time. Thus, for example, $\mathsf{bad}_{\mathsf{evb}} = \mathsf{true}$ implies that $\mathsf{bad}_{\mathsf{ext}} = \mathsf{false}$. Let $\mathcal{E}$ be the event $\mathsf{Ext}^{\mathsf{sub}}_{\mathsf{ss}}$ succeeds. Thus, $\mathcal{E}$ is the event that $\mathsf{a}(X)$, $\mathsf{b}(X)$, $\mathsf{c}(X)$, and $\mathsf{z}(X)$ are consistent with the commitments and all openings, and $\mathsf{t}(X) = (\mathsf{F}_0(X) + \alpha\mathsf{F}_1(X) + \alpha^2\mathsf{F}_2(X))/\mathsf{Z}_{\mathbb{H}}(X)$ is a polynomial. Let $\overline{\mathsf{bad}}$ be the event none of the bad flags was set. We analyze the success probability of $\mathsf{Ext}^{\mathsf{sub}}_{\mathsf{ss}}$. For this, we make the following claims.

1. **Claim 1**. $\Pr[\mathcal{E}|\overline{\mathsf{bad}}] = 1$.
   Really, assume that $\overline{\mathsf{bad}}$ holds. Since $\mathsf{bad}_{\mathsf{ext}} = \mathsf{bad}_{\mathsf{evb}} = \mathsf{false}$, we get that $\mathsf{a}(X)$, $\mathsf{b}(X)$, $\mathsf{c}(X)$, $\mathsf{z}(X)$ are consistent with the commitments and all open-

$\mathcal{C}_{\mathsf{evb}}(\mathsf{p},\mathsf{ck})$

$\hat{\mathfrak{T}}_{\beta\gamma\alpha} \leftarrow \mathcal{A}_{\mathsf{ss}}^{\mathsf{plonk}}(\mathsf{ck}); ((\mathbf{k.tr}_k)_{k\in[1,\kappa_v+2]}, \mathbf{k.tr}^{\boldsymbol{\omega}}) \leftarrow \mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{\mathfrak{T}}_{\beta\gamma\alpha});$
$(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{t}_{\mathsf{lo}}(X), \mathsf{t}_{\mathsf{mid}}(X), \mathsf{t}_{\mathsf{hi}}(X)) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}(\mathsf{ck}, \hat{\mathfrak{T}}_{\beta\gamma\alpha});$
for $i \in [\kappa_{\mathsf{kzg}}+2, \kappa_{\mathfrak{z}}]$ **do**
  if $\mathsf{z}(\mathfrak{z}_i\omega) \neq \bar{z}_{\omega i}$ **then**
    return $([z]_1, \mathfrak{z}_i\omega, \bar{z}_{\omega i}, [\mathsf{W}'_{\omega i \mathbb{j}1}]_1, \mathsf{z}(\mathfrak{z}_i\omega), [(\mathsf{z}(x)-\mathsf{z}(\mathfrak{z}_i\omega))/(x-\mathfrak{z}_i\omega)]_1)$; **fi**
  if $\mathsf{a}(\mathfrak{z}_i) \neq \bar{a}_i$ **then return** $([a]_1, \mathfrak{z}_i, \bar{a}_i, [\mathsf{W}'_{\omega i \mathbb{j}2}]_1, \mathsf{a}(\mathfrak{z}_i), [(\mathsf{a}(x)-\mathsf{a}(\mathfrak{z}_i))/(x-\mathfrak{z}_i)]_1)$; **fi**
  if $\mathsf{b}(\mathfrak{z}_i) \neq \bar{b}_i$ **then return** $([b]_1, \mathfrak{z}_i, \bar{b}_i, [\mathsf{W}'_{\omega i \mathbb{j}3}]_1, \mathsf{b}(\mathfrak{z}_i), [(\mathsf{b}(x)-\mathsf{b}(\mathfrak{z}_i))/(x-\mathfrak{z}_i)]_1)$; **fi**
  if $\mathsf{c}(\mathfrak{z}_i) \neq \bar{c}_i$ **then return** $([c]_1, \mathfrak{z}_i, \bar{c}_i, [\mathsf{W}'_{\omega i \mathbb{j}4}]_1, \mathsf{c}(\mathfrak{z}_i), [(\mathsf{c}(x)-\mathsf{c}(\mathfrak{z}_i))/(x-\mathfrak{z}_i)]_1)$; **fi**
  if $\mathsf{t}_{\mathsf{lo}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z}lo,i}$ **then return** $([t_{lo}]_1, \mathfrak{z}_i, \bar{t}_{\mathfrak{z}lo,i}, [\mathsf{W}'_{\omega i \mathbb{j}7}]_1, \mathsf{t}_{\mathsf{lo}}(\mathfrak{z}_i), [(\mathsf{t}_{\mathsf{lo}}(x)-\mathsf{t}_{\mathsf{lo}}(\mathfrak{z}_i))/(x-\mathfrak{z}_i)]_1)$; **fi**
  if $\mathsf{t}_{\mathsf{mid}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z}mid,i}$ **then return** $([t_{mid}]_1, \mathfrak{z}_i, \bar{t}_{\mathfrak{z}mid,i}, [\mathsf{W}'_{\omega i \mathbb{j}8}]_1, \mathsf{t}_{\mathsf{mid}}(\mathfrak{z}_i), [(\mathsf{t}_{\mathsf{mid}}(x)-\mathsf{t}_{\mathsf{mid}}(\mathfrak{z}_i))/(x-\mathfrak{z}_i)]_1)$; **fi**
  if $\mathsf{t}_{\mathsf{hi}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z}hi,i}$ **then return** $([t_{hi}]_1, \mathfrak{z}_i, \bar{t}_{\mathfrak{z}hi,i}, [\mathsf{W}'_{\omega i \mathbb{j}9}]_1, \mathsf{t}_{\mathsf{hi}}(\mathfrak{z}_i), [(\mathsf{t}_{\mathsf{hi}}(x)-\mathsf{t}_{\mathsf{hi}}(\mathfrak{z}_i))/(x-\mathfrak{z}_i)]_1)$; **fi**
endfor ;
for $i \in [1, \kappa_{\mathfrak{z}}]$ **do**
  $\mathsf{r}_i(X) \leftarrow \Lambda_{0i}(X) + \alpha\Lambda_{1i}(X) + \alpha^2\Lambda_{2i}(X) - \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)\cdot(\mathsf{t}_{\mathsf{lo}}(X) + \mathfrak{z}_i^n\mathsf{t}_{\mathsf{mid}}(X) + \mathfrak{z}_i^{2n}\mathsf{t}_{\mathsf{hi}}(X));$
  if $\mathsf{r}_i(\mathfrak{z}_i) \neq 0$ **then return** $([r_i]_1, \mathfrak{z}_i, 0, [\mathsf{W}'_{\mathfrak{z}i11}]_1, \mathsf{r}_i(\mathfrak{z}_i), [\mathsf{r}_i(x)-\mathsf{r}_i(\mathfrak{z}_i)/(x-\mathfrak{z}_i)]_1)$;
endfor
for $i \in [1, \kappa_{\mathfrak{z}}]$ **do**
  if $\mathsf{S}_{\sigma 1}(\mathfrak{z}_i) \neq \bar{s}_{\sigma 1 i}$ **then**
    return $([s_{\sigma 1}]_1, \mathfrak{z}_i, \bar{s}_{\sigma 1 i}, [\mathsf{W}'_{\omega i \mathbb{j}5}]_1, \mathsf{S}_{\sigma 1}(\mathfrak{z}_i), [(\mathsf{S}_{\sigma 1}(x)-\mathsf{S}_{\sigma 1}(\mathfrak{z}_i))/(x-\mathfrak{z}_i)]_1)$; **fi**
  if $\mathsf{S}_{\sigma 2}(\mathfrak{z}_i) \neq \bar{s}_{\sigma 2 i}$ **then**
    return $([s_{\sigma 2}]_1, \mathfrak{z}_i, \bar{s}_{\sigma 2 i}, [\mathsf{W}'_{\omega i \mathbb{j}6}]_1, \mathsf{S}_{\sigma 2}(\mathfrak{z}_i), [(\mathsf{S}_{\sigma 2}(x)-\mathsf{S}_{\sigma 2}(\mathfrak{z}_i))/(x-\mathfrak{z}_i)]_1)$; **fi**
endfor ;
return $\bot$;

Fig. 14. KZG's evaluation-binding adversary $\mathcal{C}_{\mathsf{evb}}$.

ings. Since $\mathsf{bad}_{\mathsf{rhino}} = \mathsf{false}$, $\mathsf{t}(X) = (\mathsf{F}_0(X) + \alpha\mathsf{F}_1(X) + \alpha^2\mathsf{F}_2(X))/\mathsf{Z}_{\mathbb{H}}(X)$ is a polynomial.

2. **Claim 2**. There exists a KZG's $(\kappa_{\mathsf{kzg}}+1)$-special-soundness adversary $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ (see Fig. 13), such that $\Pr[\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ succeeds $\mid \mathsf{bad}_{\mathsf{ext}}] = 1$.
   Recall from Item ii that $\mathsf{bad}_{\mathsf{ext}}$ is set if one of the four bad events happens. $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ just tests which of the cases is true and returns the corresponding transcript. Clearly, $\Pr[\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ succeeds $\mid \mathsf{bad}_{\mathsf{ext}}] = 1$.
3. **Claim 3**. There exists an evaluation-binding adversary $\mathcal{C}_{\mathsf{evb}}$ (see Fig. 14) for KZG, such that $\Pr[\mathcal{C}_{\mathsf{evb}}$ succeeds $\mid \mathsf{bad}_{\mathsf{evb}}] = 1$.
   Assume that $\mathsf{bad}_{\mathsf{evb}} = \mathsf{true}$. (Note that $\mathsf{bad}_{\mathsf{evb}} = \mathsf{true}$ means that $\mathsf{bad}_{\mathsf{ext}} = \mathsf{false}$, that is, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$ managed to extract all polynomials.) Then, one of the bad cases in Item iii or Item iv happens. $\mathcal{C}_{\mathsf{evb}}$ just finds out which of these events happens, and depending on the case, returns a collision. By the correctness of the extraction, and the completeness property of KZG, any of the returned values in Fig. 14 *is* a collision. Thus, $\Pr[\mathcal{C}_{\mathsf{evb}}$ succeeds $\mid \mathsf{bad}_{\mathsf{evb}}] = 1$.

4. **Claim 4**. In the case of Plonk or SmallPlonk, there exists a SplitRSDH adversary $\mathcal{D}_{\text{splitrsdh}}$ (see Fig. 15), such that $\Pr[\mathcal{D}_{\text{splitrsdh}} \text{ succeeds} \mid \mathsf{bad}_{\mathsf{rhino}}] = 1$.

In Appendix D.3, we prove this claim as a separate theorem, Theorem 8. Recalling all three bad events are disjoint,

$$\begin{aligned}\Pr[\overline{\mathcal{E}}] &= \Pr[\overline{\mathcal{E}}|\overline{\mathsf{bad}}]\Pr[\overline{\mathsf{bad}}] + \Pr[\overline{\mathcal{E}}|\mathsf{bad}_{\mathsf{ext}}]\Pr[\mathsf{bad}_{\mathsf{ext}}] + \Pr[\overline{\mathcal{E}}|\mathsf{bad}_{\mathsf{evb}}]\Pr[\mathsf{bad}_{\mathsf{evb}}] \\ &\quad + \Pr[\overline{\mathcal{E}}|\mathsf{bad}_{\mathsf{rhino}}]\Pr[\mathsf{bad}_{\mathsf{rhino}}] \\ &\leq 0 + \Pr[\mathsf{bad}_{\mathsf{ext}}] + \Pr[\mathsf{bad}_{\mathsf{evb}}] + \Pr[\mathsf{bad}_{\mathsf{rhino}}] \ .\end{aligned}$$

Since $\mathcal{C}_{\mathsf{evb}}$ succeeds whenever $\mathsf{bad}_{\mathsf{evb}}$ is set and KZG is evaluation-binding, $\Pr[\mathsf{bad}_{\mathsf{evb}}] = \mathsf{negl}(\lambda)$. Similarly, $\Pr[\mathsf{bad}_{\mathsf{ext}}] = \Pr[\mathsf{bad}_{\mathsf{rhino}}] = \mathsf{negl}(\lambda)$. Thus, $\Pr[\overline{\mathcal{E}}] \leq \mathsf{negl}(\lambda)$. This proves the claim.

Case of SanPlonk. We postpone the proof of this case to Appendix D.4. □

D.3 Rhino for Plonk

In this section, we prove Theorem 8. Recall that SplitRSDH is defined in Definition 4. Let $L_i^{\mathcal{S}}(X) = \prod_{j \neq i} \frac{X - \mathfrak{z}_j}{\mathfrak{z}_i - \mathfrak{z}_j}$ be Lagrange polynomials of $\mathcal{S} = \{\mathfrak{z}_i\}_{i=1}^{n_{\mathcal{S}}}$

Theorem 8. *Consider* Plonk *(with* $\mathsf{par} = \mathsf{par}_n^{\mathsf{plonk}}$*) or* SmallPlonk *(with* $\mathsf{par} = \mathsf{par}_n^{\mathsf{smallplonk}}$*). There exists a* par-SplitRSDH *adversary* $\mathcal{D}_{\text{splitrsdh}}$ *(see Fig. 15), such that* $\Pr[\mathcal{D}_{\text{splitrsdh}} \text{ succeeds} \mid \mathsf{bad}_{\mathsf{rhino}}] = 1$.

Proof (Proof of Theorem 8). Assume $\mathsf{bad}_{\mathsf{rhino}} = \mathsf{true}$. Let $\mathcal{D}_{\text{splitrsdh}}$ be the par-SplitRSDH adversary in Fig. 15. $\mathcal{D}_{\text{splitrsdh}}$ uses $\mathcal{A}_{\mathsf{ss}}^{\mathsf{plonk}}$ to obtain a subtree $\hat{\mathcal{T}}_{\beta\gamma\alpha} = (\mathsf{tr}_{ijk})$ (see Eq. (13); note that $\hat{\mathcal{T}}_{\beta\gamma\alpha}$ contains, say, $[t_{lo}, t_{mid}, t_{hi}]_1$ in the case of Plonk or $[t]_1$ in the case of SmallPlonk). Then, $\mathcal{D}_{\text{splitrsdh}}$ runs $\mathsf{TE}_{*\mathsf{plonk}}$ on $\hat{\mathcal{T}}_{\beta\gamma\alpha}$ to obtain several accepting KZG transcripts, and then runs $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$ on $\hat{\mathcal{T}}_{\beta\gamma\alpha}$ to obtain polynomials $(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X))$. After that, $\mathcal{D}_{\text{splitrsdh}}$ computes the polynomials $\mathsf{F}_s(X)$, $\mathsf{F}(X)$ and $\mathsf{t}(X)$ (as defined in Eq. (7)). $\mathcal{D}_{\text{splitrsdh}}$ also defines linearization polynomials $\Lambda_{0i}(X)$, $\Lambda_{1i}(X)$, $\Lambda_{2i}(X)$ (see Eq. (8)) of $\mathsf{F}_0(X)$, $\mathsf{F}_1(X)$, and $\mathsf{F}_2(X)$, and their batched sum $\Lambda_i(X)$ corresponding to $\mathsf{F}(X)$. Clearly, $[\Lambda_{si}(x)]_1 = [\Lambda_{si}]_1$ (from Fig. 6) for $s \in [0,2]$ and $i \in [1, \kappa_{\mathfrak{z}}]$. If the SplitRSDH's requirement $\mathsf{Z}_{\mathbb{H}}(X) \nmid \mathsf{F}(X)$ is satisfied (this holds always since $\mathsf{bad}_{\mathsf{rhino}} = \mathsf{true}$, see Item v), $\mathcal{D}_{\text{splitrsdh}}$ outputs a SplitRSDH adversary's output.

Let us argue that $\mathcal{D}_{\text{splitrsdh}}$ is successful, i.e., the output satisfies par-SplitRSDH's conditions. First, it is easy to see that $\mathcal{S}$ is of size $\kappa_{\mathfrak{z}}$ since the values $\mathfrak{z}_i$ output by $\mathcal{A}_{\mathsf{ss}}^{\mathsf{plonk}}$ are mutually different by the definition of special-soundness. Second, $L(X) = \sum_{i=1}^{\kappa_{\mathfrak{z}}} \mathsf{t}(\mathfrak{z}_i) L_i^{\mathcal{S}}(X)$ as defined in Fig. 15 has degree $\leq \kappa_{\mathfrak{z}} - 1$. It remains to show that $\deg(L) \geq \kappa_{\mathsf{kzg}} + n_\psi + 1$ and that

$$[t_{lo}]_1 \bullet [1]_2 + [t_{mid}]_1 \bullet [x^n]_2 + [t_{hi}]_1 \bullet [x^{2n}]_2 = [1]_1 \bullet [L(x)]_2 + [\varphi]_1 \bullet [\mathsf{Z}_{\mathcal{S}}(x)]_2 \quad (18)$$

in the case of Plonk or

$$[t]_1 \bullet [1]_2 = [1]_1 \bullet [L(x)]_2 + [\varphi]_1 \bullet [\mathsf{Z}_{\mathcal{S}}(x)]_2 \quad (19)$$

$\mathcal{D}_{\mathrm{splitrsdh}}(\mathsf{p},\mathsf{ck})$

1 : $\hat{\mathfrak{T}}_{\beta\gamma\alpha} \leftarrow \mathcal{A}_{\mathsf{ss}}^{\mathsf{plonk}}(\mathsf{ck}); ((\mathbf{k.tr}_k)_{k\in[1,\kappa_v+2]}, \mathbf{k.tr}^{\boldsymbol{\omega}}) \leftarrow \mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{\mathfrak{T}}_{\beta\gamma\alpha});$
2 : $(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}(\mathsf{ck}, \hat{\mathfrak{T}}_{\beta\gamma\alpha});$
3 : $\mathsf{F}_0(X) \leftarrow \mathsf{a}(X)\mathsf{b}(X)\mathsf{q}_{\mathsf{M}}(X) + \mathsf{a}(X)\mathsf{q}_{\mathsf{L}}(X) + \mathsf{b}(X)\mathsf{q}_{\mathsf{R}}(X) + \mathsf{c}(X)\mathsf{q}_{\mathsf{O}}(X) + \mathsf{PI}(X) + \mathsf{q}_{\mathsf{C}}(X);$
4 : $\mathsf{F}_1(X) \leftarrow (\mathsf{a}(X) + \beta X + \gamma)(\mathsf{b}(X) + \beta k_1 X + \gamma)(\mathsf{c}(X) + \beta k_2 X + \gamma)\mathsf{z}(X)$
5 : $\quad -(\mathsf{a}(X) + \beta \mathsf{S}_{\sigma 1}(X) + \gamma)(\mathsf{b}(X) + \beta \mathsf{S}_{\sigma 2}(X) + \gamma)(\mathsf{c}(X) + \beta \mathsf{S}_{\sigma 3}(X) + \gamma)\mathsf{z}(\omega X);$
6 : $\mathsf{F}_2(X) \leftarrow (\mathsf{z}(X) - 1)L_1(X);$
7 : $\mathsf{F}(X) \leftarrow \mathsf{F}_0(X) + \alpha \mathsf{F}_1(X) + \alpha^2 \mathsf{F}_2(X); (**)$
8 : $\mathsf{t}(X) \leftarrow \mathsf{F}(X)/\mathsf{Z}_{\mathbb{H}}(X);$
9 : **if** $\mathsf{Z}_{\mathbb{H}}(X) \mid \mathsf{F}(X)$ **then return** $\bot$; **fi**
10 : **for** $i \in [1, \kappa_{\mathfrak{z}}]$ **do**
11 : $\quad \Lambda_{0i}(X) \leftarrow \bar{a}_i\bar{b}_i\mathsf{q}_{\mathsf{M}}(X) + \bar{a}_i\mathsf{q}_{\mathsf{L}}(X) + \bar{b}_i\mathsf{q}_{\mathsf{R}}(X) + \bar{c}_i\mathsf{q}_{\mathsf{O}}(X) + \mathsf{PI}(\mathfrak{z}) + \mathsf{q}_{\mathsf{C}}(X);$
12 : $\quad \Lambda_{1i}(X) \leftarrow (\bar{a}_i + \beta\mathfrak{z}_i + \gamma)(\bar{b}_i + \beta k_1\mathfrak{z}_i + \gamma)(\bar{c}_i + \beta k_2\mathfrak{z}_i + \gamma)\mathsf{z}(X)$
13 : $\quad\quad - (\bar{a}_i + \beta\bar{s}_{\sigma 1} + \gamma)(\bar{b}_i + \beta\bar{s}_{\sigma 2} + \gamma)(\bar{c}_i + \beta\mathsf{S}_{\sigma 3}(X) + \gamma)\bar{z}_{\omega,i};$
14 : $\quad \Lambda_{2i}(X) \leftarrow (\mathsf{z}(X) - 1)L_1(\mathfrak{z}_i);$
15 : $\quad \Lambda_i(X) \leftarrow \Lambda_{0i}(X) + \alpha\Lambda_{1i}(X) + \alpha^2\Lambda_{2i}(X);$
16 : $\quad \chi_i'(X) \leftarrow \frac{\Lambda_i(X) - \Lambda_i(\mathfrak{z}_i)}{X - \mathfrak{z}_i};$
17 : $\quad [\mathsf{W}'_{\mathfrak{z}i1}, \ldots, \mathsf{W}'_{\mathfrak{z}i\kappa_v}]_1^{\intercal} \leftarrow \boldsymbol{V}_i^{-1}[\mathsf{W}_{\mathfrak{z}i1}, \ldots, \mathsf{W}_{\mathfrak{z}i\kappa_v}]_1^{\intercal};$ // $\boldsymbol{V}_i$ as in Eq. (15)
18 : $\quad [\chi_i]_1 \leftarrow \frac{1}{\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)}[\chi_i'(x) - \mathsf{W}'_{\mathfrak{z}i1}]_1;$ // $\mathsf{W}'_{\mathfrak{z}i1}$ is as in Line 8 in Fig. 6
19 : $\quad \Delta_i \leftarrow 1/\prod_{j\neq i}(\mathfrak{z}_i - \mathfrak{z}_j)$; **endfor**
20 : $[\varphi]_1 \leftarrow \sum_{i=1}^{\kappa_{\mathfrak{z}}} \Delta_i[\chi_i]_1;$
21 : $L(X) \leftarrow \sum_{i=1}^{\kappa_{\mathfrak{z}}} \mathsf{t}(\mathfrak{z}_i)L_i^S(X);$ // $L_i^S(X)$ are Lagrange basis polynomials
22 : // Only the return value depends on whether we have Plonk or SmallPlonk
23 : **return** $(\mathcal{S} = \{\mathfrak{z}_i\}_{i=1}^{\kappa_{\mathfrak{z}}}, L(X), [t, \varphi]_1);$ // In SmallPlonk
24 : **return** $(\mathcal{S} = \{\mathfrak{z}_i\}_{i=1}^{\kappa_{\mathfrak{z}}}, L(X), [t_{lo}, t_{mid}, t_{hi}, \varphi]_1);$ // In Plonk

Fig. 15. The SplitRSDH adversary $\mathcal{D}_{\mathrm{splitrsdh}}$.

in the case of SmallPlonk. (here, we took into account the concrete par).

Recall from the proof of Theorem 5 that $\mathsf{bad}_{\mathsf{rhino}} = \mathsf{true}$ implies $\mathsf{bad}_{\mathsf{evb}} = \mathsf{bad}_{\mathsf{ext}} = \mathsf{false}$. Thus,

1. $[z]_1 = [\mathsf{z}(x)]_1$, $[a]_1 = [\mathsf{a}(x)]_1$, $[b]_1 = [\mathsf{b}(x)]_1$, $[c]_1 = [\mathsf{c}(x)]_1$, and
2. $\mathsf{z}(\mathfrak{z}_i\omega) = \bar{z}_{\omega i}$, $\mathsf{a}(\mathfrak{z}_i) = \bar{a}_i$, $\mathsf{b}(\mathfrak{z}_i) = \bar{b}_i$, and $\mathsf{c}(\mathfrak{z}_i) = \bar{c}_i$ for $i \in [1, \kappa_{\mathfrak{z}}]$.

Thus, $\mathsf{F}_s(\mathfrak{z}_i) = \Lambda_{si}(\mathfrak{z}_i)$, $\mathsf{F}(\mathfrak{z}_i) = \Lambda_i(\mathfrak{z}_i)$, and $\mathsf{t}(\mathfrak{z}_i) = \Lambda_i(\mathfrak{z}_i)/\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)$ for $s \in \{0, 1, 2\}$ and $i \in [1, \kappa_{\mathfrak{z}}]$.

Denote $y_i := t_{lo} + \mathfrak{z}_i^n t_{mid} + \mathfrak{z}_i^{2n} t_{hi}$ in the case of Plonk and $y_i := t$ in the case of SmallPlonk. Recall

$$[r_i]_1 = [\Lambda_i(x) - \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)y_i]_1 \tag{20}$$

from the lines 3, 4, and 6 in Fig. 6. Next, we use the fact that $\mathcal{D}_{\mathrm{splitrsdh}}$ knows $\Lambda_i(X)$ as a polynomial. Since it can open both $[r_i]_1$ and $[\Lambda_i(x)]_1$ at $\mathfrak{z}_i$, it can also open $[y_i]_1$. More precisely, let

$$\chi_i'(X) = (\Lambda_i(X) - \Lambda_i(\mathfrak{z}_i))/(X - \mathfrak{z}_i) = (\Lambda_i(X) - \mathsf{F}(\mathfrak{z}_i))/(X - \mathfrak{z}_i)$$

be the KZG opening polynomial of $\Lambda_i(X)$, $i \in [1, \kappa_{\mathfrak{z}}]$, at point $\mathfrak{z}_i$. The second equality follows from $\mathsf{bad}_{\mathsf{evb}} = \mathsf{bad}_{\mathsf{ext}} = \mathsf{false}$.

Recall that as part of $\mathsf{k.tr}_{1i}$ (see line 9 in Fig. 6), $[\mathsf{W}'_{\mathfrak{z}i1}]_1$ is the opening proof of $[r_i]_1$ at $\mathfrak{z}_i$ to 0. Since $\mathsf{k.tr}_{1i}$ is accepting, Eq. (20) implies that $\mathsf{W}'_{\mathfrak{z}i1} \cdot (x - \mathfrak{z}_i) = r_i$. It follows from this, $\mathsf{bad}_{\mathsf{evb}} = \mathsf{bad}_{\mathsf{ext}} = \mathsf{false}$, and the definition of $\chi'_i(X)$ that

$$\mathsf{W}'_{\mathfrak{z}i1} \cdot (x - \mathfrak{z}_i) = r_i = \chi'_i(x)(x - \mathfrak{z}_i) + \Lambda_i(\mathfrak{z}_i) - \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) y_i$$

and thus

$$\chi'_i(x) - \mathsf{W}'_{\mathfrak{z}i1} = \tfrac{1}{x - \mathfrak{z}_i}(\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) y_i - \Lambda_i(\mathfrak{z}_i))$$

for $i \in [1, \kappa_{\mathfrak{z}}]$. Since $\Lambda_i(\mathfrak{z}_i) = \mathsf{F}(\mathfrak{z}_i)$ and $\mathsf{t}(\mathfrak{z}_i) = \mathsf{F}(\mathfrak{z}_i)/\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)$, we get $\chi'_i(x) - \mathsf{W}'_{\mathfrak{z}i1} = \frac{1}{x - \mathfrak{z}_i} \cdot (\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) y_i - \mathsf{F}(\mathfrak{z}_i)) = \frac{\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)}{x - \mathfrak{z}_i}(y_i - \mathsf{t}(\mathfrak{z}_i))$. Defining $\chi_i \leftarrow \frac{1}{\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)}(\chi'_i(x) - \mathsf{W}'_{\mathfrak{z}i1})$ as in Fig. 15 for $i \in [1, \kappa_{\mathfrak{z}}]$, we get

$$y_i - \mathsf{t}(\mathfrak{z}_i) = \chi_i(x - \mathfrak{z}_i) \ . \tag{21}$$

We multiply $\kappa_{\mathfrak{z}}$ equations Eq. (21) individually by $L^S_i(X)$ and then sum the results, getting

$$\textstyle\sum_{i=1}^{\kappa_{\mathfrak{z}}}(y_i - \mathsf{t}(\mathfrak{z}_i)) L^S_i(x) = \sum_{i=1}^{\kappa_{\mathfrak{z}}} \chi_i(x - \mathfrak{z}_i) L^S_i(x) \ . \tag{22}$$

Since $L^S_i(X) = \prod_{j \neq i} \frac{X - \mathfrak{z}_j}{\mathfrak{z}_i - \mathfrak{z}_j} = \Delta_i \prod_{j \neq i}(X - \mathfrak{z}_j) = \Delta_i \frac{\mathsf{Z}_S(X)}{X - \mathfrak{z}_i}$, $(X - \mathfrak{z}_i) L^S_i(X) = \Delta_i \mathsf{Z}_S(X)$. Let $L(X) = \sum_{i=1}^{\kappa_{\mathfrak{z}}} \mathsf{t}(\mathfrak{z}_i) L^S_i(X)$ be the interpolating polynomial of $\{(\mathfrak{z}_i, \mathsf{t}(\mathfrak{z}_i))\}_{i=1}^{\kappa_{\mathfrak{z}}}$. Denote $T(x) := t_{lo} + t_{mid} x^n + t_{hi} x^{2n}$ in the case of Plonk and $T(x) := t$ in the case of $\mathsf{SmallPlonk}$. In the case of Plonk,

$$\begin{aligned} T(x) - L(x) &= t_{lo} + t_{mid} x^n + t_{hi} x^{2n} - L(x) \\ &\overset{(*)}{=} t_{lo} + t_{mid} \textstyle\sum_{i=1}^{\kappa_{\mathfrak{z}}} \mathfrak{z}_i^n L^S_i(x) + t_{hi} \sum_{i=1}^{\kappa_{\mathfrak{z}}} \mathfrak{z}_i^{2n} L^S_i(x) - L(x) \\ &= \textstyle\sum_{i=1}^{\kappa_{\mathfrak{z}}}(y_i - \mathsf{t}(\mathfrak{z}_i)) L^S_i(x) \ , \end{aligned}$$

where (*) follows from (say) $X^n = \sum_{i=1}^{\kappa_{\mathfrak{z}}} \mathfrak{z}_i^n L^S_i(X)$. In the case of $\mathsf{SmallPlonk}$,

$$T(x) - L(x) = t - L(x) = \textstyle\sum_{i=1}^{\kappa_{\mathfrak{z}}}(y_i - \mathsf{t}(\mathfrak{z}_i)) L^S_i(x) \ .$$

Thus,

$$T(x) - L(x) \overset{22}{=} \textstyle\sum_{i=1}^{\kappa_{\mathfrak{z}}} \chi_i(x - \mathfrak{z}_i) L^S_i(x) = \mathsf{Z}_S(x) \sum_{i=1}^{\kappa_{\mathfrak{z}}} \Delta_i \chi_i = \mathsf{Z}_S(x) \varphi \ .$$

Thus, Eq. (18) (in the case of Plonk) or Eq. (19) (in the case of $\mathsf{SmallPlonk}$) holds.

Finally, clearly $\deg L(X) \leq \kappa_{\mathfrak{z}} - 1$. On the other hand, assume $\deg L(X) \leq \kappa_{\mathsf{kzg}} + n_\psi \leq \kappa_{\mathfrak{z}} - n - 1$. In the case of Plonk, the last inequality holds due to the choice of $\mathsf{par}^{\mathsf{plonk}}_n$, since $\kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}} + 1 = 4n + 21 \geq (n + 5) + 2n + n + 1 = \kappa_{\mathsf{kzg}} + n_\psi + n + 1$. In $\mathsf{SmallPlonk}$, it is even more obvious since $\kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}} + 1 = 12n + 21 \geq (3n + 5) + 0 + n + 1 = \kappa_{\mathsf{kzg}} + n_\psi + n + 1$. Since $L(X)$ and $\mathsf{F}(X)/\mathsf{Z}_{\mathbb{H}}(X)$ match on $\kappa_{\mathfrak{z}}$ points $\mathfrak{z}_i$ and both $L(X)\mathsf{Z}_{\mathbb{H}}(X)$ and $\mathsf{F}(X)$ have degree $\leq \kappa_{\mathfrak{z}} - 1$, $L(X) = \mathsf{F}(X)/\mathsf{Z}_{\mathbb{H}}(X)$, which means $\mathsf{Z}_{\mathbb{H}}(X) \mid \mathsf{F}(X)$, contradiction with $\mathsf{bad}_{\mathsf{rhino}} = \mathsf{true}$. Thus, if $\mathsf{Z}_{\mathbb{H}}(X) \nmid \mathsf{F}(X)$, then $\deg L(X) \in [\kappa_{\mathsf{kzg}} + n_\psi + 1, \kappa_{\mathfrak{z}} - 1]$. $\square$

D.4 SanPlonk's Case from Theorem 5

Proof (Finishing the proof of Theorem 5).

Case of SanPlonk. The case of SanPlonk is similar to Plonk but differs in quite many details. For the sake of clarity, we highlight additional text, but we also removed some text that is not relevant for SanPlonk. (Intuitively, the removed text corresponds to the part in Plonk's proof where one handles the reduction to SplitRSDH.)

In Fig. 12, we depict an extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$. $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$ invokes $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}^{\omega})$ and $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}_k)$ for $k \in [2,4] \cup [7,9]$, extracting polynomials $\mathsf{z}(X)$, $\mathsf{a}(X)$, $\mathsf{b}(X)$, $\mathsf{c}(X)$, $\mathsf{t}_{\mathsf{lo}}(X)$, $\mathsf{t}_{\mathsf{mid}}(X)$, and $\mathsf{t}_{\mathsf{hi}}(X)$ of at most degree $\kappa_{\mathsf{kzg}} = n+5$. After executing $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$, we use the following procedure to possibly set one of the "bad" flags:

(i') $\mathsf{bad}_{\mathsf{ext}} \leftarrow \mathsf{false}$; $\mathsf{bad}_{\mathsf{evb}} \leftarrow \mathsf{false}$;

(ii') if $([z]_1, \mathsf{z}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_1} \vee ([a]_1, \mathsf{a}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_2} \vee ([b]_1, \mathsf{b}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_3} \vee ([c]_1, \mathsf{c}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_4} \vee ([t_{lo}]_1, \mathsf{t}_{\mathsf{lo}}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_7} \vee ([t_{mid}]_1, \mathsf{t}_{\mathsf{mid}}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_8} \vee ([t_{lo}]_1, \mathsf{t}_{\mathsf{hi}}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_9}$ (see Eq. (1)) then $\mathsf{bad}_{\mathsf{ext}} \leftarrow \mathsf{true}$; abort;

(iii') for $i \in [\kappa_{\mathsf{kzg}} + 2, \kappa_{\mathfrak{z}}]$: if $\mathsf{z}(\mathfrak{z}_i\omega) \neq \bar{z}_{\omega i} \vee \mathsf{a}(\mathfrak{z}_i) \neq \bar{a}_i \vee \mathsf{b}(\mathfrak{z}_i) \neq \bar{b}_i \vee \mathsf{c}(\mathfrak{z}_i) \neq \bar{c}_i \vee \mathsf{t}_{\mathsf{lo}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z}lo,i} \vee \mathsf{t}_{\mathsf{mid}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z}mid,i} \vee \mathsf{t}_{\mathsf{hi}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z}hi,i}$ then $\mathsf{bad}_{\mathsf{evb}} \leftarrow \mathsf{true}$; abort;

(iv') for $i \in [1, \kappa_{\mathfrak{z}}]$:

- if $\mathsf{S}_{\sigma 1}(\mathfrak{z}_i) \neq \bar{s}_{\sigma 1 i} \vee \mathsf{S}_{\sigma 2}(\mathfrak{z}_i) \neq \bar{s}_{\sigma 2 i}$ then $\mathsf{bad}_{\mathsf{evb}} \leftarrow \mathsf{true}$; abort;
- $\mathsf{r}_i(X) \leftarrow \Lambda_{0i}(X) + \alpha\Lambda_{1i}(X) + \alpha^2\Lambda_{2i}(X) - \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) \cdot (\mathsf{t}_{\mathsf{lo}}(X) + \mathfrak{z}_i^n \mathsf{t}_{\mathsf{mid}}(X) + \mathfrak{z}_i^{2n} \mathsf{t}_{\mathsf{hi}}(X))$;
- if $\mathsf{r}_i(\mathfrak{z}_i) \neq 0$ then $\mathsf{bad}_{\mathsf{evb}} \leftarrow \mathsf{true}$;

Importantly, only one of the "bad" flags is set at a time. Thus, for example, $\mathsf{bad}_{\mathsf{evb}} = \mathsf{true}$ implies that $\mathsf{bad}_{\mathsf{ext}} = \mathsf{false}$. Let $\mathcal{E}$ be the event $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$ succeeds and $\overline{\mathsf{bad}}$ be the event none of the bad flags was set. Thus, $\mathcal{E}$ is the event that $\mathsf{a}(X)$, $\mathsf{b}(X)$, $\mathsf{c}(X)$, $\mathsf{z}(X)$, $\mathsf{t}_{\mathsf{lo}}(X)$, $\mathsf{t}_{\mathsf{mid}}(X)$, and $\mathsf{t}_{\mathsf{hi}}(X)$ are consistent with the commitments and all openings, and $\mathsf{t}(X) = (\mathsf{F}_0(X) + \alpha\mathsf{F}_1(X) + \alpha^2\mathsf{F}_2(X))/\mathsf{Z}_{\mathbb{H}}(X)$ is a polynomial. We analyze the success probability of $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$. For this, we make the following claims.

1. **Claim 1**. $\Pr[\mathcal{E} | \overline{\mathsf{bad}}] = 1$.
   Really, assume that $\overline{\mathsf{bad}}$ holds. Since $\mathsf{bad}_{\mathsf{ext}} = \mathsf{bad}_{\mathsf{evb}} = \mathsf{false}$, we get that $\mathsf{a}(X)$, $\mathsf{b}(X)$, $\mathsf{c}(X)$, $\mathsf{z}(X)$, $\mathsf{t}_{\mathsf{lo}}(X)$, $\mathsf{t}_{\mathsf{mid}}(X)$, $\mathsf{t}_{\mathsf{hi}}(X)$ are consistent with the commitments and all openings.
   Recall that in the case of Plonk, we deduced here that since $\mathsf{bad}_{\mathsf{rhino}} = \mathsf{false}$, $\mathsf{t}(X) = (\mathsf{F}_0(X) + \alpha\mathsf{F}_1(X) + \alpha^2\mathsf{F}_2(X))/\mathsf{Z}_{\mathbb{H}}(X)$ is a polynomial. In the case of SanPlonk, we handle this differently. Since $\mathsf{bad}_{\mathsf{evb}} = \mathsf{false}$, $\mathsf{r}_i(X) = 0$ for all $i \in [1, \kappa_{\mathfrak{z}}]$. Let $g(X) := \mathsf{F}(X) - \mathsf{Z}_{\mathbb{H}}(X)\mathsf{t}^*(X)$, where $\mathsf{t}^*(X) := \mathsf{t}_{\mathsf{lo}}(X) + X^n\mathsf{t}_{\mathsf{mid}}(X) + X^{2n}\mathsf{t}_{\mathsf{hi}}(X)$. Since for $i \in [1, \kappa_{\mathfrak{z}}]$, $\mathsf{F}_s(\mathfrak{z}_i) = \Lambda_{si}(\mathfrak{z}_i)$, we have $g(\mathfrak{z}_i) = 0$. Since this holds for $\kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}} + 1$ evaluation points and $\deg g(X) \leq 4\kappa_{\mathsf{kzg}}$, $g(X) = 0$. Thus, $\mathsf{t}(X)$ is a polynomial.

2. **Claim 2**. There exists a KZG's $(\kappa_{\mathsf{kzg}}+1)$-special-soundness adversary $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ (see Fig. 13), such that $\Pr[\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}} \text{ succeeds} \mid \mathsf{bad}_{\mathsf{ext}}] = 1$.
   Recall from Item ii that $\mathsf{bad}_{\mathsf{ext}}$ is set if one of the seven bad events happens. $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ just tests which of the cases is true and returns the corresponding transcript. Clearly, $\Pr[\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}} \text{ succeeds} \mid \mathsf{bad}_{\mathsf{ext}}] = 1$.
3. **Claim 3**. There exists an evaluation-binding adversary $\mathcal{C}_{\mathsf{evb}}$ (see Fig. 14) for KZG, such that $\Pr[\mathcal{C}_{\mathsf{evb}} \text{ succeeds} \mid \mathsf{bad}_{\mathsf{evb}}] = 1$.
   Assume that $\mathsf{bad}_{\mathsf{evb}} = \mathsf{true}$. (Note that $\mathsf{bad}_{\mathsf{evb}} = \mathsf{true}$ means that $\mathsf{bad}_{\mathsf{ext}} = \mathsf{false}$, that is, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$ managed to extract all polynomials.) Then, one of the bad cases in Item iii or Item iv happens. $\mathcal{C}_{\mathsf{evb}}$ just finds out which of these events happens, and depending on the case, returns a collision. By the correctness of the extraction, and the completeness property of KZG, any of the returned values in Fig. 14 *is* a collision. Thus, $\Pr[\mathcal{C}_{\mathsf{evb}} \text{ succeeds} \mid \mathsf{bad}_{\mathsf{evb}}] = 1$.

Thus,

$$\begin{aligned}\Pr[\overline{\mathcal{E}}] &= \Pr[\overline{\mathcal{E}}|\overline{\mathsf{bad}}]\Pr[\overline{\mathsf{bad}}] + \Pr[\overline{\mathcal{E}}|\mathsf{bad}_{\mathsf{ext}}]\Pr[\mathsf{bad}_{\mathsf{ext}}] + \Pr[\overline{\mathcal{E}}|\mathsf{bad}_{\mathsf{evb}}]\Pr[\mathsf{bad}_{\mathsf{evb}}] \\ &\leq 0 + \Pr[\mathsf{bad}_{\mathsf{ext}}] + \Pr[\mathsf{bad}_{\mathsf{evb}}]\end{aligned}$$

Since $\mathcal{C}_{\mathsf{evb}}$ succeeds whenever $\mathsf{bad}_{\mathsf{evb}}$ is set and KZG is evaluation-binding, $\Pr[\mathsf{bad}_{\mathsf{evb}}] = \mathsf{negl}(\lambda)$. Similarly, $\Pr[\mathsf{bad}_{\mathsf{ext}}] = \mathsf{negl}(\lambda)$. Thus, $\Pr[\overline{\mathcal{E}}] \leq \mathsf{negl}(\lambda)$. This proves the claim. □

D.5 Proof of Theorem 6

In this section, we restate and then prove the theorem Theorem 6 from Section 6.

Theorem 9. *Let $n \in \mathsf{poly}(\lambda)$ and κ_{kzg}, $\boldsymbol{\kappa}_{\mathsf{Plonk}}$, and $\boldsymbol{\kappa}_{\mathsf{san}}$ be as in Eq. (12).*

1. *If KZG is computational $(\kappa_{\mathsf{kzg}}+1)$-special-sound and evaluation-binding, and $\mathsf{par}_n^{\mathsf{smallplonk}}$-SplitRSDH holds, then $\mathsf{SmallPlonk}$ is computational $\boldsymbol{\kappa}_{\mathsf{Plonk}}$-special-sound.*
2. *If KZG is computational $(\kappa_{\mathsf{kzg}}+1)$-special-sound and evaluation-binding, and $\mathsf{par}_n^{\mathsf{plonk}}$-SplitRSDH holds, then Plonk is computational $\boldsymbol{\kappa}_{\mathsf{Plonk}}$-special-sound.*
3. *If KZG is computational $(\kappa_{\mathsf{kzg}}+1)$-special-sound and evaluation-binding, then $\mathsf{SanPlonk}$ is computational $\boldsymbol{\kappa}_{\mathsf{san}}$-special-sound.*

Proof. Fix $\mathscr{P} = \{\mathsf{q}_{\mathsf{M}}(X), \mathsf{q}_{\mathsf{L}}(X), \mathsf{q}_{\mathsf{R}}(X), \mathsf{q}_{\mathsf{O}}(X), \mathsf{q}_{\mathsf{C}}(X), \mathsf{S}_{\sigma 1}(X), \mathsf{S}_{\sigma 2}(X), \mathsf{S}_{\sigma 3}(X)\}$ for a relation defined by encoding a circuit as in Section 6.1. Let $\mathcal{A}_{\mathsf{ss}}$ be a PPT adversary in the computational special-soundness game that outputs a $(\kappa_\beta, \kappa_\gamma, \kappa_\alpha, \kappa_{\mathfrak{z}}, \kappa_\delta, \kappa_v)$-tree $\mathcal{T}$. We describe the special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{*\mathsf{plonk}}$ for Plonk in Fig. 16. The extractor picks an arbitrary $(1,1,1,\kappa_{\mathfrak{z}},\kappa_\delta,\kappa_v)$-subtree $\hat{\mathcal{T}} = \hat{\mathcal{T}}_{\beta\gamma\alpha}$ of $\mathcal{T}$ and runs the subtree extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$ from Theorem 5 on it to obtain the polynomials $\mathsf{z}(X)$, $\mathsf{a}(X)$, $\mathsf{b}(X)$, $\mathsf{c}(X)$, $\mathsf{t}_{\mathsf{lo}}(X)$, $\mathsf{t}_{\mathsf{mid}}(X)$, and $\mathsf{t}_{\mathsf{hi}}(X)$. In the honest protocol, the witness is encoded in $\mathsf{a}(X)$, $\mathsf{b}(X)$, and $\mathsf{c}(X)$. Namely, $\bar{w}_i \leftarrow \mathsf{a}(\omega^i)$, $\bar{w}_{n+i} \leftarrow \mathsf{b}(\omega^i)$, $\bar{w}_{2n+i} \leftarrow \mathsf{c}(\omega^i)$ for $i \in [1,n]$, and $\mathbb{w} = (\bar{w}_i)_{i=\ell+1}^{3n}$. The rest of the proof shows that $(\mathbb{x} = (\bar{w}_i)_{i=1}^{\ell}, \mathbb{w}) \in \mathcal{R}_{\mathscr{P}}$.

$\mathsf{Ext}_{\mathsf{ss}}^{*\mathsf{plonk}}(\mathsf{ck}, \mathbb{x}, \mathcal{T})$

Pick an arbitrary $(1, 1, 1, \kappa_3, \kappa_\delta, \kappa_v)$-subtree $\hat{\mathcal{T}}$ of $\mathcal{T}$.
$(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{t}_{\mathsf{lo}}(X), \mathsf{t}_{\mathsf{mid}}(X), \mathsf{t}_{\mathsf{hi}}(X)) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}(\mathsf{ck}, \hat{\mathcal{T}})$;
for $i \in [1, n]$ **do**
  $\bar{w}_i \leftarrow \mathsf{a}(\omega^i); \bar{w}_{n+i} \leftarrow \mathsf{b}(\omega^i); \bar{w}_{2n+i} \leftarrow \mathsf{c}(\omega^i)$; **endfor**
return $\mathbb{w} \leftarrow (\bar{w}_i)_{i=\ell+1}^{3n}$;

Fig. 16. The special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{*\mathsf{plonk}}$ for Plonk/SanPlonk.

To be sure we compute a correct witness, we must assume that for each β_i, we have κ_γ mutually different values γ_{ij}. Whence the double index on challenges γ_{ij}, despite they are sent in the same round by Plonk and its variants' prover. One can be implement this by rewinding the protocol with $\kappa_\beta \kappa_\gamma$ different challenges $(\beta_i \gamma_{ij})$. Alternatively, one can consider Plonk as the interactive argument where the prover first receives the challenge β from the verifier, then replies with an empty message, then receives the challenge γ, and goes on with the execution.[7]

Let $\mathsf{SubTrees}_{\mathcal{T}} := \{\hat{\mathcal{T}}_{\beta_i \gamma_{ij} \alpha_{ijk}} : i \in [1, \kappa_\beta], j \in [1, \kappa_\gamma], k \in [1, \kappa_\alpha]\}$ be the set of all $(1, 1, 1, \kappa_3, \kappa_\delta, \kappa_v)$-subtrees of $\mathcal{T}$. For each $\hat{\mathcal{T}} \in \mathsf{SubTrees}_{\mathcal{T}}$, we can apply the extractor from Theorem 5 and obtain the subtree transcript $\mathsf{s.tr}_{\hat{\mathcal{T}}} = (\mathsf{z}_{\hat{\mathcal{T}}}(X), \mathsf{a}_{\hat{\mathcal{T}}}(X), \mathsf{b}_{\hat{\mathcal{T}}}(X), \mathsf{c}_{\hat{\mathcal{T}}}(X))$, where $\hat{\mathcal{T}} = \hat{\mathcal{T}}_{\beta_i \gamma_{ij} \alpha_{ijk}}$ and $i \in [1, \kappa_\beta]$, $j \in [1, \kappa_\gamma]$, and $k \in [1, \kappa_\alpha]$. Observe that the extracted polynomials may depend on the specific subtree $\hat{\mathcal{T}}$.

Next, we argue that the extractor $\mathsf{Ext}_{\mathsf{ss}}^{*\mathsf{plonk}}$ can fail only if one of the following events happens.

$\mathsf{bad}_{\mathsf{sub}}$: For some $\hat{\mathcal{T}} \in \mathsf{SubTrees}_{\mathcal{T}}$, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}(\mathsf{ck}, \hat{\mathcal{T}})$ outputs $\mathsf{s.tr}_{\hat{\mathcal{T}}}$ such that either (1) $\mathsf{z}_{\hat{\mathcal{T}}}(X)$, $\mathsf{a}_{\hat{\mathcal{T}}}(X)$, $\mathsf{b}_{\hat{\mathcal{T}}}(X)$, and $\mathsf{c}_{\hat{\mathcal{T}}}(X)$ are inconsistent with the commitments $[z_{ij}, a, b, c]_1$, or (2) $\mathsf{t}_{\hat{\mathcal{T}} ijk}(X)$ (defined as in Eq. (7)) is not a polynomial.

$\mathsf{bad}_{\mathsf{bind}}$: (1) The event $\mathsf{bad}_{\mathsf{sub}}$ does not happen. (2) Define $\mathsf{W}_{\hat{\mathcal{T}}} := \{(\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)) : (\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}(\mathsf{ck}, \hat{\mathcal{T}})\}$. There exist two subtrees $\hat{\mathcal{T}} \neq \hat{\mathcal{T}}' \in \mathsf{SubTrees}_{\mathcal{T}}$, such that $\mathsf{W}_{\hat{\mathcal{T}}} \neq \mathsf{W}_{\hat{\mathcal{T}}'}$.

Theorem 5 implies that if KZG is evaluation-binding and computationally special-sound (and the additional assumption SplitRSDH holds in the case of Plonk or SmallPlonk), then $\Pr[\mathsf{bad}_{\mathsf{sub}}]$ is negligible. The fact that $\Pr[\mathsf{bad}_{\mathsf{bind}}]$ is negligible follows straightforwardly from the binding property of KZG. For the sake of completeness we present the binding adversary $\mathcal{A}_{\mathsf{bind}}$ in Fig. 17.

In the following, suppose that neither $\mathsf{bad}_{\mathsf{sub}}$ or $\mathsf{bad}_{\mathsf{bind}}$ happened. Thus, $\mathsf{W}_{\hat{\mathcal{T}}}(X) = \mathsf{W}_{\hat{\mathcal{T}}'}(X)$ for any $\hat{\mathcal{T}}, \hat{\mathcal{T}} \in \mathsf{SubTrees}_{\mathcal{T}}$. This justifies the notation $\mathsf{s.tr}_{ijk} = (\mathsf{z}_{ij}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X))$, where $\mathsf{a}(X)$, $\mathsf{b}(X)$, and $\mathsf{c}(X)$ do not depend on the specific subtree $\hat{\mathcal{T}}$ while $\mathsf{z}_{ij}(X)$ depends on β_i and γ_{ij}. Further-

[7] We note that this does not change actual Plonk/SanPlonk: when applying Fiat-Shamir, one defines $\beta = H(view, 0)$ and $\gamma = H(view, 1)$. To rewind only γ and not β, one can reprogram the random oracle at input $(view, 1)$ but not input $(view, 0)$.

$\mathcal{A}_{\mathsf{bind}}(\mathsf{ck})$

$(\mathbb{x}, \mathcal{T}) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck})$;
for $\hat{\mathcal{T}} \in \mathsf{SubTrees}_{\mathcal{T}}$ **do**
  $\mathsf{s.tr}_{\hat{\mathcal{T}}} \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}(\mathsf{ck}, \hat{\mathcal{T}})$; **endfor** // Extracts polynomials from each subtree
for distinct $\hat{\mathcal{T}} \neq \hat{\mathcal{T}}' \in \mathsf{SubTrees}_{\mathcal{T}}$ **do**
  $(\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)) \leftarrow \mathsf{W}_{\hat{\mathcal{T}}}; (\mathsf{a}'(X), \mathsf{b}'(X), \mathsf{c}'(X)) \leftarrow \mathsf{W}_{\hat{\mathcal{T}}'}$;
  if $\mathsf{a}(X) \neq \mathsf{a}'(X)$ **then return** $([a]_1, \mathsf{a}(X), \mathsf{a}'(X))$;
  if $\mathsf{b}(X) \neq \mathsf{b}'(X)$ **then return** $([b]_1, \mathsf{b}(X), \mathsf{b}'(X))$;
  if $\mathsf{c}(X) \neq \mathsf{c}'(X)$ **then return** $([c]_1, \mathsf{c}(X), \mathsf{c}'(X))$; **endfor**
return $\bot$;

Fig. 17. The binding adversary $\mathcal{A}_{\mathsf{bind}}$ in Theorem 6.

more, each $\mathsf{t}_{ijk}(X) := \mathsf{F}_{ijk}(X)/\mathsf{Z}_{\mathbb{H}}(X)$ is a polynomial, where $\mathsf{F}_{ijk}(X) := \mathsf{F}_0(X) + \alpha_{ijk}\mathsf{F}_{1ij}(X) + \alpha_{ijk}^2\mathsf{F}_{2ij}(X)$, and $\mathsf{F}_0(X)$, $\mathsf{F}_{1ij}(X)$ and $\mathsf{F}_{2ij}(X)$ are defined as in Eq. (7) (but they may depend on β_i and γ_{ij}). But then $\mathsf{F}_0(X) + \alpha_{ijk}\mathsf{F}_{1ij}(X) + \alpha_{ijk}^2\mathsf{F}_{2ij}(X) = \mathsf{Z}_{\mathbb{H}}(X)\mathsf{t}_{ijk}(X)$. Thus, for every $s \in [1, n]$, $\mathsf{F}_0(\omega^s) + \alpha_{ijk}\mathsf{F}_{1ij}(\omega^s) + \alpha_{ijk}^2\mathsf{F}_{2ij}(\omega^s) = 0$. Let

$$\boldsymbol{A}_{ij} = \begin{pmatrix} 1 & \alpha_{ij1} & \alpha_{ij1}^2 \\ 1 & \alpha_{ij2} & \alpha_{ij2}^2 \\ 1 & \alpha_{ij3} & \alpha_{ij3}^2 \end{pmatrix} .$$

be a Vandermonde matrix. Then, $\boldsymbol{A}_{ij} \cdot (\mathsf{F}_0(\omega^s), \mathsf{F}_{1ij}(\omega^s), \mathsf{F}_{2ij}(\omega^s))^\intercal = 0$. Since α_{ij1}, α_{ij2}, and α_{ij3} are distinct, $\boldsymbol{A}_{ij}$ is invertible. Thus, $\mathsf{F}_0(\omega^s) = \mathsf{F}_{1ij}(\omega^s) = \mathsf{F}_{2ij}(\omega^s) = 0$. We analyze these three equalities individually.

$\underline{\mathsf{F}_0(\omega^s) = 0}$. Let $\bar{w}_s := \mathsf{a}(\omega^s)$, $\bar{w}_{n+s} := \mathsf{b}(\omega^s)$, and $\bar{w}_{2n+s} := \mathsf{c}(\omega^s)$. Then, $\underline{\mathsf{F}_0(\omega^s) = 0}$ iff $\mathsf{q}_{\mathsf{M}s}\bar{w}_s\bar{w}_{n+s} + \mathsf{q}_{\mathsf{L}s}\bar{w}_s + \mathsf{q}_{\mathsf{R}s}\bar{w}_{n+s} + \mathsf{q}_{\mathsf{O}s}\bar{w}_{2n+s} + \mathsf{q}_{Cs} + \mathsf{PI}(\omega^s) = 0$. For $s > \ell$, $\mathsf{PI}(\omega^s) = 0$ and we obtain the constraint in Eq. (5). Recall that for $s \leq \ell$, $\mathsf{q}_{\mathsf{M}s} = \mathsf{q}_{\mathsf{R}s} = \mathsf{q}_{\mathsf{O}s} = \mathsf{q}_{Cs} = 0$ and $\mathsf{q}_{\mathsf{L}s} = -1$ (see Eq. (4)). Thus, $-\bar{w}_s + \mathsf{PI}(\omega^s) = -\bar{w}_s + w_s = 0$. It follows that $w_s = \bar{w}_s$ for $1 \leq s \leq \ell$. Hence, $\mathsf{a}(X)$ encoded the public statement $\mathbb{x}$ correctly.

$\underline{\mathsf{F}_{2ij}(\omega^s) = 0}$. We get $\mathsf{F}_{2ij}(\omega^s) = (\mathsf{z}_{ij}(\omega^s) - 1)L_1(\omega^s) = 0$ for all $\omega^s \in \mathbb{H}$. Thus, $\mathsf{z}_{ij}(\omega) = 1$.

$\underline{\mathsf{F}_{1ij}(\omega^s) = 0}$. We will conclude from $\mathsf{F}_{1ij}(\omega^s) = 0$ that $w_j = w_{\sigma(j)}$ for all $j \in [1, 3n]$. We will first prove a warm-up lemma (Lemma 4), which we will later expand to a more technical result Lemma 5 that better suits our needs.

Lemma 4. *Let σ be a permutation on $[1, n]$ and $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{F}$. Let $\beta_1, \ldots, \beta_{n+1} \in \mathbb{F}$ be mutually distinct and $\gamma_1, \ldots, \gamma_{n+1} \in \mathbb{F}$ be mutually distinct. If $\prod_{s=1}^{n}(a_s + \beta_i\omega^s + \gamma_j) = \prod_{s=1}^{n}(b_s + \beta_i\omega^{\sigma(s)} + \gamma_j)$ for all $i, j \in [1, n+1]$, then $b_s = a_{\sigma(s)}$ for all $s \in [1, n]$.*

Proof. Consider the polynomials

$$f(Y,Z) := \prod_{s=1}^{n}(a_s + \omega^s Y + Z)$$

and

$$g(Y,Z) := \prod_{s=1}^{n}(b_s + \omega^{\sigma(s)} Y + Z) \ ;$$

the degree of Y or Z in both f and g is at most n. Denote $B := \{\beta_s\}_{s=1}^{n+1}$ and $\Gamma := \{\gamma_s\}_{s=1}^{t+1}$.

Define the Lagrange polynomials $L_s^B(Y) := \prod_{i\neq s} \frac{Y-\beta_i}{\beta_s-\beta_i}$ and $L_s^\Gamma(Z) := \prod_{k\neq s} \frac{Z-\gamma_k}{\gamma_s-\gamma_k}$ for $s = 1,\ldots,n+1$. Clearly, $\{L_i^B(Y)\cdot L_j^\Gamma(Z)\}_{ij}$ is a basis of bivariate polynomials where each variable has at most degree n. Thus, we can express f and g uniquely as

$$f(Y,Z) := \sum_{i=1}^{n+1}\sum_{j=1}^{n+1} f_{ij} L_i^B(Y) L_j^\Gamma(Z) \ ,$$

$$g(Y,Z) := \sum_{i=1}^{n+1}\sum_{j=1}^{n+1} g_{ij} L_i^B(Y) L_j^\Gamma(Z) \ ,$$

for some $f_{ij}, g_{ij} \in \mathbb{F}$. Since by the hypothesis of the lemma, $f(\beta_i,\gamma_j) = f_{ij} = g(\beta_i,\gamma_j) = g_{ij}$ for all $i,j \in [1,n+1]$, it follows that $f(X,Y) = g(X,Y)$.

Observe that the polynomials $a_i+\omega^i Y+Z$ and $b_s+\omega^{\sigma(s)}Y+Z$ are irreducible. Thus, $f(X,Y) = g(X,Y)$ implies that for every s there exists exactly one i such that

$$a_s + \omega^s Y + Z = b_i + \omega^{\sigma(i)} Y + Z \ .$$

Thus, $\omega^i = \omega^{\sigma(s)}$, which implies that $i = \sigma(s)$, which in turn implies that $a_i = a_{\sigma(s)} = b_s$. The result follows since the above holds for all $s \in [1,n]$. □

Next, we prove a more involved version of Lemma 4, that directly applies to Plonk. Let us first define the polynomials $f_\vartheta(Y,Z) := \prod_{s=1}^{n}(w_{\vartheta n+s} + k_\vartheta \omega^s Y + Z)$ and $g_\vartheta(Y,Z) := \prod_{s=1}^{n}(w_{\vartheta n+s} + S_{\sigma,(\vartheta+1)}(\omega^s)Y + Z)$. for $\vartheta \in \{0,1,2\}$, where $k_0 := 1$ and k_1, k_2 are defined as in Section 6.1.

Lemma 5. *If* $\prod_{\vartheta=0}^{2} f_\vartheta(\beta_i,\gamma_{ij}) = \prod_{\vartheta=0}^{2} g_\vartheta(\beta_i,\gamma_{ij})$ *for all* $i,j \in [1,3n+1]$, *then* $w_s = w_{\sigma(s)}$ *for all* $s \in [1,3n]$.

Proof. Let $f(Y,Z) := \prod_{\vartheta=0}^{2} f_\vartheta(Y,Z)$ and $g(Y,Z) := \prod_{\vartheta=0}^{2} g_\vartheta(Y,Z)$. Both f and g have at most degree $3n$ in both Y and Z. Using the same reasoning as in Lemma 4, we conclude that $f(Y,Z) = g(Y,Z)$. The polynomials $w_{\vartheta n+s} + k_\vartheta\omega^s Y + Z$ are irreducible and pairwise distinct for all $\vartheta \in \{0,1,2\}$ and $s \in [1,n]$. The same holds for the polynomials $w_{\vartheta n+s} + S_{\sigma,(\vartheta+1)}(\omega^s)Y + Z$.

Recall that $\sigma^*(i) = \mathbb{H}'[\sigma(i)]$ for all $i \in [1, 3n]$ and $S_{\sigma,(\vartheta+1)}(\omega^s) = \sigma^*(\vartheta n+s) = \mathbb{H}'[\sigma(\vartheta n + s)]$. Therefore, we can express

$$f(Y, Z) = \prod_{j=1}^{3n}(w_i + Y\mathbb{H}'[i] + Z), \;\; g(Y, Z) = \prod_{i=1}^{3n}(w_i + Y\mathbb{H}'[\sigma(i)] + Z) \;\; .$$

Just as in Lemma 4, each factor $w_i + Y\mathbb{H}'[\sigma(i)] + Z$ of f is equal to exactly one factor $w_{i'} + Y\mathbb{H}'[i'] + Z$. Thus, $\mathbb{H}'[\sigma(i)] = \mathbb{H}'[i']$ and $w_i = w_{i'}$. The first identity implies that $\sigma(i) = i'$ and the second implies that $w_i = w_{i'} = w_{\sigma(i)}$. Since this holds for all $i \in [1, 3n]$, we have proven the lemma. □

We prove one more small result before proving our main result.

Lemma 6. *For all s, i, j,*

$$(w_s + \beta_i S_{\sigma 1}(\omega^s) + \gamma_{ij}) \cdot (w_{n+s} + \beta_i S_{\sigma 2}(\omega^s) + \gamma_{ij}) \cdot (w_{2n+s} + \beta_i \cdot \mathsf{S}_{\sigma 3}(\omega^s) + \gamma_{ij}) \neq 0.$$

Proof. Consider $w_s + \beta_i S_{\sigma 1}(\omega^s) + \gamma_{ij}$ as an example. We already noted that $w_s + S_{\sigma 1}(\omega^s)Y + Z$ is an irreducible polynomial, which means it has no roots in $\mathbb{F}$. Thus, (β_i, γ_{ij}) is not a root and $w_s + \beta_i S_{\sigma 1}(\omega^s) + \gamma_{ij} \neq 0$. Similarly, the other two factors are non-zero. Hence, their product is non-zero. □

We will inductively show that

$$\mathsf{z}_{ij}(\omega^{s+1}) = \prod_{t=1}^{s} \frac{(w_t + \beta_i\omega^t + \gamma_{ij}) \cdot (w_{n+t} + \beta_i k_1\omega^t + \gamma_{ij}) \cdot (w_{2n+t} + \beta_i k_2\omega^t + \gamma_{ij})}{(w_t + \beta_i S_{\sigma 1}(\omega^t) + \gamma_{ij}) \cdot (w_{n+t} + \beta_i S_{\sigma 2}(\omega^t) + \gamma_{ij}) \cdot (w_{2n+t} + \beta_i \mathsf{S}_{\sigma 3}(\omega^t) + \gamma_{ij})} \;\; . \quad (23)$$

Since we already showed that $\mathsf{z}_{ij}(\omega) = 1$, the claim holds for $s = 0$. Suppose the statement holds for $\mathsf{z}_{ij}(\omega^s)$. From $\mathsf{F}_{1ij}(\omega^i) = 0$ (see Eq. (7)), we conclude

$$\mathsf{z}_{ij}(\omega^{s+1}) = \frac{\mathsf{z}_{ij}(\omega^s) \cdot (w_s + \beta_i\omega^s + \gamma_{ij}) \cdot (w_{n+s} + \beta_i k_1\omega^s + \gamma_{ij}) \cdot (w_{2n+s} + \beta_i k_2\omega^s + \gamma_{ij})}{(w_s + \beta_i S_{\sigma 1}(\omega^s) + \gamma_{ij}) \cdot (w_{n+s} + \beta_i S_{\sigma 2}(\omega^s) + \gamma_{ij}) \cdot (w_{2n+s} + \beta_i \mathsf{S}_{\sigma 3}(\omega^s) + \gamma_{ij})} \;\; .$$

Note that the division is well-defined according to Lemma 6 and Eq. (23) follows by expanding $\mathsf{z}_{ij}(\omega^s)$.

Since $\omega^{n+1} = \omega$, we have $\mathsf{z}_{ij}(\omega^{n+1}) = 1$, implying that $\prod_{t=1}^{n}(w_t + \beta_i\omega^t + \gamma_{ij}) \cdot (w_{n+t} + \beta_i k_1\omega^t + \gamma_{ij}) \cdot (w_{2n+t} + \beta_i k_2\omega^t + \gamma_{ij}) = \prod_{t=1}^{n}(w_t + \beta_i S_{\sigma 1}(\omega^t) + \gamma_{ij}) \cdot (w_{n+t} + \beta_i S_{\sigma 2}(\omega^t) + \gamma_{ij}) \cdot (w_{2n+t} + \beta_i \cdot \mathsf{S}_{\sigma 3}(\omega^t) + \gamma_{ij})$. We can conclude from Lemma 5 that $w_i = w_{\sigma(i)}$ for all $i \in [1, 3n]$. Hence, $\mathbb{w}$ also satisfies the final constraint in Eq. (6). □

D.6 Fiat-Shamir Transform

Recall the following theorem from [AFK22].

Theorem 10 ([AFK22]). *Let Π be a $(\kappa_1, \ldots, \kappa_\mu)$-out-of-$(N_1, \ldots, N_\mu)$-special-sound interactive proof. Then, the Fiat-Shamir transformation $\mathsf{FS}[\Pi]$ of Π is knowledge-sound with knowledge error*

$$\mathsf{Er}_{\mathrm{fs}}(Q) = (Q + 1) \cdot \mathsf{Er},$$

where Q is the number of random oracle queries the adversary makes and $\mathsf{Er} = 1 - \prod_{i=1}^{\mu}\left(1 - \frac{\kappa_i - 1}{N_i}\right)$.

Importantly, in all our security proofs, k_i/N_i is negligible for all i, resulting in a negligible $\mathsf{Er}_{\mathrm{fs}}(Q)$. This holds since all verifier challenges (including the evaluation point $\mathfrak{z}$), used as branches in the transcript tree, in $\mathsf{SanPlonk}$ and Plonk are chosen randomly from $\mathbb{F}$ (or, a large subset of $\mathbb{F}$); thus, $N_i \approx |\mathbb{F}|$ and $k_i/N_i = \mathsf{negl}(\lambda)$.

While Theorem 10 applies to proof systems, it also extends to argument systems as explained in [AFKR23, Remark 1]. This is because the interactive argument system Π can be seen as a proof of knowledge for a slightly different relation: the knowledge of a witness for the underlying relation OR a solution to some computationally hard problem (in some cases more than one hard problem). Note that all of the computational special-soundness proofs in this paper showed that there exists a DPT extractor $\mathsf{Ext}_{\mathsf{ss}}$ and a PPT $\mathcal{A}$ such that given an accepting transcript tree $\mathcal{T}$, either $\mathsf{Ext}_{\mathsf{ss}}$ outputs a witness or $\mathcal{A}$ outputs a solution to some hard problem. The proof system's special soundness extractor for the OR relation runs internally both $\mathsf{Ext}_{\mathsf{ss}}$ and $\mathcal{A}$ on $\mathcal{T}$. It returns the witness if $\mathsf{Ext}_{\mathsf{ss}}$ returns the witness and otherwise returns the output of $\mathcal{A}$. Applying now the Fiat-Shamir transform, we obtain a non-interactive knowledge-sound proof system $\mathsf{FS}[\Pi]$ for the OR-relation with the knowledge error $\mathsf{Er}_{\mathrm{fs}}(Q)$ as described in Theorem 10. Furthermore, the obtained $\mathsf{FS}[\Pi]$ is also an argument system for the original relation since we can reduce the security to the underlying assumption with a factor $\mathsf{Er}_{\mathrm{fs}}(Q)$ loss.

E Zero-Knowledge of SanPlonk

We recall the zero-knowledge property of a non-interactive argument.[8] Below $\mathcal{UR}_{\mathsf{p},n}$ is a family of binary relations parameterized by the system parameters p and an integer n. For $\mathcal{R} \in \mathcal{UR}_{\mathsf{p},n}$ and $\mathtt{srs} \in \mathrm{range}(\mathsf{KGen}(\mathsf{p}, n))$.

A non-interactive argument is (statistical) *zero-knowledge* if there exists a PPT simulator Sim, s.t. for all unbound $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, all $\mathsf{p} \in \mathrm{range}(\mathsf{Pgen})$, all $n \in \mathsf{poly}(\lambda)$,

$$\Pr\left[\begin{array}{c} \mathcal{A}_2(st,\pi)=1 \wedge \\ \mathcal{R}(\mathbb{x},\mathbb{w}) \wedge \mathcal{R} \in \mathcal{UR}_{\mathsf{p},n}; \end{array} \middle| \begin{array}{c} (\mathtt{srs},\mathtt{td}_{\mathtt{srs}}) \leftarrow \mathsf{KGen}(\mathsf{p},n); \\ (\mathcal{R},\mathbb{x},\mathbb{w},st) \leftarrow \mathcal{A}_1(\mathtt{srs}); \\ \pi \leftarrow \mathsf{P}(\mathcal{R},\mathtt{srs},\mathbb{x},\mathbb{w}) \end{array} \right] \approx_s$$
$$\Pr\left[\begin{array}{c} \mathcal{A}_2(st,\pi)=1 \wedge \\ \mathcal{R}(\mathbb{x},\mathbb{w}) \wedge \mathcal{R} \in \mathcal{UR}_{\mathsf{p},n} \end{array} \middle| \begin{array}{c} (\mathtt{srs},\mathtt{td}_{\mathtt{srs}}) \leftarrow \mathsf{KGen}(\mathsf{p},n); \\ (\mathcal{R},\mathbb{x},\mathbb{w},st) \leftarrow \mathcal{A}_1(\mathtt{srs}); \\ \pi \leftarrow \mathsf{Sim}(\mathcal{R},\mathtt{srs},\mathtt{td}_{\mathtt{srs}},\mathbb{x}) \end{array} \right].$$

Here, $\approx_s$ denotes the statistical distance as a function of λ. Π is *perfect zero-knowledge* if the above probabilities are equal.

[8] We proved the special soundness for interactive arguments, and since then, we could use Theorem 10 (which has a pretty complicated proof) to obtain knowledge-soundness for non-interactive arguments. For zero-knowledge, it is easy to prove a direct result for non-interactive arguments, so we do that.

Recently, Sefranek [Sef24] showed that an earlier version of Plonk did not satisfy statistical zero-knowledge since the polynomials $\mathsf{t}_{\mathsf{lo}}(X)$, $\mathsf{t}_{\mathsf{mid}}(X)$, $\mathsf{t}_{\mathsf{hi}}(X)$ were not randomized. He corrects this mistake and proves statistical zero-knowledge of the corrected Plonk. The correction is a part of the Plonk now, [GWC19]. Sefranek also mentions that SmallPlonk, which does not split $\mathsf{t}(X)$, is simulatable. Our paper contains the current (corrected) version of Plonk, so we will not repeat the zero-knowledge proofs of Plonk and SmallPlonk and instead refer the reader to [Sef24]. However, we provide a similar proof of zero-knowledge for SanPlonk.

We recall the following well-known lemma; see, e.g., [Sef24].

Lemma 7 ([Sef24]). *Let $f(X) \in \mathbb{F}[X]$ and $x_1, \ldots, x_k$ be distinct values in $\mathbb{F} \setminus \mathbb{H}$. Assume $\tilde{f}(X) = \mathsf{f}(X) + \varrho(X)\mathsf{Z}_{\mathbb{H}}(X)$ for $\varrho(X) \leftarrow_{\$} \mathbb{F}_{\leq k-1}[X]$. Then, $(\tilde{f}(x_1), \ldots, \tilde{f}(x_k))$ is distributed uniformly over $\mathbb{F}^k$,*

The same claim also holds for $\tilde{f}(X) = \mathsf{f}(X) + \varrho(X)$; moreover, then it is true even when $x_1, \ldots, x_k \in \mathbb{F}$, not just in $\mathbb{F} \setminus \mathbb{H}$ (in Lemma 7, we need that $\mathsf{Z}_{\mathbb{H}}(x_i) \neq 0$). We use both results to prove the statistical zero-knowledge of SanPlonk.

Theorem 11. SanPlonk *has statistical zero-knowledge.*

Proof. Consider the following simulation strategy. Recall that SanPlonk's transcript is $([a, b, c]_1; \beta, \gamma; [z]_1; \alpha; [t_{lo}, t_{mid}, t_{hi}]_1; \mathfrak{z}; \bar{a}, \bar{b}, \bar{c}, \bar{s}_{\sigma 1}, \bar{s}_{\sigma 2}, \bar{z}_{\omega}; \delta; \bar{t}_{\mathfrak{z}}; v; [\mathsf{W}_{\mathfrak{z}}, \mathsf{W}_{\mathfrak{z}\omega}]_1)$. Similarly to Plonk's simulator in [Sef24], given the verifier's random challenges, SanPlonk's simulator works like an honest prover with four crucial differences. First, it creates the commitments a, b, c, z and their openings by sampling them randomly. Second, since $\mathsf{t}(X)$, $\mathsf{W}_{\mathfrak{z}}(X)$, $\mathsf{W}_{\mathfrak{z}\omega}(X)$ might not be polynomials, it computes $[\mathsf{t}(x), \mathsf{W}_{\mathfrak{z}}(x), \mathsf{W}_{\mathfrak{z}\omega}(x)]_1$ by using the trapdoor. Third, it computes $[t_{lo}, t_{mid}, t_{hi}]_1$ by sampling two of the values at random and setting the third one so that $[t_{lo}, t_{mid}, t_{hi}]_1$ agrees with the previously computed value of $[\mathsf{t}(x)]_1$. Fourth, the sanitization value $\bar{t}_{\mathfrak{z}}$ is sampled randomly. In Fig. 18, we present the full simulator for SanPlonk as a zk-SNARK, including the definition of the verifier's challenges as the outputs of the random oracle. (For brevity, we refer to Eq. (7) for the formulas of $\mathsf{F}_i(X)$ and $\mathsf{F}(X)$.)

From Lemma 7 it follows that in the honest proof $\mathsf{a}(x)$, $\mathsf{b}(x)$, $\mathsf{c}(x)$, $\mathsf{z}(x)$, $\mathsf{a}(\mathfrak{z})$, $\mathsf{b}(\mathfrak{z})$, $\mathsf{c}(\mathfrak{z})$, $\mathsf{z}(\mathfrak{z}\omega)$, and $\mathsf{z}(x\omega)$ are distributed uniformly randomly and independently. Here, the last element $\bar{z}_{x\omega} = \mathsf{z}(x\omega)$ is not part of the proof transcript, but it is necessary for determining a unique $[\mathsf{t}(x)]_1$. Furthermore, in the honest protocol $\mathsf{t}_{\mathsf{hi}}(X) := \mathsf{t}_{\mathsf{hi}}{}'(X) - b_{11}$, $\mathsf{t}_{\mathsf{mid}}(X) := \mathsf{t}_{\mathsf{mid}}{}'(X) - b_{10} - b_{12}X + b_{11}X^n$, and $\mathsf{t}_{\mathsf{lo}}(X) := \mathsf{t}_{\mathsf{lo}}{}'(X) + b_{10}X^n + b_{12}X^{n+1}$ (as always, we hilight the changes compared to Plonk). Thus, $\mathsf{t}_{\mathsf{hi}}(x)$, $\mathsf{t}_{\mathsf{mid}}(x)$, and $\mathsf{t}_{\mathsf{lo}}(\mathfrak{z})$ are distributed uniformly at random and independently since (resp.) b_{11}, b_{10}, and b_{12} are sampled uniformly at random. In the case of $\mathsf{t}_{\mathsf{lo}}(\mathfrak{z})$, it holds under the assumption that $\mathfrak{z}^{n+1} \neq 0$ (otherwise $b_{12}\mathfrak{z}^{n+1} = 0$). Note that the proof does not reveal $t_{lo}(\mathfrak{z})$. However, $t_{lo}(\mathfrak{z})$ being uniformly random and independent of other elements guarantees that $\bar{t}_{\mathfrak{z}} = \mathsf{t}_{\mathsf{lo}}(\mathfrak{z}) + \delta\mathsf{t}_{\mathsf{mid}}(\mathfrak{z}) + \delta^2\mathsf{t}_{\mathsf{hi}}(\mathfrak{z})$ is uniformly random and independent. In the simulator, we also pick all the mentioned random elements uniformly at random and independently.

1. $a, b, c \leftarrow_\$ \mathbb{F}$.
2. $\beta \leftarrow H(\mathtt{srs}, \mathbb{x}, [a, b, c]_1, 0)$; $\gamma \leftarrow H(\mathtt{srs}, \mathbb{x}, [a, b, c]_1, 1)$.
3. $z \leftarrow_\$ \mathbb{F}$.
4. $\alpha \leftarrow H(\mathtt{srs}, \mathbb{x}, [a, b, c, z]_1)$.
5. Sample $\bar{z}_{x\omega} \leftarrow_\$ \mathbb{F}$ (a candidate value for $\mathsf{z}(x\omega)$ used to compute $\mathsf{F}_1(x)$ in the next step).
6. For $\mathsf{F}_i(X)$ defined as in Eq. (7), set $[\mathsf{t}(x)]_1 \leftarrow \frac{1}{\mathsf{Z}_{\mathbb{H}}(x)}[\mathsf{F}_0(x)+\alpha\mathsf{F}_1(x)+\alpha^2\mathsf{F}_2(x)]_1$.
7. $t_{hi}, t_{mid} \leftarrow_\$ \mathbb{F}$; $[t_{lo}]_1 \leftarrow [\mathsf{t}(x) - t_{mid}x^n - t_{hi}x^{2n}]_1$. Abort if $\mathsf{Z}_{\mathbb{H}}(x) = 0$.
8. $\mathfrak{z} \leftarrow H(\mathtt{srs}, \mathbb{x}, [a, b, c, z, t_{lo}, t_{mid}, t_{hi}]_1)$.
9. Abort if $\mathfrak{z} = x$ or $\mathfrak{z}\omega = x$.
10. $\bar{a}, \bar{b}, \bar{c}, \bar{z}_\omega \leftarrow_\$ \mathbb{F}$; $\bar{s}_{\sigma 1} \leftarrow \mathsf{S}_{\sigma 1}(\mathfrak{z})$; $\bar{s}_{\sigma 2} \leftarrow \mathsf{S}_{\sigma 2}(\mathfrak{z})$.
11. $\delta \leftarrow H(\mathtt{srs}, \mathbb{x}, [a, b, c, z, t_{lo}, t_{mid}, t_{hi}]_1, \bar{a}, \bar{b}, \bar{c}, \bar{s}_{\sigma 1}, \bar{s}_{\sigma 2}, \bar{z}_\omega)$.
12. $\bar{t}_{\mathfrak{z}} \leftarrow_\$ \mathbb{F}$.
13. $v \leftarrow H(\mathtt{srs}, \mathbb{x}, [a, b, c, z, t_{lo}, t_{mid}, t_{hi}]_1, \bar{a}, \bar{b}, \bar{c}, \bar{s}_{\sigma 1}, \bar{s}_{\sigma 2}, \bar{z}_\omega, \bar{t}_{\mathfrak{z}})$.
14. $\mathsf{W} \leftarrow r + v(a - \bar{a}) + v^2(b - \bar{b}) + v^3(c - \bar{c}) + v^4(s_{\sigma 1} - \bar{s}_{\sigma 1}) + v^5(s_{\sigma 2} - \bar{s}_{\sigma 2})$.
15. $[\mathsf{W}_{\mathfrak{z}}]_1 \leftarrow \frac{1}{x-\mathfrak{z}}[\mathsf{W} + v^6(t_{lo} + \delta t_{mid} + \delta^2 t_{hi} - \bar{t}_{\mathfrak{z}})]_1$.
16. $[\mathsf{W}_{\mathfrak{z}\omega}]_1 \leftarrow \frac{1}{x-\mathfrak{z}\omega}[z - \bar{z}_\omega]_1$.
17. Return $([a, b, c]_1; \beta, \gamma; [z]_1; \alpha; [t_{lo}, t_{mid}, t_{hi}]_1; \mathfrak{z}; \bar{a}, \bar{b}, \bar{c}, \bar{s}_{\sigma 1}, \bar{s}_{\sigma 2}, \bar{z}_\omega; \delta; \bar{t}_{\mathfrak{z}}; v; [\mathsf{W}_{\mathfrak{z}}, \mathsf{W}_{\mathfrak{z}\omega}]_1)$.

Fig. 18. SanPlonk's simulator.

The remaining proof elements have only one possible satisfiable value. Namely, there is precisely one possible value of $t_{lo}(x)$ such that $\mathsf{t}_{\mathsf{lo}}(x)+x^n\mathsf{t}_{\mathsf{mid}}(x)+x^{2n}\mathsf{t}_{\mathsf{hi}}(x) = t(x)$ and only one possible value for opening proofs $[\mathsf{W}_{\mathfrak{z}}, \mathsf{W}_{\mathfrak{z}\omega}]_1$ such that the verification equation is satisfied. The simulator computes these elements accordingly. The public polynomials are evaluated honestly as $\bar{s}_{\sigma 1} \leftarrow \mathsf{S}_{\sigma 1}(\mathfrak{z})$ and $\bar{s}_{\sigma 2} \leftarrow \mathsf{S}_{\sigma 2}(\mathfrak{z})$. Also, the challenges $\beta, \gamma, \ldots$ are computed from the correct distribution. The simulator fails when either $x^{n+1} = 0$, $\mathfrak{z} = x$, $\mathfrak{z}\omega = x$, or $\mathsf{Z}_{\mathbb{H}}(x) = 0$ (the last three conditions are needed to avoid division by 0 in the simulator); this happens only with a negligible probability. □