On Knowledge-Soundness of Plonk in ROM from Falsifiable Assumptions

June 20, 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 TriRSDH. We also prove that a minor modification of the interactive Plonk has computational special-soundness under only the ARSDH assumption. 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

Succinct non-interactive arguments of knowledge (SNARKs, [Kil94,Mic94,DL08,Gro10,Lip12,Gro16]) allow us to prove the validity of mathematical statements with a succinct proof. When additionally accompanied by the zero-knowledge property (zk-SNARKs), the proof leaks no information besides the statement's validity. SNARKs and zk-SNARKs have recently found wide-scale adoption in many applications such as scalability [BMRS20] and privacy [BCG$^+$14] of blockchains.

Almost all known zk-SNARKs with constant argument size (with respect to the length of the prover's witness $\mathbb{w}$) use the KZG polynomial commitment scheme [KZG10] (either explicitly or implicitly). This includes Plonk [GWC19], Marlin [CHM$^+$20], and others [CFF$^+$21,RZ21,LSZ22], but also many related schemes like lookup arguments [EFG22,CFF$^+$24]. KZG allows the prover to make a constant-size commitment to a polynomial $\mathsf{f}(X)$ and to open later $\mathsf{f}(\mathfrak{z})$ for some point $\mathfrak{z}$ chosen by the verifier. Crucially, KZG facilitates aggressive optimizations and batching techniques, which makes it ideal for many applications.

In the knowledge-soundness proof of KZG-based zk-SNARKs, it is necessary to extract $\mathsf{f}(X)$ from the commitment. Up until recently, it was only known how to extract using a knowledge assumption [CHM+20] or an idealized group model [FKL18,LPS23], both of which have known undesirable features. Knowledge assumptions are non-falsifiable and not even computational [Nao03]. Pass [Pas16] defines an hierarchy of non-falsifiable intractability assumptions and states that knowledge assumptions are outside this hierarchy. Furthermore, knowledge assumption cannot exist when the adversary receives auxiliary input of the unbounded polynomial size and the indistinguishability obfuscation exists [BCPR14]. For idealized group models such as the generic group model [Sho97,Mau05] and algebraic group model [FKL18], there exist contrived schemes that are secure in the idealized group model, but insecure when instantiated with any group [Den02,Zha22]. In addition, one uses the Fiat-Shamir heuristic [FS87] to make the argument system non-interactive, requiring the random oracle model (ROM). Thus, one relies on two uncomparable idealized models: an idealized group model (or a knowledge assumption) and the ROM.

This situation changed with the recent work of Lipmaa, Parisella, and Siim [LPS24]. First, they proved that the KZG polynomial commitment scheme has computational special-soundness under a falsifiable assumption ARSDH. Their notion of special-soundness closely follows the one for proof systems [CDS94,AFK22]. Suppose an efficient adversary produces $n + 1$ accepting KZG transcripts[3] $([C]_1, \mathfrak{z}_i, \bar{f}_i, [\pi_i]_1)$, where $[C]_1$ is a commitment, $\mathfrak{z}_i$ is an evaluation point, $\bar{f}_i$ is an evaluation, and $[\pi_i]_1$ is a proof. Assume $\mathfrak{z}_i$ are mutually distinct and that the ARSDH assumption (a falsifiable assumption defined in [LPS24], see Definition 1) holds in the respective bilinear group. Lipmaa et al. [LPS24] prove that one can efficiently extract a polynomial $\mathsf{f}(X)$ of degree $\leq n$ that is consistent with the commitment $[C]_1$ and satisfies $\mathsf{f}(\mathfrak{z}_i) = \bar{f}_i$ for all $i = 1, \dots, n+1$. They show that when casting the KZG as an interactive protocol between the committer and the verifier, where the evaluation point $\mathfrak{z}$ is chosen randomly, it is possible to extract $\mathsf{f}(X)$ by rewinding the committer (under the ARSDH assumption). They call this security notion black-box extractability.

Second, they construct a compiler that transforms a polynomial interactive oracle proof (IOP) [BFS20] into a public-coin interactive argument system. Recall that a polynomial IOP is an interactive information-theoretic proof system where the prover sends polynomial oracles to the verifier, who replies with random strings and (at the end) queries the oracles. Their compiler commits to each polynomial oracle, opens the polynomials at points the polynomial IOP verifier requires, and additionally opens each polynomial at a new random point. The latter extra step guarantees, using black-box extractability, that the polynomials can be extracted. By picking an efficient polynomial IOP, such as the one under-

[3] We use bilinear pairings $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are additive groups of prime order p. We denote by $[1]_\iota$ a generator of $\mathbb{G}_\iota$ and $[a]_\iota := a\,[1]_\iota$ for $a \in \mathbb{Z}_p$ and $\iota \in \{1, 2, T\}$. We also follow the notation of Plonk [GWC19].

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]	$23\lvert\mathbb{F}\rvert + 30\lvert\mathbb{G}_1\rvert$ (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)	$7\lvert\mathbb{F}\rvert + 9\lvert\mathbb{G}_1\rvert$ (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)	$6\lvert\mathbb{F}\rvert + 9\lvert\mathbb{G}_1\rvert$ (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, TriRSDH

lying Plonk or Marlin, one can construct a public-coin constant-size interactive argument. Applying the Fiat-Shamir transform, [LPS24] achieves a constant-size SNARK in the ROM under the falsifiable ARSDH assumption.

Unfortunately, after compiling the polynomial IOPs of (say) Plonk or Marlin with [LPS24]'s approach, one obtains notably less efficient succinct interactive arguments than their fully-optimized counterparts in [GWC19,CHM$^+$20]. More precisely, [LPS24]'s compiler introduces the following overheads, making Plonk's argument about 3.5 times longer (see Table 1).[4]

First, each commitment is opened at an additional random point, adding an overhead of one evaluation of the polynomial and one opening proof per each polynomial sent by the prover in the polynomial IOP. For example, in Plonk's polynomial IOP, the prover sends 7 polynomials, thus inducing an overhead of 7 field elements and 7 group elements in the compiled argument.

Second, [LPS24] proves security only when each polynomial is opened separately; that is, the openings are not batched. Optimized argument constructions, such as Plonk, aggressively batch the opening proofs. For instance, Plonk's prover sends only 2 opening proofs, while the prover resulting from [LPS24] sends an opening proof for each polynomial evaluation in the underlying IOP, on top of the openings for the extra evaluations introduced by the compiler.

[LPS23] observed that the linearization trick (a well-known batching technique, where the prover opens a linearized polynomial instead of the original polynomial, see Section 3.2), used in Plonk and Marlin, is not secure in the AGMOS (Algebraic Group Model with Oblivious Sampling), a more realistic variant of the algebraic group model (AGM, [FKL18]). This follows from the ability of AGMOS (and real-world) adversaries to sample group elements without knowing their discrete logarithm. As noted in [LPS23], Lemma 3.3 from Section 3.1 in [GWC19], used in the soundness proof of Plonk, suffers from the same issue. It

[4] Over time, Plonk's description in [GWC19] has changed due to optimizations and bug fixes. In the current paper, we refer to the variant of Plonk available at the moment of writing (namely, the update of [GWC19] from 2024-02-23).

holds in AGM but not in the AGMOS, thus necessitating a new security proof for Plonk. [LPS23,LPS24] left open the problem of defining a different batching technique that would result in a secure variant of the linearization trick. Moreover, the linearization trick is insecure; it is unclear if this affects the security of real-life zk-SNARKs. Thus, it is unknown if (say) Plonk is secure in the ROM under falsifiable assumptions.

Finally, in most applications, one casts the public-coin interactive argument into a non-interactive one using the Fiat-Shamir heuristic. The resulting non-interactive argument is secure in the ROM under the same security assumptions as the initial interactive argument.

One can apply the Fiat-Shamir transform to a knowledge-sound interactive argument, as in, say, [GWC19,CHM$^+$20,RZ21,CFF$^+$21,LSZ22,LPS24], or to an interactive argument that satisfies a more stringent property like special-soundness. Suppose a $(2\mu + 1)$-move knowledge-sound, but not special-sound, interactive proof has knowledge error ϵ. The Fiat-Shamir transformed argument admits a probability of cheating of at most $(Q+1)^\mu \cdot \epsilon$. Attema et al. [AFK22] provided examples showing that this bound is nearly optimal. Attema et al. also showed that special-soundness results in a significantly smaller security loss: in the case of special-sound proofs, the knowledge error is bounded by $(Q+1)\epsilon$. As explained [AFK22], their result also applies to special-sound argument systems since one can view them as proof systems for an OR language where the extractor either outputs a witness or a solution to a computational assumption. Thus, computational special-soundness is the preferred notion for public-coin interactive argument systems.

The interactive variants of Plonk [GWC19] and other KZG-based zk-SNARKs like [CHM$^+$20,RZ21,CFF$^+$21,LSZ22] were proven to be knowledge-sound in the AGM and in the AGMOS [FFR24], but it is not known if they are computationally special-sound in the standard-model under falsifiable assumptions. Of such zk-SNARKs, Plonk is most widely used due to its ("Plonkish") arithmetization that allows to compactify arithmetic circuits and thus significantly decreases the prover's computation. In addition, Plonk is updatable and has a small proof and efficient verification. Thus, in the current paper, we concentrate on Plonk.

Our Results. We improve on [LPS24] by proving that various KZG-based batch-opening protocols have computational special-soundness, assuming that KZG satisfies evaluation-binding[5] and computational special-soundness. As shown in [LPS24], KZG satisfies the latter properties under the falsifiable ARSDH assumption. Importantly, we prove that interactive Plonk has computational special-soundness under falsifiable assumptions. On top of KZG's evaluation-binding and computational special-soundness, Plonk relies on a new falsifiable assumption, n-TriRSDH. We prove a slightly modified variant of Plonk is secure without relying on TriRSDH (see Table 1). The results

[5] Recall that evaluation-binding means that it is computationally hard to find two opening proofs and two different evaluations $\bar{f}_1, \bar{f}_2$ such that they will verify for the same commitment $[C]_1$ and evaluation point $\mathfrak{z}$.

of [AFK22,DG23,AFKR23] imply that (non-interactive) Plonk is knowledge-sound, after applying the Fiat-Shamir transform, under the same falsifiable assumptions. Moreover, applying the Fiat-Shamir transform is (optimally) tight. This results in the first tight security proof of a constant-length zk-SNARK in the ROM under falsifiable assumptions.

The specific protocols we focus on already open each commitment at a random point. It allows us to avoid the inefficiency from [LPS24], where the PIOP compiler added an additional random opening to each commitment.

Special-Sound Subroutines. We recall the notion of computational $\boldsymbol{\kappa}$-special-soundness for argument systems, where $\boldsymbol{\kappa} = (\kappa_1, \dots, \kappa_\mu)$. A $\boldsymbol{\kappa}$-tree of transcripts contains the prover's messages in the nodes and the verifier's challenges on the edges. Each node at depth $i-1$ has exactly κ_i edges with distinct challenges, and every path from the root to a leaf defines one accepting transcript. An argument satisfies computational $\boldsymbol{\kappa}$-special-soundness when there exists an efficient extractor that can extract a witness from a $\boldsymbol{\kappa}$-tree of accepting transcripts produced by an efficient adversary (except with negligible probability).

We show that the following three multi-round interactive arguments are computationally $\boldsymbol{\kappa}$-special-sound for some vector $\boldsymbol{\kappa}$, assuming that KZG is computationally special-sound and evaluation-binding.

<u>First, Batch:</u> A standard 5-round interactive argument for batch-proving that m KZG commitments open to some values at a joint (randomly sampled) opening point $\mathfrak{z}$. Batch opens a linear combination of the KZG commitments. We construct a tight security reduction (independent of m) to the computational special-soundness of the KZG. Batch is an important example since it is used in most KZG-based zk-SNARKs and one of the sources of inefficiency in [LPS24] comes from not handling batching. It also showcases our proof techniques.

<u>Second, SanLin:</u> A 5-round interactive argument that employs the standard linearization trick to prove that three KZG-committed polynomials $\mathsf{a}(X)$, $\mathsf{b}(X)$, and $\mathsf{c}(X)$ satisfy $\mathsf{a}(X)\mathsf{b}(X) = \mathsf{c}(X)$. Lipmaa et al. [LPS23] showed that the basic variant Lin of this argument (introduced in [CHM$^+$20]) is insecure in the AGMOS, and thus insecure in the standard model. Slightly more involved variants of Lin are used in well-known zk-SNARKs such as Marlin, Plonk, and Lunar. In Lin, the prover opens polynomials $\mathsf{a}(X)$ and $\Lambda(X) := \mathsf{a}(\mathfrak{z})\mathsf{b}(X) - \mathsf{c}(X)$ at $\mathfrak{z}$ to $\mathsf{a}(\mathfrak{z})$ and 0. We show that Lin is not special-sound since the commitment to $\mathsf{b}(X)$ can be obliviously sampled (without knowing its discrete logarithm). Intuitively, the variant $\Lambda_i(X)$ of $\Lambda(X)$ in ith branch of the transcript tree depends on the ith value $\mathfrak{z}_i$ of $\mathfrak{z}$. Thus, in the special-soundness proof, one has many different transcripts corresponding to the openings of different polynomial commitments $[\Lambda_i]_1$, each at a single point $\mathfrak{z}_i$. While one can use KZG's special-soundness to open the polynomial commitment $[a]_1$ (which stays the same in all transcripts), corresponding to $\mathsf{a}(X)$, one cannot open $[\Lambda_i]_1$ due to our attack.

To correct this, we add a *sanitization* step, opening $\mathsf{b}(X)$ at a random point. After that, one can use KZG's special-soundness to extract $\mathsf{b}(X)$ and commence with the special-soundness proof. Compared to the insecure non-sanitized version, the sanitized variant SanLin has communication only bigger by one field element. We prove SanLin's security under computational special-soundness and evaluation-binding of KZG. While SanLin is only minimally more efficient than known alternatives, it is a simple example of the novel sanitization technique (we apply similar sanitization in other protocols). Moreover, it can be used in any zk-SNARKs that use the linearization trick, thus resulting in more efficient, secure variants under falsifiable assumptions.

Third, $\mathsf{SanLinGen}$: A 7-round interactive argument that employs the linearization trick to prove that KZG-committed polynomials $\mathsf{a}_i(X)$ and $\mathsf{b}_i(X)$, $i \in [1, m]$, satisfy $\sum_{i=1}^{m} \mathsf{a}_i(X)\mathsf{b}_i(X) = 0$. We show that this argument's natural variant LinGen is not computationally special-sound in the standard model. To correct this, we add a sanitization step that shows that the prover knows how to open a random linear combination of $\mathsf{b}_i(X)$. Sanitization increases communication by one field element, compared to the insecure non-sanitized version, resulting in significant savings compared to known secure alternatives (see Section 3.3). We give a tight reduction (again, independent of m) to computational special-soundness and evaluation-binding of the KZG. $\mathsf{SanLinGen}$ is an example of how sanitization introduces minimal overhead in real-world applications.

Special-Soundness of Plonk. We prove that Plonk [GWC19] has computational $\boldsymbol{\kappa}$-special-soundness under falsifiable assumptions. To maximize efficiency, Plonk relies heavily on the KZG polynomial commitment scheme's batching capabilities, including a variant of the linearization trick. Due to the latter, it is not obvious that Plonk is special-sound. We use the tools developed for Batch and $\mathsf{SanLinGen}$ to extract polynomials corresponding to most of KZG commitments made by Plonk's prover. We then use the extracted polynomials to prove that Plonk has computational special-soundness under falsifiable assumptions.

However, we face a significant obstacle compared to the AGM knowledge-soundness proof in [GWC19]. Namely, an AGM extractor can extract the polynomials corresponding to all KZG commitments made by Plonk's prover, but there are three polynomials that our standard-model special-soundness extractor cannot extract. For those familiar with Plonk, the corresponding polynomial commitments are $[t_{lo}, t_{mid}, t_{hi}]_1$; they are essentially a trifurcation of a single polynomial commitment $[t]_1$. Trifurcation results in an optimization: without trifurcation, Plonk's SRS would be three times longer, resulting in higher prover costs. The inability to extract comes from the fact that one of the polynomials that are opened during a Plonk run ($\mathsf{r}(X)$, for those familiar with Plonk) depends on the evaluation point. That is, Plonk uses the linearization trick. When we try to prove special-soundness, we encounter the same problem as in unsanitized Lin.

We propose two different solutions to this challenge. We compare different versions of Plonk in Table 1.

First, (interactive) sanitized Plonk. Using the sanitization techniques developed for SanLin and $\mathsf{SanLinGen}$, we propose $\mathsf{SanPlonk}$, which differs from Plonk by batch opening the commitments $[t_{lo}]_1$, $[t_{mid}]_1$, and $[t_{hi}]_1$. The verifier sends an additional random challenge δ and the prover reveals $\bar{t}_{\mathfrak{z}} = \mathsf{t}_{\mathsf{lo}}(\mathfrak{z}) + \delta \mathsf{t}_{\mathsf{mid}}(\mathfrak{z}) + \delta^2 \mathsf{t}_{\mathsf{hi}}(\mathfrak{z})$, where $\mathfrak{z}$ is an evaluation point used in Plonk and (say) $\mathsf{t}_{\mathsf{lo}}(X)$ is a polynomial that was committed to in $[t_{lo}]_1$. We show this modification is sufficient to extract $\mathsf{t}_{\mathsf{lo}}(X)$, $\mathsf{t}_{\mathsf{mid}}(X)$, and $\mathsf{t}_{\mathsf{hi}}(X)$. Thus, we can extract all polynomials that are extracted in a typical AGM proof of Plonk and can commence with a special-soundness proof, assuming that KZG is evaluation-binding and has special-soundness. Sanitization adds minimal overhead in real protocols. After applying Fiat-Shamir, $\mathsf{SanPlonk}$'s argument has only one more field element compared to Plonk. See Table 1 for a comparison.

Second, (interactive) Plonk under a novel assumption. Adding an extra element to the argument of Plonk is non-ideal; in particular, it would break many existing implementations. We showed that sanitization is necessary for Lin and LinGen by demonstrating an attack against their non-sanitized variants. It is natural to expect that Plonk as a much more complicated protocol cannot have special-soundness either. Perhaps surprisingly, this is not the case. We prove that Plonk has computationally special-soundness under an additional falsifiable assumption n-TriRSDH (*Trifurcation Rational SDH*, where trifurcation refers to the division of a particular polynomial commitment to $[t_{lo}, t_{mid}, t_{hi}]_1$).

More precisely, instead of opening three polynomial commitments $[t_{lo}, t_{mid}, t_{hi}]_1$, we define a rational function $\mathsf{t}(X)$ that we *can* open. We show that if $\mathsf{t}(X)$ is a polynomial, then the prover was honest and thus, Plonk has computationally special-soundness. On the other hand, if $\mathsf{t}(X)$ is not a polynomial, then we construct a reduction to the TriRSDH assumption.

Now, n-TriRSDH is a novel assumption, essentially stating that Plonk's optimization trick (trifurcation) retains the security. It states that it should be difficult to output a tuple of evaluation points $\mathfrak{z}_i$ together with opening proofs $[\chi_i]_1$, group elements $[t_{lo}, t_{mid}, t_{hi}]_1$, and a rational, non-polynomial function $\mathsf{t}(X)$, such that for every i, $[\chi_i]_1$ is an accepting proof that $[t_{lo} + \mathfrak{z}_i^n t_{mid} + \mathfrak{z}_i^{2n} t_{lo}]_1$ opens to $\mathsf{t}(\mathfrak{z}_i)$ at $\mathfrak{z}_i$. Here, n is the length of the SRS while the number of evaluation points and the numerator and denominator of $\mathsf{t}(X)$ must satisfy certain additional conditions, see Definition 2. While TriRSDH is non-standard, it is falsifiable. Moreover, we prove that TriRSDH is secure in the AGMOS [LPS23], which is probably the most realistic variant of AGM. It also minimally depends on the structure of Plonk, only ascertaining that it is "ok" to do the trifurcation. TriRSDH is definitely not a tautological assumption for Plonk: in fact, the reduction to TriRSDH is the most involved reduction in the current paper.

The combined analysis of Plonk and $\mathsf{SanPlonk}$ is involved, taking about ten pages of the current paper. (If $\mathsf{SanPlonk}$ were omitted, the analysis would shorten slightly.) Part of the reason comes from the fact that we formalized every step of the proof. Given Plonk's importance in practice, giving independent (and different, since we prove special-soundness) and more thorough proofs is crucial.

Fiat-Shamir. We proved that *interactive* Plonk and $\mathsf{SanPlonk}$ have computational $\boldsymbol{\kappa}$-special-soundness (for a slightly different $\boldsymbol{\kappa}$) under falsifiable assumptions. One can apply the Fiat-Shamir transformation to obtain a zk-SNARK that is knowledge-sound in the ROM under the same assumptions. The tightness 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 B.5 for a brief discussion.

Zero-Knowledge. Up to now, we ignored the issue of zero knowledge. Since sanitization means 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 standard (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 D, we then prove that SanPlonk has zero knowledge.

2 Preliminaries

Let λ denote the security parameter. $f(\lambda) \approx_\lambda 0$ means 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 $\mathbb{F}_{\leq n}[X] \subset \mathbb{F}[X]$ is the set of polynomials of at most degree n. We denote $[a, b] := \{a, a+1, \ldots, b\}$, where $a \leq b$ are integers. Our notation is inspired by Plonk [GWC19] (for example, we denote polynomials by using SansSerif), but we do not follow it universally.

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 standard 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 [LPS24].

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

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

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

2.1 Polynomial Commitment Schemes

In a (univariate) polynomial commitment scheme (PCS, [KZG10]), the prover commits to a polynomial $\mathsf{f} \in \mathbb{F}_{\le n}[X]$ 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 $\mathsf{f} \in \mathbb{F}_{\le n}[X]$, 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 $\mathsf{f} \in \mathbb{F}_{\le n}[X]$, return $(\bar{f}, \pi)$, where $\bar{f} \leftarrow \mathsf{f}(\mathfrak{z})$ and π is an evaluation 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 $\bar{f} =^? \mathsf{f}(\mathfrak{z})$, and an evaluation 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, $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, $n \in \mathsf{poly}(\lambda)$, $\mathfrak{z} \in \mathbb{F}$, $\mathsf{f} \in \mathbb{F}_{\le n}[X]$,

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

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

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

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 $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{evb}}_{\mathsf{Pgen},\mathsf{PC},n,\mathcal{A}}(\lambda) :=$

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

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 $f(X) \ne f'(X)$. Then, we can find a point $\mathfrak{z}$, such that $f(\mathfrak{z}) \ne f'(\mathfrak{z})$, 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 $\mathbf{tr} = (\mathsf{tr}_0, \dots, \mathsf{tr}_n)$ 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 $\mathcal{A}_{\mathsf{ss}}$, $\mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen}, \mathsf{PC}, \mathsf{Ext}_{\mathsf{ss}}, n, \mathcal{A}_{\mathsf{ss}}}(\lambda) :=$

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

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 $(\sum s_j C_j, \mathfrak{z}, \sum s_j \bar{f}_j, \sum s_j \pi_j)$ 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 denote the set of valid witnesses for $(\mathtt{srs}, \mathbb{x})$ by $\mathcal{R}(\mathtt{srs}, \mathbb{x}) = \{\mathbb{w} : (\mathtt{srs}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$. A statement that has a witness is said to be true. We denote the set of true statements by $\mathcal{L}_{\mathcal{R}} = \{\mathbb{x} : \exists \mathbb{w}, \mathtt{srs}$ s.t. $(\mathtt{srs}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$. The relation $\mathcal{R}$ is an NP-*relation* if the validity of a witness $\mathbb{w}$ can be verified in time polynomial in $|\mathbb{x}| + |\mathtt{srs}|$. From now on, we assume all relations to be NP-relations. 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 relation $\mathcal{R}$ is an interactive protocol between two probabilistic machines, a prover P, and a polynomial time 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 $\mathbb{w} \in \mathcal{R}(\mathtt{srs}, \mathbb{x})$. The verifier V either accepts or rejects. Accordingly, we say the transcript (all messages exchanged in the protocol execution) is accepting or rejecting.

Let $\boldsymbol{\kappa} = (\kappa_1, \dots, \kappa_\mu) \in \mathbb{N}^\mu$. 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.

Let $\boldsymbol{\kappa} = (\kappa_1, \dots, \kappa_\mu), \boldsymbol{N} = (N_1, \dots, N_\mu) \in \mathbb{N}^\mu$. A $(2\mu + 1)$-move public-coin interactive argument $\Pi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ for relation $\mathcal{R}$, where V samples the ith challenge from a set of cardinality $N_i \geq \kappa_i$ for $1 \leq i \leq \mu$, is $\boldsymbol{\kappa}$*-out-of-*$\boldsymbol{N}$ *special-sound* if there exists a DPT extractor $\mathsf{Ext}_{\mathsf{ss}}$ such that for any PPT $\mathcal{A}_{\mathsf{ss}}$, $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{Pgen}, \Pi, \mathsf{Ext}_{\mathsf{ss}}, \boldsymbol{\kappa}, \mathcal{A}}(\lambda) :=$

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

In most of the current paper, $N_1 = \cdots = N_\mu = |\mathbb{F}|$. Then, we say for simplicity that Π has computational $\boldsymbol{\kappa}$-special-soundness.

3 Special Soundness of KZG Batching Protocols

It is a standard practice to prove knowledge-soundness of KZG-based interactive arguments in idealized group models (AGM, AGMOS). Stretching the techniques of [LPS24], we prove the computational special-soundness of such arguments under falsifiable assumptions. The earlier (easier) proofs demonstrate our proof techniques. We show that several known arguments are not special-sound; then, we apply a novel technique of sanitization to obtain special-soundness.

3.1 Special-Soundness of Batch-KZG

Assume the usual setting of KZG with trapdoor x and degree bound n. Consider the standard interactive protocol Batch from Fig. 1, where the prover has committed to m polynomials and then engages in a single batch proof

```
KGen(p): $x \leftarrow_\$ \mathbb{F}^*$; return $\mathtt{srs} \leftarrow ([(x^s)_{s=0}^n]_1, [1, x]_2)$ and $\mathtt{td}_{\mathtt{srs}} \leftarrow x$;
P($\mathtt{srs}, \mathbb{w} = (\mathsf{f}_s(X))_{s=1}^m$): for $s \in [1, m]$, $[f_s]_1 \leftarrow [\mathsf{f}_s(x)]_1$; return $[(f_s)_{s=1}^m]_1$;
V: return $\mathfrak{z} \leftarrow_\$ \mathbb{F}$;    // Evaluation point
P: for $s \in [1, m]$, $\bar{f}_s \leftarrow \mathsf{f}_s(\mathfrak{z})$; return $(\bar{f}_s)_{s=1}^m$;
V: return $v \leftarrow_\$ \mathbb{F}$;    // Batch coefficient
P: $h(X) \leftarrow \left(\sum_{s=1}^m v^{s-1}(\mathsf{f}_s(X) - \bar{f}_s)\right) / (X - \mathfrak{z})$; return $[h]_1 \leftarrow [h(X)]_1$;
V: check $[\sum_{s=1}^m v^{s-1}(f_s - \bar{f}_s)]_1 \bullet [1]_2 = [h]_1 \bullet [x - \mathfrak{z}]_2$;
```

Fig. 1. The protocol Batch.

```
$\mathsf{TE}_{\mathsf{batch}}(\mathsf{ck}, T)$
Parse $T = (\mathsf{tr}_{ij})_{i \in [1,n+1], j \in [1,m]}$;    // $\mathsf{tr}_{ij} = ([(f_s)_{s=1}^m]_1, \mathfrak{z}_i, (\bar{f}_{si})_{s=1}^m, v_{ij}, [h_{ij}]_1)$
$[h'_{i1}, \dots, h'_{im}]_1^\intercal \leftarrow \boldsymbol{V}_i^{-1} [h_{i1}, \dots, h_{im}]_1^\intercal$;  $(*)$    // See $\boldsymbol{V}_i$ in Eq. (2)
for $j \in [1, m]$ do
    for $i \in [1, n+1]$ do $\mathsf{k.tr}'_{ij} \leftarrow ([f_j]_1, \mathfrak{z}_i, \bar{f}_{ji}, [h'_{ij}]_1)$; endfor $(**)$
    $\mathbf{k.tr}'_j \leftarrow (\mathsf{k.tr}'_{1j}, \dots, \mathsf{k.tr}'_{n+1,j})$; endfor
return $(\mathbf{k.tr}'_j)_{j=1}^m$;
```

Fig. 2. The subroutine $\mathsf{TE}_{\mathsf{batch}}$.

that opens all m polynomials simultaneously at the same point $\mathfrak{z}$. Here, $[f_s]_1$ are commitments to some polynomials, $\mathfrak{z}$ is the common evaluation point, $\bar{f}_s$ are purported evaluations of $[f_s]_1$ at $\mathfrak{z}$, and $[h]_1$ is the batched opening of all m commitments. Note that Batch's verifier essentially checks that $\mathsf{k.tr} = (\sum_{s=1}^m v^{s-1}[f_s]_1, \mathfrak{z}, \sum_{s=1}^m v^{s-1}\bar{f}_s, [h]_1)$ is an accepting KZG transcript.

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 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 $T = (\mathsf{tr}_{ij})$ be an $(n+1, m)$-tree of Batch's accepting transcripts, where tr_{ij} are as in Fig. 2. The DPT algorithm $\mathsf{TE}_{\mathsf{batch}}(\mathsf{ck}, T)$ in Fig. 2 computes a tuple of accepting KZG transcripts $(\mathbf{k.tr}'_j)_{j=1}^m$, such that $\mathsf{k.tr}'_{ij} = ([f_j]_1, \mathfrak{z}_i, \dots)$, with mutually different $\mathfrak{z}_i$ for $i \in [1, n+1]$.*

Proof. Let T be the given accepting tree of transcripts and

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

be a Vandermonde matrix. Given 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 $i \in [1, n+1]$.

```
Ext_ss^batch(ck, T)
    $(\mathbf{k.tr}'_s)_{s=1}^m \leftarrow \mathsf{TE}_{\mathsf{batch}}(\mathsf{ck}, T)$;
    for $s \in [1, m]$ do
        $\mathsf{f}^*_s(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}'_s)$; endfor
    return $(\mathsf{f}^*_s(X))_{s=1}^m$;

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

Fig. 3. 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.

Define $[h'_{i1}, \dots, h'_{im}]_1^\intercal$ as in (*) in Fig. 2. As noted above, by the definition of Batch (see Fig. 1), 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 $\varphi_{ij} := \sum_{s=1}^m v_{ij}^{s-1} f_s$, $\Phi_{ij} := \sum_{s=1}^m v_{ij}^{s-1} \bar{f}_{si}$, and $h_{ij} = \sum_{s=1}^m v_{ij}^{s-1} h'_{is}$. 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 $\mathsf{k.tr}'_{ij}$ (see (**) in Fig. 2) is an accepting KZG transcript. Thus, $\mathsf{TE}_{\mathsf{batch}}$ returns accepting KZG transcripts $\mathbf{k.tr}'_j = (\mathsf{k.tr}'_{1j}, \dots, \mathsf{k.tr}'_{n+1,j})$ for $j \in [1, m]$, of the claimed form. □

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. 3, 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 $T = (\mathsf{tr}_{ij})_{i \in [1, n+1], j \in [1, m]}$ of accepting Batch transcripts, where tr_{ij} is defined as in Fig. 2. $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}$ calls the (deterministic) algorithm $\mathsf{TE}_{\mathsf{batch}}$ (see Fig. 2) to compute $n+1$ valid transcripts $\mathsf{k.tr}'_{ij} = ([f_j]_1, \mathfrak{z}_i, \dots)$ 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 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 s, $([f_s]_1, \mathsf{f}^*_s(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{k.tr}'_s}$. 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, $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{Pgen}, \mathsf{KZG}, \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}, n+1, \mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}}(\lambda) = \Pr[\mathsf{bad}] = \mathsf{Adv}^{\mathsf{ss}}_{\mathsf{Pgen}, \mathsf{Batch}, \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{batch}}, n+1, m, \mathcal{A}_{\mathsf{ss}}^{\mathsf{batch}}}(\lambda)$. This concludes the proof. □

Here, as in the rest of the paper, we exploit that the special-soundness extractor for KZG, defined in [LPS24], is deterministic. Thus, the adversary $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ does not have to guess for which s the event bad happened. As a result, all our reductions to KZG's $(n+1)$-special-soundness are tight, with a loss independent from the number m of polynomials whose openings are batched together.

3.2 Lin: Common Linearization Trick

In many zk-SNARKs, one needs to test quadratic equations of type $\mathsf{f}(X) := \mathsf{a}(X)\mathsf{b}(X) - \mathsf{c}(X) = 0$, where $\mathsf{a}(X)$, $\mathsf{b}(X)$, and $\mathsf{c}(X)$ are committed polynomials of low degree. A straightforward way of testing this is by opening $\mathsf{a}(X)$, $\mathsf{b}(X)$, and $\mathsf{c}(X)$ at a random point $\mathfrak{z}$ and checking that $\mathsf{f}(\mathfrak{z}) = 0$. Clearly, $\mathfrak{z}$ is a root of non-zero f with probability at most $\deg(\mathsf{f})/|\mathbb{F}|$. Since f is also a low-degree polynomial, $\mathsf{f}(X) \equiv 0$ with an overwhelming probability when $\deg(\mathsf{f}) \ll |\mathbb{F}|$. When using the KZG commitment scheme, the prover in the straightforward protocol has to send three field and three group elements. One can use Batch to batch the openings, resulting in three field elements and one group element.

Alternatively, one can batch-open two polynomials, $\mathsf{a}(X)$ and $\Lambda(X) := \mathsf{a}(\mathfrak{z})\mathsf{b}(X) - \mathsf{c}(X)$ to $\mathsf{a}(\mathfrak{z})$ and 0 at a random point $\mathfrak{z}$. The resulting protocol Lin (see Fig. 4) is sometimes known as the *linearization trick*. A variant of Lin was first used in [CHM$^+$20]; variants of Lin occurs in almost all modern KZG-based zk-SNARKs. Notably, in Lin, the opening consists of a single field element and (after using batching) a single group element. See Table 3 for comparison.

Lin is well-known to be knowledge-sound in the AGM. However, as shown in [LPS23], Lin is not knowledge-sound in the plain model.[6] Crucially, the adversary can use oblivious sampling, that is, creating a group element without knowing its discrete logarithm. We give a variation of their attack in Appendix A.2.

Since Lin is widely used, making it special-sound in plain model with the smallest possible overhead is an important independent question. While the win cannot be large (the difference of Lin and batch-opening is only two field elements, see Table 3), any gain of efficiency is important. Moreover, Lin is a good toy example of our new proof technique that we will use in subsequent protocols.

We modify Lin to become special-sound in the plain model. The resulting protocol SanLin (see Fig. 4) adds *sanitization*, asking the prover to batch-open the polynomial $\mathsf{b}(X)$ together with $\mathsf{a}(X)$ and $\Lambda(X)$. The latter convinces the verifier that the prover can open $\mathsf{a}(X)$ and $\mathsf{b}(X)$. Since $\mathsf{c}(X) = \mathsf{a}(X)\mathsf{b}(X)$, the prover also knows how to open $\mathsf{c}(X)$. Sanitization increases the communication by just one field element, resulting in a smaller cost than batch-opening $[a, b, c]_1$ (see Table 3). Clearly, the SanLin verifier checks that $([a + v(\bar{a}b - c) + v^2 b]_1, \mathfrak{z}, \bar{a} + v^2\bar{b}, [h]_1)$ is an accepting KZG transcript.

Theorem 3. *Let $n \in \mathsf{poly}(\lambda)$ and $\boldsymbol{\kappa} = (2n + 1, 3)$. If KZG for degree $\leq n$ polynomials has computational $(n+1)$-special-soundness and evaluation-binding, then SanLin has computational $\boldsymbol{\kappa}$-special-soundness.*

[6] As [LPS23] pointed out, even though there is an attack against Lin, the zk-SNARKs which use Lin (or some variation of it) may still be secure.

```
KGen: $x \leftarrow_{\$} \mathbb{F}^*$; return $\mathtt{srs} \leftarrow ([(x^s)_{s=0}^n]_1, [1, x]_2)$ and $\mathtt{td}_{\mathtt{srs}} \leftarrow x$;
P($\mathtt{srs}, \mathbb{w} = (\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X))$): $[a, b, c]_1 \leftarrow [\mathsf{a}(x), \mathsf{b}(x), \mathsf{c}(x)]_1$; return $[a, b, c]_1$;
V: return $\mathfrak{z} \leftarrow_{\$} \mathbb{F}$;
P: $\bar{a} \leftarrow \mathsf{a}(\mathfrak{z})$; $\bar{b} \leftarrow \mathsf{b}(\mathfrak{z})$; $\Lambda(X) \leftarrow \bar{a}\mathsf{b}(X) - \mathsf{c}(X)$; return $\bar{a}, \bar{b}$;
V: return $v \leftarrow_{\$} \mathbb{F}$;
P: $[h]_1 \leftarrow [(\mathsf{a}(x) + v\Lambda(x) + v^2\mathsf{b}(X) - (\bar{a} + v^2\bar{b}))/(x - \mathfrak{z})]_1$; return $[h]_1$;
V: $[\Lambda]_1 \leftarrow [\bar{a}b - c]_1$; check $[a + v\Lambda + v^2 b - (\bar{a} + v^2\bar{b})]_1 \bullet [1]_2 = [h]_1 \bullet [x - \mathfrak{z}]_2$;
```

Fig. 4. Lin (without highlighted parts) and SanLin (with highlighted parts).

```
KGen: $x \leftarrow_{\$} \mathbb{F}^*$; return $\mathtt{srs} \leftarrow ([(x^s)_{s=0}^n]_1, [1, x]_2)$ and $\mathtt{td}_{\mathtt{srs}} \leftarrow x$;
P($\mathtt{srs}, \mathbb{w} = (\mathsf{a}_s(X), \mathsf{b}_s(X))_{s=1}^m$):
    for $s \in [1, m]$: $[a_s, b_s]_1 \leftarrow [\mathsf{a}_s(x), \mathsf{b}_s(x)]_1$; return $[(a_s, b_s)_{s=1}^m]_1$;
V: return $\mathfrak{z} \leftarrow_{\$} \mathbb{F}$;
P: for $s \in [1, m]$: $\bar{a}_s \leftarrow \mathsf{a}_s(\mathfrak{z})$; return $(\bar{a}_s)_{s=1}^m$;
V: return $\gamma \leftarrow_{\$} \mathbb{F}$;
P: return $\bar{b} \leftarrow \sum_{s=1}^m \gamma^{s-1}\mathsf{b}_s(\mathfrak{z})$;
V: return $\beta \leftarrow_{\$} \mathbb{F}$;
P: $\mathsf{H}(X) \leftarrow \sum_{s=1}^m \beta^{s-1}(\mathsf{a}_s(X) - \bar{a}_s) + \beta^m \cdot \sum_{s=1}^m \bar{a}_s\mathsf{b}_s(X) + \beta^{m+1} \cdot \left(\sum_{s=1}^m \gamma^{s-1}\mathsf{b}_s(X) - \bar{b}\right)$;
    $h(X) \leftarrow \mathsf{H}(X)/(X - \mathfrak{z})$; return $[h]_1 \leftarrow [h(x)]_1$;
V: check
    $\left[\sum_{s=1}^m \beta^{s-1}(a_s - \bar{a}_s) + \beta^m \sum_{s=1}^m \bar{a}_s b_s + \beta^{m+1}\left(\sum_{s=1}^m \gamma^{s-1} b_s - \bar{b}\right)\right]_1 \bullet [1]_2 = [h]_1 \bullet [x - \mathfrak{z}]_2$;
```

Fig. 5. LinGen (without highlighted parts) and SanLinGen (with highlighted parts).

Compared to Theorem 2, Theorem 3 additionally relies on KZG's evaluation-binding. We need evaluation-binding to argue that openings of different polynomial commitments are consistent with each other. Note that KZG is evaluation-binding under the SDH assumption which follows from the ARSDH assumption. We postpone the proof to Appendix A.4.

3.3 SanLinGen: Generalized SanLin

Consider LinGen, a natural generalization of Lin, where the prover aims to show that $\sum_{s=1}^m \mathsf{a}_s(X)\mathsf{b}_s(X) = 0$, where $\mathsf{a}_s(X)$ and $\mathsf{b}_s(X)$ are $2m$ committed polynomials. Similarly to Lin, one can use a linearization trick to obtain a simple protocol LinGen for this task (see Fig. 5). However, since LinGen is a generalization of Lin, unsurprisingly LinGen is not extractable in the AGMOS or the standard model. For sake of compleness, we present an attack in Appendix A.3.

As with Lin, we overcome this issue by employing sanitization, which here means a batched opening of all polynomials $\mathsf{b}_s(X)$ at a random point. Crucially, we are not interested in the actual evaluations $\mathsf{b}_s(\mathfrak{z})$. Thus, it suffices for the prover to send $\bar{b} \leftarrow \sum_{s=1}^m \gamma^{s-1}\mathsf{b}_s(\mathfrak{z})$, for a batching coefficient γ, adding a single

Table 2. Comparison of different protocols for the relation $\mathcal{R}_{\mathsf{ck}}^{\mathsf{LinGen}}$. KS stands for the knowledge-soundness and SS for the special-soundness. The number of bits are given for the BLS381-12 pairing.

Method	$\lvert\pi\rvert$ (bits)	KS in AGM	KS and SS in plain-model
Opening $\mathsf{a}_s, \mathsf{b}_s$ separately	$2m\lvert\mathbb{F}\rvert + 2m\lvert\mathbb{G}_s\rvert$ $(1280m)$	✓	✓
Batch-opening a_s, b_s	$2m\lvert\mathbb{F}\rvert + \lvert\mathbb{G}_s\rvert$ $(512m + 384)$	✓	✓
LinGen	$m\lvert\mathbb{F}\rvert + \lvert\mathbb{G}_s\rvert$ $(256m + 384)$	✓	✗
SanLinGen (the current paper)	$(m+1)\lvert\mathbb{F}\rvert + \lvert\mathbb{G}_s\rvert$ $(256m + 640)$	✓	✓

field element to LinGen's communication. We depict SanLinGen in Fig. 5 and compare it to more simplistic protocols in Table 2. SanLinGen's computational special-soundness proof (see Appendix A.5) is inspired by the proof of SanLin.

Theorem 4. *Let $n, m \in \mathsf{poly}(\lambda)$ and $\boldsymbol{\kappa} = (2n+1, m, m+2)$. If KZG for degree $\leq n$ polynomials has computational $(n+1)$-special-soundness and evaluation-binding, then* SanLinGen *has computational $\boldsymbol{\kappa}$-special-soundness.*

4 Special-Soundness of Plonk

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 TriRSDH holds. Recall that KZG is evaluation-binding and specially sound under the ARSDH assumption. In addition, we prove that a sanitized variant SanPlonk (with one field element of extra communication) of Plonk has special-soundness without relying on TriRSDH. By applying the Fiat-Shamir transform to either of the two constructions, one can obtain a zk-SNARKs secure in the ROM under the same assumptions.

4.1 Preliminaries For Plonk

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

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.

Due to the lack of the space, we describe the polynomials (like $\mathsf{q_M}(X)$) that define a specific circuit in Appendix B.1. (They are exactly the same as in [GWC19].)

The SNARK proof relation. We use the notation $\mathsf{q}_{\mathsf{X}i} := \mathsf{q_X}(\omega^i)$ for the polynomials defined above. Define $\mathcal{P} = \{\mathsf{q_M}(X), \mathsf{q_L}(X), \mathsf{q_R}(X), \mathsf{q_O}(X), \mathsf{q_C}(X),$

$\mathsf{S}_{\sigma1}(X), \mathsf{S}_{\sigma2}(X), \mathsf{S}_{\sigma3}(X)\}$, where the polynomials satisfy the above conditions. Thus, $\mathcal{P}$ is the set of polynomials that defines a given circuit. Given $\ell \le n$ 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 $i \in [1, \ell]$: $\mathsf{q}_{\mathsf{M}i} = \mathsf{q}_{\mathsf{R}i} = \mathsf{q}_{\mathsf{O}i} = \mathsf{q}_{Ci} = 0$ and $\mathsf{q}_{\mathsf{L}i} = -1$, which guarantees
$$\mathsf{q}_{\mathsf{M}i} w_i w_{n+i} + \mathsf{q}_{\mathsf{L}i} w_i + \mathsf{q}_{\mathsf{R}i} w_{n+i} + \mathsf{q}_{\mathsf{O}i} w_{2n+i} + \mathsf{q}_{Ci} = -w_i \ . \tag{3}$$
We see later that this is needed to force the prover to use the correct $\mathbb{x}$.
2. For all $i \in [\ell + 1, n]$:
$$\mathsf{q}_{\mathsf{M}i} w_i w_{n+i} + \mathsf{q}_{\mathsf{L}i} w_i + \mathsf{q}_{\mathsf{R}i} w_{n+i} + \mathsf{q}_{\mathsf{O}i} w_{2n+i} + \mathsf{q}_{Ci} = 0 \ , \tag{4}$$
3. For all $i \in [1, 3n]$:
$$w_i = w_{\sigma(i)} \ . \tag{5}$$

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

4.2 Plonk And SanPlonk

We present Plonk and its sanitized variant SanPlonk. While we describe their interactive versions, to save space, we will omit the adjective "interactive". We describe them in parallel, highlighting SanPlonk's additional sanitization steps. Compared to Plonk, the SanPlonk batch opens $[\mathsf{t}_{\mathsf{lo}}(\mathfrak{z})]_1$, $[\mathsf{t}_{\mathsf{mid}}(\mathfrak{z})]_1$, and $[\mathsf{t}_{\mathsf{hi}}(\mathfrak{z})]_1$ (three group elements sent in Plonk), applying the sanitization technique from Section 3.2. This results in an interactive argument with two additional rounds and one more field element sent by the prover, compared to Plonk. We prove the computational special-soundness of Plonk assuming that (1) KZG is evaluation-binding and special sound, and (2) a new falsifiable assumption TriRSDH holds. We prove the computational special-soundness of SanPlonk solely under (1).

Common preprocessed input: n, $[x, \dots, x^{n+5}]_1$, $(q_{\mathsf{M}i}, q_{\mathsf{L}i}, q_{\mathsf{R}i}, q_{\mathsf{O}i}, q_{\mathsf{C}i})_{i=1}^{n}$, σ^*, $\mathsf{q}_{\mathsf{M}}(X) = \sum_{i=1}^{n} q_{\mathsf{M}i} L_i(X)$, $\mathsf{q}_{\mathsf{L}}(X) = \sum_{i=1}^{n} q_{\mathsf{L}i} L_i(X)$, $\mathsf{q}_{\mathsf{R}}(X) = \sum_{i=1}^{n} q_{\mathsf{R}i} \cdot L_i(X)$, $\mathsf{q}_{\mathsf{O}}(X) = \sum_{i=1}^{n} q_{Oi} L_i(X)$, $\mathsf{q}_{\mathsf{C}}(X) = \sum_{i=1}^{n} q_{Ci} L_i(X)$, $\mathsf{S}_{\sigma1}(X) = \sum_{i=1}^{n} \sigma^*(i) \cdot L_i(X)$, $\mathsf{S}_{\sigma2}(X) = \sum_{i=1}^{n} \sigma^*(n+i) L_i(X)$, $\mathsf{S}_{\sigma3}(X) = \sum_{i=1}^{n} \sigma^*(2n+i) L_i(X)$.

Verifier preprocessed input: $[\mathsf{q}_{\mathsf{M}}]_1 := \mathsf{q}_{\mathsf{M}}(x) \cdot [1]_1$, $[\mathsf{q}_{\mathsf{L}}]_1 := \mathsf{q}_{\mathsf{L}}(x) \cdot [1]_1$, $[\mathsf{q}_{\mathsf{R}}]_1 := \mathsf{q}_{\mathsf{R}}(x) \cdot [1]_1$, $[\mathsf{q}_{\mathsf{O}}]_1 := \mathsf{q}_{\mathsf{O}}(x) \cdot [1]_1$, $[\mathsf{q}_{\mathsf{C}}]_1 := \mathsf{q}_{\mathsf{C}}(x) \cdot [1]_1$, $[s_{\sigma1}]_1 := \mathsf{S}_{\sigma1}(x) \cdot [1]_1$, $[s_{\sigma2}]_1 := \mathsf{S}_{\sigma2}(x) \cdot [1]_1$, $[s_{\sigma3}]_1 := \mathsf{S}_{\sigma3}(x) \cdot [1]_1$, $x \cdot [1]_2$,

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

First round. On input $(w_i)_{i=1}^{3n}$, the prover does the following. Sample $(b_1, \dots, b_9) \leftarrow_{\$} \mathbb{F}$. Compute wire polynomials $\mathsf{a}(X) \leftarrow \sum_{i=1}^{n} w_i L_i(X) + (b_1 X + b_2) \mathsf{Z}_{\mathbb{H}}(X)$, $\mathsf{b}(X) \leftarrow \sum_{i=1}^{n} w_{n+i} L_i(X) + (b_3 X + b_4) \mathsf{Z}_{\mathbb{H}}(X)$, $\mathsf{c}(X) \leftarrow \sum_{i=1}^{n} w_{2n+i} \cdot L_i(X) + (b_5 X + b_6) \mathsf{Z}_{\mathbb{H}}(X)$. 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 $\mathsf{z}(X) \leftarrow L_1(X) + \sum_{i=1}^{n-1} \left(\prod_{j=1}^{i} \frac{(w_j+\beta\omega^j+\gamma)(w_{n+j}+\beta k_1 \omega^j+\gamma)(w_{2n+j}+\beta k_2 \omega^j+\gamma)}{(w_j+\sigma^*(j)\omega^j+\gamma)(w_{n+j}+\sigma^*(n+j)\omega^j+\gamma)(w_{2n+j}+\sigma^*(2n+j)\omega^j+\gamma)} \right) L_{i+1}(X) + (b_7 X^2 + b_8 X + b_9)\mathsf{Z}_{\mathbb{H}}(X)$ and sends $[\mathsf{z}(x)]_1$ to V. V replies with $\alpha \leftarrow_{\$} \mathbb{F}$.

Third round. The prover does the following. Sample $\alpha \leftarrow \mathbb{F}$. Compute

$$\begin{aligned}
\mathsf{F}_0(X) &:= \mathsf{a}(X)\mathsf{b}(X)\mathsf{q}_\mathsf{M}(X) + \mathsf{a}(X)\mathsf{q}_\mathsf{L}(X) + \mathsf{b}(X)\mathsf{q}_\mathsf{R}(X) + \mathsf{c}(X)\mathsf{q}_\mathsf{O}(X) + \mathsf{PI}(X) + \mathsf{q}_\mathsf{C}(X) \\
\mathsf{F}_1(X) &:= (\mathsf{a}(X) + \beta X + \gamma)(\mathsf{b}(X) + \beta k_1 X + \gamma)(\mathsf{c}(X) + \beta k_2 X + \gamma)\mathsf{z}(X) \\
&\quad - (\mathsf{a}(X) + \beta \mathsf{S}_{\sigma 1}(X) + \gamma)(\mathsf{b}(X) + \beta \mathsf{S}_{\sigma 2}(X) + \gamma)(\mathsf{c}(X) + \beta \mathsf{S}_{\sigma 3}(X) + \gamma)\mathsf{z}(X\omega) \\
\mathsf{F}_2(X) &:= (\mathsf{z}(X) - 1)L_1(X) \\
\mathsf{F}(X) &:= \mathsf{F}_0(X) + \alpha \mathsf{F}_1(X) + \alpha^2 \mathsf{F}_2(X) \ , \\
\mathsf{t}(X) &:= \frac{\mathsf{F}(X)}{\mathsf{Z}_{\mathbb{H}}(X)} \ .
\end{aligned} \tag{6}$$

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 $\mathsf{t}(X) = \mathsf{t}'_\mathsf{lo}(X) + X^n \mathsf{t}'_\mathsf{mid}(X) + X^{2n}\mathsf{t}'_\mathsf{hi}(X)$. [7] 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*) 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*) or sixth (*SanPlonk*) round.* Prover does the following. Compute the linearization polynomial $\mathsf{r}(X)$:

$$\begin{aligned}
\Lambda_0(X) &= \bar{a}\bar{b} \cdot \mathsf{q}_\mathsf{M}(X) + \bar{a} \cdot \mathsf{q}_\mathsf{L}(X) + \bar{b} \cdot \mathsf{q}_\mathsf{R}(X) + \bar{c} \cdot \mathsf{q}_\mathsf{O}(X) + \mathsf{PI}(\mathfrak{z}) + \mathsf{q}_\mathsf{C}(X) \ , \\
\Lambda_1(X) &= (\bar{a} + \beta\mathfrak{z} + \gamma)(\bar{b} + \beta k_1 \mathfrak{z} + \gamma)(\bar{c} + \beta k_2 \mathfrak{z} + \gamma) \cdot \mathsf{z}(X) \\
&\quad - (\bar{a} + \beta \bar{s}_{\sigma 1} + \gamma)(\bar{b} + \beta \bar{s}_{\sigma 2} + \gamma)(\bar{c} + \beta \cdot \mathsf{S}_{\sigma 3}(X) + \gamma)\bar{z}_\omega \ , \\
\Lambda_2(X) &= (\mathsf{z}(X) - 1)L_1(\mathfrak{z}) \ , \\
\mathsf{r}(X) &= \Lambda_0(X) + \alpha\Lambda_1(X) + \alpha^2\Lambda_2(X) \\
&\quad - \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}) \cdot (\mathsf{t}_\mathsf{lo}(X) + \mathfrak{z}^n \mathsf{t}_\mathsf{mid}(X) + \mathfrak{z}^{2n}\mathsf{t}_\mathsf{hi}(X)) \ .
\end{aligned} \tag{7}$$

Let

$$\begin{aligned}
\mathsf{H}(X) &:= \mathsf{r}(X) + v\mathsf{a}(X) + v^2\mathsf{b}(X) + v^3\mathsf{c}(X) + v^4\mathsf{S}_{\sigma 1}(X) + v^5\mathsf{S}_{\sigma 2}(X) \\
&\quad + v^6(\mathsf{t}_\mathsf{lo}(X) + \delta\mathsf{t}_\mathsf{mid}(X) + \delta^2\mathsf{t}_\mathsf{hi}(X)) \ , \\
\bar{H} &:= v\bar{a} + v^2\bar{b} + v^3\bar{c} + v^4\bar{s}_{\sigma 1} + v^5\bar{s}_{\sigma 2} + v^6\bar{t}_\mathfrak{z} \ .
\end{aligned} \tag{8}$$

[7] The polynomial is split since we want to avoid committing to $\mathsf{t}(X)$ of degree $\approx 3n$, which would force us to increase the SRS size.

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 $([a]_1, [b]_1, [c]_1, [z]_1, [t_{lo}]_1, [t_{mid}]_1, [t_{hi}]_1, [\mathsf{W}_{\mathfrak{z}}]_1, [\mathsf{W}_{\mathfrak{z}\omega}]_1) \in \mathbb{G}_1^9$.
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: $\mathsf{r}_0 := \mathsf{PI}(\mathfrak{z}) - L_1(\mathfrak{z})\alpha^2 - \alpha(\bar{a} + \beta\bar{s}_{\sigma 1} + \gamma)(\bar{b} + \beta\bar{s}_{\sigma 2} + \gamma)(\bar{c} + \gamma)\bar{z}_\omega$, and let $\mathsf{r}'(X) := \mathsf{r}(X) - \mathsf{r}_0$.
5. Sample $u \leftarrow_{\$} \mathbb{F}$ and compute the first part of the batched polynomial commitment $[D]_1 := [\mathsf{r}']_1 + u \cdot [z]_1$:

$$\begin{aligned}[D]_1 := &\bar{a}\bar{b} \cdot [\mathsf{q_M}]_1 + \bar{a} \cdot [\mathsf{q_L}]_1 + \bar{b} \cdot [\mathsf{q_R}]_1 + \bar{c} \cdot [\mathsf{q_O}]_1 + [\mathsf{q_C}]_1 \\ &+ \left((\bar{a} + \beta\mathfrak{z} + \gamma)(\bar{b} + \beta k_1\mathfrak{z} + \gamma)(\bar{c} + \beta k_2\mathfrak{z} + \gamma)\alpha + L_1(\mathfrak{z})\alpha^2 + u\right) \cdot [z]_1 \\ &- (\bar{a} + \beta\bar{s}_{\sigma 1} + \gamma)\ (\bar{b} + \beta\bar{s}_{\sigma 2} + \gamma)\alpha\beta\bar{z}_\omega \cdot [s_{\sigma 3}]_1 \\ &- \mathsf{Z}_{\mathbb{H}}(\mathfrak{z})([t_{lo}]_1 + \mathfrak{z}^n \cdot [t_{mid}]_1 + \mathfrak{z}^{2n} \cdot [t_{hi}]_1)\ .\end{aligned}$$

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:

$$([\mathsf{W}_{\mathfrak{z}}]_1 + u \cdot [\mathsf{W}_{\mathfrak{z}\omega}]_1) \bullet [x]_2 \stackrel{?}{=} (\mathfrak{z} \cdot [\mathsf{W}_{\mathfrak{z}}]_1 + u\mathfrak{z}\omega \cdot [\mathsf{W}_{\mathfrak{z}\omega}]_1 + [F]_1 - [E]_1) \bullet [1]_2\ . \quad (9)$$

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

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

We state the theorem that Plonk/SanPlonk (both described in the previous section) have computational special-soundness. Our proof of Plonk uses the following novel falsifiable assumption n-TriRSDH. Intuitively, TriRSDH states that it must be hard to come up with three polynomial commitments $[t_{lo}, t_{mid}, t_{hi}]_1$, differing evaluation points $\mathfrak{z}_i$ and a rational non-polynomial function $\mathsf{F}(X)/\mathsf{Z}_{\mathbb{H}}(X)$, and a proof that for every i, $[t_{lo} + \mathfrak{z}_i t_{mid} + \mathfrak{z}_i^{2n} t_{hi}]_1$ opens to $\mathsf{F}(\mathfrak{z}_i)/\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)$. That is, TriRSDH formalizes the fact that Plonk's optimization of not opening $[t_{lo}, t_{mid}, t_{hi}]_1$ does not compromise soundness. Importantly, TriRSDH minimizes the dependency on any other details; moreover, it is a falsifiable assumption. We refer to Appendix C for TriRSDH's security proof in the AGMOS.

Definition 2 (n-TriRSDH). *Let $\mathbb{H}$ be a multiplicative subgroup of $\mathbb{F}^*$ of order n. Let $\kappa_{\mathsf{kzg}} = n + 5$ and $\kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}} + 1$. For any PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{trirsdh}}_{\mathsf{Pgen},\mathcal{A},n,\mathbb{H}}(\lambda) :=$*

$$\Pr\left[\begin{array}{l|l} \forall i \neq i'. \mathfrak{z}_i \neq \mathfrak{z}_{i'} \in \mathbb{F} \wedge & x \leftarrow_{\$} \mathbb{F}; \\ \mathsf{F}(X) \in \mathbb{F}_{\leq \kappa_{\mathfrak{z}}-1}[X] \wedge (\mathsf{Z}_{\mathbb{H}}(X) \nmid \mathsf{F}(X)) \wedge & \mathsf{ck} \leftarrow ([1, x, \ldots, x^{\kappa_{\mathsf{kzg}}}]_1, [1, x]_2); \\ \forall i \in [1, \kappa_{\mathfrak{z}}]. \left[t_{lo} + \mathfrak{z}_i^n t_{mid} + \mathfrak{z}_i^{2n} t_{hi} - \frac{\mathsf{F}(\mathfrak{z}_i)}{\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)}\right]_1 \bullet [1]_2 & \begin{pmatrix} (\mathfrak{z}_i, [\chi_i]_1)_{i=1}^{\kappa_{\mathfrak{z}}}, \\ [t_{lo}, t_{mid}, t_{hi}]_1, \\ \mathsf{F}(X) \end{pmatrix} \leftarrow \mathcal{A}(\mathsf{p}, \mathsf{ck}) \\ \qquad = [\chi_i]_1 \bullet [x - \mathfrak{z}_i]_2 & \end{array}\right] \approx_\lambda 0 \enspace .$$

Before going on, we note that the Plonk verifier performs batch verification, using a batching coefficient u created after the prover's last message. Clearly, one can unbatch the two verification equations, without having to rewind the prover. Batching with u just introduces a soundness error $1/|\mathbb{F}|$. We say that a Plonk transcript is accepting (Plonk transcript) if it is accepting in the unbatched case by Plonk's verifier. That is, Eq. (9) is replaced by two checks, $[\mathsf{W}_{\mathfrak{z}}]_1 \bullet [x]_2 = (\mathfrak{z} \cdot [\mathsf{W}_{\mathfrak{z}}]_1 + [F]_1 - [E_0]_1) \bullet [1]_2$ and $[\mathsf{W}_{\mathfrak{z}\omega}]_1 \bullet [x]_2 = (\mathfrak{z}\omega \cdot [\mathsf{W}_{\mathfrak{z}\omega}]_1 - [E_1]_1) \bullet [1]_2$.

We divide the proof into several smaller lemmas. In Section 4.4, we will analyze a subtree of Plonk's and SanPlonk's accepting transcripts for fixed β, γ, and α, showing that from it, one can extract certain polynomials. In Section 4.5, we use that result to prove the special-soundness of Plonk and SanPlonk.

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

$$\begin{aligned} \kappa_{\mathsf{kzg}} &= n + 5 \enspace , \\ \boldsymbol{\kappa}_{\mathsf{Plonk}} &= (\kappa_\beta = 3n + 1, \kappa_\gamma = 3n + 1, \kappa_\alpha = 3, \kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}} + 1, \kappa_v^{\mathsf{Plonk}} = 6) \enspace , \qquad (10) \\ \boldsymbol{\kappa}_{\mathsf{san}} &= (\kappa_\beta = 3n + 1, \kappa_\gamma = 3n + 1, \kappa_\alpha = 3, \kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}} + 1, \kappa_\delta = 3, \kappa_v^{\mathsf{san}} = 7) \end{aligned}$$

corresponding to the branching factors of KZG, Plonk, and SanPlonk. 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.

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

$$\mathsf{tr}_{ijk} = \begin{pmatrix} [a, b, c]_1, \beta, \gamma, [z]_1, \alpha, [t_{lo}, t_{mid}, t_{hi}]_1, \mathfrak{z}_i, \bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{s}_{\sigma 1i}, \bar{s}_{\sigma 2i}, \bar{z}_{\omega i}, \\ \delta_{ij}, \bar{t}_{\mathfrak{z}ij}, v_{ijk}, [\mathsf{W}_{\mathfrak{z}ijk}, \mathsf{W}_{\mathfrak{z}\omega ijk}]_1 \end{pmatrix} \enspace . \qquad (11)$$

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

```
$\mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{T}_{\beta\gamma\alpha})$
1 : Parse $\hat{T}_{\beta\gamma\alpha} = (\mathsf{tr}_{ijk})_{i\in[1,\kappa_{\mathfrak{z}}],j\in[1,\kappa_\delta],k\in[1,\kappa_v]}$;   // $\mathsf{tr}_{ijk}$ as in Eq. (11); $\kappa_\delta = 1$ in Plonk
2 : for $i \in [1, \kappa_{\mathfrak{z}}]$ do
3 :     $[\Lambda_{0i}]_1 \leftarrow \bar{a}_i\bar{b}_i[\mathsf{q_M}]_1 + \bar{a}_i[\mathsf{q_L}]_1 + \bar{b}_i[\mathsf{q_R}]_1 + \bar{c}_i[\mathsf{q_O}]_1 + \mathsf{PI}(\mathfrak{z})[1]_1 + [\mathsf{q_C}]_1$;
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 :             $- (\bar{a}_i + \beta\bar{s}_{\sigma 1} + \gamma)(\bar{b}_i + \beta\bar{s}_{\sigma 2} + \gamma)(\bar{c}_i[1]_1 + \beta[\mathsf{S}_{\sigma 3}(x)]_1 + \gamma[1]_1)\bar{z}_{\omega,i}$;
6 :     $[\Lambda_{2i}]_1 \leftarrow [z-1]_1 L_1(\mathfrak{z}_i)$;
7 :     $[\mathsf{r}_i]_1 \leftarrow [\Lambda_{0i}(x)]_1 + \alpha[\Lambda_{1i}]_1 + \alpha^2[\Lambda_{2i}]_1 - \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)\cdot([t_{lo}]_1 + \mathfrak{z}_i^n[t_{mid}]_1 + \mathfrak{z}_i^{2n}[t_{hi}]_1)$;
8 :     $[\mathsf{W}'_{\mathfrak{z}i11}, \dots, \mathsf{W}'_{\mathfrak{z}i1\kappa_v}]_1^\intercal \leftarrow \boldsymbol{V}_{i1}^{-1}[\mathsf{W}_{\mathfrak{z}i11}, \dots, \mathsf{W}_{\mathfrak{z}i1\kappa_v}]_1^\intercal$;
9 :     $\mathsf{k.tr}_{1i} \leftarrow ([\mathsf{r}_i]_1, \mathfrak{z}_i, 0, [\mathsf{W}'_{\mathfrak{z}i11}]_1)$; $\mathsf{k.tr}_{2i} \leftarrow ([a]_1, \mathfrak{z}_i, \bar{a}_i, [\mathsf{W}'_{\mathfrak{z}i12}]_1)$;
10 :    $\mathsf{k.tr}_{3i} \leftarrow ([b]_1, \mathfrak{z}_i, \bar{b}_i, [\mathsf{W}'_{\mathfrak{z}i13}]_1)$; $\mathsf{k.tr}_{4i} \leftarrow ([c]_1, \mathfrak{z}_i, \bar{c}_i, [\mathsf{W}'_{\mathfrak{z}i14}]_1)$;
11 :    $\mathsf{k.tr}_{5i} \leftarrow ([s_{\sigma 1}]_1, \mathfrak{z}_i, \bar{s}_{\sigma 1i}, [\mathsf{W}'_{\mathfrak{z}i15}]_1)$; $\mathsf{k.tr}_{6i} \leftarrow ([s_{\sigma 2}]_1, \mathfrak{z}_i, \bar{s}_{\sigma 2i}, [\mathsf{W}'_{\mathfrak{z}i16}]_1)$;
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 :    $\mathsf{k.tr}_i^\omega \leftarrow ([z]_1, \mathfrak{z}_i\omega, \bar{z}_{\omega i}, [\mathsf{W}_{\mathfrak{z}\omega i11}]_1)$; 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)* $[\mathsf{r}_i]_1$, $[a]_1$, $[b]_1$, $[c]_1$, $[s_{\sigma 1}]_1$ *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 T be a $(\kappa_\beta, \kappa_\gamma, \kappa_\alpha, \dots)$-tree of accepting transcripts. We fix a $(1, 1, 1, \kappa_{\mathfrak{z}}, \kappa_\delta, \kappa_v^{\mathsf{san}})$-subtree $\hat{T}_{\beta\gamma\alpha}$ of T for some β, γ, and α. Then, $\hat{T}_{\beta\gamma\alpha} = \{\mathsf{tr}_{ijk}\}$ contains accepting transcripts given in Eq. (11) with mutually different $\mathfrak{z}_i$.

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

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

$$
\begin{aligned}
\bar{H}_{ijk} &\leftarrow v_{ijk}^0 \cdot 0 + v_{ijk}^1\bar{a}_i + v_{ijk}^2\bar{b}_i + v_{ijk}^3\bar{c}_i + v_{ijk}^4\bar{s}_{\sigma 1i} + v_{ijk}^5\bar{s}_{\sigma 2i} + v_{ijk}^6\bar{t}_{\mathfrak{z}ij} \ , \\
[\mathsf{t}_{\delta_{ij}}]_1 &\leftarrow [t_{lo} + \delta_{ij}t_{mid} + \delta_{ij}^2 t_{hi}]_1 \ , \qquad (12) \\
[\mathsf{H}_{ijk}]_1 &\leftarrow v_{ijk}^0[\mathsf{r}_i]_1 + v_{ijk}[a]_1 + v_{ijk}^2[b]_1 + v_{ijk}^3[c]_1 + v_{ijk}^4[s_{\sigma 1}]_1 + v_{ijk}^5[s_{\sigma 2}]_1 + v^6[\mathsf{t}_{\delta_{ij}}]_1 \ .
\end{aligned}
$$

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 Plonk and $\mathsf{SanPlonk}$. However, the subroutine on Fig. 6 corresponds to both.

<u>Plonk.</u> Here,

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

is an invertible Vandermonde matrix. According to Eq. (12), $(\bar{H}_{i1}, \dots, \bar{H}_{i6})^\intercal = \boldsymbol{V}_i \cdot (0, \bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{s}_{\sigma 1i}, \bar{s}_{\sigma 2i})^\intercal$ and $[\mathsf{H}_{i1}, \dots, \mathsf{H}_{i6}]_1 = \boldsymbol{V}_i \cdot [\mathsf{r}_i, a, b, c, s_{\sigma 1}, s_{\sigma 2}]_1^\intercal$. 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.

<u>SanPlonk.</u> Here,

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

are invertible Vandermonde matrices. According to Eq. (12), $[\mathsf{H}_{ij1}, \dots, \mathsf{H}_{ij7}]_1 = \boldsymbol{V}_{ij} \cdot [\mathsf{r}_i, a, b, c, s_{\sigma 1}, s_{\sigma 2}, \mathsf{t}_{\delta_{ij}}]_1^\intercal$ and $(\bar{H}_{ij1}, \dots, \bar{H}_{ij7})^\intercal = \boldsymbol{V}_{ij} \cdot (0, \bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{s}_{\sigma 1i}, \bar{s}_{\sigma 2i}, \bar{t}_{\mathfrak{z}ij})^\intercal$. 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, $\mathsf{k.tr}_{7i}$, $\mathsf{k.tr}_{7i}$, and $\mathsf{k.tr}_{7i}$ are accepting KZG transcripts for every i. $\square$

Theorem 5 (Subtree extractor). *Let T be a $\boldsymbol{\kappa}_{\mathsf{Plonk}}$-tree of* Plonk*'s (resp., $\boldsymbol{\kappa}_{\mathsf{san}}$-tree of* $\mathsf{SanPlonk}$*'s) accepting transcripts. Let $\hat{T}_{\beta\gamma\alpha} = (\mathsf{tr}_{ijk})$ be a subtree of T for any fixed β, γ, and α, where tr_{ijk} are as in Eq. (11). Assume that KZG is evaluation-binding and computational $(\kappa_{\mathsf{kzg}} + 1)$-special-sound. In the case of* Plonk*, assume the n-*TriRSDH *assumption holds. There exists a DPT extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$ that, given $\hat{T}_{\beta\gamma\alpha}$, 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. (6)) is a polynomial.*

Proof. Let T be a $(\kappa_\beta, \kappa_\gamma, \kappa_\alpha, \kappa_{\mathfrak{z}}, \kappa_\delta, \kappa_v)$-tree of accepting transcripts and let $\hat{T}_{\beta\gamma\alpha}$ be its $(1, 1, 1, \kappa_{\mathfrak{z}}, \kappa_\delta, \kappa_v)$-subtree for any fixed α, β, γ. By Lemma 2, $\mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{T}_{\beta\gamma\alpha})$ 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 Plonk and $\mathsf{SanPlonk}$. However, the extractors and adversaries on figures (say, Fig. 7) correspond to both.

<u>Case of Plonk.</u> In Fig. 7, 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 $\kappa_{\mathsf{kzg}} = n + 5$. After executing $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$, we use the following procedure to possibly set one of the "bad" flags:

```
$\mathsf{Ext}^{\mathsf{sub}}_{\mathsf{ss}}(\mathsf{ck}, \hat{T}_{\beta\gamma\alpha})$
$((\mathbf{k.tr}_k)_{k\in[1,\kappa_v+2]}, \mathbf{k.tr}^{\boldsymbol{\omega}}) \leftarrow \mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{T}_{\beta\gamma\alpha})$;
$\mathsf{z}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}^{\boldsymbol{\omega}})$; $\mathsf{a}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_2)$;
$\mathsf{b}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_3)$; $\mathsf{c}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_4)$;
$\mathsf{t}_{\mathsf{lo}}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_7)$; $\mathsf{t}_{\mathsf{mid}}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_8)$; $\mathsf{t}_{\mathsf{hi}}(X) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck}, \mathbf{k.tr}_9)$;
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. 7. Plonk's/SanPlonk's special-soundness extractor $\mathsf{Ext}^{\mathsf{sub}}_{\mathsf{ss}}$.

```
$\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck})$
$\hat{T}_{\beta\gamma\alpha} \leftarrow \mathcal{A}^{\mathsf{plonk}}_{\mathsf{ss}}(\mathsf{ck})$;
$((\mathbf{k.tr}_k)_{k\in[1,\kappa_v+2]}, \mathbf{k.tr}^{\boldsymbol{\omega}}) \leftarrow \mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{T}_{\beta\gamma\alpha})$;
$(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{t}_{\mathsf{lo}}(X), \mathsf{t}_{\mathsf{mid}}(X), \mathsf{t}_{\mathsf{hi}}(X)) \leftarrow \mathsf{Ext}^{\mathsf{sub}}_{\mathsf{ss}}(\mathsf{ck}, \hat{T}_{\beta\gamma\alpha})$;
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 $\perp$;
```

Fig. 8. Plonk's/SanPlonk's KZG's special-soundness adversary $\mathcal{B}^{\mathsf{kzg}}_{\mathsf{ss}}$.

(i) $\mathsf{bad}_{\mathsf{ext}} \leftarrow \mathsf{false}$; $\mathsf{bad}_{\mathsf{evb}} \leftarrow \mathsf{false}$; $\mathsf{bad}_{\mathrm{trirsdh}} \leftarrow \mathsf{false}$;
(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 $\mathsf{bad}_{\mathrm{trirsdh}} \leftarrow \mathsf{true}$;

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

```
$\mathcal{C}_{\mathsf{evb}}(\mathsf{p}, \mathsf{ck})$
$\hat{T}_{\beta\gamma\alpha} \leftarrow \mathcal{A}_{\mathsf{ss}}^{\mathsf{plonk}}(\mathsf{ck}); ((\mathbf{k.tr}_k)_{k \in [1, \kappa_v + 2]}, \mathbf{k.tr}^{\boldsymbol{\omega}}) \leftarrow \mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{T}_{\beta\gamma\alpha})$;
$(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{t_{lo}}(X), \mathsf{t_{mid}}(X), \mathsf{t_{hi}}(X)) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}(\mathsf{ck}, \hat{T}_{\beta\gamma\alpha})$;
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 $([z]_1, \mathfrak{z}_i\omega, \bar{z}_{\omega i}, [\mathsf{W}'_{\omega ij1}]_1, \mathsf{z}(\mathfrak{z}_i\omega), [(\mathsf{z}(x) - \mathsf{z}(\mathfrak{z}_i\omega))/(x - \mathfrak{z}_i\omega)]_1)$; fi
   if $\mathsf{a}(\mathfrak{z}_i) \neq \bar{a}_i$ then return $([a]_1, \mathfrak{z}_i, \bar{a}_i, [\mathsf{W}'_{\omega ij2}]_1, \mathsf{a}(\mathfrak{z}_i), [(\mathsf{a}(x) - \mathsf{a}(\mathfrak{z}_i))/(x - \mathfrak{z}_i)]_1)$; fi
   if $\mathsf{b}(\mathfrak{z}_i) \neq \bar{b}_i$ then return $([b]_1, \mathfrak{z}_i, \bar{b}_i, [\mathsf{W}'_{\omega ij3}]_1, \mathsf{b}(\mathfrak{z}_i), [(\mathsf{b}(x) - \mathsf{b}(\mathfrak{z}_i))/(x - \mathfrak{z}_i)]_1)$; fi
   if $\mathsf{c}(\mathfrak{z}_i) \neq \bar{c}_i$ then return $([c]_1, \mathfrak{z}_i, \bar{c}_i, [\mathsf{W}'_{\omega ij4}]_1, \mathsf{c}(\mathfrak{z}_i), [(\mathsf{c}(x) - \mathsf{c}(\mathfrak{z}_i))/(x - \mathfrak{z}_i)]_1)$; fi
   if $\mathsf{t_{lo}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z}lo,i}$ then return $([t_{lo}]_1, \mathfrak{z}_i, \bar{t}_{\mathfrak{z}lo,i}, [\mathsf{W}'_{\omega ij7}]_1, \mathsf{t_{lo}}(\mathfrak{z}_i), [(\mathsf{t_{lo}}(x) - \mathsf{t_{lo}}(\mathfrak{z}_i))/(x - \mathfrak{z}_i)]_1)$; fi
   if $\mathsf{t_{mid}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z}mid,i}$ then return $([t_{mid}]_1, \mathfrak{z}_i, \bar{t}_{\mathfrak{z}mid,i}, [\mathsf{W}'_{\omega ij8}]_1, \mathsf{t_{mid}}(\mathfrak{z}_i), [(\mathsf{t_{mid}}(x) - \mathsf{t_{mid}}(\mathfrak{z}_i))/(x - \mathfrak{z}_i)]_1)$; fi
   if $\mathsf{t_{hi}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z}hi,i}$ then return $([t_{hi}]_1, \mathfrak{z}_i, \bar{t}_{\mathfrak{z}hi,i}, [\mathsf{W}'_{\omega ij9}]_1, \mathsf{t_{hi}}(\mathfrak{z}_i), [(\mathsf{t_{hi}}(x) - \mathsf{t_{hi}}(\mathfrak{z}_i))/(x - \mathfrak{z}_i)]_1)$; fi
endfor ;
for $i \in [1, \kappa_{\mathfrak{z}}]$ do
   $\mathsf{r}_i(X) \leftarrow \Lambda_{0i}(X) + \alpha\Lambda_{1i}(X) + \alpha^2\Lambda_{2i}(X) - \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) \cdot (\mathsf{t_{lo}}(X) + \mathfrak{z}_i^n \mathsf{t_{mid}}(X) + \mathfrak{z}_i^{2n} \mathsf{t_{hi}}(X))$;
   if $\mathsf{r}_i(\mathfrak{z}_i) \neq 0$ then return $([\mathsf{r}_i]_1, \mathfrak{z}_i, 0, [\mathsf{W}'_{\mathfrak{z}i11}]_1, \mathsf{r}_i(\mathfrak{z}_i), [\mathsf{r}_i(x) - \mathsf{r}_i(\mathfrak{z}_i)/(x - \mathfrak{z}_i)]_1)$;
endfor
for $i \in [1, \kappa_{\mathfrak{z}}]$ do
   if $\mathsf{S}_{\sigma 1}(\mathfrak{z}_i) \neq \bar{s}_{\sigma 1i}$ then
      return $([s_{\sigma 1}]_1, \mathfrak{z}_i, \bar{s}_{\sigma 1i}, [\mathsf{W}'_{\omega ij5}]_1, \mathsf{S}_{\sigma 1}(\mathfrak{z}_i), [(\mathsf{S}_{\sigma 1}(x) - \mathsf{S}_{\sigma 1}(\mathfrak{z}_i))/(x - \mathfrak{z}_i)]_1)$; fi
   if $\mathsf{S}_{\sigma 2}(\mathfrak{z}_i) \neq \bar{s}_{\sigma 2i}$ then
      return $([s_{\sigma 2}]_1, \mathfrak{z}_i, \bar{s}_{\sigma 2i}, [\mathsf{W}'_{\omega ij6}]_1, \mathsf{S}_{\sigma 2}(\mathfrak{z}_i), [(\mathsf{S}_{\sigma 2}(x) - \mathsf{S}_{\sigma 2}(\mathfrak{z}_i))/(x - \mathfrak{z}_i)]_1)$; fi
endfor ;
return $\perp$;
```

Fig. 9. KZG's evaluation-binding adversary $\mathcal{C}_{\mathsf{evb}}$.

2. **Claim 2**. There exists a KZG's $(\kappa_{\mathsf{kzg}} + 1)$-special-soundness adversary $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ (see Fig. 8), 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. 9) 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. 9 *is* a collision. Thus, $\Pr[\mathcal{C}_{\mathsf{evb}}$ succeeds $\mid \mathsf{bad}_{\mathsf{evb}}] = 1$.
4. **Claim 4**. There exists an n-TriRSDH adversary $\mathcal{D}_{\mathrm{trirsdh}}$ (see Fig. 10), such that $\Pr[\mathcal{D}_{\mathrm{trirsdh}}$ succeeds $\mid \mathsf{bad}_{\mathrm{trirsdh}}] = 1$.
   Since the proof of this claim is more complicated, we postpone its proof to a separate lemma (see Lemma 3).

```
$\mathcal{D}_{\mathrm{trirsdh}}(\mathsf{p}, \mathsf{ck})$
$\hat{T}_{\beta\gamma\alpha} \leftarrow \mathcal{A}_{\mathsf{ss}}^{\mathsf{plonk}}(\mathsf{ck}); ((\mathbf{k.tr}_k)_{k \in [1, \kappa_v + 2]}, \mathbf{k.tr}^{\boldsymbol{\omega}}) \leftarrow \mathsf{TE}_{*\mathsf{plonk}}(\mathsf{ck}, \hat{T}_{\beta\gamma\alpha});$
$(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}(\mathsf{ck}, \hat{T}_{\beta\gamma\alpha});$
$\mathsf{F}_0(X) \leftarrow \mathsf{a}(X)\mathsf{b}(X)\mathsf{q_M}(X) + \mathsf{a}(X)\mathsf{q_L}(X) + \mathsf{b}(X)\mathsf{q_R}(X) + \mathsf{c}(X)\mathsf{q_O}(X) + \mathsf{PI}(X) + \mathsf{q_C}(X);$
$\mathsf{F}_1(X) \leftarrow (\mathsf{a}(X) + \beta X + \gamma)(\mathsf{b}(X) + \beta k_1 X + \gamma)(\mathsf{c}(X) + \beta k_2 X + \gamma)\mathsf{z}(X)$
          $-(\mathsf{a}(X) + \beta \mathsf{S}_{\sigma 1}(X) + \gamma)(\mathsf{b}(X) + \beta \mathsf{S}_{\sigma 2}(X) + \gamma)(\mathsf{c}(X) + \beta \mathsf{S}_{\sigma 3}(X) + \gamma)\mathsf{z}(\omega X);$
$\mathsf{F}_2(X) \leftarrow (\mathsf{z}(X) - 1)L_1(X);$
$\mathsf{F}(X) \leftarrow \mathsf{F}_0(X) + \alpha \mathsf{F}_1(X) + \alpha^2 \mathsf{F}_2(X); (**)$
$\mathsf{t}(X) \leftarrow \mathsf{F}(X)/\mathsf{Z}_{\mathbb{H}}(X);$
if $\mathsf{Z}_{\mathbb{H}}(X) \nmid \mathsf{F}(X)$ then
  for $i \in [1, \kappa_{\mathfrak{z}}]$ do
    $\Lambda_{0i}(X) \leftarrow \bar{a}_i \bar{b}_i \mathsf{q_M}(X) + \bar{a}_i \mathsf{q_L}(X) + \bar{b}_i \mathsf{q_R}(X) + \bar{c}_i \mathsf{q_O}(X) + \mathsf{PI}(\mathfrak{z}) + \mathsf{q_C}(X);$
    $\Lambda_{1i}(X) \leftarrow (\bar{a}_i + \beta \mathfrak{z}_i + \gamma)(\bar{b}_i + \beta k_1 \mathfrak{z}_i + \gamma)(\bar{c}_i + \beta k_2 \mathfrak{z}_i + \gamma)\mathsf{z}(X)$
             $- (\bar{a}_i + \beta \bar{s}_{\sigma 1} + \gamma)(\bar{b}_i + \beta \bar{s}_{\sigma 2} + \gamma)(\bar{c}_i + \beta \mathsf{S}_{\sigma 3}(X) + \gamma)\bar{z}_{\omega,i};$
    $\Lambda_{2i}(X) \leftarrow (\mathsf{z}(X) - 1)L_1(\mathfrak{z}_i);$
    $\Lambda_i(X) \leftarrow \Lambda_{0i}(X) + \alpha \Lambda_{1i}(X) + \alpha^2 \Lambda_{2i}(X);$
    $\chi_i'(X) \leftarrow \frac{\Lambda_i(X) - \Lambda_i(\mathfrak{z}_i)}{X - \mathfrak{z}_i};$
    $[\chi_i]_1 \leftarrow \frac{1}{\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)}[\chi_i'(x) - \mathsf{W}_{\mathfrak{z}i1}']_1;$ endfor
  return $((\mathfrak{z}_i, [\chi_i]_1)_{i=1}^{\kappa_{\mathfrak{z}}}, [t_{lo}, t_{mid}, t_{hi}]_1, \mathsf{F}(X));$ fi
return $\perp;$
```

Fig. 10. The TriRSDH adversary $\mathcal{D}_{\mathrm{trirsdh}}$.

Recalling all three bad events are disjoint,

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

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, $\Pr[\mathsf{bad}_{\mathsf{ext}}] = \Pr[\mathsf{bad}_{\mathrm{trirsdh}}] = \mathsf{negl}(\lambda)$. Thus, $\Pr[\overline{\mathcal{E}}] \le \mathsf{negl}(\lambda)$. This proves the claim.

Case of $\mathsf{SanPlonk}$. We postpone the proof of this case to Appendix B.2. □

We are left to prove the following lemma. Recall that TriRSDH is defined in Definition 2.

Lemma 3. *There exists an n-*TriRSDH *adversary $\mathcal{D}_{\mathrm{trirsdh}}$ (see Fig. 10), such that* $\Pr[\mathcal{D}_{\mathrm{trirsdh}}$ *succeeds* $\mid \mathsf{bad}_{\mathrm{trirsdh}}] = 1$.

Proof (Proof of Lemma 3). Let $\mathcal{D}_{\mathrm{trirsdh}}$ be the n-TriRSDH adversary in Fig. 10. $\mathcal{D}_{\mathrm{trirsdh}}$ uses $\mathcal{A}_{\mathsf{ss}}^{\mathsf{plonk}}$ to obtain a subtree $\hat{T}_{\beta\gamma\alpha}$, runs $\mathsf{TE}_{*\mathsf{plonk}}$ on $\hat{T}_{\beta\gamma\alpha}$ to obtain a number of accepting KZG transcripts, and then runs $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}$ on $\hat{T}_{\beta\gamma\alpha}$ to obtain polynomials $(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X))$. After that, $\mathcal{D}_{\mathrm{trirsdh}}$ computes the polynomials $\mathsf{F}_s(X)$, $\mathsf{F}(X)$ and $\mathsf{t}(X)$ (as defined in Eq. (6)). If it satisfies TriRSDH's requirement $\mathsf{Z}_{\mathbb{H}}(X) \nmid \mathsf{F}(X)$, $\mathcal{D}_{\text{trirsdh}}$ computes the required KZG opening proofs $[\chi_i]_1$ and outputs a correctly formed TriRSDH adversary's output. Let us now argue that the output satisfies TriRSDH's conditions.

The first three conditions are straightforward. First, the values $\mathfrak{z}_i$ output by $\mathcal{A}_{\mathsf{ss}}^{\mathsf{plonk}}$ are mutually different by the definition of special-soundness. Second, $\mathsf{F}(X) = \mathsf{t}(X)\mathsf{Z}_{\mathbb{H}}(X)$ from line 7 is a polynomial of degree $\deg \mathsf{F}(X) \leq \max(\deg(\mathsf{F}_1), \deg(\mathsf{F}_2), \deg(\mathsf{F}_3)) \leq \deg \mathsf{a} + \deg \mathsf{b} + \deg \mathsf{c} + \deg \mathsf{z} \leq 4(n+5) = 4\kappa_{\mathsf{kzg}} = \mathfrak{z} - 1$. Third, from $\mathsf{bad}_{\text{trirsdh}} = \mathsf{true}$ it follows that $\mathsf{Z}_{\mathbb{H}}(X) \nmid \mathsf{F}(X)$.

To finish the proof, we now have to verify the fourth condition of TriRSDH that for all $i \in [1, \kappa_{\mathfrak{z}}]$, $[t_{lo} + \mathfrak{z}_i^n t_{mid} + \mathfrak{z}_i^{2n} t_{hi} - \mathsf{F}(\mathfrak{z}_i)/\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)]_1 \bullet [1]_2 = [\chi_i]_1 \bullet [x - \mathfrak{z}_i]_2$. TriRSDH defines linearization polynomials $\Lambda_{0i}(X)$, $\Lambda_{1i}(X)$, $\Lambda_{2i}(X)$ (see Eq. (7)) 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 $i \in [1, \kappa_{\mathfrak{z}}]$.

We want to show that $\mathcal{D}_{\text{trirsdh}}$ is successful whenever, given $\hat{T}_{\beta\gamma\alpha} \leftarrow \mathcal{A}_{\mathsf{ss}}^{\mathsf{plonk}}(\mathsf{ck})$ and $(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}(\mathsf{ck}, \hat{T}_{\beta\gamma\alpha})$, $\mathsf{bad}_{\text{trirsdh}} = \mathsf{true}$. Recall from the proof of Theorem 5 that then $\mathsf{bad}_{\mathsf{evb}} = \mathsf{bad}_{\mathsf{ext}} = \mathsf{false}$. 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 $i \in [1, \kappa_{\mathfrak{z}}]$. Hence, $\mathsf{F}_s(\mathfrak{z}_i) = \Lambda_{si}(\mathfrak{z}_i)$, $\mathsf{F}(\mathfrak{z}_i) = \Lambda_i(\mathfrak{z}_i)$, 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 $i \in [1, \kappa_{\mathfrak{z}}]$.

$\mathcal{D}_{\text{trirsdh}}$ defines $\chi'_i(X) = (\Lambda_i(X) - \Lambda_i(\mathfrak{z}_i))/(X - \mathfrak{z}_i) = (\Lambda_i(X) - \mathsf{F}(\mathfrak{z}_i))/(X - \mathfrak{z}_i)$ to be the KZG opening polynomial of $\Lambda_i(X)$, $i \in [1, \kappa_{\mathfrak{z}}]$, at point $\mathfrak{z}_i$. Here, the second equality follows from $\mathsf{bad}_{\mathsf{evb}} = \mathsf{bad}_{\mathsf{ext}} = \mathsf{false}$. Recall

$$[\mathsf{r}_i]_1 = [\Lambda_i(x) - \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) \cdot (t_{lo} + \mathfrak{z}^n t_{mid} + \mathfrak{z}^{2n} t_{hi})]_1 \tag{13}$$

from the lines 3, 4, and 6 in Fig. 6. Since $\mathsf{k.tr}_{1i}$ (see line 9 in Fig. 6) is accepting, it follows from Eq. (13), $\mathsf{bad}_{\mathsf{evb}} = \mathsf{bad}_{\mathsf{ext}} = \mathsf{false}$, and the definition of $\chi'_i(X)$ that

$$\mathsf{W}'_{\mathfrak{z}i1} \cdot (x - \mathfrak{z}_i) = \mathsf{r}_i = \chi'_i(x)(x - \mathfrak{z}_i) + \Lambda_i(\mathfrak{z}_i) - \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) \cdot (t_{lo} + \mathfrak{z}^n t_{mid} + \mathfrak{z}^{2n} t_{hi})$$

for $i \in [1, \kappa_{\mathfrak{z}}]$. Defining $[\chi_i]_1$ as in Fig. 10, we get

$$\left[t_{lo} + \mathfrak{z}^n t_{mid} + \mathfrak{z}^{2n} t_{hi} - \frac{\Lambda_i(\mathfrak{z}_i)}{\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i)}\right]_1 \bullet [1]_2 = [\chi_i]_1 \bullet [x - \mathfrak{z}_i]_2 \enspace ,$$

for $i \in [1, \kappa_{\mathfrak{z}}]$. Now, note that $\Lambda_i(\mathfrak{z}_i)/\mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) = \mathsf{t}(\mathfrak{z}_i)$. This proves the lemma. $\square$

4.5 Full Special-Soundness Proof

Finally, we are ready to prove the special-soundness of Plonk and SanPlonk. For this, we combine the subtree analysis of Section 4.4, 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 $\kappa_{\mathsf{kzg}} := n + 5$, $\boldsymbol{\kappa}_{\mathsf{Plonk}}$, and $\boldsymbol{\kappa}_{\mathsf{san}}$ be as in Eq.* (10)*.*

1. *If KZG is computational $(\kappa_{\mathsf{kzg}}+1)$-special-sound and evaluation-binding, and n-*TriRSDH *holds, then* Plonk *is $\boldsymbol{\kappa}_{\mathsf{Plonk}}$-special-sound.*

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

Pick an arbitrary $(1, 1, 1, \kappa_{\mathfrak{z}}, \kappa_\delta, \kappa_v)$-subtree $\hat{T}$ of $T$.
$(\mathsf{z}(X), \mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X), \mathsf{t}_{\mathsf{lo}}(X), \mathsf{t}_{\mathsf{mid}}(X), \mathsf{t}_{\mathsf{hi}}(X)) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}(\mathsf{ck}, \hat{T})$;
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. 11. The special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{*\mathsf{plonk}}$ for Plonk/SanPlonk.

2. *If KZG is computational* $(\kappa_{\mathsf{kzg}} + 1)$*-special-sound and evaluation-binding, then* SanPlonk *is computational* $\boldsymbol{\kappa}_{\mathsf{san}}$*-special-sound.*

Proof. Fix $\mathcal{P} = \{\mathsf{q_M}(X), \mathsf{q_L}(X), \mathsf{q_R}(X), \mathsf{q_O}(X), \mathsf{q_C}(X), \mathsf{S}_{\sigma 1}(X), \mathsf{S}_{\sigma 2}(X), \mathsf{S}_{\sigma 3}(X)\}$ for a relation defined by encoding a circuit as in Section 4.1. 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 T. We describe the special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{*\mathsf{plonk}}$ for Plonk in Fig. 11. The extractor picks an arbitrary $(1, 1, 1, \kappa_{\mathfrak{z}}, \kappa_\delta, \kappa_v)$-subtree $\hat{T} = \hat{T}_{\beta\gamma\alpha}$ of 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 $(\mathbb{x} = (\bar{w}_i)_{i=1}^{\ell}, \mathbb{w}) \in \mathcal{R}_{\mathcal{P}}$.

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's 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.[8]

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

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 $\hat{T} \in \mathsf{SubTrees}_T$, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sub}}(\mathsf{ck}, \hat{T})$ outputs $\mathsf{s.tr}_{\hat{T}}$ such that either (1) $\mathsf{z}_{\hat{T}}(X)$, $\mathsf{a}_{\hat{T}}(X)$, $\mathsf{b}_{\hat{T}}(X)$, and $\mathsf{c}_{\hat{T}}(X)$ are inconsistent with the commitments $[z_{ij}, a, b, c]_1$, or (2) $\mathsf{t}_{\hat{T}ijk}(X)$ (defined as in Eq. (6)) is not a polynomial.

[8] 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 $(view, 0)$.

```
$\mathcal{A}_{\mathsf{bind}}(\mathsf{ck})$
$(\mathbb{x}, T) \leftarrow \mathsf{Ext}^{\mathsf{kzg}}_{\mathsf{ss}}(\mathsf{ck})$;
for $\hat{T} \in \mathsf{SubTrees}_T$ do
    $\mathsf{s.tr}_{\hat{T}} \leftarrow \mathsf{Ext}^{\mathsf{sub}}_{\mathsf{ss}}(\mathsf{ck}, \hat{T})$; endfor    // Extracts polynomials from each subtree
for distinct $\hat{T} \neq \hat{T}' \in \mathsf{SubTrees}_T$ do
    $(\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)) \leftarrow \mathsf{W}_{\hat{T}}$; $(\mathsf{a}'(X), \mathsf{b}'(X), \mathsf{c}'(X)) \leftarrow \mathsf{W}_{\hat{T}'}$;
    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 $\perp$;
```

Fig. 12. The binding adversary $\mathcal{A}_{\mathsf{bind}}$ in Theorem 6.

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

Theorem 5 implies that if KZG is evaluation-binding and computationally special-sound (and n-TriRSDH holds in the case of Plonk), 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. 12.

In the following, suppose that neither $\mathsf{bad}_{\mathsf{sub}}$ or $\mathsf{bad}_{\mathsf{bind}}$ happened. Thus, $\mathsf{W}_{\hat{T}}(X) = \mathsf{W}_{\hat{T}'}(X)$ for any $\hat{T}, \hat{T} \in \mathsf{SubTrees}_T$. 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 $\hat{T}$ while $\mathsf{z}_{ij}(X)$ depends on β_i and γ_{ij}. Furthermore, each $\mathsf{t}_{ijk}(X) := \mathsf{F}_{ijk}(X)/\mathsf{Z}_{\mathbb{H}}(X)$ is a polynomial, where $\mathsf{F}_{ijk}(X) := \mathsf{F}_0(X) + \alpha_{ijk}\mathsf{F}_{1ij}(X) + \alpha^2_{ijk}\mathsf{F}_{2ij}(X)$, and $\mathsf{F}_0(X)$, $\mathsf{F}_{1ij}(X)$ and $\mathsf{F}_{2ij}(X)$ are defined as in Eq. (6) (but they may depend on β_i and γ_{ij}). But then $\mathsf{F}_0(X) + \alpha_{ijk}\mathsf{F}_{1ij}(X) + \alpha^2_{ijk}\mathsf{F}_{2ij}(X) = \mathsf{Z}_{\mathbb{H}}(X)\mathsf{t}_{ijk}(X)$. Thus, for every $s \in [1, n]$, $\mathsf{F}_0(\omega^s) + \alpha_{ijk}\mathsf{F}_{1ij}(\omega^s) + \alpha^2_{ijk}\mathsf{F}_{2ij}(\omega^s) = 0$. Let

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

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, $\mathsf{F}_0(\omega^s) = 0$ 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. (4). 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. (3)). 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. See Appendix B.3 for the proof.

Lemma 4. *Let σ be a permutation on $[1, n]$ and $a_1, \dots, a_n, b_1, \dots, b_n \in \mathbb{F}$. Let $\beta_1, \dots, \beta_{n+1} \in \mathbb{F}$ be mutually distinct and $\gamma_1, \dots, \gamma_{n+1} \in \mathbb{F}$ 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 $s \in [1, n]$.*

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 4.1. See Appendix B.4 for the proof.

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 $s \in [1, 3n]$.*

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

$$\mathsf{z}_{ij}(\omega^{s+1}) = \prod_{t=1}^{s} \frac{(w_t + \beta_i\omega^t + \gamma_{ij}) \cdot (w_{n+t} + \beta_i k_1\omega^t + \gamma_{ij}) \cdot (w_{2n+t} + \beta_i k_2\omega^t + \gamma_{ij})}{(w_t + \beta_i S_{\sigma 1}(\omega^t) + \gamma_{ij}) \cdot (w_{n+t} + \beta_i S_{\sigma 2}(\omega^t) + \gamma_{ij}) \cdot (w_{2n+t} + \beta_i \mathsf{S}_{\sigma 3}(\omega^t) + \gamma_{ij})} \ . \quad (14)$$

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. (6)), 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. (14) 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. (5). □

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, Heidelberg, November 2022. doi:10.1007/978-3-031-22318-1_5. 1, 1, 1, B.5, 7

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, B.5

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. 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. 2.1

BCG$^+$14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014. doi:10.1109/SP.2014.36. 1

BCPR14. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014. doi:10.1145/2591796.2591859. 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, Heidelberg, May 2020. doi:10.1007/978-3-030-45721-1_24. 1, 2.1

BMRS20. Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. Coda: Decentralized cryptocurrency at scale. Cryptology ePrint Archive, Report 2020/352, 2020. https://eprint.iacr.org/2020/352. 1

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994. doi:10.1007/3-540-48658-5_19. 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, Heidelberg, December 2021. doi:10.1007/978-3-030-92078-4_1. 1, 1, 3.1

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, Heidelberg, April 2024. doi:10.1007/978-3-031-57722-2_11. 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, Heidelberg, May 2020. doi:10.1007/978-3-030-45721-1_26. 1, 1, 1, 2.1, 3.1, 3.2

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, 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, Heidelberg, April 2023. doi:10.1007/978-3-031-30617-4_18. 1, 1

DL08. Giovanni Di Crescenzo and Helger Lipmaa. Succinct NP Proofs from an Extractability Assumption. In Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe, editors, *Computability in Europe, CIE 2008*, volume 5028 of *LNCS*, pages 175–185, Athens, Greece, June 15–20, 2008. Springer, Heidelberg. 1

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

FFR24. Antonio Faonio, Dario Fiore, and Luigi Russo. Real-world Universal zk-SNARKs are non-malleable. Technical Report 2024/721, IACR, May 11 2024. URL: https://eprint.iacr.org/2024/721. 1, A.3

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96881-0_2. 1, 1, 2.1

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_12. 1

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_19. 1

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_11. 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. https://eprint.iacr.org/2019/953. 1, 3, 1, 4, 1, 2, 3.1, 4, 4.1, 4.1, B.1, D

Kil94. Joe Kilian. On the complexity of bounded-interaction and noninteractive zero-knowledge proofs. In *35th FOCS*, pages 466–477. IEEE Computer Society Press, November 1994. doi:10.1109/SFCS.1994.365744. 1

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_11. 1, 2.1, 2.1

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, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_10. 1, 2.1, C.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, Heidelberg, November / December 2023. doi:10.1007/978-3-031-48624-1_14. 1, 1, 1, 1, 2.1, 3.2, 6, C, C.1, C.1

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, Heidelberg, May 2024. doi:10.1007/978-3-031-58751-1_2. 1, 1, 1, 1, 2, 2.1, 2.1, 3, 3.1

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, Heidelberg, December 2022. doi:10.1007/978-3-031-22966-4_9. 1, 1, 3.1

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, Heidelberg, December 2005. 1

Mic94. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994. doi:10.1109/SFCS.1994.365746. 1

Nao03. Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_6. 1

Pas16. Rafael Pass. Unprovable Security of Perfect NIZK and Non-interactive Non-malleable Commitments. *Computational Complexity*, 25(3):607–666, 2016. 1

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, 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, Heidelberg. doi:10.1007/978-3-030-84242-0_27. 1, 1, 3.1

Sef24. Marek Sefranek. How (not) to simulate PLONK. Technical Report 2024/848, IACR, May 31, 2024. URL: https://eprint.iacr.org/2024/848. D, 9, D

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, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18. 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, Heidelberg, August 2022. doi:10.1007/978-3-031-15982-4_3. 1

A Postponed Material from Section 3

A.1 Comparison of Lin And Competitors

We compare SanLin to related approaches in Table 3.

Table 3. Comparison of different arguments to test $\mathsf{a}(X)\mathsf{b}(X) - \mathsf{c}(X)$. KS stands for the knowledge-soundness and SS for the special-soundness. The number of bits is given for the BLS381-12 curve.

Method	$\lvert\pi\rvert$ (bits)	KS in AGM	KS and SS in plain-model
Opening $[a, b, c]_1$ separately	$3\lvert\mathbb{F}\rvert + 3\lvert\mathbb{G}_\iota\rvert$ (1920)	✓	✓
Batch-opening $[a, b, c]_1$ together	$3\lvert\mathbb{F}\rvert + \lvert\mathbb{G}_\iota\rvert$ (1152)	✓	✓
Lin	$\lvert\mathbb{F}\rvert + \lvert\mathbb{G}_\iota\rvert$ (640)	✓	✗
SanLin (the current paper)	$2\lvert\mathbb{F}\rvert + \lvert\mathbb{G}_\iota\rvert$ (896)	✓	✓

A.2 Attacks Against Special-Soundness of Lin

For the following examples, see the description of Lin in Fig. 4.

*Example 1 (*Lin *is not knowledge-sound).* The knowledge-soundness adversary $\mathcal{A}$ chooses any $\bar{a} \in \mathbb{F}$, and samples $[q]_1 \leftarrow_{\$} \mathbb{G}_1$ obliviously. $\mathcal{A}$ sets $\mathsf{a}(X) \leftarrow \bar{a}$, $[b]_1 \leftarrow [q]_1$, and $[c]_1 \leftarrow \bar{a}[b]_1$. Since $\mathsf{a}(X)$ is a constant function, $\mathsf{a}(\mathfrak{z}) = \bar{a}$. Then, $[\Lambda(x)]_1 = [\bar{a}\mathsf{b}(x) - \mathsf{c}(x)]_1 = [0]_1$ and $[h]_1 = [0]_1$ and the adversary succeeds in cheating. However, $\mathcal{A}$ does not know $\mathsf{b}(X)$ and $\mathsf{c}(X)$ as polynomials.

One may ask if knowledge-soundness can be achieved by checking that $[h]_1 \neq [0]_1$. However, this is not always sufficient.

Example 2. Assume $\mathcal{A}$ knows $\mathfrak{z}$ (e.g., $\mathfrak{z}$ has low entropy and $\mathcal{A}$ can guess $\mathcal{A}$ with a non-negligible probability). $\mathcal{A}$ sets $\mathsf{a}(X) \leftarrow \bar{a}$ for any $\bar{a} \in \mathbb{F}$ and picks any polynomials $\mathsf{b}'(X)$ and $\mathsf{c}'(X)$ such that

$$\bar{a}\mathsf{b}'(X) - \mathsf{c}'(X) = q(X)(X - \mathfrak{z}) \ ,$$

where $q(X)$ is a non-zero quotient polynomial. Then, $\mathcal{A}$ samples obliviously $[r]_1$ and sets $[a]_1 \leftarrow [\bar{a}]_1$, $[b]_1 \leftarrow [\mathsf{b}'(x) + r]_1$, $[c]_1 \leftarrow [\mathsf{c}'(x) + \bar{a}r]_1$. For the verifier to accept, the adversary sets (see the definition of $[h]_1$ in Fig. 4)

$$\begin{aligned}[h]_1 &\leftarrow [v(\bar{a}b - c)/(x - \mathfrak{z})]_1 = v[(\bar{a}(\mathsf{b}'(x) + r) - (\mathsf{c}'(x) + \bar{a}r))/(x - \mathfrak{z})]_1 \\ &= v[(q(x)(x - \mathfrak{z}))/(x - \mathfrak{z})]_1 = v[q(x)]_1 \ .\end{aligned}$$

The latter is non-zero when v and $q(x)$ are non-zero. However, $\mathcal{A}$ cannot open $[b]_1$ and $[c]_1$.

A.3 Attacks Against Special-Soundness of LinGen

Example 3. The adversary $\mathcal{A}$ sets each $\mathsf{a}_s(X)$ to be equal to an arbitrary field element $\bar{a}_s^* \in \mathbb{F}$. Let $\bar{\boldsymbol{b}}^*$ be any vector orthogonal to $\bar{\boldsymbol{a}}^*$, $\sum \bar{a}_s^* \bar{b}_s^* = 0$. $\mathcal{A}$ samples $[b]_1 \leftarrow_\$ \mathbb{G}_1$ obliviously, and then sets $[b_s]_1 \leftarrow \bar{b}_s^*[b]_1$. $\mathcal{A}$ sets $\bar{a}_s \leftarrow \bar{a}_s^*$ and $[h]_1 \leftarrow [0]_1$. Clearly,

$$\sum_s ([a_s]_1 \bullet [b_s]_1) = \sum_s \left[\bar{a}_s^* \bar{b}_s^* b\right]_T = \left(\sum_s \bar{a}_s^* \bar{b}_s^*\right) [b]_T = [0]_T \enspace ,$$

and thus

$$\left[\sum_{s=1}^{m} \beta^{s-1}(a_s - \bar{a}_s) + \beta^m \sum_{s=1}^{m} \bar{a}_s b_s\right]_1 \bullet [1]_2 = \left[\sum_{s=1}^{m} \beta^{s-1} 0 + \beta^m \cdot 0\right]_1 = [0]_T \enspace .$$

Thus, the LinGen verifier accepts. However, $\mathcal{A}$ does not know how to open $[b]_1$ and thus any of $[b_s]_1$.

Faonio et al. [FFR24] (a concurrent work) independently found and described a more general version of this attack. They showed that LinGen is knowledge-sound exactly if the "left polynomials" $\mathsf{a}_s(X)$ are linearly independent.

A.4 Proof of Theorem 3 (Special-Soundness of SanLin)

We start again with a tree extractor lemma.

Lemma 7. *Let $T = (\mathsf{tr}_{ij})$ be an $(2n+1, 3)$-tree of* Batch*'s accepting transcripts, where tr_{ij} are as in Fig. 13. The DPT algorithm $\mathsf{TE}_{\mathsf{sanlin}}(\mathsf{ck}, T)$ in Fig. 13 computes a tuple of accepting KZG transcripts $(\mathbf{k.tr}'_j)_{j=1}^3$, such that $\mathsf{k.tr}'_{i1} = ([a]_1, \mathfrak{z}_i, \ldots)$, $\mathsf{k.tr}'_{i2} = ([c]_1, \mathfrak{z}_i, \ldots)$, and $\mathsf{k.tr}'_{i3} = ([b]_1, \mathfrak{z}_i, \ldots)$, with mutually different $\mathfrak{z}_i$ for $i \in [1, 2n+1]$.*

Proof (Lemma 7). Let T be the given accepting tree of transcripts and

$$\boldsymbol{V}_i := \begin{pmatrix} 1 & v_{i1} & v_{i1}^2 \\ 1 & v_{i2} & v_{i3}^2 \\ 1 & v_{i3} & v_{i3}^2 \end{pmatrix} \tag{15}$$

be a Vandermonde matrix. Given 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 $i \in [1, 2n+1]$.

Define $(h'_{i1}, h^*_{i2}, h'_{i3})$ as in (*) in Fig. 13. Since the SanLin verifier accepts tr_{ij} for each $i \in [1, 2n+1]$ and $j \in [1, 3]$, the KZG verifier accepts each $\mathsf{k.tr}_{ij} :=$

```
$\mathsf{TE}_{\mathsf{sanlin}}(\mathsf{ck}, T)$
Parse $T = (\mathsf{tr}_{ij})_{i \in [1,2n+1], j \in [1,3]}$;   // $\mathsf{tr}_{ij} = ([a,b,c]_1, \mathfrak{z}_i, \bar{a}_i, \bar{b}_i, v_i, [h_{ij}]_1)$
for $i \in [1, 2n+1]$ do
    Parse $\mathsf{tr}_{ij} = ([a,b,c]_1, \mathfrak{z}_i, \bar{a}_i, \bar{b}_i, v_{ij}, [h_{ij}]_1)$;
    $[h'_{i1}, h^*_{i2}, h'_{i3}]_1^\intercal \leftarrow \boldsymbol{V}_i^{-1} [h_{i1}, h_{i2}, h_{i3}]_1^\intercal$;   $(*)$   // See $\boldsymbol{V}_i$ in Eq. (15)
    $[h'_{i2}]_1 \leftarrow \bar{a}_i [h'_{i3}]_1 - [h^*_{i2}]_1$;
    $\mathsf{k.tr}'_{i1} \leftarrow ([a]_1, \mathfrak{z}_i, \bar{a}_i, [h'_{i1}]_1)$; $\mathsf{k.tr}'_{i2} \leftarrow ([c]_1, \mathfrak{z}_i, \bar{a}_i \bar{b}_i, [h'_{i2}]_1)$;
    $\mathsf{k.tr}'_{i3} \leftarrow ([b]_1, \mathfrak{z}_i, \bar{b}_i, [h'_{i3}]_1)$;
endfor
for $j \in \{1,2,3\}$ do $\mathbf{k.tr}'_j \leftarrow (\mathsf{k.tr}'_{1j}, \ldots, \mathsf{k.tr}'_{n+1,j})$; endfor
return $(\mathbf{k.tr}'_1, \mathbf{k.tr}'_2, \mathbf{k.tr}'_3)$;
```

Fig. 13. The subroutine $\mathsf{TE}_{\mathsf{sanlin}}$

$([\varphi_{ij}]_1, \mathfrak{z}_i, \Phi_{ij}, [h_{ij}]_1)$, where $\varphi_{ij} := \mathsf{a} + v_{ij}\Lambda_i + v_{ij}^2 \mathsf{b}$ for $\Lambda_i = \bar{a}_i \mathsf{b} - \mathsf{c}$, $\Phi_{ij} := \bar{a}_i + v_{ij}^2 \bar{b}_i$, and $h_{ij} = h'_{i1} + v_{ij} h^*_{i2} + v_{ij}^2 h'_{i3}$. Clearly,

$$\begin{bmatrix} \mathsf{a} \\ \Lambda_i \\ \mathsf{b} \end{bmatrix}_1 = \boldsymbol{V}_i^{-1} \begin{bmatrix} \varphi_{i1} \\ \varphi_{i2} \\ \varphi_{i3} \end{bmatrix}_1 , \quad \begin{pmatrix} \bar{a}_i \\ 0 \\ \bar{b}_i \end{pmatrix} = \boldsymbol{V}_i^{-1} \begin{bmatrix} \Phi_{i1} \\ \Phi_{i2} \\ \Phi_{i3} \end{bmatrix}_1 .$$

Since KZG is triply homomorphic,

$$\begin{pmatrix} \mathsf{k.tr}'_{i1} \\ \mathsf{k.tr}^*_{i2} \\ \mathsf{k.tr}'_{i3} \end{pmatrix} := \begin{pmatrix} ([a]_1, \mathfrak{z}_i, \bar{a}_i, [h'_{i1}]_1) \\ ([\Lambda_i]_1, \mathfrak{z}_i, 0, [h^*_{i2}]_1) \\ ([b]_1, \mathfrak{z}_i, \bar{b}_i, [h'_{i3}]_1) \end{pmatrix}$$

are accepting KZG transcripts. Since $\mathsf{c} = \bar{a}_i \mathsf{b} - \Lambda_i$, by triple homomorphism, $\mathsf{k.tr}'_{i2} := ([c]_1, \mathfrak{z}_i, \bar{a}_i \bar{b}_i, [h'_{i2}]_1)$ is an accepting KZG transcript, where $[h'_{i2}]_1 := \bar{a}_i [h'_{i3}]_1 - [h^*_{i2}]_1$. Thus, $\mathsf{TE}_{\mathsf{sanlin}}$ returns accepting KZG transcripts $\mathbf{k.tr}'_1$, $\mathbf{k.tr}'_2$, and $\mathbf{k.tr}'_3$ of the claimed form. □

Proof (Theorem 3). Let $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$ be the promised $(n+1)$-special-soundness extractor of KZG and let $\mathcal{A}_{\mathsf{ss}}^{\mathsf{lin}}$ be any SanLin $\boldsymbol{\kappa}$-special-soundness adversary. In Fig. 14, we depict a $\boldsymbol{\kappa}$-special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$ for SanLin and an $(n+1)$-special-soundness adversary $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ for KZG. Here, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$ has an oracle access to $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$ and $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ has an oracle access to $\mathcal{A}_{\mathsf{ss}}^{\mathsf{lin}}$ and $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$.

$\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$ inputs ck and a $\boldsymbol{\kappa}$-tree $T = (\mathsf{tr}_{ij})_{i \in [1,2n+1], j \in [1,3]}$ of accepting SanLin transcripts, where tr_{ij} is defined as in Fig. 13. $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$ calls the (deterministic) algorithm $\mathsf{TE}_{\mathsf{sanlin}}$ (see Fig. 13) to compute three valid transcripts $\mathbf{k.tr}'_1 = ([a]_1, \mathfrak{z}_i, \ldots)$, $\mathbf{k.tr}'_2 = ([c]_1, \mathfrak{z}_i, \ldots)$, and $\mathbf{k.tr}'_3 = ([b]_1, \mathfrak{z}_i, \ldots)$ for three polynomials that have to be extracted. Then, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$ calls three times the $(n+1)$-special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$ to compute the witness $(\mathsf{a}^*(X), \mathsf{c}^*(X), \mathsf{b}^*(X))$.

Let us bound the advantage of SanLin's adversary $\mathcal{A}_{\mathsf{ss}}^{\mathsf{lin}}$ against the extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$. Assume that T is a $\boldsymbol{\kappa}$-tree of SanLin transcripts, each accepted by the SanLin verifier. Let us consider the following two events. The event $\mathsf{bad}_{\mathsf{ext}}$ happens when $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$ computes $\mathsf{a}^*(X)$, $\mathsf{b}^*(X)$, $\mathsf{c}^*(X)$ such that $([a]_1, \mathsf{a}^*(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{k.tr}'_1}$

```
$\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}(\mathsf{ck}, T)$
$(\mathbf{k.tr}'_j)_{j=1}^3 \leftarrow \mathsf{TE}_{\mathsf{sanlin}}(\mathsf{ck}, T)$;
$\mathsf{a}^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}'_1)$;
$\mathsf{c}^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}'_2)$;
$\mathsf{b}^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}'_3)$;
return $(\mathsf{a}^*(X), \mathsf{b}^*(X), \mathsf{c}^*(X))$;
```

```
$\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck})$
$T \leftarrow \mathcal{A}_{\mathsf{ss}}^{\mathsf{lin}}(\mathsf{ck})$; $(\mathbf{k.tr}'_j)_{j=1}^3 \leftarrow \mathsf{TE}_{\mathsf{sanlin}}(\mathsf{ck}, T)$;
$\mathsf{a}^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}'_1)$;
if $([a]_1, \mathsf{a}^*(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{k.tr}'_1}$ then
   return $\mathbf{k.tr}'_1$; fi
$\mathsf{c}^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}'_2)$;
if $([c]_1, \mathsf{c}^*(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{k.tr}'_2}$ then
   return $\mathbf{k.tr}'_2$; fi
$\mathsf{b}^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}'_3)$;
if $([b]_1, \mathsf{b}^*(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{k.tr}'_3}$ then
   return $\mathbf{k.tr}'_3$; fi
return $\perp$;
```

```
$\mathcal{C}_{\mathsf{evb}}(\mathsf{ck})$
$T \leftarrow \mathcal{A}_{\mathsf{ss}}^{\mathsf{lin}}(\mathsf{ck})$; $(\mathbf{k.tr}'_j)_{j=1}^3 \leftarrow \mathsf{TE}_{\mathsf{sanlin}}(\mathsf{ck}, T)$;
$\mathsf{a}^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathtt{srs}, \mathbf{k.tr}'_1)$; $\mathsf{c}^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}'_2)$; $\mathsf{b}^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{k.tr}'_3)$;
for $i \in [n+2, 2n+1]$ do   // Using the notation from Fig. 13
   if $\bar{a}_i \neq \mathsf{a}^*(\mathfrak{z}_i)$ then
      return $\left([a]_1, \mathfrak{z}_i, \bar{a}_i, [h'_{i1}]_1, \mathsf{a}^*(\mathfrak{z}_i), [(\mathsf{a}^*(x) - \mathsf{a}^*(\mathfrak{z}_i))/(x - \mathfrak{z}_i)]_1\right)$; fi
   if $\bar{b}_i \neq \mathsf{b}^*(\mathfrak{z}_i)$ then
      return $\left([b]_1, \mathfrak{z}_i, \bar{b}_i, [h'_{i3}]_1, \mathsf{b}^*(\mathfrak{z}_i), [(\mathsf{b}^*(x) - \mathsf{b}^*(\mathfrak{z}_i))/(x - \mathfrak{z}_i)]_1\right)$; fi
   if $\bar{a}_i\bar{b}_i \neq \mathsf{c}^*(\mathfrak{z}_i)$ then
      return $\left([c]_1, \mathfrak{z}_i, \bar{a}_i\bar{b}_i, [h'_{i2}]_1, \mathsf{c}^*(\mathfrak{z}_i), [(\mathsf{c}^*(x) - \mathsf{c}^*(\mathfrak{z}_i))/(x - \mathfrak{z}_i)]_1\right)$; fi
endfor
return $\perp$;
```

Fig. 14. The extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$, the KZG special-soundness adversary $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$, and the evaluation-binding adversary $\mathcal{C}_{\mathsf{evb}}$.

or $([c]_1, \mathsf{c}^*(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{k.tr}'_2}$ or $([b]_1, \mathsf{b}^*(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{k.tr}'_3}$. The event $\mathsf{bad}_{\mathsf{evb}}$ happens when the event $\mathsf{bad}_{\mathsf{ext}}$ did not happen, but for some $i > n+1$, either $\mathsf{a}^*(\mathfrak{z}_i) \neq \bar{a}_i$, $\mathsf{b}^*(\mathfrak{z}_i) \neq \bar{b}_i$, or $\mathsf{c}^*(\mathfrak{z}_i) \neq \bar{a}_i\bar{b}_i$.

If neither of the events happens, then $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$ has extracted polynomials $\mathsf{a}^*(X)$, $\mathsf{b}^*(X)$, $\mathsf{c}^*(X)$ of degree at most n, which satisfy $\mathsf{a} = \mathsf{a}^*(x)$, $\mathsf{b} = \mathsf{b}^*(x)$, $\mathsf{c} = \mathsf{c}^*(x)$. Furthermore, $\mathsf{f}(X) := \mathsf{a}^*(X)\mathsf{b}^*(X) - \mathsf{c}^*(X)$ is at most degree $2n$ polynomial, which satisfies $\mathsf{f}(\mathfrak{z}_i) = \bar{a}_i\bar{b}_i - \bar{a}_i\bar{b}_i = 0$ for $i \in [1, 2n+1]$. Thus, $\mathsf{f}(X) \equiv 0$ and $\mathsf{a}^*(X)\mathsf{b}^*(X) = \mathsf{c}^*(X)$, which is what we needed to show.

To bound $\Pr[\mathsf{bad}_{\mathsf{ext}}]$ and $\Pr[\mathsf{bad}_{\mathsf{evb}}]$, we construct adversaries $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ and $\mathcal{C}_{\mathsf{evb}}$ described in Fig. 14. The KZG $(n+1)$-special-soundness adversary $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ follows the structure of $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$, but when the extraction of one of the polynomials fails (which can be deterministically tested), $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ outputs the respective transcript vector $\mathbf{k.tr}'_j$ for $j \in \{1, 2, 3\}$. $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ wins the $(n+1)$-special-soundness game exactly when $\mathsf{bad}_{\mathsf{evb}}$ happens. Thus,

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

Finally, we construct the evaluation-binding adversary $\mathcal{C}_{\mathsf{evb}}$. Observe that when $\mathsf{bad}_{\mathsf{ext}}$ does not happen, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$ knows $\mathsf{a}^*(X)$, which satisfies $\mathsf{a} = \mathsf{a}^*(x)$. This means, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}}$ knows how to compute opening of the commitment $[a]_1$ at any point $\mathfrak{z}_i$ for $i > n+1$. Since

$$h(X) := \frac{\mathsf{a}^*(X) - \mathsf{a}^*(\mathfrak{z}_i)}{X - \mathfrak{z}_i}$$

is a polynomial, $\mathcal{C}_{\mathsf{evb}}$ can compute an opening proof $[h(x)]_1$ for the evaluation $\mathsf{a}^*(\mathfrak{z}_i)$. In addition to this opening, we know from the construction of $\mathsf{TE}_{\mathsf{sanlin}}$ that $\mathsf{k.tr}'_{i1} = ([a]_1, \mathfrak{z}_i, \bar{a}_i, [h'_{i1}]_1)$ is an accepting KZG transcript. If $\bar{a}_i \neq \mathsf{a}^*(\mathfrak{z}_i)$, we have found a collision which breaks the evaluation-binding. With the same reasoning, the conditions $V_i \neq \mathsf{b}^*(\mathfrak{z}_i)$ and $\bar{a}_i\bar{b}_i \neq \mathsf{c}^*(\mathfrak{z}_i)$ will lead to breaking the evaluation-binding. This is precisely the strategy that $\mathcal{C}_{\mathsf{evb}}$ follows in Fig. 14. Thus,

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

Thus, we have shown that

$$\begin{aligned}\mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen},\mathsf{SanLin},\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{sanlin}},\boldsymbol{\kappa},\mathcal{A}_{\mathsf{ss}}^{\mathsf{lin}}}(\lambda) &= \Pr[\mathsf{bad}_{\mathsf{ext}}] + \Pr[\mathsf{bad}_{\mathsf{evb}}] \\ &= \mathsf{Adv}^{\mathrm{ss}}_{\mathsf{Pgen},\mathsf{KZG},\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}},n+1,\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}}(\lambda) + \\ &\quad \mathsf{Adv}^{\mathrm{evb}}_{\mathsf{Pgen},\mathsf{KZG},n,\mathcal{C}_{\mathsf{evb}}}(\lambda) \ .\end{aligned}$$

□

A.5 Proof of Theorem 4 (Special-Soundness of SanLinGen)

Before proving Theorem 4, we state the following lemma.

Lemma 8. *Assume that $T = (\mathsf{tr}_{ijk})$ is a $(2n+1, m, m+2)$-tree of* SanLinGen*'s accepting transcripts, where tr_{ijk} are as in step 17 in Fig. 15. Then the PPT algorithm $\mathsf{TE}_{\mathsf{lingen}}(\mathsf{ck}, T)$ in Fig. 15 computes accepting KZG transcripts $(\mathbf{tr}_{\boldsymbol{a_s}}, \mathbf{tr}_{\boldsymbol{b_s}})_{s=1}^m$, such that $\mathsf{tr}_{a_s,i} = ([a_s]_1, \mathfrak{z}_i, \ldots)$ and $\mathsf{tr}_{b_s,i} = ([b_s]_1, \mathfrak{z}_i, \ldots)$ for $i \in [1, n+1]$, and $\mathfrak{z}_i$ are mutually distinct.*

Proof. Let T be the given accepting tree of transcripts. Let

$$\boldsymbol{B}_{ij} = \begin{pmatrix} 1 & \beta_{ij1} & \cdots & \beta_{ij1}^{m+1} \\ 1 & \beta_{ij2} & \cdots & \beta_{ij2}^{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \beta_{ij,m+2} & \cdots & \beta_{ij,m+2}^{m+1} \end{pmatrix} \quad \text{and} \quad \boldsymbol{C}_i = \begin{pmatrix} 1 & \gamma_{i1} & \gamma_{i1}^2 & \vdots & \gamma_{i1}^{m-1} \\ 1 & \gamma_{i2} & \gamma_{i2}^2 & \vdots & \gamma_{i2}^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma_{im} & \gamma_{im}^2 & \vdots & \gamma_{im}^{m-1} \end{pmatrix} \qquad (16)$$

be Vandemonde matrices. Since T is a tree of transcripts, $\mathfrak{z}_i$ are all distinct, and for each $\mathfrak{z}_i$, all γ_{ij} are disctinct and for each γ_{ij} all β_{ijk} are distinct. Thus, for each $i \in [1, 2n+1]$ and $j \in [1, m]$, $\boldsymbol{B}_{ij}$ and $\boldsymbol{C}_i$ are non-singular.

By the construction of $\mathsf{SanLinGen}$, tr_{ijk} is an accepting $\mathsf{SanLinGen}$ transcript iff

$$\mathsf{k.tr}_{ijk} = ([\varphi_{ijk}]_1, \mathfrak{z}_i, \Phi_{ijk}, [h_{ijk}]_1)$$

```
$\mathsf{TE}_{\mathsf{lingen}}(\mathsf{ck}, T)$
1 : Parse $T = (\mathsf{tr}_{ijk})_{i\in[1,2n+1],j\in[1,m],k\in[1,m+2]}$;
2 : Parse $\mathsf{tr}_{ijk} = ([(a_s, b_s)_{s=1}^m]_1, \mathfrak{z}_i, (\bar{a}_{is})_{s=1}^m, \gamma_{ij}, \bar{b}_{ij}, \beta_{ijk}, [h_{ijk}]_1)$;
3 : for $(i, j) \in [1, 2n+1] \times [1, m]$ do
4 :     $[h^*_{ij1}, \dots, h^*_{ij,m+2}]_1^\intercal \leftarrow \boldsymbol{B}_{ij}^{-1}[h_{ij1}, \dots, h_{ij,m+2}]_1^\intercal$;
5 :     $\begin{pmatrix} \mathsf{k.tr}^*_{ij1} \\ \vdots \\ \mathsf{k.tr}^*_{ijm} \\ \mathsf{k.tr}^*_{ij,m+1} \\ \mathsf{k.tr}^*_{ij,m+2} \end{pmatrix} \leftarrow \begin{pmatrix} ([a_1]_1, \mathfrak{z}_i, \bar{a}_{i1}, [h^*_{ij1}]_1) \\ \vdots \\ ([a_m]_1, \mathfrak{z}_i, \bar{a}_{im}, [h^*_{ijm}]_1) \\ ([\sum_{s=1}^m \bar{a}_{is} b_s]_1, \mathfrak{z}_i, 0, [h^*_{ij,m+1}]_1) \\ ([\sum_{s=1}^m \gamma_{ij}^{s-1} b_s]_1, \mathfrak{z}_i, \bar{b}_{ij}, [h^*_{ij,m+2}]_1) \end{pmatrix}$;
6 : endfor
7 : for $s \in [1, m]$ do
8 :     for $i \in [1, 2n+1]$ do
9 :         $\begin{pmatrix} \bar{b}'_{i1} \\ \cdots \\ \bar{b}'_{im} \end{pmatrix} \leftarrow \boldsymbol{C}_i^{-1} \begin{pmatrix} \bar{b}_{i1} \\ \cdots \\ \bar{b}_{im} \end{pmatrix}$; $\begin{bmatrix} h'_{i1} \\ \cdots \\ h'_{im} \end{bmatrix}_1 \leftarrow \boldsymbol{C}_i^{-1} \begin{bmatrix} h^*_{i1,m+2} \\ \cdots \\ h^*_{im,m+2} \end{bmatrix}_1$;
10 :        $\mathsf{k.tr}_{a_s,i} \leftarrow \mathsf{k.tr}^*_{i1s}$; $\mathsf{k.tr}_{b_s,i} \leftarrow ([b_s]_1, \mathfrak{z}_i, \bar{b}'_{is}, [h'_{is}]_1)$;
11 :    endfor
12 :    $\mathbf{tr}_{a_s} \leftarrow (\mathsf{k.tr}_{a_s,1}, \dots, \mathsf{k.tr}_{a_s,n+1})$; $\mathbf{tr}_{b_s} \leftarrow (\mathsf{k.tr}_{b_s,1}, \dots, \mathsf{k.tr}_{b_s,n+1})$;
13 : endfor
14 : return $(\mathbf{tr}_{a_s}, \mathbf{tr}_{b_s})_{s=1}^m$; ./
```

Fig. 15. The $\mathsf{TE}_{\mathsf{lingen}}$ subroutine.

is an accepting KZG transcript, where

$$\varphi_{ijk} := \sum_{s=1}^m \beta_{ijk}^{s-1} a_s + \beta_{ijk}^m \sum_{s=1}^m \bar{a}_{is} b_s + \beta_{ijk}^{m+1} \sum_{s=1}^m \gamma_{ij}^{s-1} b_s$$

and

$$\varPhi_{ijk} := \sum_{s=1}^m \beta_{ijk}^{s-1} \bar{a}_{is} + \beta_{ijk}^{m+1} \bar{b}_{ij} \quad .$$

That is,

$$(\varphi_{ij1}, \dots, \varphi_{ij,m+2})^\intercal = \boldsymbol{B}_{ij} \cdot (a_1, \dots, a_m, \sum_{s=1}^m \bar{a}_{is} b_s, \sum_{s=1}^m \gamma_{ij}^{s-1} b_s)^\intercal \quad ,$$
$$(\varPhi_{ij1}, \dots, \varPhi_{ij,m+2})^\intercal = \boldsymbol{B}_{ij} \cdot (\bar{a}_{i1}, \dots, \bar{a}_{im}, 0, \bar{b}_{ij})^\intercal \quad .$$

Let $h^*_{ijk}, \dots, h^*_{ij,m+2}$ be defined as in step 4. Then,

$$(a_1, \dots, a_m, \sum_{s=1}^m \bar{a}_{is} b_s, \sum_{s=1}^m \gamma_{ij}^{s-1} b_s)^\intercal = \boldsymbol{B}_{ij}^{-1} \cdot (\varphi_{ij1}, \dots, \varphi_{ij,m+2})^\intercal \quad ,$$
$$(\bar{a}_{i1}, \dots, \bar{a}_{im}, 0, \bar{b}_{ij})^\intercal = \boldsymbol{B}_{ij}^{-1} \cdot (\varPhi_{ij1}, \dots, \varPhi_{ij,m+2})^\intercal \quad .$$

Since KZG is triply homomorphic, for all i, j, k, $\mathsf{k.tr}^*_{ijk}$ (see step 5) are accepting KZG transcripts. Thus, one can define $\mathsf{tr}_{a_s,i} = \mathsf{k.tr}^*_{ijs}$ for $j = 1$ (one can choose any value of j).

Finally, one can apply $\boldsymbol{C}_i^{-1}$ to obtain $(\bar{b}'_{i1}, \dots, \bar{b}'_{im})^\intercal \leftarrow \boldsymbol{C}_i^{-1}(\bar{b}_{i1}, \dots, \bar{b}_{im})^\intercal$ and $(h'_{i1}, \dots, h'_{im})^\intercal \leftarrow \boldsymbol{C}_i^{-1}(h^*_{i1,m+2}, \dots, h^*_{im,m+2})^\intercal$. Since KZG is triple homomorphic and the first elements of $\mathsf{k.tr}^*_{ij,m+2}$ result from $\boldsymbol{C}_i(v_1, \dots, v_m)^\intercal$, $\mathsf{k.tr}_{b_s i} \leftarrow ([b_s]_1, \mathfrak{z}_i, \bar{b}'_{is}, [h'_{is}]_1)$ is an accepting KZG transcript. □

```
$\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{SanLinGen}}(\mathsf{ck}, T)$
$(\mathbf{tr}_{a_s}, \mathbf{tr}_{b_s})_{s=1}^{m} \leftarrow \mathsf{TE}_{\mathsf{lingen}}(\mathsf{ck}, T)$;
for $s \in [1, m]$ do
    $\mathsf{a}_s^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{tr}_{a_s})$;
    $\mathsf{b}_s^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{tr}_{b_s})$;
endfor
return $(\mathsf{a}_s^*(X), \mathsf{b}_s^*(X))_{s=1}^{m}$;
```

```
$\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck})$
$T \leftarrow \mathcal{A}(\mathsf{ck})$;
$(\mathbf{tr}_{a_s}, \mathbf{tr}_{b_s})_{s=1}^{m} \leftarrow \mathsf{TE}_{\mathsf{lingen}}(\mathsf{ck}, T)$;
for $s \in [1, m]$ do
    $\mathsf{a}_s^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{tr}_{a_s})$;
    if $([a_s]_1, \mathsf{a}_s^*(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{tr}_{a_s}}$ then return $\mathbf{tr}_{a_s}$;
    $\mathsf{b}_s^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{tr}_{b_s})$;
    if $([b_s]_1, \mathsf{b}_s^*(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{tr}_{b_s}}$ then return $\mathbf{tr}_{a_s}$;
return $\perp$;
```

```
$\mathcal{C}_{\mathsf{evb}}(\mathsf{ck})$
$T \leftarrow \mathcal{A}(\mathsf{ck})$; $(\mathbf{tr}_{a_s}, \mathbf{tr}_{b_s})_{s=1}^{m} \leftarrow \mathsf{TE}_{\mathsf{lingen}}(\mathsf{ck}, T)$;
for $s \in [1, m]$ do $\mathsf{a}_s^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{tr}_{a_s})$; $\mathsf{b}_s^*(X) \leftarrow \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{tr}_{b_s})$; endfor
Parse $(\mathbf{tr}_{a_s}, \mathbf{tr}_{b_s})_{s=1}^{m}$ as in Fig. 15;
for $(s, i) \in [1, m] \times [n+2, 2n+1]$ do
    if $\bar{a}_{is} \neq \mathsf{a}_s^*(\mathfrak{z}_i)$ then return $\left([a_s]_1, \mathfrak{z}_i, \bar{a}_{is}, [h_{jki}^*]_1, \mathsf{a}_s^*(\mathfrak{z}_i), \left[\frac{\mathsf{a}_s^*(x) - \mathsf{a}_s^*(\mathfrak{z}_i)}{x - \mathfrak{z}_i}\right]_1\right)$;
    if $\bar{b}_{is}' \neq \mathsf{b}_s^*(\mathfrak{z}_i)$ then return $\left([b_s]_1, \mathfrak{z}_i, \bar{b}_{is}', [h_{is}']_1, \mathsf{b}_s^*(\mathfrak{z}_i), \left[\frac{\mathsf{b}_s^*(x) - \mathsf{b}_s^*(\mathfrak{z}_i)}{x - \mathfrak{z}_i}\right]_1\right)$;
endfor
for $i \in [1, 2n+1]$ do
    if $\sum_{s=1}^{m} \bar{a}_{is} \bar{b}_{is}' \neq 0$
        then return $\left([0]_1, \mathfrak{z}_i, \sum_{s=1}^{m} \bar{a}_{is} \bar{b}_{is}', [\sum_{s=1}^{m} \bar{a}_{is} h_{is}' - h_{i1,m+1}^*]_1, 0, [0]_1\right)$;
return $\perp$;
```

Fig. 16. The extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{SanLinGen}}$, the KZG special soundness-adversary $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$, and the KZG evaluation-binding adversary $\mathcal{C}_{\mathsf{evb}}$.

Next, we prove Theorem 4.

Proof (Theorem 4). Recall that a valid witness contains $2m$ polynomials $(\mathsf{a}_s^*(X), \mathsf{b}_s^*(X))_{s=1}^{m}$ of degree at most n, such that $\sum_{s=1}^{m} \mathsf{a}_s^*(X)\mathsf{b}_s^*(X) \equiv 0$ and $\mathsf{a}_s^*(x) = a_s$, $\mathsf{b}_s^*(x) = b_s$ for all $s \in [1, m]$.

Let $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$ be a $(n+1)$-special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$ of KZG. We depict the $(2n+1, m, m+2)$-special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{SanLinGen}}$ for $\mathsf{SanLinGen}$ in Fig. 16. It has blackbox access to $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$. On input a $(2n+1, m, m+2)$-tree of $\mathsf{SanLinGen}$ accepting transcripts, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{SanLinGen}}$ calls $\mathsf{TE}_{\mathsf{lingen}}$ (from Fig. 15) that computes accepting KZG transcripts $\mathbf{tr}_{a_s}, \mathbf{tr}_{b_s}$ for $s \in [1, m]$. As stated in Lemma 8, each $\mathsf{tr}_{a_s i}$ (respectively $\mathsf{tr}_{b_s i}$) contains a transcript for the commitment $[a_s]_1$ with a distinct evaluation point $\mathfrak{z}_i$. We feed them separately nto KZG's $(n+1)$-special-soundness extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}$ to compute all the polynomials $\mathsf{a}_s^*(X)$ and $\mathsf{b}_s^*(X)$.

Next, we show that $\mathsf{Adv}_{\mathsf{Pgen}, \mathsf{SanLinGen}, \mathsf{Ext}_{\mathsf{ss}}^{\mathsf{SanLinGen}}, (2n+1, m, m+2), \mathcal{A}}^{\mathsf{ss}}(\lambda)$ is negligible for any PPT adversary $\mathcal{A}$. That is, given $\mathcal{A}$'s output, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{SanLinGen}}$ fails to extract

$(\mathsf{a}^*_s(X), \mathsf{b}^*_s(X))_{s=1}^m$, such that

$$(([a_s, b_s]_1, \bar{a}_s, \bar{b}_s)_{s=1}^m, (\mathsf{a}^*_s(X), \mathsf{b}^*_s(X))_{s=1}^m) \in \mathcal{R}_{\mathsf{ck}}^{\mathsf{LinGen}} \enspace .$$

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

1. The event $\mathsf{bad}_{\mathsf{ext}}$ happens when $([a_s]_1, \mathsf{a}^*_s(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{tr}_{a_s}}$ or $([b_s]_1, \mathsf{a}^*_s(X)) \notin \mathcal{R}_{\mathsf{ck}, \mathbf{tr}_{b_s}}$ for some $s \in [1, m]$.
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 $s \in [1, m]$, either $\mathsf{a}^*_s(\mathfrak{z}_i) \neq \bar{a}_{is}$ or $\mathsf{b}^*_s(\mathfrak{z}_i) \neq \bar{b}'_{is}$ for some $i \in [n+2, 2n+1]$.
   (b) $\sum_{s=1}^m \bar{a}_{is} \bar{b}'_{is} \neq 0$ for any $i \in [1, 2n+1]$.

Consider the scenario where neither $\mathsf{bad}_{\mathsf{ext}}$ nor $\mathsf{bad}_{\mathsf{evb}}$ occurs. Then, we have extracted polynomials $\mathsf{a}^*_s(X)$ and $\mathsf{b}^*_s(X)$ for $s \in [1, m]$ of degree at most n which are consistent with the commitments. Denote $g(X) := \sum_{s=1}^m \mathsf{a}^*_s(X)\mathsf{b}^*_s(X)$. For any $i \in [1, 2n+1]$,

$$g(\mathfrak{z}_i) = \sum_{s=1}^m \mathsf{a}^*_s(\mathfrak{z}_i)\mathsf{b}^*_s(\mathfrak{z}_i) = \sum_{s=1}^m \bar{a}_{is}\bar{b}'_{is} = 0 \enspace .$$

Since $g(X)$ is at most degree $2n$, it follows that $g(X) = 0$ and consequently $\sum_{s=1}^m \mathsf{a}^*_s(X)\mathsf{b}^*_s(X) = 0$. Therefore, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{SanLinGen}}$ 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 $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ (see Fig. 16) that breaks the special-soundness of KZG when the event $\mathsf{bad}_{\mathsf{ext}}$ happens. $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ runs $\mathcal{A}(\mathsf{ck})$ to recover the tree T. Then, it obtains transcript vectors $(\mathbf{tr}_{a_s}, \mathbf{tr}_{b_s})_{s=1}^m \leftarrow \mathsf{TE}_{\mathsf{lingen}}(\mathsf{ck}, T)$. For each transcript vector $\mathbf{tr}_f$, where $f \in \{a_s, b_s\}_{s=1}^m$, $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ runs the deterministic extractor $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{kzg}}(\mathsf{ck}, \mathbf{tr}_f)$ to some polynomial $\mathsf{f}^*(X)$. $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ returns the first 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 $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ breaks the special soundness of KZG,

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

Second, we bind the probability $\Pr[\mathsf{bad}_{\mathsf{evb}}]$. We construct a PPT evaluation-binding adversary $\mathcal{C}_{\mathsf{evb}}$, depicted in Fig. 16. $\mathcal{C}_{\mathsf{evb}}$ runs $\mathcal{A}$, $\mathsf{TE}_{\mathsf{lingen}}$, $\mathsf{Ext}_{\mathsf{ss}}^{\mathsf{SanLinGen}}$ to obtain T and corresponding $(\mathsf{a}^*_s(X), \mathsf{b}^*_s(X))_{s=1}^m$. If the event $\mathsf{bad}_{\mathsf{evb}}$ happens, then for all s, $([a_s]_1, \mathsf{a}^*_s(X)) \in \mathcal{R}_{\mathsf{ck}, \mathbf{tr}_{a_s}}$ and $([b_s]_1, \mathsf{a}^*_s(X)) \in \mathcal{R}_{\mathsf{ck}, \mathbf{tr}_{b_s}}$ and thus, $\mathsf{a}^*_s(x) = a_s$ and $\mathsf{b}^*_s(x) = b_s$. Note that

$$h_{u_o}(X) := (\mathsf{a}^*_s(X) - \mathsf{a}^*_s(\mathfrak{z}_i))/(X - \mathfrak{z}_i)$$

is always a polynomial. Hence, $\mathcal{C}_{\mathsf{evb}}$ can compute $[h_{a_s}(x)]_1$ that satisfies

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

Thus,

$$([a_s]_1, \mathfrak{z}_i, \mathsf{a}^*_s(\mathfrak{z}_i), [h_{a_s}(x)]_1)$$

is an accepting transcript. However, $\mathsf{k.tr}^*_{ijs} = ([a_s]_1, \mathfrak{z}_i, \bar{a}_{is}, [h^*_{i1s}]_1)$ is also an accepting transcript. If $\mathsf{a}^*_s(\mathfrak{z}_i) \neq \bar{a}_{is}$, $\mathcal{C}_{\mathsf{evb}}$ has broken evaluation-binding by finding a collision. Analogously, one can show that $\mathsf{b}^*_s(\mathfrak{z}_i) \neq \bar{b}'_{is}$ implies that $\mathcal{C}_{\mathsf{evb}}$ broke evaluation-binding.

We also need that $g(\mathfrak{z}_i) = \sum_{s=1}^m \mathsf{a}^*_s(\mathfrak{z}_i)\mathsf{b}^*_s(\mathfrak{z}_i) = 0$. From the above,

$$([v_j]_1, \mathfrak{z}_i, \bar{b}'_{ij}, [h'_{ij}]_1)$$

and

$$\left(\left[\sum_{s=1}^m \bar{a}_{is} b_s\right]_1, \mathfrak{z}_i, 0, \left[h^*_{ij,m+1}\right]_1\right)$$

are accepting transcripts for all $i \in [1, 2n+1]$.

Since KZG is triply homomorphic,

$$([\textstyle\sum_{s=1}^m \bar{a}_{is} b_s]_1, \mathfrak{z}_i, \sum_{s=1}^m \bar{a}_{is}\bar{b}'_{is}, [\sum_{s=1}^m \bar{a}_{is} h'_{is}]_1)$$

is an accepting transcript. If $\sum_{s=1}^m \bar{a}_{is}\bar{b}'_{is} \neq 0$, $\mathcal{C}_{\mathsf{evb}}$ has found a collision and thus broken 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

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

□

B Postponed Material from Section 4

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

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

- $\mathsf{q_M}(X), \mathsf{q_L}(X), \mathsf{q_R}(X), \mathsf{q_O}(X), \mathsf{q_C}(X)$ 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 $\mathsf{S}_{\sigma 3}(X) := \sum_{i=1}^{n} \sigma^*(2n+i) L_i(X)$.

B.2 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 TriRSDH.)

In Fig. 7, 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 $k \in [2, 4] \cup [7, 9]$, extracting polynomials $\mathsf{z}(X)$, $\mathsf{a}(X)$, $\mathsf{b}(X)$, $\mathsf{c}(X)$, $\mathsf{t_{lo}}(X)$, $\mathsf{t_{mid}}(X)$, and $\mathsf{t_{hi}}(X)$ 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') $\mathsf{bad_{ext}} \leftarrow \mathsf{false}$; $\mathsf{bad_{evb}} \leftarrow \mathsf{false}$;

(ii') if $([z]_1, \mathsf{z}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_1} \vee ([a]_1, \mathsf{a}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_2} \vee ([b]_1, \mathsf{b}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_3} \vee ([c]_1, \mathsf{c}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_4} \vee ([t_{lo}]_1, \mathsf{t_{lo}}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_7} \vee ([t_{mid}]_1, \mathsf{t_{mid}}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_8} \vee ([t_{lo}]_1, \mathsf{t_{hi}}(X)) \notin \mathcal{R}_{\mathsf{ck},\mathbf{k.tr}_9}$ (see Eq. (1)) then $\mathsf{bad_{ext}} \leftarrow \mathsf{true}$; abort;

(iii') for $i \in [\kappa_{\mathsf{kzg}} + 2, \kappa_{\mathfrak{z}}]$: if $\mathsf{z}(\mathfrak{z}_i \omega) \neq \bar{z}_{\omega i} \vee \mathsf{a}(\mathfrak{z}_i) \neq \bar{a}_i \vee \mathsf{b}(\mathfrak{z}_i) \neq \bar{b}_i \vee \mathsf{c}(\mathfrak{z}_i) \neq \bar{c}_i \vee \mathsf{t_{lo}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z} lo,i} \vee \mathsf{t_{mid}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z} mid,i} \vee \mathsf{t_{hi}}(\mathfrak{z}_i) \neq \bar{t}_{\mathfrak{z} hi,i}$ then $\mathsf{bad_{evb}} \leftarrow \mathsf{true}$; 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 $\mathsf{bad_{evb}} \leftarrow \mathsf{true}$; abort;
 – $\mathsf{r}_i(X) \leftarrow \Lambda_{0i}(X) + \alpha \Lambda_{1i}(X) + \alpha^2 \Lambda_{2i}(X) - \mathsf{Z}_{\mathbb{H}}(\mathfrak{z}_i) \cdot (\mathsf{t_{lo}}(X) + \mathfrak{z}_i^n \mathsf{t_{mid}}(X) + \mathfrak{z}_i^{2n} \mathsf{t_{hi}}(X))$;
 – if $\mathsf{r}_i(\mathfrak{z}_i) \neq 0$ then $\mathsf{bad_{evb}} \leftarrow \mathsf{true}$;

Importantly, only one of the "bad" flags is set at a time. Thus, for example, $\mathsf{bad_{evb}} = \mathsf{true}$ implies that $\mathsf{bad_{ext}} = \mathsf{false}$. 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 $\mathsf{a}(X)$, $\mathsf{b}(X)$, $\mathsf{c}(X)$, $\mathsf{z}(X)$, $\mathsf{t_{lo}}(X)$, $\mathsf{t_{mid}}(X)$, and $\mathsf{t_{hi}}(X)$ 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 $\mathsf{bad_{ext}} = \mathsf{bad_{evb}} = \mathsf{false}$, we get that $\mathsf{a}(X)$, $\mathsf{b}(X)$, $\mathsf{c}(X)$, $\mathsf{z}(X)$, $\mathsf{t_{lo}}(X)$, $\mathsf{t_{mid}}(X)$, $\mathsf{t_{hi}}(X)$ are consistent with the commitments and all openings.
 Recall that in the case of Plonk, we deduced here that since $\mathsf{bad}_{\mathrm{trirsdh}} = \mathsf{false}$, $\mathsf{t}(X) = (\mathsf{F}_0(X) + \alpha \mathsf{F}_1(X) + \alpha^2 \mathsf{F}_2(X)) / \mathsf{Z}_{\mathbb{H}}(X)$ is a

polynomial. In the case of $\mathsf{SanPlonk}$, 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 $\deg g(X) \le 4\kappa_{\mathsf{kzg}}$, $g(X) = 0$. 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. 8), 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 seven bad events happens. $\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ just tests which of the cases is true and returns the corresponding transcript. Clearly, $\Pr[\mathcal{B}_{\mathsf{ss}}^{\mathsf{kzg}}$ succeeds $\mid \mathsf{bad}_{\mathsf{ext}}] = 1$.
3. **Claim 3**. There exists an evaluation-binding adversary $\mathcal{C}_{\mathsf{evb}}$ (see Fig. 9) 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. 9 *is* a collision. Thus, $\Pr[\mathcal{C}_{\mathsf{evb}}$ succeeds $\mid \mathsf{bad}_{\mathsf{evb}}] = 1$.

Thus,

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

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, $\Pr[\overline{\mathcal{E}}] \le \mathsf{negl}(\lambda)$. This proves the claim. □

B.3 Proof of Lemma 4

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

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

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

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

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

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 $s \in [1, n]$. □

B.4 Proof of Lemma 5

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

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

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. □

B.5 Fiat-Shamir Transform

Recall the following theorem from [AFK22].

Theorem 7 ([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 7 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 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 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 7. 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.

C Security of TriRSDH in the AGMOS

We prove that n-TriRSDH is secure in the AGMOS (AGM with oblivious sampling, [LPS23]). For this, in Appendix C.1, we first recall the definition of AGMOS.

C.1 Preliminaries: AGMOS

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. As shown in [LPS23], certain uses of KZG are secure in AGM but not in AGMOS. Since AGMOS is a new model, we will give a longer description of AGMOS and TOFR (an underlying security assumption); our description follows closely [LPS23].

Fix a pairing description $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$. Let $\mathcal{EF}_{\mathsf{p},\iota}$ be a set of (polynomially many) functions $\mathbb{F} \to \mathbb{G}_\iota$. Let $\mathcal{DF}_{\mathsf{p}}$ be a family of distributions over $\mathbb{F}$. We introduce two oracles $\mathcal{O}_1$ and $\mathcal{O}_2$, one for each group $\mathbb{G}_1$ and $\mathbb{G}_2$. Let $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$. The ith query (E, D) to $\mathcal{O}_\iota$ consists of a function $E \in \mathcal{EF}_{\mathsf{p},\iota}$ and a distribution $D \in \mathcal{DF}_{\mathsf{p}}$. The oracle samples a random field element $s_i \leftarrow_{\$} D$ and returns $[\mathbb{q}_{\iota i}]_\iota \leftarrow E(s_i)$ and s_i.

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 $\mathbb{x} = ([\mathbb{x}_1]_1, [\mathbb{x}_2]_2)$. The

```
$\mathcal{O}_\iota(E, D)$
**if** $E \notin \mathcal{EF}_{\mathsf{p},\iota} \vee D \notin \mathcal{DF}_{\mathsf{p}}$ **then return** $\perp$; **fi**
$s \leftarrow_\$ D; [\mathbb{q}]_\iota \leftarrow E(s)$; **return** $([\mathbb{q}]_\iota, s)$;
```

Fig. 17. The description of the oblivious sampling oracle $\mathcal{O}_\iota$, where $\iota \in \{1, 2\}$.

adversary's view consists of all group elements that the adversary has seen up to the given moment. This includes the adversary's initial input, elements sent by other parties during the interaction, and oracle answers.

Let $\mathcal{O}$ be as above. We require that for any PPT oracle adversary $\mathcal{A}^{\mathcal{O}}$, there exists a (non-uniform) PPT extractor $\mathsf{Ext}_{\mathcal{A}}^{\mathcal{O}}$, such that: if $\mathcal{A}^{\mathcal{O}}(\mathbb{x})$ 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, $\mathsf{Ext}_{\mathcal{A}}^{\mathcal{O}}$ outputs field-element matrices $\boldsymbol{\gamma}$, $\boldsymbol{\delta}$, and $[\mathbb{q}_\iota]_\iota$ ($\mathcal{O}_\iota$'s answer vector), such that

$$\mathbb{y} = \boldsymbol{\gamma}^\intercal \mathbb{x}_\iota + \boldsymbol{\delta}^\intercal \mathbb{q}_\iota \ . \tag{17}$$

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 $\mathbb{x} = (\mathbb{x}_1, \mathbb{x}_2)$,* $\mathsf{Adv}^{\mathrm{agmos}}_{\mathsf{Pgen},\mathcal{EF},\mathcal{DF},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) :=$

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

$\mathcal{O}$ is the non-programmable oracle depicted in Fig. 17. Here, $[\mathbb{q}_\iota]_\iota$ is required to be the tuple of elements output by $\mathcal{O}_\iota$. We denote by il_ι the number of $\mathcal{O}_\iota$ calls.

Many AGMOS proofs rely on the following two assumptions.

Let $d_1(\lambda), d_2(\lambda) \in \mathsf{poly}(\lambda)$. Pgen is $(d_1(\lambda), d_2(\lambda))$-$\mathsf{PDL}$ *(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 $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{tofr}}_{\mathsf{Pgen},\mathcal{A}}(\lambda) :=$

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

Here, $\mathcal{O}$, $\mathbb{q}_1$, and $\mathbb{q}_2$ are as in Definition 3.

C.2 TriRSDH's Security Proof

Theorem 8. *Let $n = \mathsf{poly}(\lambda)$; then n-*TriRSDH *holds in the AGMOS under the* PDL *and* TOFR *assumptions.*

Proof (Sketch). Let $\mathcal{A}$ be a AGMOS adversary against the TriRSDH assumption. Thus, with non-negligible probability, on input $\mathsf{ck} = ([1, x, \dots, x^{\kappa_{\mathsf{kzg}}}]_1, [1, x]_2)$, $\mathcal{A}$ outputs $(\mathfrak{z}_i, [\chi_i]_1)_{i=1}^{\kappa_\mathfrak{z}}$, $[t_{lo}, t_{mid}, t_{hi}]_1$, and $\mathsf{F}(X)$ such that

$$\begin{aligned}&\forall i \neq i'. \mathfrak{z}_i \neq \mathfrak{z}_{i'} \in \mathbb{F} \wedge \mathsf{F}(X) \in \mathbb{F}_{\leq \kappa_\mathfrak{z} - 1}[X] \wedge (\mathsf{Z}_\mathbb{H}(X) \nmid \mathsf{F}(X)) \wedge \\ &\forall i \in [1, \kappa_\mathfrak{z}]. \left[t_{lo} + \mathfrak{z}_i^n t_{mid} + \mathfrak{z}_i^{2n} t_{hi} - \tfrac{\mathsf{F}(\mathfrak{z}_i)}{\mathsf{Z}_\mathbb{H}(\mathfrak{z}_i)}\right]_1 \bullet [1]_2 = [\chi_i]_1 \bullet [x - \mathfrak{z}_i]_2\end{aligned} .$$

Then, there exists an extractor $\mathsf{Ext}_\mathcal{A}^\mathcal{O}$ that, on input ck and $\mathcal{A}$'s random coins, outputs $(\{\mathsf{t}'_s, \hat{\mathbf{t}}_s\}_{s \in \{lo, mid, hi\}}, \{\chi'_i, \hat{\boldsymbol{\chi}}_i\}_{i \in [1, \kappa_\mathfrak{z}]})$ such that, except with negligible probability, we have

$$\mathsf{t}_s = \mathsf{t}'_s(x) + \hat{\mathbf{t}}_s^\top \mathbb{q} \ , \quad \chi_i = \chi'_i(x) + \hat{\boldsymbol{\chi}}_i^\top \mathbb{q} \ .$$

Here $\mathbb{q}$ is a vector of discrete logarithms of the answers returned by the oblivious sampling oracle $\mathcal{O}$.

Let us define $\mathsf{t}_s(X, \mathbb{Q}) = \mathsf{t}'_s(X) + \hat{\mathbf{t}}_s^\top \mathbb{Q}$ and $\chi_i(X, \mathbb{Q}) = \chi'_i(X) + \hat{\boldsymbol{\chi}}_i^\top \mathbb{Q}$ for $s \in \{lo, mid, hi\}$ and $i \in [1, \kappa_\mathfrak{z}]$. Here, say, $\chi'_i(X) \in \mathbb{F}_{\leq \kappa_{\mathsf{kzg}}}[X]$ are polynomials and, say, $\hat{\chi}_{ij} \in \mathbb{F}$ are field elements. Let $\mathsf{t}(X) = \mathsf{F}(X)/\mathsf{Z}_\mathbb{H}(X)$. Let il_1 denote the length of the vector $\mathbb{Q}$. For each $i \in [1, \kappa_\mathfrak{z}]$, we define the verification polynomial as

$$\begin{aligned}V_i(X, \mathbb{Q}) :=& V_{i0}(X) + \sum_{j=1}^{\mathsf{il}_1} V_{ij}(X) Q_j \\ =& (\mathsf{t}'_{lo}(X) + \hat{\mathbf{t}}_{lo}^\top \mathbb{Q}) + \mathfrak{z}_i^n (\mathsf{t}'_{mid}(X) + \hat{\mathbf{t}}_{mid}^\top \mathbb{Q}) + \mathfrak{z}_i^{2n} (\mathsf{t}'_{hi}(X) + \hat{\mathbf{t}}_{hi}^\top \mathbb{Q}) \\ &- \mathsf{t}(\mathfrak{z}_i) - (\chi'_i(X) + \hat{\boldsymbol{\chi}}_i^\top \mathbb{Q})(X - \mathfrak{z}_i) \ .\end{aligned}$$

One of the following three cases can happen:

1. For each $i \in [1, \kappa_\mathfrak{z}]$, $V_i(X, \mathbb{Q}) \equiv 0$ as a polynomial. We show that this is impossible.
2. For at least one i, $V_i(X, \mathbb{Q}) \not\equiv 0$. However, $\sum_{j=1}^{\mathsf{il}_1} V_{ij}(x) Q_j \equiv 0$. In this case we have a successful reduction to the PDL assumption.
3. For at least one i, we have $V_i(X, \mathbb{Q}) \not\equiv 0$ and $\sum_{j=1}^{\mathsf{il}_1} V_{ij}(x) Q_j \not\equiv 0$. In this case we break the TOFR assumption.

Case 1. In the first case of an AGMOS proof, we get that for any $i \in [1, \kappa_\mathfrak{z}]$,

$$\begin{aligned}&(\mathsf{t}'_{lo}(X) + \hat{\mathbf{t}}_{lo}^\top \mathbb{Q}) + \mathfrak{z}_i^n (\mathsf{t}'_{mid}(X) + \hat{\mathbf{t}}_{mid}^\top \mathbb{Q}) + \\ &\qquad \mathfrak{z}_i^{2n} (\mathsf{t}'_{hi}(X) + \hat{\mathbf{t}}_{hi}^\top \mathbb{Q}) - \mathsf{t}(\mathfrak{z}_i) \equiv (\chi'_i(X) + \hat{\boldsymbol{\chi}}_i^\top \mathbb{Q})(X - \mathfrak{z}_i) \ ,\end{aligned}$$

where the latter is an equality of polynomials.

Let us set $X = \mathfrak{z}_i$ on the left and right hand side. Then, we obtain the equation,

$$\left(\mathsf{t}'_{lo}(\mathfrak{z}_i) + \mathfrak{z}_i^n \mathsf{t}'_{mid}(\mathfrak{z}_i) + \mathfrak{z}_i^{2n}(\mathsf{t}'_{hi}\mathfrak{z}_i) - \mathsf{t}(\mathfrak{z}_i)\right) + \left(\hat{\mathbf{t}}_{lo} + \mathfrak{z}_i^n \hat{\mathbf{t}}^{\intercal}_{mid} + \hat{\mathbf{t}}_{hi}\right) \mathbb{Q} = 0 \enspace .$$

This implies $\mathsf{t}'_{lo}(\mathfrak{z}_i) + \mathfrak{z}_i^n \mathsf{t}'_{mid}(\mathfrak{z}_i) + \mathfrak{z}_i^{2n} \mathsf{t}'_{hi}(\mathfrak{z}_i) = \mathsf{t}(\mathfrak{z}_i)$. Observe that the polynomial $\mathsf{F}(X)$ does not depend on $\mathbb{Q}$ by definition. Thus, the polynomial

$$T(X) := \mathsf{t}'_{lo}(X) + X^n \mathsf{t}'_{mid}(X) + X^{2n} \mathsf{t}'_{hi}(X) \in \mathbb{F}_{\leq \kappa_{\mathsf{kzg}}+2n}[X]$$

and the rational function $\mathsf{t}(X) = \mathsf{F}(X)/\mathsf{Z}_{\mathbb{H}}(X)$ agree on $\kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}} + 1$ points $\{\mathfrak{z}_i\}_i$. Hence, the polynomials $T(X)\mathsf{Z}_{\mathbb{H}}(X) \in \mathbb{F}_{\leq \kappa_{\mathsf{kzg}}+3n}[X]$ and

$$\mathsf{F}(X) = \mathsf{t}(X)\mathsf{Z}_{\mathbb{H}}(X) \in \mathbb{F}_{\leq 4\kappa_{\mathsf{kzg}}}[X]$$

agree on $\kappa_{\mathfrak{z}} = 4\kappa_{\mathsf{kzg}} + 1$ points $\{\mathfrak{z}_i\}_i$. Since both polynomials have degree $\leq 4\kappa_{\mathsf{kzg}}$, they coincide as polynomials and thus $T(X) = \mathsf{t}(X)$ is a polynomial. Contradiction.

Case 2. Exists $\hat{i} \in [1, \kappa_{\mathfrak{z}}]$, such that $V_{\hat{i}}(X, \mathbb{Q}) \not\equiv 0$, and $\sum_{j=1}^{\mathsf{il}_1} V_{\hat{i}j}(x) Q_j \equiv 0$. In this case we have that either

$$V_{\hat{i}0}(X) = \mathsf{t}'_{lo}(X) + \mathfrak{z}_{\hat{i}}^n \mathsf{t}'_{mid}(X) + \mathfrak{z}_{\hat{i}}^{2n} \mathsf{t}'_{hi}(X) - \mathsf{t}(\mathfrak{z}_{\hat{i}}) - \chi'_{\hat{i}}(X)(X - \mathfrak{z}_{\hat{i}}) \not\equiv 0 \enspace ,$$

or there exists $j \in [1, \mathsf{il}_1]$ such that

$$V_{\hat{i}j}(X) = \hat{\mathsf{t}}_{lo:j} + \mathfrak{z}_{\hat{i}}^n \hat{\mathsf{t}}_{mid:j} + \mathfrak{z}_{\hat{i}}^{2n} \hat{\mathsf{t}}_{mid:j} - \hat{\chi}_{\hat{i}j}(X - \mathfrak{z}_{\hat{i}}) \not\equiv 0 \enspace .$$

Furthermore, both $V_{\hat{i}0}(x) = 0$ and $V_{\hat{i}j}(x) = 0$.

Let $\mathsf{Tar}_{\hat{i}}(X)$ be one of such polynomials. Note that $\mathsf{Tar}_{\hat{i}}(X)$ is univariate (independent from the oracle calls), and of degree at most κ_{kzg}. Then we have $\mathsf{Tar}_{\hat{i}}(X) \not\equiv 0$, but $\mathsf{Tar}_{\hat{i}}(x) = 0$. The PDL adversary can now compute x as one of the roots of the univariate $\mathsf{Tar}_{\hat{i}}(X)$ and win the game. Thus, the probability of being in case 2 must be negligible.

Case 3. Exists $i \in [1, \kappa_{\mathfrak{z}}]$, such that $V_i(X, \mathbb{Q}) \not\equiv 0$ and $\sum_{j=1}^{\mathsf{il}_1} V_{ij}(x) Q_j \not\equiv 0$. Therefore, we have that

$$\begin{aligned}(\mathsf{t}'_{lo}(x) + \hat{\mathbf{t}}^{\intercal}_{lo}\mathbb{Q}) + \mathfrak{z}_i^n(\mathsf{t}'_{mid}(x) + \hat{\mathbf{t}}^{\intercal}_{mid}\mathbb{Q}) + \mathfrak{z}_i^{2n}(\mathsf{t}'_{hi}(x) + \hat{\mathbf{t}}^{\intercal}_{hi}\mathbb{Q}) - \mathsf{t}(\mathfrak{z}_i) \\ - (\chi'_i(x) + \hat{\boldsymbol{\chi}}_i^{\intercal}\mathbb{Q})(x - \mathfrak{z}_i) \not\equiv 0 \enspace .\end{aligned}$$

In this case we construct an adversary that breaks the TOFR assumption. The adversary samples ck by itself, and then, knowing the trapdoor x, it outputs the vector $\boldsymbol{v}$ defined as

$$\begin{aligned} v_0 &= \mathsf{t}'_{lo}(x) + \mathfrak{z}_i^n \mathsf{t}'_{mid}(x) + \mathfrak{z}_i^{2n} \mathsf{t}'_{hi}(x) - \mathsf{t}(\mathfrak{z}_i) - \chi'_i(x)(x - \mathfrak{z}_i) \enspace , \\ \forall j \in [1, \mathsf{il}_1]. v_j &= \hat{\mathsf{t}}_{lo:j} + \mathfrak{z}_{\hat{i}}^n \hat{\mathsf{t}}_{mid:j} + \mathfrak{z}_{\hat{i}}^{2n} \hat{\mathsf{t}}_{hi:j} - \hat{\chi}_{\hat{i}j}(x)(x - \mathfrak{z}_{\hat{i}}) \enspace . \end{aligned}$$

Such adversary is successful in breaking the TOFR assumption. Thus, the probability of being in case 3 is negligible. $\square$

D Zero-Knowledge of SanPlonk

We recall the zero-knowledge property of a non-interactive argument.[9] 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.

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 standard Plonk now, [GWC19]. Our paper contains the current (corrected) version of Plonk, so we will not repeat the zero-knowledge proof of Plonk 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 9 ([Sef24]). *Let $f(X) \in \mathbb{F}[X]$ and $x_1, \dots, x_k$ be distinct values in $\mathbb{F} \setminus \mathbb{H}$. Assume $\tilde{f}(X) = f(X) + \varrho(X)\mathsf{Z}_{\mathbb{H}}(X)$ for $\varrho(X) \leftarrow_{\$} \mathbb{F}_{\leq k-1}[X]$. Then, $(\tilde{f}(x_1), \dots, \tilde{f}(x_k))$ is distributed uniformly over $\mathbb{F}^k$,*

The same claim also holds for $\tilde{f}(X) = f(X) + \varrho(X)$; moreover, then it is true even when $x_1, \dots, x_k \in \mathbb{F}$, not just in $\mathbb{F} \setminus \mathbb{H}$ (in Lemma 9, we need that $\mathsf{Z}_{\mathbb{H}}(x_i) \neq 0$). We use both results to prove the statistical zero-knowledge of SanPlonk.

Theorem 9. 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

[9] We proved the special soundness for interactive arguments, and since then, we could use Theorem 7 (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.

```
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. (6), 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.

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. (6) for the formulas of $\mathsf{F}_i(X)$ and $\mathsf{F}(X)$.)

From Lemma 9 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 $[\mathsf{t}(x)]_1$. Furthemore, 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.

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. $\square$