Simulation-Extractable KZG Polynomial Commitments and Applications to HyperPlonk

Benoît Libert

Zama, France

Abstract. HyperPlonk is a recent SNARK proposal (Eurocrypt'23) that features a linear-time prover and supports custom gates of larger degree than Plonk. For the time being, its instantiations are only proven to be knowledge-sound (meaning that soundness is only guaranteed when the prover runs in isolation) while many applications motivate the stronger notion of simulation-extractability (SE). Unfortunately, the most efficient SE compilers are not immediately applicable to multivariate polynomial interactive oracle proofs. To address this problem, we provide an instantiation of HyperPlonk for which we can prove simulation-extractability in a strong sense. As a crucial building block, we describe KZG-based commitments to multivariate polynomials that also provide simulation-extractability while remaining as efficient as malleable ones. Our proofs stand in the combined algebraic group and random oracle model and ensure straight-line extractability (i.e., without rewinding).

Keywords: Polynomial commitments, SNARKs, zero-knowledge, simulation-extractability

1 Introduction

A standard technique to obtain succinct non-interactive arguments of knowledge (SNARKs) is to compile a polynomial interactive oracle proof (PIOP) [11] using a polynomial commitment scheme (PCS) [39]. The resulting interactive argument system can then be made non-interactive using the Fiat-Shamir heuristic [26]. Many popular SNARKs (including Sonic [46], Plonk [30], STARK [9], Marlin [19] or Gemini [13]) were designed using this methodology. In order to obtain concretely efficient SNARKs, two widely used polynomial commitment schemes (PCS) are KZG [39] and FRI [8].

Most SNARKs are proven to be knowledge-sound, which only guarantees soundness against a stand-alone prover. In practical applications, however, cheating provers can copy proofs from one another and even tamper with proofs generated by honest parties in an attempt to prove a related statement without knowing underlying witnesses. The notions of simulation-soundness [49] and simulation-extractability [22] rule out these malleability attacks. In particular, simulation-extractability (SE) ensures knowledge-soundness even when the adversary can observe proofs generated by honest parties and try to create a proof of its own by mauling honestly generated proofs. Simulation-extractability is thus

an important security property in all applications where succinct arguments are widely available online.

The most efficient simulation-extractable SNARKs often require a scheme-specific analysis [38,3] or, in instantiations of the PIOP paradigm, require specific conditions on the underlying PIOP and polynomial commitments [31,24,41]. To our knowledge, existing generic approaches either fail to preserve succinctness [43] or introduce significant overhead [2] by relying on additional primitives.

In this paper, we further investigate the (non-black-box) simulation-extractability of SNARKs obtained by compiling multivariate PIOPs using pairing-based polynomial commitments like KZG. We focus on the recent HyperPlonk construction of Chen *et al.* [18], which improves Plonk so as to obtain a linear-time prover and larger-degree custom gates by working over the Boolean hypercube. We provide a simulation-extractable instantiation of HyperPlonk in the combined algebraic group and random oracle (AGM+ROM) model using appropriate polynomial commitments that are themselves proven simulation-extractable. Our security proof features a straight-line knowledge extractor that extracts witnesses without rewinding in order to make the reduction tighter.

To this end, we build suitable simulation-extractable polynomial commitments to multivariate polynomials since HyperPlonk heavily relies on multilinear polynomial commitments over the Boolean hypercube.

1.1 Contributions

We first construct pairing-based simulation-extractable polynomial commitments (SE-PCS) in the combined algebraic group and random oracle model. For multivariate and univariate polynomials, we obtain these by tweaking KZG-based commitments [39,56] and prove simulation-extractability in a strong sense.

Our multivariate PCS is obtained by modifying a multivariate PCS suggested by Zhang *et al.* [56], which is itself a randomized variant the multivariate extension of KZG from [47]. Just like its deterministic version [47], the multivariate PCS of [56] is actually malleable. However, we show that it can be made simulation-extractable (by introducing a random oracle) at a moderate cost using an approach of "proving knowledge of an evaluation proof" via the Fiat-Shamir paradigm [26]. This approach only slightly increases the complexity of the scheme (by introducing only one additional scalar in the proof without significantly affecting computational costs) and leverages the non-malleability properties [25] of the Fiat-Shamir transform. Despite the use of Fiat-Shamir, we can prove simulation-extractability in the AGM+ROM without rewinding.

Along the way, we also provide a new security proof (in the algebraic group model) for the multivariate PCS of Zhang *et al.* [56], in its simplest variant where commitments only consist of one group element. While the original proof was given under a knowledge assumption (more precisely, a parameterized variant of the knowledge-of-exponent assumption [20]) in the standard model and for a less efficient variant of the scheme, it was recently shown to be incorrect [42]. Our new proof stands in the algebraic group model and generalizes the result Kohrita and Towa [42] who gave a new proof in the univariate case.

Using our simulation-extractable multivariate PCS, we then give a simulation-extractable variant of HyperPlonk which preserves its linear complexity at the prover. Again, we can prove security in the AGM+ROM without rewinding.

As a result of independent interest, we describe a new randomized variant of univariate KZG commitments, which is more efficient than the standard randomized variant (described in [39]) of KZG commitments. In particular, its hiding and zero-knowledge properties do not require the commitment randomness to be as large as the degree of the committed polynomial, even when many evaluations are given out. This variant is proven simulation-extractable and is more efficient than the one implied by our simulation-extractable variant of [56].

1.2 Technical Overview

In SNARK constructions relying on a trusted setup, proving SE via the modular frameworks of [31,41] requires to provide a trapdoor-less zero-knowledge simulator, which does not use the trapdoor of the SRS but proceeds by only programming random oracles. In the zero-knowledge version of HyperPlonk [18, Appendix A], the simulator uses a ZK simulator described in [54] for sumcheck protocols [45]. However, the latter zero-knowledge sumcheck simulator assumes that the underlying multivariate PCS has zero-knowledge evaluation proofs (see [18, Lemma A.2] and [54, Theorem 3]). While the commitment scheme of Zhang *et al.* [56] satisfies this property, its simulator is not trapdoor-less. Moreover, it does not provide the weak unique response property required by [41] because its evaluation proofs are publicly randomizable. For the same reason, we cannot easily extend the ideas of [24] to the multivariate setting since they require an even stronger property on behalf of the polynomial commitment.

To overcome the above hurdle, we construct a variant of the multivariate PCS of [56] which is trapdoor-less zero-knowledge and satisfies a strong flavor of simulation-extractability.

Our multivariate PCS system commits to polynomials in the same way as its malleable counterpart [56]. However, we use a different non-interactive evaluation protocol, which proceeds by having the committer prove knowledge of an evaluation proof of the original PCS scheme [56]. This can be done efficiently by exploiting the malleability of the initial evaluation proofs. In groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ endowed with a bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$, the malleable PCS of [56] commits to polynomials $f \in \mathbb{Z}_p[X_1, \ldots, X_\ell]$ via commitments of the form $C = g^{f(\alpha_1,\ldots,\alpha_\ell)+r\cdot\alpha_r}$, for a random $r \in \mathbb{Z}_p$ and where $(\alpha_1, \ldots, \alpha_\ell, \alpha_r)$ are secrets hidden in the structured reference string (SRS). The correctness of evaluations $y = f(z_1, \ldots, z_\ell)$ is proven via group elements $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_\ell, \pi_r) \in \mathbb{G}^{\ell+1}$ satisfying a pairing-product equation of the form

$$F(\mathsf{srs}, C, y) = \prod_{i=1}^{\ell} e(\pi_i, \hat{g}^{\alpha_i} \cdot \hat{g}^{-z_i}) \cdot e(\pi_r, \hat{g}^{\alpha_r}), \tag{1}$$

where F is a function of the SRS, C is the commitment and $y \in \mathbb{Z}_p$ is the claimed output. Instead of revealing $\boldsymbol{\pi}$ as the original scheme does, we exploit

the linearity properties of equation (1) which make it possible to efficiently prove knowledge of $(\pi_1, \ldots, \pi_\ell, \pi_r) \in \mathbb{G}^{\ell+1}$ satisfying (1) via a standard Schnorr-like Σ-protocol [50] allowing to prove knowledge of homomorphism pre-images. Although the product (1) allows randomizing the underlying $\boldsymbol{\pi}$, we can achieve simulation-extractability by exploiting the non-malleability [25] of Fiat-Shamir when $(C, y, \boldsymbol{z} = (z_1, \ldots, z_\ell))$ are included in the inputs of the random oracle.

While the above idea is simple, proving the simulation-extractability of the Fiat-Shamir-compiled evaluation proof is non-trivial when it comes to achieving straight-line extraction. To this end, we extend an observation from [29] which shows that, in the AGM, Schnorr signatures can be proven secure without rewinding. In our SE-PCS, in order to prove knowledge of $(\pi_1, \ldots, \pi_\ell, \pi_r)$ satisfying (1), we reveal $(\pi_1, \ldots, \pi_\ell)$ in the clear but we prove knowledge of π_r. The Σ-protocol has a verification equation of the form

$$R = e(S_\pi, \hat{g}^{\alpha_r}) \cdot \left(F(\mathsf{srs}, C, y) \cdot \prod_{i=1}^{\ell} e(\pi_i, \hat{g}^{\alpha_i} \cdot \hat{g}^{-z_i})^{-1} \right)^{-c},$$

where S_π is a blinded version of π_r and $c \in \mathbb{Z}_p$ is a Fiat-Shamir challenge obtained by hashing $R \in \mathbb{G}_T$ along with $(C, y, \boldsymbol{z})$ and $(\pi_1, \ldots, \pi_\ell)$. In the combined AGM+ROM model, when the adversary makes a random oracle query involving a tuple (R, C, y, z), it has to provide algebraic representations of group elements R and C with respect to previously observed elements of the same group. From these representations, the knowledge extractor can infer polynomials $R[X_1, \ldots, X_\ell, X_r]$, $\{A_i[X_1, \ldots, X_\ell, X_r]\}_{i=1}^{\ell}$ and $f[X_1, \ldots, X_\ell]$ such that $C = g^{f(\alpha_1, \ldots, \alpha_\ell) + r \cdot \alpha_r}$, $R = e(g, \hat{g})^{R(\alpha_1, \ldots, \alpha_\ell, \alpha_r)}$, and $\pi_i = g^{A_i(\alpha_1, \ldots, \alpha_\ell, \alpha_r)}$ for each $i \in [\ell]$. If $y \neq f(z_1, \ldots, z_\ell)$, we can use an argument reminiscent of Katz-Wang signatures [40] and argue that a certain $(\ell+1)$-variate polynomial – which is computable from $R[X_1, \ldots, X_\ell, X_r]$, $\{A_i[X_1, \ldots, X_\ell, X_r]\}_{i=1}^{\ell}$ and $f[X_1, \ldots, X_\ell]$ – is non-zero with overwhelming probability. In turn, this non-zero polynomial determines a univariate polynomial whose roots contain the discrete logarithm $\alpha \in \mathbb{Z}_p$ hidden in the structured CRS $(g, (g^{(\alpha^i)})_{i=1}^{m}, (\hat{g}^{(\alpha^i)})_{i=1}^{n})$.

In order to build an SE variant of HyperPlonk, we use our SE-PCS in the following way. Like Plonk [30], HyperPlonk proceeds by having the prover define an $N \times 3$ matrix of which the rows $\mathbf{M} = \{(L_i, R_i, O_i)\}_{i=1}^{N}$ contain the left/right inputs and the output of each gate. It commits to a multilinear $M[X_1, \ldots, X_\ell]$ whose evaluations over the Boolean hypercube are the entries of $\mathbf{M}$. Then, it proves that $M[\boldsymbol{X}]$ satisfies a "gate identity" by showing that some polynomial $f[\boldsymbol{X}]$ defined as a function of $M[\boldsymbol{X}]$ and circuit-dependent polynomials cancels everywhere on the hypercube $\{0,1\}^\ell$. Next, in order to prove that $\mathbf{M}$ is consistent with the wiring of the circuit, it proves that $M[\boldsymbol{X}]$ satisfies a "wiring identity" $M(\boldsymbol{x}) = M(\hat{\sigma}(\boldsymbol{x}))$ for all $x \in \{0,1\}^\ell$, for some circuit-dependent permutation $\hat{\sigma}$. This is done by showing that $M[\boldsymbol{X}]$ satisfies a product argument over $\{0,1\}^\ell$, by adapting a technique used in Plonk [30] and inspired from [7]. In turn, the product relation is proven by showing that another polynomial vanishes over $\{0,1\}^\ell$ and that some related polynomial $\tilde{v}[\boldsymbol{X}]$ evaluates to 1 on the input $(1, 1, \ldots, 1, 0)$.

In order to obtain a trapdoor-less simulator for HyperPlonk, we choose a multilinear polynomial $\hat{M}[\boldsymbol{X}]$ that satisfies the gate identity, but not the wiring identity (which is always possible without knowing a witness). Then, we can easily prove that the appropriate polynomials vanish everywhere on $\{0,1\}^{\ell}$ (since $\hat{M}[\boldsymbol{X}]$ is a valid witness for the zerocheck arguments) but, since $\hat{M}[\boldsymbol{X}]$ does not satisfy the wiring identity and the underlying product relation, the corresponding polynomial $\tilde{v}[\boldsymbol{X}]$ does not evaluate to 1 on $(1,1,\ldots,1,0)$. However, we can simulate a proof of the latter false statement using our SE-PCS construction.

1.3 Related Work

Simulation-soundness was first introduced by Sahai [49] and strengthened by De Santis *et al.* [22] so as to further ensure extractability. Several techniques based on OR proofs [22,36] have been proposed to generically build simulation-extractable NIZK proofs by upgrading NIZK proofs satisfying the standard notion of soundness. Among these, the technique of De Santis *et al.* [22] makes non-black-box use of additional building blocks such as one-time signatures and pseudorandom functions, which is generally very expensive. Optimizations of this approach were considered in [43,2,1]. The compiler of Kosba *et al.* [43] fails to preserve succinctness in the SNARK setting. While the lifting techniques of [2,1] retain succcinctness, they introduce additional components such as key-homomorphic signatures, which still introduce a significant overhead in terms of proving/verification time or SRS size. The compilers of [1,32] further achieve universal composability [17]. Ganesh *et al.* [32] notably achieve full succinctness via a generic construction allowing to achieve UC security on top of simulation-extractable SNARKs (in the non-programmable ROM).

A different approach proceeds via direct, scheme-specific security analyzes. Groth and Maller [38] gave a simulation-extractable SNARK where proofs only consist of 3 group elements. SE variants of Groth16 [37] were given in [14,4,6,3] while the original version of the scheme [37] was proven [5] weakly simulation-extractable (meaning that the adversary can randomize existing proofs but not come up with a fake proof for a new statement). Lipmaa [44] gave a general framework for constructing SE-SNARKs for R1CS statements [35] in an extension of the AGM allowing to oblivious sample group elements. The constructions of [38,37,14,4,44,3] rely on a non-universal (i.e., circuit-dependent) CRS.

BulletProofs [15] was shown to provide simulation-extractability in the AGM [34] and in the random oracle model [33,21]. Ganesh *et al.* [31] proved the simulation-extractability of Plonk [30], Sonic [46] and Marlin [19] in the combined random oracle and algebraic group model (AGM) [28] using a rewinding-based proof. One disadvantage of their framework is the use of rewinding, which results in non-tight reductions. Also, the proof of simulation-extractability given in [31] for Plonk was recently found [41] to be flawed since the original Plonk uses a deterministic variant of KZG commitments [39]. Specifically, the trapdoor-less simulator of [31] was shown [41] to not provide statistical zero-knowledge.[1] In

[1] This does not contradict the stated zero-knowledge property of Plonk since the simulator outlined in [30] uses the trapdoor of the CRS (unlike the one from [31]).

[41], the authors showed that this problem can be fixed using a randomized version of KZG commitments. A closer inspection suggests that, when instantiated with randomized KZG commitments, Plonk can be proven simulation-extractable without rewinding by applying the compiler of [41] since its knowledge soundness can be argued via a straight-line extractor in the AGM.

A third approach builds SE-SNARKs using the PIOP paradigm by formalizing specific requirements on the underlying PIOP and PCS systems. A recent work of Faonio *et al.* [24] suggests to use a weak form of simulation-extractable polynomial commitments. While they prove such a weak form of simulation-extractability for randomized KZG commitments, they only do it in the univariate case and their techniques are not known to carry over to existing multivariate PCS. The same holds for the framework of [41] as it requires a PCS satisfying a notion of weak uniqueness, meaning that no PPT adversary should be able to randomize a proof obtained from the simulator for the same evaluation pair $(\boldsymbol{z}, y)$. Unfortunately, the most widely used multivariate extensions [47,56] of KZG commitments do not satisfy the latter property since their proofs are publicly randomizable (as explained in Supplementary Material E).

We note that Virgo [55] provides a construction of multilinear PCS from univariate PCS using the univariate sumcheck protocol of Aurora [10]. However, their transformation induces a substantial overhead and requires a prover of super-linear (in fact, quasi-linear) time in order to obtain poly-logarithmic verification time. While Gemini [13] does imply a multilinear-to-univariate conversion that preserves the prover's linear time complexity, it still increases the prover's overhead by a factor 3 (or even 4 according to [42, Table 1]) at each evaluation proof. In [42], Kohrita and Towa [42] gave a more efficient generic construction of multilinear PCS from additively homomorphic univariate PCS. When instantiated with KZG-based commitments, their multilinear PCS scheme has a faster verification algorithm than [56] with only 3 pairing evaluations instead of $O(\mu)$, where μ is the number of variables. On the downside, it increases the number of exponentiations at the prover (which can be as large as 2^{25} for large circuits) by a factor 2.5 at each evaluation proof. Kohrita and Towa [42] instantiate their construction using a randomized variant of KZG commitments where the prover's randomness consists of a constant number of field elements. The latter construction is obtained from [56] and inherits its randomizable proofs. It may be possible to combine the univariate-to-multilinear transformation of [42] with our simulation-extractable univariate PCS of Section C in order to obtain a more efficient multilinear SE-PCS with a constant number of pairing evaluations at the verifier. This would require a simulation-extractable variant of the batch-degre-check argument of [42] and we leave it for future work.

As of today, the results of [31,24,41] are not known to apply to linear-time-prover multilinear PIOPs like [18]. Moreover, even if they were, they would incur a significant overhead when transforming multilinear PCS into univariate ones.

In the relaxed notion of SE-PCS formalized by Faonio *et al.* [24], simulation-extractability is defined with respect to a policy. They showed that KZG commitments satisfy their relaxed security notion for random evaluation inputs, which

suffices to build simulation-extractable SNARKs. Here, we show that a stronger and simpler-to-state flavor of SE-PCS is achievable at a quite moderate cost in the combined algebraic group and random oracle model. Compared to the underlying malleable randomized commitment (of which we prove knowledge of an evaluation proof), we just need to introduce one more scalar in the evaluation proof and the number of pairings at the verifier remains exactly the same.

Unlike [24,41], our approach is not generic as we only apply it to HyperPlonk, which is one of the most appealing candidates in terms of efficiency. However, we believe that our multivariate SE-PCS can be applied to other PIOPs that proceed by proving sumcheck relations. A common feature of our construction and [24] is that they both impose stronger (yet, efficiently achievable) requirements on behalf of the underlying PCS instead of relying on PIOPs satisfying unique response properties. At the same time, we depart from [24] in that our SE-PCS candidates have a trapdoor-less simulator while [24] simulates evaluation proofs using "programmable" trapdoors in randomized KZG commitments.

Campanelli *et al.* [16] recently described a SNARK featuring a linear-time prover, $O(1)$-size proofs and a universal SRS whose size only grows with the square root of the circuit size. Their approach consists in proving knowledge of a Spartan proof [51] using Groth16 [37]. It is plausible that existing techniques [14,6,3] allowing to make Groth16 simulation-extractable carry over to achieve simulation-extractability in Testudo [16]. Compared to HyperPlonk, Testudo [16] provides shorter proof/SRS sizes. On the other hand, its R1CS arithmetization does not immediately support custom gates as HyperPlonk does.

In the context of polynomial commitments, Kohrita and Towa [42, Appendix B.2.2] recently reported an error in the original security proof of Zhang *et al.*'s randomized variant [56] of [47]. A new security proof in the algebraic group model was given in [42, Appendix B] in the univariate case. Our security proof differs from theirs (but relies on the same assumption in the algebraic group model) and carries over to the multivariate case. Another difference with [42] is that we provide a simulation-extractable variant of the multilinear PCS from [56] whereas [42] does not consider simulation-extractability.

In a concurrent work, Fleischhacker *et al.* [27] consider simulation-extractable aggregatable vector commitments as a building block for non-interactive aggregatable lotteries. They require a strong form of simulation-extractability while allowing to aggregate opening proofs for multiple commitments on the same vector positions. They realized their primitive using randomized KZG commitments and enforce simulation-extractability by requiring each KZG commitment to come with an evaluation proof on a random input. Our constructions differ from theirs in their design and goals since they intentionally preserve some homomorphism on evaluation proofs (but require each KZG commitment to come with a proof of "input awareness") whereas we leave commitments unchanged but disallow mix-and-matches on evaluation proofs.

2 Background and Definitions

For any positive integer ℓ, we denote by $[\ell]$ the set $\{1, \ldots, \ell\}$. For a positive integer x, we denote by $\langle x \rangle$ its binary representation. When $\boldsymbol{x} = x_1 \ldots x_\mu \in \{0,1\}^\mu$, we denote by $[x]$ the integer $[x] = \sum_{i=1}^{\mu} 2^{i-1} \cdot x_i$.

For a positive integer μ, we denote by $B_\mu = \{0,1\}^\mu$ the Boolean hypercube of dimension μ. For an integer $d > 0$, we define $\mathcal{W}_{d,\ell} = \{0, \ldots, d\}^\ell$. For convenience, we also define the set

$$\mathcal{U}_{d,\ell} = \big\{(i_1, \ldots, i_\ell) \in \{0, \ldots, d\} \times \{0,1\}^{\ell-3} \times \{0,1,2\} \times \{0,1,2,3\}\big\}.$$

2.1 Definitions for Polynomials

In the following, we denote by $\mathbb{Z}_p^{(\leq d)}[X]$ the set of polynomials of degree at most $d \in \mathbb{N}$ with coefficients in $\mathbb{Z}_p$. In the case of ℓ-variate polynomials, we denote by $\mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\ell]$ the set of polynomials that have degree $\leq d$ in each variable.

Definition 1. *For every function $f : B_\mu \to \mathbb{Z}_p$, there exists a unique multilinear polynomial $\tilde{f} \in \mathbb{Z}_p^{(\leq 1)}[X_1, \ldots, X_\mu]$ such that $\tilde{f}(\boldsymbol{b}) = f(\boldsymbol{b})$ for all $\boldsymbol{b} \in B_\mu$. This polynomial $\tilde{f}$ is called the multilinear extension of the function f and it is obtained as $\tilde{f}[X_1, \ldots, X_\mu] = \sum_{\boldsymbol{b} \in B_\mu} f(\boldsymbol{b}) \cdot eq_{\boldsymbol{b}}[X_1, \ldots, X_\mu]$, for the multilinear polynomial $eq_{\boldsymbol{b}}[X_1, \ldots, X_\mu] = \prod_{i=1}^{\mu}(b_i \cdot X_i + (1 - b_i) \cdot (1 - X_i))$*

2.2 Hardness Assumptions

Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be cyclic groups of prime order p that are equipped with a bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. We rely on the hardness of the following problem, which has been widely used in pairing-based SNARKs.

Definition 2 ([28]). *Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be asymmetric bilinear groups of prime order p. For integers m, n, the (m,n)-**Discrete Logarithm** $((m,n)$-DLOG) problem is, given $(g, g^\alpha, g^{(\alpha^2)}, \ldots, g^{(\alpha^m)}, \hat{g}, \hat{g}^\alpha, \ldots, \hat{g}^{(\alpha^n)})$ where $\alpha \xleftarrow{R} \mathbb{Z}_p$, $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$, to compute $\alpha \in \mathbb{Z}_p$.*

2.3 Succinct Non-interactive Arguments

An indexed relation is a set of triples $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ where $\mathbb{i}$ is the index, $\mathbb{x}$ is the instance and $\mathbb{w}$ is the witness. Given an index $\mathbb{i}$, $\mathcal{R}_\mathbb{i}$ denotes the restriction of $\mathcal{R}$ to $\{(\mathbb{x}, \mathbb{w}) : (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$. Typically, $\mathbb{i}$ is an arithmetic circuit over a finite field, $\mathbb{x}$ is the public input and $\mathbb{w}$ is the private input.

A (preprocessing) succinct non-interactive zero-knowledge argument of knowledge (SNARK) consists of algorithms $(\mathsf{CRS\text{-}Gen}, \mathsf{PreProcess}, \mathsf{Prove}, \mathsf{Verify})$ with the following specifications. On input of a security parameter $\lambda \in \mathbb{N}$ and (optionally) other parameters, $\mathsf{CRS\text{-}Gen}$ generates a universal common reference string srs and a simulation trapdoor τ; $\mathsf{PreProcess}$ is a deterministic algorithm that

takes as input the universal common references string srs and an index $\mathbb{i}$ describing a circuit $\mathcal{C}$ in order to generate a short circuit-dependent reference string vp (which can be seen as a digest of $\mathbb{i}$) and a longer one pp; Algorithm Prove takes as input the common reference string pp, an index $\mathbb{i}$, a statement $\mathbb{x}$ and a witness $\mathbb{w}$ and outputs a proof π; Verify takes as input vp, a statement $\mathbb{x}$ and a proof π and returns 0 or 1. Correctness requires that proofs honestly generated by the prover are always (or with overwhelming probability) accepted by the verifier.

The $\mathsf{PreProcess}$ algorithm is run exactly once for each circuit (in contrast with the $\mathsf{CRS\text{-}Gen}$ algorithm which generates a universal SRS that can be re-used for any circuit of a priori bounded size). The verifier's preprocessed reference string vp is required to have at most poly-logarithmic length in the size of $\mathcal{C}$. We assume that pp and vp are uniquely determined by srs and $\mathcal{C}$.

From a security point of view, NIZK argument systems should satisfy two properties. The *zero-knowledge* property requires that proofs leak no information about the witness. This is formalized by asking that the trapdoor τ (hidden in pp) allows simulating proofs that are (statistically or computationally) indistinguishable from real proofs. The (non-black-box) *knowledge-soundness* property requires that there exists an extractor that can compute a witness whenever the adversary generates a valid proof. The extractor has access to the adversary's internal state, including its random coins. In a NIZK argument for a relation $\mathcal{R}$, these properties are defined below.

Completeness: For any $\lambda \in \mathbb{N}$, any index $\mathbb{i}$, and any statement-witness pair $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathbb{i}}$, we have

$$\Pr\left[\mathsf{Verify}_{\mathsf{vp}}(\mathbb{i}, \mathbb{x}, \pi) = 1 \mid (\mathsf{srs}, \tau) \leftarrow \mathsf{CRS\text{-}Gen}(1^\lambda), \pi \leftarrow \mathsf{Prove}_{\mathsf{pp}}(\mathbb{i}, \mathbb{x}, \mathbb{w})\right] = 1 - \mathsf{negl}(\lambda)$$

for some negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{N}$.

Knowledge-Soundness: For any PPT adversary $\mathcal{A}$, there is a PPT extractor $\mathcal{E}_{\mathcal{A}}$ that has access to $\mathcal{A}$'s internal state and random coins ρ such that

$$\Pr\left[\mathsf{Verify}_{\mathsf{vp}}(\mathbb{i}, \mathbb{x}, \pi) = 1 \ \wedge \ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R} \ \mid \ (\mathsf{srs}, \tau) \leftarrow \mathsf{CRS\text{-}Gen}(1^\lambda), (\mathbb{i}, \mathbb{x}, \pi) \leftarrow \mathcal{A}(\mathsf{srs}; \rho), \ w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{srs}, (\mathbb{i}, \mathbb{x}, \pi), \rho)\right] = \mathsf{negl}(\lambda).$$

(Statistical) Zero-knowledge: There exists a PPT simulator Sim such that, for any $\lambda \in \mathbb{N}$, and any pair $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, the distributions $D_0 = \{\pi \leftarrow \mathsf{Prove}_{\mathsf{pp}}(\mathbb{i}, \mathbb{x}, \mathbb{w}) : (\mathsf{srs}, \tau) \leftarrow \mathsf{CRS\text{-}Gen}(1^\lambda)\}$ and $D_1 = \{\pi \leftarrow \mathsf{Sim}(\mathsf{srs}, \tau, \mathbb{i}, \mathbb{x}) : (\mathsf{srs}, \tau) \leftarrow \mathsf{CRS\text{-}Gen}(1^\lambda)\}$ are statistically close.

Simulation-extractability strengthens knowledge-soundness by giving the adversary access to an oracle that simulates proofs for possibly false statements.

Simulation-Extractability: For any PPT adversary $\mathcal{A}$, there is a PPT extractor $\mathcal{E}_{\mathcal{A}}$ that has access to $\mathcal{A}$'s internal state/randomness ρ such that

$$\Pr\Big[\mathsf{Verify}_{\mathsf{vp}}(\mathbb{i},\mathbb{x},\pi)=1 \;\wedge\; (\mathbb{i},\mathbb{x},\mathbb{w})\notin\mathcal{R} \;\wedge\; (\mathbb{i},\mathbb{x},\pi)\notin Q \mid (\mathsf{srs},\tau)\leftarrow\mathsf{CRS\text{-}Gen}(1^\lambda),\ (\mathbb{i},\mathbb{x},\pi,\mathsf{lbl})\leftarrow\mathcal{A}^{\mathsf{SimProve}}(\mathsf{srs};\rho),\ \mathbb{w}\leftarrow\mathcal{E}_{\mathcal{A}}(\mathsf{srs},(\mathbb{i},\mathbb{x},\pi),\rho,Q)\Big]=\mathsf{negl}(\lambda)\ ,$$

where $\mathsf{SimProve}(\mathsf{srs},\tau,\cdot,\cdot)$ is an oracle that returns a simulated proof $\pi\leftarrow\mathsf{Sim}(\mathsf{srs},\tau,\mathbb{i},\mathbb{x})$ for a given $(\mathbb{i},\mathbb{x})$ and $Q=\{(\mathbb{i}_j,\mathbb{x}_j,\pi_j)\}_j$ denotes the set of queried statements and the simulated proofs returned by $\mathsf{SimProve}$.

2.4 Algebraic Group Model

The algebraic group model (AGM) [28] is an idealized model, where the adversary is modeled as an algebraic algorithm. Algebraic algorithms generalize generic algorithms in the sense that they only compute group elements as linear combinations of group elements observed so far. Therefore, whenever they output a group element $X\in\mathbb{G}$, they also provide a representation $\{\alpha_i\}_{i=1}^N$ of $X=\prod_{i=1}^N g_i^{\alpha_i}$ as a function of previously seen elements $(g_1,\ldots,g_N)\in\mathbb{G}^N$ of the same group. Unlike generic algorithms, algebraic ones can freely exploit the structure of the group. Due to its generality and because it provides a powerful framework that simplifies the security analyzes of complex protocols, the AGM has been widely used to prove the security of SNARKs.

2.5 Polynomial Commitments

We first recall the syntax of polynomial commitments [39]. We restrict ourselves to polynomials over a field and where the evaluation protocol Eval is non-interactive. We allow Eval and Verify to take as input a label consisting of public data that should be bound to evaluation proofs in a non-malleable way.

Definition 3. *A polynomial commitment scheme (PCS)* $\Gamma=(\mathsf{CRS\text{-}Gen},\mathsf{Com},\mathsf{Eval},\mathsf{Verify})$ *is a tuple of (possibly randomized) algorithms where:*

- $\mathsf{CRS\text{-}Gen}$ *inputs a security parameters and (optionally) the number* ℓ *of variables and an upper bound* d *on the degree of committed polynomials in each variable. It outputs a common reference string* srs *that specifies the field* $\mathbb{F}$ *for which committed polynomials live in* $\mathbb{F}^{(\leq d)}[X_1,\ldots,X_\ell]$ *and (optionally) a simulator trapdoor* τ*. The reference string* srs *is implicitly taken as input by all other algorithms hereunder.*
- $\mathsf{Com}_{\mathsf{srs}}$ *is a (possibly randomized) algorithm that takes as input a polynomial* $f\in\mathbb{F}^{(\leq d)}[X_1,\ldots,X_\ell]$ *and outputs a commitment* C *to* f*, together with the state information* aux *allowing to open* C*. We assume that* aux *contains the randomness allowing to compute the commitment* C*.*

- $\mathsf{Eval}_{\mathsf{srs}}$ *is a (possibly randomized) algorithm that inputs a commitment* C *together its the corresponding state information* aux*, an input* $\boldsymbol{z} \in \mathbb{F}^{\ell}$*, an output* $y \in \mathbb{F}$ *and (optionally) a label* lbl*. If* $y = f(\boldsymbol{z})$*, it outputs a proof* $\boldsymbol{\pi}$ *that* $y = f(\boldsymbol{z})$*. If* $y \neq f(\boldsymbol{z})$*, it returns* $\perp$*.*
- $\mathsf{Verify}_{\mathsf{srs}}$ *is a (usually deterministic) algorithm that inputs a commitment* C*, an input* $\boldsymbol{z} \in \mathbb{F}^{\ell}$*, a claimed output* $y \in \mathbb{F}$*, a candidate proof* $\boldsymbol{\pi}$*, and (optionally) a label* lbl*. It outputs* 0 *or* 1*.*

A PCS is called *succinct* if the size of commitments C and evaluation proofs $\boldsymbol{\pi}$ grows at most logarithmically with the degree d of committed polynomials.

In terms of security, a PCS should satisfy the binding property of standard commitment schemes, which is the infeasiliby of opening a given commitment to distinct polynomials f, f'. The construction of SNARKs requires a PCS satisfying a notion of knowledge-soundness, which is formalized in the same way as in Section 2.3 but we explicitly write it in order to clearly parse statements and witnesses. Together with the binding property, knowledge soundness implies *evaluation-binding*, which captures the adversary's inability to prove two distinct evaluations of a committed polynomial on a given input.

Definition 4. *A PCS is* **knowledge-sound** *if, for any* $\mathsf{srs} \leftarrow \mathsf{CRS\text{-}Gen}(1^{\lambda}, 1^{\ell}, 1^{d})$*,* Eval *is a non-interactive argument of knowledge for the relation*

$$\mathcal{R}_{\mathsf{Eval}}(\mathsf{srs}, 1^{\ell}, 1^{d}) := \big\{ \big(\underbrace{(C, y, \boldsymbol{z})}_{\triangleq\, \mathtt{x}}, \underbrace{(f, \mathsf{aux})}_{\triangleq\, \mathtt{w}} \big) : f \in \mathbb{F}^{(\leq d)}[X_1, \ldots, X_{\ell}] \\ \wedge \quad f(\boldsymbol{z}) = y \quad \wedge \quad C = \mathsf{Com}_{\mathsf{srs}}(f; r) \big\}$$

where r *is the randomness contained in* aux*.*

For our purposes, we also consider a notion of *extended knowledge-soundness* that stands between knowledge-soundness and simulation-extractability, which will be defined shortly.

In order to build *zero-knowledge* SNARKs, it is useful to have PCS constructions satisfying the hiding property.

Definition 5. *A PCS is* **hiding** *if, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*,*

$$\Big| \Pr \Big[b' = b : \mathsf{srs} \leftarrow \mathsf{CRS\text{-}Gen}(1^{\lambda}, 1^{\ell}, 1^{d});\ (st, f_0, f_1) \leftarrow \mathcal{A}_0(\mathsf{srs}); \\ b \xleftarrow{R} \{0,1\}; C \leftarrow \mathsf{Com}_{\mathsf{srs}}(f_b); b' \leftarrow \mathcal{A}_1(st, C) \Big] - 1/2 \Big| \leq \mathsf{negl}(\lambda)$$

A PCS is zero-knowledge if its evaluation protocol Eval is zero-knowledge. In the case of a non-interactive Eval, there is a simulator that can use a trapdoor hidden in pp to simulate proofs without using witnesses. Our definition of zero-knowledge is almost identical to the notion called *hiding* in [41] (the main difference is that their definition also involves a simulated setup algorithm).

Definition 6. *A PCS for parameters $d, \ell \in \mathsf{poly}(\lambda)$ is* **zero-knowledge** *if, for any polynomial $Q \in \mathsf{poly}(\lambda)$, and any adversary $\mathcal{A}$, there is a simulator $\mathcal{S} = (\mathsf{SimCom}, \mathsf{Sim})$ s.t. $|\Pr[\mathbf{Real}_{\mathcal{A},Q}(1^\lambda) \Rightarrow 1] - \Pr[\mathbf{Ideal}_{\mathcal{A},Q}(1^\lambda) \Rightarrow 1]| \leq \mathsf{negl}(\lambda)$ for the following experiments.*

$\mathbf{Real}_{\mathcal{A},Q}(1^\lambda)$:

1. $(\mathsf{srs}, \tau) \leftarrow \mathsf{CRS\text{-}Gen}(1^\lambda, 1^\ell, 1^d)$
2. $ctr = 0$; $st = \varepsilon$
3. $(f, st) \leftarrow \mathcal{A}(\mathsf{srs}, st)$
   If $f \notin \mathbb{F}^{(\leq d)}[X_1, \ldots, X_\ell]$ *return* 0.
4. $(C, \mathsf{aux}) \leftarrow \mathsf{Com}_{\mathsf{srs}}(f; r_f)$
5. $(k, st) \leftarrow \mathcal{A}(\mathsf{srs}, C, st)$
6. *For* $i = 1$ *to* k
   a. $(\boldsymbol{z}_i, \mathsf{lbl}_i, st)$
      $\leftarrow \mathcal{A}(\mathsf{srs}, \{f(\boldsymbol{z}_j), \pi_j\}_{j=1}^{i-1}, st)$
   b. $\pi_i \leftarrow \mathsf{Eval}_{pp}(C, \boldsymbol{z}_i, f(\boldsymbol{z}_i), \mathsf{aux}, \mathsf{lbl}_i)$
7. *If* $ctr < Q$, *set* $ctr = ctr + 1$
   and go to step 3
8. $d \leftarrow \mathcal{A}(C, f(\boldsymbol{z}_k), \pi_k, st)$
9. *Return* d.

$\mathbf{Ideal}_{\mathcal{A},Q}(1^\lambda)$:

1. $(\mathsf{srs}, \tau) \leftarrow \mathsf{CRS\text{-}Gen}(1^\lambda, 1^\ell, 1^d)$
2. $ctr = 0$; $st = \varepsilon$
3. $(f, st) \leftarrow \mathcal{A}(\mathsf{srs}, st)$
   If $f \notin \mathbb{F}^{(\leq d)}[X_1, \ldots, X_\ell]$ *return* 0.
4. $(C, \widetilde{\mathsf{aux}}) \leftarrow \mathsf{SimCom}_{\mathsf{srs}}(\tau)$
5. $(k, st) \leftarrow \mathcal{A}(\mathsf{srs}, C, st)$
6. *For* $i = 1$ *to* k
   a. $(\boldsymbol{z}_i, \mathsf{lbl}_i, st)$
      $\leftarrow \mathcal{A}(\mathsf{srs}, \{f(\boldsymbol{z}_j), \pi_j\}_{j=1}^{i-1}, st)$
   b. $\pi_i \leftarrow \mathsf{Sim}_{pp}(\tau, \boldsymbol{z}_i, f(\boldsymbol{z}_i), \widetilde{\mathsf{aux}}, \mathsf{lbl}_i)$
7. *If* $ctr < Q$, *set* $ctr = ctr + 1$
   and go to step 3
8. $d \leftarrow \mathcal{A}(C, f(\boldsymbol{z}_k), \pi_k, st)$
9. *Return* d.

Remark. In the ideal experiment of Definition 6, we assume that the output of SimCom contains the state information $\widetilde{\mathsf{aux}}$ allowing to re-compute C. If $(C, \widetilde{\mathsf{aux}})$ is a possible output of SimCom, we say that aux is *consistent* with C.

The notion of simulation-extractability is formalized for PCS in the same way as in general succinct NIZK arguments.

Definition 7. *A PCS is* **simulation-extractable** *if, for any PPT adversary $\mathcal{A}$, there is a PPT extractor $\mathcal{E}_\mathcal{A}$ that has access to $\mathcal{A}$'s internal state/randomness ρ such that*

$$\begin{aligned}\Pr\big[&\mathsf{Verify}_{\mathsf{srs}}(C, \boldsymbol{z}, y, \pi, \mathsf{lbl}) = 1 \\ &\wedge \quad \big((C, y, \boldsymbol{z}), (f, \mathsf{aux})\big) \notin \mathcal{R}_{\mathsf{Eval}}(\mathsf{srs}, 1^\ell, 1^d) \quad \wedge \quad \big((C, y, \boldsymbol{z}), \pi, \mathsf{lbl}\big) \notin Q \mid \\ &(\mathsf{srs}, \tau) \leftarrow \mathsf{CRS\text{-}Gen}(1^\lambda),\ \big((C, y, \boldsymbol{z}), \pi, \mathsf{lbl}\big) \leftarrow \mathcal{A}^{\mathsf{Sim}_{\mathsf{srs}}(\tau, \cdot, \cdot, \cdot)}(\rho), \\ &\qquad (f, \mathsf{aux}) \leftarrow \mathcal{E}_\mathcal{A}(\mathsf{srs}, ((C, y, z), \pi, \mathsf{lbl}), \rho, Q)\big] = \mathsf{negl}(\lambda)\ ,\end{aligned}$$

where $\mathsf{Sim}_{\mathsf{srs}}(\tau, \cdot, \cdot, \cdot)$ is a simulation oracle that takes as input a statement-label pair $(\mathbb{x} = (C, \boldsymbol{z}, y), \mathsf{lbl})$ and an auxiliary information $\widetilde{\mathsf{aux}}$. If $\widetilde{\mathsf{aux}}$ is inconsistent with C, it returns $\perp$. Otherwise, it returns a simulated proof $\pi \leftarrow \mathsf{Sim}_{\mathsf{srs}}(\tau, (C, \boldsymbol{z}, y), \mathsf{lbl}, \widetilde{\mathsf{aux}})$ and $Q = \{(\mathbb{x}_i = (C_i, \boldsymbol{z}_i, y_i), \pi_i, \mathsf{lbl}_i)\}_i$ denotes the set of queried statements and the simulated proofs returned by $\mathsf{Sim}_{\mathsf{srs}}$.

Faonio *et al.* [24] considered relaxed notion of simulation-extractable PCS, which is defined w.r.t. policies. In their definition, the forgery is required to be made

on a random evaluation point $\boldsymbol{z}$ (derived from a random oracle) and simulation-queries are made selectively, before the generation of the commitment key. Although we could also use a relaxed notion of SE for our purposes, we chose to work with Definition 7 since it can be achieved efficiently in the AGM+ROM.

Extended Knowledge-Soundness. The notion of extended knowledge-soundness is weaker than simulation-extractability and only preserves extractability when the adversary can see honestly generated proofs. It is thus similar to *true simulation-extractability* [23], except that the adversary obtains real proofs (instead of simulated ones) of true statements. The definition is identical to Definition 7 but, instead of having oracle access to a simulator, the adversary is given a $\mathsf{Prove}(\cdot)$ oracle taking as input a polynomial f and a set $S \subset \mathbb{F}^\ell$ of evaluation inputs and returning a commitment C to f (obtained as $(C, \mathsf{aux}) \leftarrow \mathsf{Com}_{\mathsf{srs}}(f)$) together with real evaluation proofs $\{\pi_i \leftarrow \mathsf{Eval}_{\mathsf{srs}}(C, z_i, f(z_i), \mathsf{aux})\}_{z_i \in S}$. In the AGM, it is easy to see that extended knowledge-soundness is equivalent to standard knowledge-soundness when Com and Eval are algebraic algorithms that compute linear combinations of group elements contained in the SRS.

3 Commitments to Multivariate Polynomials

We first prove the security of the multivariate PCS of Zhang *et al.* [56] in the AGM under the (m, n)-DLOG assumption. In [56], a less simple variant of the scheme was considered. In this variant (which was also used in [54]), each commitment consists of a pair $(g^{f(\alpha_1,\ldots,\alpha_\ell)+r\cdot\alpha_{\ell+1}}, h^{f(\alpha_1,\ldots,\alpha_\ell)+r\cdot\alpha_{\ell+1}})$, where g, h are public generators, in order to extract the coefficients of the committed $f[X_1, \ldots, X_\ell]$ using a variant of the knowledge-of-exponent assumption [20]. In the AGM, we give a security proof for the simpler scheme where each commitment consists of only one group element. Our proof generalizes the one given by Kohrita and Towa [42] to the multivariate case. In Section 4, we will modify the scheme to achieve simulation-extractability. We assume that committed polynomials have bounded degree $\leq d$ in each variable.

3.1 The Multivariate PCS of Zhang *et al.*

When it comes to committing to polynomials in $\mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\ell]$, the multivariate PCS of [56] can be described as follows.

In order to use the scheme in HyperPlonk, we need to allow the prover to commit to (and generate proofs for) polynomials where the number of variables may be smaller than the maximal number ℓ of variables allowed by the SRS. For this reason, when the actual number of variables μ is strictly smaller than ℓ in the verification algorithm, the knowledge extractor does not extract an opening of the commitment C to an μ-variate polynomial $f[X_1, \ldots, X_\mu]$ such that $f(\boldsymbol{z}) = y$ in general. It only extracts an opening of C to an ℓ-variate polynomial $f \in \mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\ell]$ such that $f(\boldsymbol{z}, x_{\mu+1}, \ldots, x_\ell) = y$ for all assignments of $(x_{\mu+1}, \ldots, x_\ell) \in \mathbb{Z}_p^{\ell-\mu}$ (i.e., $f[\boldsymbol{z}, X_{\mu+1}, \ldots, X_\ell] - y$ is the zero polynomial). However, it still ensures that the prover knows a polynomial

$\bar{f}[X_1, \ldots, X_\mu] = f[X_1, \ldots, X_\mu, 0, \ldots, 0]$ such that $\bar{f}(\boldsymbol{z}) = y$. Moreover, when $\boldsymbol{z} \in \mathbb{Z}_p^\mu$ is a random evaluation point (as is the case in sumcheck protocols), the knowledge extractor *does* extract (with overwhelming probability) an opening of C to an μ-variate polynomial $f[X_1, \ldots, X_\mu]$ such that $f(\boldsymbol{z}) = y$.

The multivariate PCS of [56] is a probabilistic version of PST commitments [47]. Its simplified version goes as follows.

CRS-Gen$(1^\lambda, 1^d, 1^\ell)$**:** On input of a security parameter λ, a number of variables ℓ and a degree d such that $(d+1)^\ell \in \mathsf{poly}(\lambda)$, generate the SRS as follows:

1. Choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^{l(\lambda)}$, for some function $l : \mathbb{N} \to \mathbb{N}$, and $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$.
2. Pick $\alpha_1, \ldots, \alpha_\ell, \alpha_r \xleftarrow{R} \mathbb{Z}_p$ and compute $\hat{g}_1, \ldots, \hat{g}_\ell \in \mathbb{G}$, where $\hat{g}_i = \hat{g}^{\alpha_i}$ for each $i \in [\ell]$. Compute $g_r = g^{\alpha_r}$ and $\hat{g}_r = \hat{g}^{\alpha_r}$.
3. For each tuple $\mathcal{I} = (i_1, \ldots, i_\ell) \in \mathcal{W}_{d,\ell}$, compute $g_{\mathcal{I}} = g^{\prod_{j=1}^{\ell} \alpha_j^{i_j}}$.

The public parameters are defined to be

$$\mathsf{srs} = \Big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), \{g_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{d,\ell}}, g_r, \hat{g}_r, \hat{g}, \{\hat{g}_i\}_{i \in [\ell]} \Big)$$

Com$_{\mathsf{srs}}(f)$**:** To commit to $f[X_1, \ldots, X_\mu] \in \mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\mu]$, where $\mu \leq \ell$, choose a random $r \xleftarrow{R} \mathbb{Z}_p$ and compute $C = g^{f(\alpha_1, \ldots, \alpha_\mu) + r \cdot \alpha_r}$ using $\{g_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{d,\ell}}$ and g_r. Then, output C and $(\mathsf{aux}, f) = (r, f)$.

Eval$_{\mathsf{srs}}\big(C, \boldsymbol{z}, y, \mathsf{aux}\big)$**:** given a commitment C, a witness $\mathsf{aux} = (r, f)$, an input $\boldsymbol{z} = (z_1, \ldots, z_\mu) \in \mathbb{Z}_p^\mu$ of dimension $\mu \leq \ell$, and an output $y = f(\boldsymbol{z}) \in \mathbb{Z}_p$, return $\perp$ if $y \neq f(\boldsymbol{z})$. Otherwise, do the following:

1. Using [47, Lemma 1], compute $\{Q_i[X_1, \ldots, X_\mu]\}_{i=1}^\mu$ such that

$$f[X_1, \ldots, X_\mu] - y = \sum_{i=1}^{\mu} Q_i[X_1, \ldots, X_\mu] \cdot (X_i - z_i)$$

2. Choose $s_1, \ldots, s_\mu \xleftarrow{R} \mathbb{Z}_p$ and compute $\pi_{1,i} = g^{Q_i(\alpha_1, \ldots, \alpha_\mu) + s_i \cdot \alpha_r}$ for each $i \in [\mu]$ together with $\pi_2 = g^{r - \sum_{i=1}^{\mu} s_i \cdot (\alpha_i - z_i)}$.

Return the proof $\boldsymbol{\pi} = \big((\pi_{1,i})_{i=1}^\mu, \pi_2\big) \in \mathbb{G}^{\mu+1}$.

Verify$_{\mathsf{srs}}\big(C, y, \boldsymbol{z}, \boldsymbol{\pi}\big)$**:** Given $C \in \mathbb{G}$, an input $\boldsymbol{z} = (z_1, \ldots, z_\mu) \in \mathbb{Z}_p^\mu$, a claimed evaluation $y \in \mathbb{Z}_p$, and a purported proof $\boldsymbol{\pi}$, return 0 if $\boldsymbol{\pi}$ does not parse properly. Return 1 if the following equality holds and 0 otherwise:

$$e(C \cdot g^{-y}, \hat{g}) = \prod_{i=1}^{\mu} e(\pi_{i,1}, \hat{g}_i \cdot \hat{g}^{-z_i}) \cdot e(\pi_2, \hat{g}_r) \tag{2}$$

The scheme is correct since equation (2) uses the pairing to check the equality

$$f(\alpha_1, \ldots, \alpha_\mu) + r \cdot \alpha_r - y = \sum_{i=1}^{\mu} (\alpha_i - z_i) \cdot (Q_i(\alpha_1, \ldots, \alpha_\mu) + s_i \cdot \alpha_r) + \alpha_r \cdot \Big(r - \sum_{i=1}^{\mu} s_i \cdot (\alpha_i - z_i) \Big).$$

We now prove knowledge-soundness under the $(d\ell, d\ell)$-DLOG assumption in the AGM. The proof of Theorem 1 (which can be found in Supplementary Material A.1) differs from the one given by [42] in the univariate case.

Theorem 1. *In the AGM and under the $(d\ell, d\ell)$-DLOG assumption, the scheme is an (extended) knowledge-sound argument of knowledge of a polynomial $f \in \mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\ell]$ such that $f(z_1, \ldots, z_\mu, X_{\mu+1}, \ldots, X_\ell) = y$ for any assignment of $(X_{\mu+1}, \ldots, X_\ell)$, where $(y, \boldsymbol{z}) \in \mathbb{Z}_p \times \mathbb{Z}_p^\mu$. Moreover, if $\boldsymbol{z} = (z_1, \ldots, z_\mu)$ is a random input, the knowledge extractor outputs $f \in \mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\mu]$ and $r \in \mathbb{Z}_p$ such that $C = g^{f(\alpha_1, \ldots, \alpha_\mu) + r \cdot \alpha_r}$ with overwhelming probability.*

Using standard batching techniques, multiple evaluations of committed polynomials can be proven at once on a common input. The scheme and the proof of Theorem 1 easily extend, as explained in Supplementary Material A.2.

Zero-knowledge. We note that the trapdoor $\alpha_r \in \mathbb{Z}_p^*$ can be used to simulate proofs for a given commitment C and a given pair $(y, \boldsymbol{z})$. Indeed, the simulator can choose $\theta_{i,1} \xleftarrow{R} \mathbb{Z}_p$ for each $i \in [\mu]$ in order to compute $\pi_{i,1} = g^{\theta_{i,1}}$ for all $i \in [\mu]$ and $\pi_2 = \left(C \cdot g^{-y} \cdot \prod_{i=1}^{\mu} (g_i \cdot g^{-z_i})^{-\theta_{i,1}}\right)^{1/\alpha_r}$. This yields a simulated $\left((\pi_{1,i})_{i=1}^{\mu}, \pi_2\right)$ that is distributed as a real proof.

While the use of α_r yields a zero-knowledge simulator, it is not trapdoor-less and cannot be used to obtain simulation-extractability Moreover, the scheme is clearly not simulation-extractable as it is since a proof for $(C, y, \boldsymbol{z})$ is also a proof for $(C \cdot g, y + 1, \boldsymbol{z})$. In Section 4, we show how to thwart such attacks.

3.2 Enforcing a Special Shape for Committed Polynomials

In order to achieve zero-knowledge at the PIOP level, HyperPlonk [18, Appendix A] suggests to transform ℓ-variate multilinear polynomials into polynomials that agree with the original polynomials everywhere on the hypercube B_ℓ but evaluate to random-looking values outside B_ℓ. These "almost multilinear" polynomials (where all variables have degree 1, except one) can be written as a sum

$$f[X_1, \ldots, X_\ell] = f'[X_1, \ldots, X_\ell] + R[X_\mu] \cdot X_\mu \cdot (X_\mu - 1), \tag{3}$$

for some $\mu \in [\ell]$, where $f' \in \mathbb{Z}_p^{(\leq 1)}[X_1, \ldots, X_\ell]$ is the original multilinear polynomial and $R[X_\mu]$ is univariate of degree t for some $t \geq 0$. In order to achieve zero-knowledge in sumcheck-based protocols, polynomials of the form (3) have the properties that: (i) f and f' agree everywhere on B_ℓ; (ii) Any set of $t + 1$ evaluations of f outside B_ℓ are random and independent (since $R[X_\mu]$ has degree t) and can be easily simulated in zero-knowledge.

In the case of HyperPlonk, the verifier has to obtain evaluation proofs on correlated inputs sharing the same value of X_μ, in which case the above masking technique does not quite suffice to ensure that evaluations look independent.

Therefore we need to slightly modify the masking technique (3) and encode the original $f' \in \mathbb{Z}_p^{(\leq 1)}[X_1, \ldots, X_\ell]$ as

$$f[X_1, \ldots, X_\ell] = f'[X_1, \ldots, X_\ell] + R[X_1 + X_{\mu+1}] \cdot X_\mu \cdot (X_\mu - 1), \qquad (4)$$

where $\mu + 1 < \ell$ (the reason why this modification works will become clear in the proof of Theorem 5 in Section 5.1 and in the proof of Lemma 3).

In the following, we need to consider multivariate PCS schemes where the SRS allows committing to polynomials of the form (4). For the soundness analysis, the verifier should be convinced that a committed polynomial is really of this form. This can be ensured by a careful choice of the group elements available in the SRS. At the same time, we need to preserve the prover's ability to commit to both univariate and multivariate polynomials of larger degree $d > 1$. Our solution is to have the prover commit to these polynomials in $\mathbb{G}$ (instead of $\hat{\mathbb{G}}$).

The Modified PCS. We now describe a variant of the scheme in Section 3.1 where we force each committed polynomial to be of the form (4), where the number of evaluations t is set to $t = 3$. Although, the committer is meant to commit to almost multilinear polynomials, the $\mathsf{CRS\text{-}Gen}$ algorithm still inputs a parameter $\bar{d} > 1$ which is the maximal variable-degree of commitments to be committed in the first source group $\mathbb{G}$. We also assume that $\bar{d} \geq t + 2$ in order to allow the prover to compute evaluation proofs in $\mathbb{G}$.

In the notations, the set $\mathcal{U}_{\bar{d},\ell} \subset \mathcal{W}_{\bar{d},\ell}$ determines a set of monomials in the exponent and is defined in Section 2 to retain a linear-size SRS in Section 5.

$\mathsf{CRS\text{-}Gen}(1^\lambda, 1^d, 1^\ell)$: On input of a security parameter λ, a number of variables ℓ and a degree $\bar{d}$ such that $\bar{d} \cdot 2^\ell \in \mathsf{poly}(\lambda)$, generate the SRS as follows:

1. Choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^{l(\lambda)}$, for some function $l : \mathbb{N} \to \mathbb{N}$, and $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$.
2. Pick $\alpha_1, \ldots, \alpha_\ell, \alpha_r \xleftarrow{R} \mathbb{Z}_p$. Compute $g_r = g^{\alpha_r}$ and $\hat{g}_r = \hat{g}^{\alpha_r}$.
3. For each $\mathcal{I} = (i_1, \ldots, i_\ell) \in \mathcal{U}_{\bar{d},\ell}$, compute $g_{\mathcal{I}} = g^{\prod_{j=1}^{\ell} \alpha_j^{i_j}}$.
4. For each $\mathcal{I} = (i_1, \ldots, i_\ell) \in \mathcal{W}_{1,\ell}$, compute $\hat{g}_{\mathcal{I}} = \hat{g}^{\prod_{j=1}^{\ell} \alpha_j^{i_j}}$. Compute $\left\{\hat{g}^{(\alpha_1+\alpha_{\mu+1})^i \cdot \alpha_\mu \cdot (\alpha_\mu - 1)}\right\}_{i=0}^{3}$, where $\mu = \ell - 1$.

The public parameters are defined to be

$$\mathsf{srs} = \Big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g_r, \hat{g}_r, \{g_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{U}_{\bar{d},\ell}}, \{\hat{g}_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{1,\ell}}, \left\{\hat{g}^{(\alpha_1+\alpha_{\mu+1})^i \cdot \alpha_\mu \cdot (\alpha_\mu - 1)}\right\}_{i=0}^{3} \Big).$$

$\mathsf{Com}_{\mathsf{srs}}(f)$: To commit to a polynomial $f[X_1, \ldots, X_{\mu'}] \in \mathbb{Z}_p[X_1, \ldots, X_{\mu'}]$ of the form (4), where $\mu' \leq \ell$, choose $r \xleftarrow{R} \mathbb{Z}_p$ and compute $\hat{C} = \hat{g}^{f(\alpha_1, \ldots, \alpha_{\mu'}) + r \cdot \alpha_r}$. Then, output $\hat{C}$ and $(\mathsf{aux}, f) = (r, f)$.

$\mathsf{Eval}_{\mathsf{srs}}(\hat{C}, \boldsymbol{z}, y, \mathsf{aux})$: given a commitment $\hat{C}$, a witness $\mathsf{aux} = (r, f)$, an input $\boldsymbol{z} = (z_1, \ldots, z_{\mu'}) \in \mathbb{Z}_p^{\mu'}$ of dimension $\mu' \leq \ell$, and an output $y = f(\boldsymbol{z}) \in \mathbb{Z}_p$, return $\perp$ if $y \neq f(\boldsymbol{z})$. Otherwise, do the following:

1. Using [47, Lemma 1], compute polynomials $\{Q_i[X_1,\ldots,X_{\mu'}]\}_{i=1}^{\mu'}$ such that $f[X_1,\ldots,X_{\mu'}] - y = \sum_{i=1}^{\mu'} Q_i[X_1,\ldots,X_{\mu'}]\cdot(X_i - z_i)$.
2. Choose $s_1,\ldots,s_{\mu'} \xleftarrow{R} \mathbb{Z}_p$ and compute $\pi_{1,i} = g^{Q_i(\alpha_1,\ldots,\alpha_{\mu'})+s_i\cdot\alpha_r}$ for each $i\in[\mu']$ together with $\pi_2 = g^{r-\sum_{i=1}^{\mu'} s_i\cdot(\alpha_i - z_i)}$.

Return the proof $\boldsymbol{\pi} = \big((\pi_{1,i})_{i=1}^{\mu'}, \pi_2\big) \in \mathbb{G}^{\mu'+1}$.

$\mathsf{Verify}_{\mathsf{srs}}(\hat{C}, y, \boldsymbol{z}, \boldsymbol{\pi})$: Given $\hat{C}\in\hat{\mathbb{G}}$, an input $\boldsymbol{z} = (z_1,\ldots,z_{\mu'}) \in \mathbb{Z}_p^{\mu'}$, a claimed evaluation $y\in\mathbb{Z}_p$, and a candidate proof $\boldsymbol{\pi}$, return 0 if $\boldsymbol{\pi}$ does not parse properly. Then, return 1 if the following equation holds and 0 otherwise:

$$e(g, \hat{C}\cdot\hat{g}^{-y}) = \prod_{i=1}^{\mu'} e(\pi_{i,1}, \hat{g}^{\alpha_i}\cdot\hat{g}^{-z_i})\cdot e(\pi_2, \hat{g}_r) \tag{5}$$

Although the commitment lives in a different group than evaluation proofs, the proof of Theorem 1 easily extends to give the following result.

Theorem 2. *In the AGM+ROM, under the $(d\ell, d\ell)$-DLOG assumption, the scheme is a knowledge-sound argument of knowledge of an ℓ-variate polynomial $f\in\mathbb{Z}_p[X_1,\ldots,X_\ell]$ of the form (4) such that $f(z_1,\ldots,z_{\mu'},X_{\mu'+1},\ldots,X_\ell) = y$ for any assignment of $(X_{\mu'+1},\ldots,X_\ell)$, where $(y,\boldsymbol{z})\in\mathbb{Z}_p\times\mathbb{Z}_p^{\mu'}$. Moreover, if the tuple $\boldsymbol{z} = (z_1,\ldots,z_{\mu'})$ is a random evaluation input, the knowledge extractor outputs an f of the form (4) and $r\in\mathbb{Z}_p$ such that $\hat{C} = \hat{g}^{f(\alpha_1,\ldots,\alpha_{\mu'})+r\cdot\alpha_r}$ with overwhelming probability.* (The proof is given in Supplementary Material A.3.)

4 A Simulation-Extractable Variant of Zhang *et al.*'s Polynomial Commitment

We now apply the Fiat-Shamir transform to build a simulation-extractable multivariate PCS. We apply the idea to the scheme of Section 3.2 because it is the version that we need to compile the HyperPlonk PIOP in its zero-knowledge version. However, the same technique applies to the PCS of Section 3.1 and the batch version of Supplementary Material A.2.

4.1 Description

The construction is almost as efficient as the one of Section 3.1. Indeed, proofs are only longer by one element of $\mathbb{Z}_p$ and the verifier computes the same number of pairings and exponentiations.

$\mathsf{CRS\text{-}Gen}(1^\lambda, 1^{\bar{d}}, 1^\ell)$: On input of a security parameter λ, a number of variables ℓ and a degree d such that $\bar{d}\cdot 2^\ell \in \mathsf{poly}(\lambda)$, generate the SRS as follows:

1. Choose asymmetric bilinear groups $(\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T)$ of prime order $p > 2^{l(\lambda)}$, for some function $l:\mathbb{N}\to\mathbb{N}$, and $g\xleftarrow{R}\mathbb{G}$, $\hat{g}\xleftarrow{R}\hat{\mathbb{G}}$.
2. Pick $\alpha_1,\ldots,\alpha_\ell,\alpha_r\xleftarrow{R}\mathbb{Z}_p$. Compute $g_r = g^{\alpha_r}$ and $\hat{g}_r = \hat{g}^{\alpha_r}$.

3. For each $\mathcal{I} = (i_1, \ldots, i_\ell) \in \mathcal{U}_{\bar{d},\ell}$, compute $g_{\mathcal{I}} = g^{\prod_{j=1}^{\ell} \alpha_j^{i_j}}$, where $\mathcal{U}_{\bar{d},\ell}$ is defined as in Section 2.
4. For each $\mathcal{I} = (i_1, \ldots, i_\ell) \in \mathcal{W}_{1,\ell}$, compute $\hat{g}_{\mathcal{I}} = \hat{g}^{\prod_{j=1}^{\ell} \alpha_j^{i_j}}$. Compute $\left\{\hat{g}^{(\alpha_1+\alpha_{\mu+1})^i \cdot \alpha_\mu \cdot (\alpha_\mu - 1)}\right\}_{i=0}^{3}$ for an arbitrary index $\mu \in [2, \ell-1]$.
5. Choose a hash function $H_{\mathsf{PCS}} : \{0,1\}^* \to \mathbb{Z}_p$ modeled as a random oracle.

The public parameters are defined to be

$$\mathsf{srs} = \Big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g_r, \hat{g}_r, \{g_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{U}_{\bar{d},\ell}}, \{\hat{g}_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{1,\ell}}, \left\{\hat{g}^{(\alpha_1+\alpha_{\mu+1})^i \cdot \alpha_\mu \cdot (\alpha_\mu - 1)}\right\}_{i=0}^{3}, H_{\mathsf{PCS}} \Big)$$

$\mathsf{Com}_{\mathsf{srs}}(f)$: To commit to a polynomial $f[X_1, \ldots, X_{\mu'}] \in \mathbb{Z}_p[X_1, \ldots, X_{\mu'}]$ of the form (4), for some $\mu' \leq \ell$, choose $r \xleftarrow{R} \mathbb{Z}_p$ and compute $\hat{C} = \hat{g}^{f(\alpha_1, \ldots, \alpha_{\mu'}) + r \cdot \alpha_r}$. Then, output $\hat{C}$ and $(\mathsf{aux}, f) = (r, f)$.

$\mathsf{Eval}_{\mathsf{srs}}\big(\hat{C}, \boldsymbol{z}, y, \mathsf{aux}, \mathsf{lbl}\big)$: given a commitment $\hat{C}$, a witness $\mathsf{aux} = (r, f)$, an input $\boldsymbol{z} = (z_1, \ldots, z_{\mu'}) \in \mathbb{Z}_p^{\mu'}$, an output $y = f(\boldsymbol{z}) \in \mathbb{Z}_p$, and a label lbl, return $\perp$ if $y \neq f(\boldsymbol{z})$. Otherwise, do the following:

1. Compute $\big((\pi_{1,i})_{i=1}^{\mu'}, \pi_2\big) \in \mathbb{G}^{\mu'+1}$ satisfying (5) by running the Eval algorithm of Section 3.2.
2. Generate a NIZK proof of knowledge of $\pi_2 \in \mathbb{G}$ satisfying

$$\frac{e(g, \hat{C} \cdot \hat{g}^{-y})}{\prod_{i=1}^{\mu'} e(\pi_{i,1}, \hat{g}_i \cdot \hat{g}^{-z_i})} = e(\pi_2, \hat{g}_r) \tag{6}$$

   Namely,
   a. Choose $R_\pi \xleftarrow{R} \mathbb{G}$ and compute[2] $R = e(R_\pi, \hat{g}_r)$.
   b. Compute $c = H_{\mathsf{PCS}}(\mathsf{lbl}, C, y, \boldsymbol{z}, (\pi_{1,i})_{i=1}^{\mu'}, R) \in \mathbb{Z}_p$.
   c. Compute the response $S_\pi = R_\pi \cdot \pi_2^c$.

Return the proof $\boldsymbol{\pi} = \big(c, (\pi_{1,i})_{i=1}^{\mu'}, S_\pi\big) \in \mathbb{Z}_p \times \mathbb{G}^{\mu'+1}$.

$\mathsf{Verify}_{\mathsf{srs}}\big(\hat{C}, y, \boldsymbol{z}, \boldsymbol{\pi}, \mathsf{lbl}\big)$: Given $\hat{C} \in \hat{\mathbb{G}}$, an input $\boldsymbol{z} = (z_1, \ldots, z_{\mu'}) \in \mathbb{Z}_p^{\mu'}$, a purported evaluation $y \in \mathbb{Z}_p$, and a candidate proof $\boldsymbol{\pi}$ with a label lbl, return 0 if $\boldsymbol{\pi}$ does not parse correctly. Otherwise, compute

$$R = e(S_\pi, \hat{g}_r) \cdot \left(\frac{e(g, \hat{C} \cdot \hat{g}^{-y})}{\prod_{i=1}^{\mu'} e(\pi_{i,1}, \hat{g}^{\alpha_i} \cdot \hat{g}^{-z_i})} \right)^{-c}. \tag{7}$$

If $c = H_{\mathsf{PCS}}(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, (\pi_{1,i})_{i=1}^{\mu'}, R)$, return 1. Otherwise, return 0.

We now prove that the above variant provides (straight-line) simulation-extractability in the combined algebraic group and random oracle model.

[2] The pairing evaluation can be avoided by computing $R_\pi = g^{r_\pi}$ and $R = e(g, \hat{g}_r)^{r_\pi}$.

Theorem 3. *In the AGM+ROM model and under the $(d\ell, d\ell)$-DLOG assumption, the scheme is a simulation-extractable argument of knowledge of a polynomial $f \in \mathbb{Z}_p[X_1, \ldots, X_\ell]$ of the form (4) such that $\hat{C} = \hat{g}^{f(\alpha_1,\ldots,\alpha_\ell)+\alpha_r \cdot r}$ and $f(\boldsymbol{z}, x_{\mu'+1}, \ldots, x_\ell) = y$ for any $(x_{\mu'+1}, \ldots, x_\ell) \in \mathbb{Z}_p^{\ell-\mu'}$, where $(y, \boldsymbol{z}) \in \mathbb{Z}_p \times \mathbb{Z}_p^{\mu'}$.*

Proof. Given an algebraic adversary $\mathcal{A}$ in the experiment of Definition 7, we build an algorithm $\mathcal{B}$ that either extracts a witness or, if the extraction fails, solves a $(\bar{d}\bar{\ell}, \bar{d}\bar{\ell})$-DLOG instance $\mathsf{inst} = \big(g, \{g^{(\alpha^i)}\}_{i=1}^{\bar{d}\bar{\ell}}, \hat{g}, \{\hat{g}^{(\alpha^i)}\}_{i=1}^{\bar{d}\bar{\ell}}\big)$ w.h.p.

Algorithm $\mathcal{B}$ first chooses $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_\ell, \rho_r) \xleftarrow{R} \mathbb{Z}_p^{\ell+1}$, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_\ell, \theta_r) \xleftarrow{R} \mathbb{Z}_p^{\ell+1}$. It implicitly sets $\alpha_i = \rho_i \cdot \alpha + \theta_i$ for each $i \in [\ell]$ and $\alpha_r = \rho_r \cdot \theta + \theta_r$. It can simulate srs from inst and $\{(\rho_i, \theta_i)\}_{i=1}^{\ell}$, (ρ_r, θ_r).

Queries: When $\mathcal{A}$ queries the random oracle, $\mathcal{B}$ returns the previously defined output if it exists and a random element of $\mathbb{Z}_p$ otherwise.

At any time, $\mathcal{A}$ can choose a commitment $\hat{C} \in \mathbb{G}$ and a pair $(y, \boldsymbol{z}) \in \mathbb{Z}_p \times \mathbb{Z}_p^{\mu'}$ and ask for a simulated proof, for some label lbl, that $\hat{C}$ commits to some polynomial $f \in \mathbb{Z}_p[X_1, \ldots, X_{\mu'}]$ of the form (4) such that $f[\boldsymbol{z}, X_{\mu'+1}, \ldots, X_\ell] - y$ is the zero-polynomial. Then, $\mathcal{B}$ simulates a proof by running the HVZK simulator of the Σ-protocol. Namely, it samples $c, t_0, \ldots, t_{\mu'} \xleftarrow{R} \mathbb{Z}_p$, computes

$$S_\pi = g^{t_0} \qquad \text{and} \qquad \pi_{1,i} = g^{t_i} \qquad \forall i \in [\mu']$$

together with $R = e(S_\pi, \hat{g}_r) \cdot \Big(e(g, \hat{C} \cdot \hat{g}^{-y}) / \prod_{i=1}^{\mu'} e(\pi_{i,1}, \hat{g}_i \cdot \hat{g}^{-z_i})\Big)^{-c}$ before programming $c = H_{\mathsf{PCS}}(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, (\pi_{1,i})_{i=1}^{\mu'}, R) \in \mathbb{Z}_p$. If H_{PCS} was already defined for this input, $\mathcal{B}$ aborts. Since R is uniformly distributed over $\mathbb{G}_T$, this only happens with probability $< (Q_H + Q_S)/p$ if Q_H (resp. Q_S) denotes the number of random oracle (resp. simulation) queries. Unless a collision occurs on H_{PCS} in a simulation query, the simulation is perfect since the simulated $\big(c, (\pi_{1,i})_{i=1}^{\mu'}, S_\pi\big) \in \mathbb{Z}_p \times \mathbb{G}^{\mu'+1}$ has the same distribution as a real proof. The probability of a such a collision during the entire game is at most $Q_S(Q_S + Q_H)/p$.

Since $\mathcal{A}$ is algebraic, at each hash query $H_{\mathsf{PCS}}(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, (\pi_{1,i})_{i=1}^{\mu'}, R)$, it must provide a representation of $\hat{C}$ w.r.t. the $\hat{\mathbb{G}}$-elements contained in srs as well as an algebraic representation of each $\{\pi_{1,i}\}_{i=1}^{\ell}$ w.r.t. the $\mathbb{G}$-elements of srs and a representation $\{\omega_{\mathcal{I},\hat{\mathcal{I}}}\}_{\mathcal{I}\in\mathcal{G},\hat{\mathcal{I}}\in\hat{\mathcal{G}}}$ of R as

$$R = \prod_{\mathcal{I}\in\mathcal{G},\hat{\mathcal{I}}\in\hat{\mathcal{G}}} e(g_{\mathcal{I}}, \hat{g}_{\hat{\mathcal{I}}})^{\omega_{\mathcal{I},\hat{\mathcal{I}}}} \tag{8}$$

where $\mathcal{G}$ (resp. $\hat{\mathcal{G}}$) denotes the set of $\mathbb{G}$-elements (resp. $\hat{\mathbb{G}}$-elements) contained in srs. While the representation given by $\mathcal{A}$ can also depend on elements of $\mathbb{G}$ contained in simulated proofs, $\mathcal{B}$ can always find a representation of the form (8) since it computes S_π and $\{\pi_{1,i}\}_{i=1}^{\mu}$ by sampling their logarithms w.r.t. g.

Output: When $\mathcal{A}$ halts, it outputs a commitment $\hat{C}$ and a pair $(y, \boldsymbol{z}) \in \mathbb{Z}_p \times \mathbb{Z}_p^{\mu'}$, for some $\mu' \in [\ell]$, together with a label lbl and a verifying proof $\boldsymbol{\pi}$ that $\hat{C}$ commits to $f[X_1, \ldots, X_\ell]$ such that $y = f[\boldsymbol{z}, X_{\mu'+1}, \ldots, X_\ell]$ for all assignments of

$(X_{\mu'+1}, \ldots, X_\ell)$. The winning conditions of the simulation-extractability game impose that the tuple $(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, \boldsymbol{\pi})$ be different from those defined by inputs/outputs of all simulation queries. Together with its output, $\mathcal{A}$ also provides a representation of $\hat{C}$ w.r.t. the group elements in $\hat{\mathbb{G}}$ that are contained in srs. From this representation, $\mathcal{B}$ can infer a polynomial

$$F[X_1, \ldots, X_\ell] = f[X_1, \ldots, X_\ell] + r \cdot X_r$$

such that $\hat{C} = \hat{g}^{F(\alpha_1, \ldots, \alpha_\ell)}$ and where f is of the form (4). At this point, if $f[\boldsymbol{z}, X_{\mu'+1}, \ldots, X_\ell] - y$ is the zero polynomial, then $\mathcal{B}$ is a successful extractor (meaning that $\mathcal{A}$ did not succeed in the experiment) since $(f[X_1, \ldots, X_\ell], r)$ is a valid witness. We henceforth assume that $f[\boldsymbol{z}, X_{\mu'+1}, \ldots, X_\ell] - y$ is non-zero.

Along with its forgery $\boldsymbol{\pi}$, $\mathcal{A}$ also outputs representations of $\{\pi_{1,i}\}_{i=1}^{\mu'}$ and S_π, which define polynomials $\{A_i[X_1, \ldots, X_\ell, X_r]\}_{i \in [\mu']}$, and $S[X_1, \ldots, X_\ell, X_r]$ such that $S_\pi = g^{S(\alpha_1, \ldots, \alpha_\ell, \alpha_r)}$ and $\pi_{1,i} = g^{A_i(\alpha_1, \ldots, \alpha_\ell, \alpha_r)}$ for each $i \in [\mu']$. Let

$$R = e(S_\pi, \hat{g}_r) \cdot \left(\frac{e(g, \hat{C} \cdot \hat{g}^{-y})}{\prod_{i=1}^{\mu'} e(\pi_{i,1}, \hat{g}_i \cdot \hat{g}^{-z_i})} \right)^{-c}. \tag{9}$$

If $\boldsymbol{\pi}$ verifies, $c = H_{\mathsf{PCS}}(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, (\pi_{1,i})_{i=1}^{\mu'}, R)$. If the latter hash query was not made, $\mathcal{B}$ aborts. However, $\boldsymbol{\pi}$ can only be valid with probability $1/p$ in this case.

We now distinguish two cases: (i) The tuple $(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, c, (\pi_{1,i})_{i=1}^{\mu'})$ was recycled from simulation query involving a different Fiat-Shamir response $\bar{S}_\pi$; (ii) $(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, c, (\pi_{1,i})_{i=1}^{\mu'})$ is a fresh tuple. In case (i), we would have a collision

$$c = H_{\mathsf{PCS}}(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, (\pi_{1,i})_{i=1}^{\mu'}, R) = H_{\mathsf{PCS}}(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, (\pi_{1,i})_{i=1}^{\mu'}, \bar{R})$$

where $\bar{R} = e(\bar{S}_\pi, \hat{g}_r) \cdot \big(e(g, \hat{C} \cdot \hat{g}^{-y}) / \prod_{i=1}^{\mu'} e(\pi_{i,1}, \hat{g}_i \cdot \hat{g}^{-z_i})\big)^{-c}$ and

$$R = e(S_\pi, \hat{g}_r) \cdot \big(e(g, \hat{C} \cdot \hat{g}^{-y}) / \prod_{i=1}^{\mu'} e(\pi_{i,1}, \hat{g}_i \cdot \hat{g}^{-z_i})\big)^{-c}$$

since, for a given $(\hat{C}, R, y, \boldsymbol{z}, c, (\pi_{1,i})_{i=1}^{\mu'})$, there is a unique S_π satisfying (9). Since H_{PCS} is a random oracle, case (i) happens with probability $(Q_H + Q_S)^2/p$.

In case (ii), $c = H_{\mathsf{PCS}}(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, (\pi_{1,i})_{i=1}^{\mu'}, R)$ must be fresh as well except with probability $(Q_H + Q_S)^2/p$ since, if it had been a hash value programmed by the simulator, we would have a collision

$$c = H_{\mathsf{PCS}}(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, (\pi_{1,i})_{i=1}^{\mu'}, R) = H_{\mathsf{PCS}}(\mathsf{lbl}^{(0)}, \hat{C}^{(0)}, y^{(0)}, \boldsymbol{z}^{(0)}, (\pi_{1,i}^{(0)})_{i=1}^{\mu'}, R^{(0)}).$$

If c is fresh (i.e, it is not a programmed random oracle value), $\mathcal{B}$ can solve its $(\bar{d\ell}, \bar{d\ell})$-DLOG instance by recalling the representation of R as $\{\omega_{\mathcal{I},\hat{\mathcal{I}}}\}_{\mathcal{I} \in \mathcal{G}, \hat{\mathcal{I}} \in \hat{\mathcal{G}}}$ satisfying (8), which must have been supplied by $\mathcal{A}$ when it queried the hash value $H_{\mathsf{PCS}}(\mathsf{lbl}, \hat{C}, y, \boldsymbol{z}, (\pi_{1,i})_{i=1}^{\mu'}, R)$. From this representation, $\mathcal{B}$ can compute an

$(\ell+1)$-variate polynomial $R[X_1,\ldots,X_\ell,X_r]$ such that $R = e(g,\hat{g})^{R(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$. The verification equation (9) then implies

$$R(\alpha_1,\ldots,\alpha_\ell,\alpha_r) = S(\alpha_1,\ldots,\alpha_\ell,\alpha_r)\cdot\alpha_r - c\cdot\Big(F(\alpha_1,\ldots,\alpha_\ell,\alpha_r) - y - \sum_{i=1}^{\mu'} A_i(\alpha_1,\ldots,\alpha_\ell,\alpha_r)\cdot(\alpha_i - z_i)\Big), \quad (10)$$

which means that the $(\ell+1)$-variate polynomial

$$T[X_1,\ldots,X_\ell,X_r] \triangleq R[X_1,\ldots,X_\ell,X_r] - S[X_1,\ldots,X_\ell,X_r]\cdot X_r + c\cdot\Big(F[X_1,\ldots,X_\ell,X_r] - y - \sum_{i=1}^{\mu'} A_i[X_1,\ldots,X_\ell,X_r]\cdot(X_i - z_i)\Big), \quad (11)$$

vanishes on $(\alpha_1,\ldots,\alpha_\ell,\alpha_r)\in\mathbb{Z}_p^{\ell+1}$. We argue that this polynomial is non-zero w.h.p. if $f[\boldsymbol{z},X_{\mu'+1},\ldots,X_\ell]-y$ is a non-zero polynomial. If $T[X_1,\ldots,X_\ell,X_r]$ is identically zero, so is $T[z_1,\ldots,z_{\mu'},X_{\mu'+1},\ldots,X_\ell,0]$, in which case we have

$$R[z_1,\ldots,z_{\mu'},X_{\mu'+1},\ldots,X_\ell,0] = -c\cdot(f[\boldsymbol{z},X_{\mu'+1},\ldots,X_\ell]-y) \quad (12)$$

since $F[\boldsymbol{z},X_{\mu'+1},\ldots,X_\ell,0] = f[\boldsymbol{z},X_{\mu'+1},\ldots,X_\ell]$. We claim that, if the polynomial $f[\boldsymbol{z},X_{\mu'+1},\ldots,X_\ell]-y$ is non-zero, the identity (12) holds with probability at most Q_H/p. Indeed, $c = H_{\mathsf{PCS}}(\mathsf{lbl},\hat{C},y,\boldsymbol{z},(\pi_{1,i})_{i=1}^{\mu'},R)$ is a non-programmed random oracle output and is thus defined after R (whose algebraic representation uniquely determines the coefficients of $R[X_1,\ldots,X_\ell,X_r]$), the extracted polynomial $f[X_1,\ldots,X_\ell]$ (which is determined by the representation of $\hat{C}$) and the statement $(\boldsymbol{z},y)$. So, for any non-programmed $H_{\mathsf{PCS}}(\mathsf{lbl},\hat{C},y,\boldsymbol{z},(\pi_{1,i})_{i=1}^{\mu'},R)$ where the polynomials $(f[X_1,\ldots,X_\ell],R[X_1,\ldots,X_\ell,X_r])$ defined by $(\hat{C},R)$ are such that $f[\boldsymbol{z},X_{\mu'+1},\ldots,X_\ell]-y\not\equiv 0$, (12) holds for at most one scalar $c\in\mathbb{Z}_p$. The probability that the output of H_{PCS} hits this bad $c\in\mathbb{Z}_p$ is thus $1/p$.

We now assume that $T[X_1,\ldots,X_\ell,X_r]$ is non-zero and define the univariate

$$L[X] \triangleq T[\rho_1 X+\theta_1,\ldots,\rho_\ell X+\theta_\ell,\rho_r X+\theta_r]$$

Note that $L(\alpha) = T(\alpha_1,\ldots,\alpha_\ell,\alpha_r) = 0$, so that $\alpha\in\mathbb{Z}_p$ is computable by factoring $L[X]$ if it is a non-zero polynomial. Let zero_L the event that the polynomial $L[X]$ is identically zero given that $T[X_1,\ldots,X_{\ell+1}]$ is not. We show that $\Pr[\mathsf{zero}_L]$ is negligible.

If $L[X]$ is identically zero, we have $L(0) = T(\theta_1,\ldots,\theta_\ell,\theta_r) = 0$. However, $(\theta_1,\ldots,\theta_\ell,\theta_r)$ was sampled uniformly in $\mathbb{Z}_p^{\ell+1}$ and remains independent of $\mathcal{A}$'s view during the entire experiment. Indeed, srs only depends on $\alpha_r = \rho_r\cdot\alpha+\theta_r$ and $\{\alpha_i = \rho_i\cdot\alpha+\theta_i\}_{i=1}^r$ and $\mathcal{A}$ never gets to see any information about $\{\theta_i\}_{i\in[\ell]}$ nor θ_r. Since the total degree of $T[X_1,\ldots,X_\ell,X_r]$ is $\leq d\ell+1$, the Schwartz-Zippel lemma implies $\Pr[\mathsf{zero}_L]\leq\Pr_{(\theta_1,\ldots,\theta_\ell,\theta_r)}[T(\theta_1,\ldots,\theta_\ell,\theta_r)=0]\leq(d\ell+1)/p$. □

4.2 Extensions

The scheme and the proof of Theorem 3 easily extend to the batch evaluation setting when we prove evaluations of multiple committed polynomials on a common input. The details are given in Supplementary Material A.4.

In the univariate case, we can construct a simulation-extractable variant of KZG commitments with a better efficiency than by setting $\ell = 1$ in the above construction. The details are given in Supplementary Material C.

5 A Simulation-Extractable Variant of HyperPlonk

Let $\mathcal{C}[G] : \mathbb{Z}_p^{n+m} \to \mathbb{Z}_p$ an arithmetic circuit of N gates, where each gate has fan-in two and can be an addition gate, a multiplication gate, or a custom gate $G : \mathbb{Z}_p^2 \to \mathbb{Z}_p$. The public input of the circuit is denoted by $\mathbb{x} \in \mathbb{Z}_p^n$ and we assume as in [18] that $n + N + 1 = 2^\mu$ is a power of 2. The Plonk arithmetization [30] represents the trace of the computation by a set $\hat{M}$ of triples $\{(L_i, R_i, O_i) \in \mathbb{Z}_p^3\}_{i=0}^{n+N}$, where (L_i, R_i, O_i) contains the left input, the right input and the output of the i-th gate. In HyperPlonk [18], the prover interpolates $\hat{M}$ by defining a multilinear polynomial $M \in \mathbb{Z}_p^{(\leq 1)}[X_1, \ldots, X_{\mu+2}]$ such that, for each $i \in \{0, \ldots, n+N\}$,

$$M(\langle i \rangle, 0, 0) = L_i, \qquad M(\langle i \rangle, 0, 1) = R_i, \qquad M(\langle i \rangle, 1, 0) = O_i.$$

In a pre-processing phase, the $\mathsf{PreProcess}$ algorithm takes as input the circuit and creates the verifier's public parameters vp. These parameters consist of deterministic commitments to selector polynomials $S_1, S_2, S_3 \in \mathbb{Z}_p^{(\leq 1)}[X_1, \ldots, X_\mu]$, and a wiring polynomial $\sigma \in \mathbb{Z}_p^{(\leq 1)}[X_1, \ldots, X_{\mu+2}]$, which is actually encoded as 3 partial polynomials in (15). These polynomials only depend on the circuit and are computed only once by the verifier.

Each verification requires to compute an input-dependent multilinear polynomial $I[X_1, \ldots, X_\mu]$ such that $I(\langle i \rangle, 0, \ldots, 0) = \mathbb{x}_i$ for each $i \in [n]$ and $I(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in B_\mu \setminus \{(\langle i \rangle, 0, \ldots, 0)\}_{i \in [n]}$.

To generate a proof, the prover defines the virtual μ-variate polynomial

$$\begin{aligned} f[\boldsymbol{X}] = S_1[\boldsymbol{X}] \cdot \big(M[\boldsymbol{X}, 0, 0] + M_1[\boldsymbol{X}, 0, 1]\big) \\ + S_2[\boldsymbol{X}] \cdot \big(M[\boldsymbol{X}, 0, 0] \cdot M_1[\boldsymbol{X}, 0, 1]\big) \\ + S_3[\boldsymbol{X}] \cdot \boldsymbol{G}\big(M[\boldsymbol{X}, 0, 0], M_1[\boldsymbol{X}, 0, 1]\big) - M[\boldsymbol{X}, 1, 0] + I[\boldsymbol{X}], \end{aligned} \tag{13}$$

and convinces the verifier that $f[\boldsymbol{X}]$ vanishes everywhere on the hypercube B_μ.

The selector polynomials are defined such that, for extremal gates $i < n$ and $i = n + N$, we have $S_1(\langle i \rangle) = S_2(\langle i \rangle) = S_3(\langle i \rangle) = 0$ so that $M(\langle i \rangle, 1, 0) = I(\langle i \rangle)$ for each $i \in [0, n-1]$ while the constraint $M(\langle n + N \rangle) = 0$ ensures that the arithmetic circuit outputs 0. For internal gates $i \in [n, n + N - 1]$, we have $S_1(\langle i \rangle) = 1$ and $S_2(\langle i \rangle) = S_3(\langle i \rangle) = 0$ for each addition gate i; $S_2(\langle i \rangle) = 1$ and $S_1(\langle i \rangle) = S_3(\langle i \rangle) = 0$ for each multiplication gate i; $S_3(\langle i \rangle) = 1$ and $S_1(\langle i \rangle) =$

$S_2(\langle i \rangle) = 0$ for each custom gate i.

Then, the prover must also convince the verifier that $M[X_1, \ldots, X_\ell]$ correctly encodes the circuit. This is done by defining the "extended hypercube" $\mathcal{H} \triangleq B_\mu \times \{\langle i \rangle\}_{i=0}^{2}$ and proving that the wiring identity $M(\boldsymbol{x}) = M(\hat{\sigma}(\boldsymbol{x}))$ holds for all $\boldsymbol{x} \in \mathcal{H}$, where $\hat{\sigma} : \mathcal{H} \to \mathcal{H}$ is the circuit-dependent permutation.

The pre-processing algorithm thus commits to an additional multilinear polynomial $s_\sigma[X_1, \ldots, X_{\mu+2}]$ such that, for all $(\boldsymbol{x}, \langle i \rangle) \in \mathcal{H}$,

$$s_\sigma(\boldsymbol{x}, \langle i \rangle) = s_{\mathrm{id}}(\hat{\sigma}(\boldsymbol{x}, \langle i \rangle)) \tag{14}$$

where $s_{\mathrm{id}}[X_1, \ldots, X_{\mu+2}] = \sum_{i=1}^{\mu+2} 2^{i-1} \cdot X_i$.

In order to prove the wiring identity, HyperPlonk relies on a product argument over the Boolean hypercube which builds on the following lemma. Given commitments $C_1 = g^{f_1(\alpha_1,\ldots,\alpha_\mu)+r_1\cdot\alpha_r}$, $C_2 = g^{f_2(\alpha_1,\ldots,\alpha_\mu)+r_2\cdot\alpha_r}$ to polynomials $f_1, f_2 \in \mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\mu]$, it allows a prover to convince the verifier that $f_b[X_1, \ldots, X_\mu] \in \mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\mu]$ for each $b \in \{1, 2\}$ and $\prod_{\boldsymbol{x} \in B_\mu} f'(\boldsymbol{x}) = s$, for a given $s \in \mathbb{Z}_p$, where $f'(\boldsymbol{x}) = f_1(\boldsymbol{x})/f_2(\boldsymbol{x})$.

Lemma 1 ([52, Lemma 5.1]). *For a rational function $f' : \mathbb{Z}_p^\mu \to \mathbb{Z}_p$ and a scalar $s \in \mathbb{Z}_p$, the equality $s = \prod_{\boldsymbol{x} \in B_\mu} f'(\boldsymbol{x})$ holds if and only if there exists a multilinear polynomial $\tilde{v} \in \mathbb{Z}_p^{(\leq 1)}[X_1, \ldots, X_{\mu+1}]$ such that $\tilde{v}(1, 1, \ldots, 1, 0) = s$ and, for all $\boldsymbol{x} \in B_\mu$, $\tilde{v}(0, \boldsymbol{x}) = f'(\boldsymbol{x})$ and $\tilde{v}(1, \boldsymbol{x}) = \tilde{v}(\boldsymbol{x}, 0) \cdot \tilde{v}(\boldsymbol{x}, 1)$.*

5.1 Description

The description below is almost identical to the original HyperPlonk [18]. The main difference is that some multivariate PCS evaluation proofs have to be generated with the simulation-extractable polynomial commitment of Section 4.

Also, in order to obtain straight-line simulation-extractability in the AGM (and even knowledge-soundness), we need to carefully choose the groups where the different commitments live. Specifically, the knowledge soundness analysis sometimes requires committed polynomials to be of the form described in Section 3.2. For this reason, we chose to have those commitments live in the second source group $\hat{\mathbb{G}}$ for which the available generators contained in srs ensure that the AGM-extractor will be able to extract a polynomial of the form described in Section 3.2. As for commitments to univariate polynomials and the group elements contained in PCS evaluation proofs, they can live in $\mathbb{G}$.

As in [18], the batched zero-check protocol applies the sumcheck protocol to

$$F_{\boldsymbol{\gamma}}[X_1, \ldots, X_\mu] = f[X_1, \ldots, X_\mu] \cdot eq_{\boldsymbol{\gamma}}[X_1, \ldots, X_\mu],$$

using the multilinear $eq_{\boldsymbol{\gamma}}[X_1, \ldots, X_\mu] = \prod_{i=1}^{\mu} \big(\gamma_i \cdot X_i + (1 - \gamma_i) \cdot (1 - X_i)\big)$ where $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_\mu)$ is a random vector obtained by hashing the transcript so far.

CRS-Gen$(1^\lambda, 1^d, 1^\mu)$**:** On input of λ, a parameter $\mu = \log(n + N + 1)$ specifying the circuit size, and a bound d on the degree of custom gates such that $d \cdot 2^\mu \in \mathsf{poly}(\lambda)$, set $\ell = \mu + 1$ and $\bar{d} = \max(2d + 1, 9)$. Then,

1. Choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^{l(\lambda)}$, for some function $l : \mathbb{N} \to \mathbb{N}$, and $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$.
2. Pick $\alpha_1, \ldots, \alpha_\ell, \alpha_r \xleftarrow{R} \mathbb{Z}_p$. Compute $g_r = g^{\alpha_r}$ and $\hat{g}_r = \hat{g}^{\alpha_r}$.
3. For each $\mathcal{I} = (i_1, \ldots, i_\ell) \in \mathcal{U}_{\bar{d},\ell}$, compute $g_{\mathcal{I}} = g^{\prod_{j=1}^{\ell} \alpha_j^{i_j}}$.
4. For each $\mathcal{I} = (i_1, \ldots, i_\ell) \in \mathcal{W}_{1,\ell}$, compute $\hat{g}_{\mathcal{I}} = \hat{g}^{\prod_{j=1}^{\ell} \alpha_j^{i_j}}$. Then, compute $\big\{\hat{g}^{(\alpha_1+\alpha_{\mu+1})^i \cdot \alpha_\mu \cdot (\alpha_\mu - 1)}\big\}_{i=0}^{3}$.
5. Choose hash functions $H, H_{\mathsf{PCS}}, H_{\mathsf{batch}}, H_\delta : \{0,1\}^* \to \mathbb{Z}_p$, $H_\xi : \{0,1\}^* \to \mathbb{Z}_p^*$, $H_\beta, H_\zeta : \{0,1\}^* \to \mathbb{Z}_p^2$ and $H_\gamma : \{0,1\}^* \to \mathbb{Z}_p^\mu$.

Let $\mathbf{H} = \{H, H_{\mathsf{PCS}}, H_{\mathsf{batch}}, H_\beta, H_\zeta, H_\xi, H_\delta\}$ and define the universal SRS

$$\mathsf{srs} = \Big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g_r, \hat{g}_r, \{g_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{U}_{\bar{d},\ell}}, \{\hat{g}_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{1,\ell}}, \big\{\hat{g}^{(\alpha_1+\alpha_{\mu+1})^i \cdot \alpha_\mu \cdot (\alpha_\mu - 1)}\big\}_{i=0}^{3}, \mathbf{H}\Big).$$

$\mathsf{PreProcess}_{\mathsf{srs}}(\mathbb{i})$: On input of the universal SRS srs and an arithmetic circuit $\mathbb{i} = \mathcal{C}[G] : \mathbb{Z}_p^{n+m} \to \mathbb{Z}_p$ of size N such that $n + N + 1 = 2^\mu$,

1. Let the selector polynomials $S_1, S_2, S_3 : \{0,1\}^\mu \to \mathbb{Z}_p$. Deterministically compute a multilinear polynomial $s_\sigma \in \mathbb{Z}_p^{(\leq 1)}[X_1, \ldots, X_\ell]$ that encodes the wiring of $\mathcal{C}$. Define the multilinear $s_{\mathrm{id}}[X_1, \ldots, X_\ell] = \sum_{i=1}^{\ell} 2^{i-1} \cdot X_i$. Let the partial permutation polynomials

$$s_\sigma[X_1, \ldots, X_\mu, 0, 0], \quad s_\sigma[X_1, \ldots, X_\mu, 0, 1], \quad s_\sigma[X_1, \ldots, X_\mu, 1, 0] \tag{15}$$

2. For each $i \in \{0,1,2\}$, compute $C_{\sigma,\langle i \rangle} = g^{s_\sigma(\alpha_1, \ldots, \alpha_\mu, \langle i \rangle)}$.
3. For each $i \in \{1,2,3\}$, compute $C_{s,i} = g^{S_i(\alpha_1, \ldots, \alpha_\mu)}$.

Return $\mathsf{pp} = \mathsf{srs}$ and

$$\mathsf{vp} = \Big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}, \hat{g}_r, \{\hat{g}_i = \hat{g}^{\alpha_i}\}_{i=1}^{\mu}, \{C_{\sigma,\langle i \rangle}\}_{i=0}^{2}, \{C_{s,i}\}_{i=1}^{3}, \mathbf{H}\Big).$$

$\mathsf{Prove}_{\mathsf{pp}}(\mathbb{i}, \mathbb{x}, \mathbb{w})$: Given an arithmetic circuit $\mathbb{i} = \mathcal{C}[G] : \mathbb{Z}_p^n \to \mathbb{Z}_p$, a public input $\mathbb{x}$ and a witness $\mathbb{w}$ consisting of a wire assignment leading to the output 0,

1. Compute $I[\boldsymbol{X}]$, which encodes $\mathbb{x}$, as well as the circuit-dependent polynomials $\{S_i[\boldsymbol{X}]\}_{i=1}^{3}$, $\{s_\sigma[\boldsymbol{X}, \langle i \rangle]\}_{i=0}^{2}$.
2. Compute the polynomial $M \in \mathbb{Z}_p^{(\leq 1)}[X_1, \ldots, X_\ell]$ satisfying the gate identity (13). For each $i \in [0,2]$, choose $R_i \xleftarrow{R} \mathbb{Z}_p$ (viewed as a constant univariate polynomial) and commit to

$$\bar{M}_i[X_1, \ldots, X_\mu] \triangleq M[X_1, \ldots, X_\mu, \langle i \rangle] + R_i \cdot X_\mu \cdot (X_\mu - 1),$$

by choosing $r_{M,i} \xleftarrow{R} \mathbb{Z}_p$ and computing

$$\hat{C}_{M,i} = \hat{g}^{M(\alpha_1, \ldots, \alpha_\mu, \langle i \rangle) + R_i \cdot \alpha_\mu \cdot (\alpha_\mu - 1) + r_{M,i} \cdot \alpha_r}.$$

3. Compute the challenge $(\beta_1, \beta_2) = H_\beta(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^2) \in \mathbb{Z}_p^2$. Compute a multilinear $\tilde{v} \in \mathbb{Z}_p^{(\leq 1)}[X_1, \ldots, X_{\mu+1}]$ satisfying the conditions of Lemma 1 and such that

$$\tilde{v}(0, \boldsymbol{x}) = \prod_{i=0}^{2} \frac{\bar{M}_i(\boldsymbol{x}) + \beta_2 \cdot s_{\mathrm{id}}(\boldsymbol{x}, \langle i \rangle) + \beta_1}{\bar{M}_i(\boldsymbol{x}) + \beta_2 \cdot s_\sigma(\boldsymbol{x}, \langle i \rangle) + \beta_1} \qquad \forall \boldsymbol{x} \in B_\mu,$$

where $\langle i \rangle \in \{0,1\}^2$ is the binary representation of $i \in [0, 2]$.

4. Choose a random degree-3 polynomial $R_v \in \mathbb{Z}_p[X]$ and define

$$\bar{v}[X_1, \ldots, X_{\mu+1}] = \tilde{v}[X_1, \ldots, X_{\mu+1}] + R_v[X_1 + X_{\mu+1}] \cdot X_\mu \cdot (X_\mu - 1)$$

Commit to $\bar{v}[X_1, \ldots, X_{\mu+1}]$ by choosing $r_v \xleftarrow{R} \mathbb{Z}_p$ and computing

$$\hat{C}_v = \hat{g}^{\tilde{v}(\alpha_1, \ldots, \alpha_{\mu+1}) + R_v(\alpha_1 + \alpha_{\mu+1}) \cdot \alpha_\mu \cdot (\alpha_\mu - 1) + r_v \cdot \alpha_r}$$

5. Compute $(\zeta_1, \zeta_2) = H_\zeta(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^2, \hat{C}_v) \in \mathbb{Z}_p^2$. Define the following virtual (i.e., not explicitly computed) polynomials in $\boldsymbol{X} = (X_1, \ldots, X_\mu)$:

$$\begin{aligned} Q_1[\boldsymbol{X}] &= \bar{v}[1, \boldsymbol{X}] - \bar{v}[\boldsymbol{X}, 0] \cdot \bar{v}[\boldsymbol{X}, 1] \\ Q_2[\boldsymbol{X}] &= \prod_{i=0}^{2} \big(\bar{M}_i[\boldsymbol{X}] + \beta_2 \cdot s_{\mathrm{id}}[\boldsymbol{X}, \langle i \rangle] + \beta_1\big) \\ &\qquad - \bar{v}[0, \boldsymbol{X}] \cdot \prod_{i=0}^{2} \big(\bar{M}_i[\boldsymbol{X}] + \beta_2 \cdot s_\sigma[\boldsymbol{X}, \langle i \rangle] + \beta_1\big) \\ f[\boldsymbol{X}] &= S_1[\boldsymbol{X}] \cdot \big(\bar{M}_0[\boldsymbol{X}] + \bar{M}_1[\boldsymbol{X}]\big) \\ &\quad + S_2[\boldsymbol{X}] \cdot \big(\bar{M}_0[\boldsymbol{X}] \cdot \bar{M}_1[\boldsymbol{X}]\big) \\ &\quad + S_3[\boldsymbol{X}] \cdot \boldsymbol{G}\big(\bar{M}_0[\boldsymbol{X}], \bar{M}_1[\boldsymbol{X}]\big) - \bar{M}_2[\boldsymbol{X}] + I[\boldsymbol{X}], \end{aligned} \tag{16}$$

6. Prove that $F[\boldsymbol{X}] = f[\boldsymbol{X}] + \zeta_1 \cdot Q_1[\boldsymbol{X}] + \zeta_2 \cdot Q_2[\boldsymbol{X}]$ vanishes over B_μ. To this end, compute $\boldsymbol{\gamma} = H_\gamma(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^2, \hat{C}_v) \in \mathbb{Z}_p^\mu$ and prove the statement $\sum_{\boldsymbol{x} \in B_\mu} F(\boldsymbol{x}) \cdot eq_{\boldsymbol{\gamma}}(\boldsymbol{x}) = 0$ via a zero-knowledge sumcheck protocol. Namely, let $\bar{d}' = \min(\bar{d}, 3) = 3$ and do the following.
   a. Choose a polynomial $a[X_1, \ldots, X_\mu] = a_0 + \sum_{i=1}^{\mu} a_i[X_i]$ such that, for each $i \in [\mu]$, $a_i[X_i] = \sum_{j=1}^{\bar{d}'} a_{i,j} \cdot X_i^j$, for random a_0, $\{a_{i,j}\}_{i,j}$, and compute a commitment $C_a = g^{a(\alpha_1, \ldots, \alpha_\mu) + r_a \cdot \alpha_r}$, where $r_a \xleftarrow{R} \mathbb{Z}_p$, using $\{g_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{\bar{d},\ell}}$. Compute $y_a = \sum_{\boldsymbol{x} \in B_\mu} a(\boldsymbol{x})$.
   b. Compute $\xi = H_\xi(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^2, \hat{C}_v, C_a, y_a, (\zeta_1, \zeta_2), (\beta_1, \beta_2), \boldsymbol{\gamma}) \in \mathbb{Z}_p^*$. Let the virtual polynomial (of degree $\leq \bar{d} = \max(2d + 1, 9)$)

$$\begin{aligned} F_{zk}[X_1, \ldots, X_\mu] \triangleq F[X_1, \ldots, X_\mu] \cdot eq_{\boldsymbol{\gamma}}[X_1, \ldots, X_\mu] \\ + \xi \cdot a[X_1, \ldots, X_\mu] \end{aligned}$$

c. Prove that $\sum_{\boldsymbol{x}\in B_\mu} F_{zk}(\boldsymbol{x}) = \xi \cdot y_a$. Namely, for $i = \mu$ to 1,

1. Compute a commitment $C_{\theta,i} = g^{\theta_i(\alpha_1)}$ to the univariate

$$\begin{aligned}\theta_i[X] = \sum_{\boldsymbol{b}\in B_{i-1}} \Big(& f[\boldsymbol{b}, X, r_{i+1}, \ldots, r_\mu] + \zeta_1 \cdot Q_1[\boldsymbol{b}, X, r_{i+1}, \ldots, r_\mu] \\ & + \zeta_2 \cdot Q_2[\boldsymbol{b}, X, r_{i+1}, \ldots, r_\mu] \Big) \cdot eq_{\boldsymbol{\gamma}}[\boldsymbol{b}, X, r_{i+1}, \ldots, r_\mu] \\ & + \xi \cdot \sum_{\boldsymbol{b}\in B_{i-1}} a[\boldsymbol{b}, X, r_{i+1}, \ldots, r_\mu]\end{aligned} \tag{17}$$

2. Compute

$$\begin{aligned}r_i = H\big(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^2, \hat{C}_v, C_a, y_a, \boldsymbol{\gamma}, (\beta_1, \beta_2), (\zeta_1, \zeta_2), \\ \{C_{\theta,j}, \theta_j(0), \theta_j(1)\}_{j\in[i,\mu]}, (r_j)_{j\in[i+1,\mu]}\big) \in \mathbb{Z}_p.\end{aligned}$$

d. Generate a batch proof $\boldsymbol{\pi}_{\mathsf{batch}}$ that $(C_{\theta,i})_{i\in[\mu]}$ commit to polynomials $(\theta_i[X])_{i\in[\mu]}$ evaluating to $\{\theta_i(0), \theta_i(1), \theta_{i,\boldsymbol{r}} \triangleq \theta_i(r_i)\}$ for the inputs $\Omega_i = \{0, 1, r_i\}$.

e. Generate multivariate PCS evaluation proofs. Namely,

1. Using the PCS of Section 3.1, generate a batch proof $\boldsymbol{\pi}_{C,a}$ for the commitments $\big\{\{C_{s,i}\}_{i=1}^3, \{C_{\sigma,\langle i\rangle}\}_{i=0}^2, C_a\big\}$ showing that
   - For each $i \in [3]$, the selector polynomial $S_i[\boldsymbol{X}]$ evaluates to $s_{i,\boldsymbol{r}} = S_i(r_1, \ldots, r_\mu)$
   - For each $i \in \{0, 1, 2\}$, the partial permutation polynomial $s_\sigma[\boldsymbol{X}, \langle i\rangle]$ evaluates to $\sigma_{i,\boldsymbol{r}} \triangleq s_\sigma(r_1, \ldots, r_\mu, \langle i\rangle)$.
   - C_a commits to a polynomial that evaluates to $a_{\boldsymbol{r}} \triangleq a(r_1, \ldots, r_\mu)$.
2. Using the PCS of Section 4, generate a batch proof $\boldsymbol{\pi}_M$ showing that $\{\hat{C}_{M,i}\}_{i=0}^2$ commit to polynomials $\{\bar{M}_i[\boldsymbol{X}]\}_{i=0}^2$ that evaluate to $m_{i,\boldsymbol{r}} \triangleq \bar{M}_i(r_1, \ldots, r_\mu)$ for each $i \in [0, 2]$. This proof is generated for a label lbl_1 containing the entire transcript so far.
3. For the commitment $\hat{C}_v$, compute simulation-extractable proofs $(\boldsymbol{\pi}_{v,x,b}, \boldsymbol{\pi}_{v,b,x})_{b=0}^1$ that

$$\forall b \in \{0,1\} \ : \ \bar{v}(r_1, \ldots, r_\mu, b) = v_{\boldsymbol{r},b}, \quad \wedge \quad \bar{v}(b, r_1, \ldots, r_\mu) = v_{b,\boldsymbol{r}},$$

   for labels $\{\mathsf{lbl}_{2,i}\}_{i=1}^4$ containing the entire transcript so far.
4. For $\hat{C}_v$, prove that $\bar{v}(1, \ldots, 1, 0) = 1$ via a simulation-extractable proof $\boldsymbol{\pi}_{v,s}$ for a label lbl_3 containing the entire transcript so far.

Let

$$\boldsymbol{\pi}_{\boldsymbol{r}} = \Big(\boldsymbol{\pi}_{C,a}, (\boldsymbol{\pi}_M, (\boldsymbol{\pi}_{v,x,b}, \boldsymbol{\pi}_{v,b,x})_{b=0}^1, \boldsymbol{\pi}_{v,s}\Big) \tag{18}$$

the vector of evaluation proofs and let the batched zero-check proof

$$\begin{aligned}\boldsymbol{\pi}_{\mathsf{zero}} = \big(C_a, a_{\boldsymbol{r}}, y_a, \{m_{i,\boldsymbol{r}}, \sigma_{i,\boldsymbol{r}}\}_{i=0}^2, \{s_{i,\boldsymbol{r}}\}_{i=1}^3, \{v_{\boldsymbol{r},b}, v_{b,\boldsymbol{r}}\}_{b=0}^1, \\ \{C_{\theta,i}, \theta_i(0), \theta_i(1)\}_{i\in[\mu]}, \boldsymbol{\pi}_{\mathsf{batch}}, \boldsymbol{\pi}_{\boldsymbol{r}}\big)\end{aligned}$$

Return the proof $\boldsymbol{\pi} = \big((\hat{C}_{M,i})_{i=0}^{2}, \hat{C}_v, \boldsymbol{\pi}_{\mathsf{zero}}\big)$.

$\mathsf{Verify}_{\mathsf{vp}}\big(\mathbb{i}, \mathbb{x}, \boldsymbol{\pi}\big)$: Given a statement $(\mathbb{i}, \mathbb{x})$, a compressed version vp of $\mathbb{i} = \mathcal{C}[G]$, and a candidate proof $\boldsymbol{\pi}$, return 0 if $\boldsymbol{\pi}$ does not parse properly. Otherwise,

1. Compute $\boldsymbol{\gamma} = H_\gamma\big(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^{2}, \hat{C}_v\big) \in \mathbb{Z}_p^{\mu}$ and

$$(\beta_1, \beta_2) = H_\beta\big(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^{2}\big), \qquad (\zeta_1, \zeta_2) = H_\zeta\big(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^{2}, \hat{C}_v\big)$$

2. Compute the polynomial $I[X_1, \ldots, X_\mu]$ such that $I(\langle i \rangle, 0, \ldots, 0) = \mathbb{x}[i]$ for each $i \in [0, n-1]$ and $I(\boldsymbol{x}) = 0$ everywhere else.
3. For each $i = \mu$ to 1, compute

$$r_i = H\big(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^{2}, \hat{C}_v, C_a, y_a, \boldsymbol{\gamma}, (\beta_1, \beta_2), (\zeta_1, \zeta_2), \{C_{\theta,j}, \theta_j(0), \theta_j(1)\}_{j\in[i,\mu]}, (r_j)_{j\in[i+1,\mu]}\big) \in \mathbb{Z}_p. \quad (19)$$

4. Compute $\xi = H_\xi\big(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^{2}, \hat{C}_v, C_a, y_a, (\zeta_1, \zeta_2), (\beta_1, \beta_2), \boldsymbol{\gamma}\big) \in \mathbb{Z}_p^*$ and return 0 if $\theta_\mu(0) + \theta_\mu(1) \neq \xi \cdot y_a$.
5. Let $I_{\boldsymbol{r}} = I(r_1, \ldots, r_\mu)$ and

$$\begin{aligned}\theta_{1,\boldsymbol{r}} \triangleq \Big[\Big(&s_{1,\boldsymbol{r}} \cdot (m_{0,\boldsymbol{r}} + m_{1,\boldsymbol{r}}) + s_{2,\boldsymbol{r}} \cdot (m_{0,\boldsymbol{r}} \cdot m_{1,\boldsymbol{r}}) + s_{3,\boldsymbol{r}} \cdot \boldsymbol{G}(m_{0,\boldsymbol{r}}, m_{1,\boldsymbol{r}}) - m_{2,\boldsymbol{r}} + I_{\boldsymbol{r}}\Big) \\ &+ \zeta_1 \cdot \Big(v_{1,\boldsymbol{r}} - v_{\boldsymbol{r},0} \cdot v_{\boldsymbol{r},1}\Big) + \zeta_2 \cdot \Big(\prod_{i=0}^{2}(m_{i,\boldsymbol{r}} + \beta_2 \cdot s_{\mathrm{id}}(\boldsymbol{r}, \langle i \rangle) + \beta_1) \\ &- v_{0,\boldsymbol{r}} \cdot \prod_{i=0}^{2}(m_{i,\boldsymbol{r}} + \beta_2 \cdot \sigma_{i,\boldsymbol{r}} + \beta_1)\Big)\Big] \cdot eq_{\boldsymbol{\gamma}}(r_1, \ldots, r_\mu) + \xi \cdot a_{\boldsymbol{r}}\end{aligned}$$

   Then, for each $i \in [2, \mu]$, define $\theta_{i,\boldsymbol{r}} \triangleq \theta_{i-1}(0) + \theta_{i-1}(1)$. Return 0 if $\boldsymbol{\pi}_{\mathsf{batch}}$ is not a valid univariate batch proof for commitments $(C_{\theta,i})_{i\in[\mu]}$, evaluations $(\theta_i(0), \theta_i(1), \theta_{i,\boldsymbol{r}})_{i=1}^{\mu}$, and inputs $\{\Omega_i = \{0, 1, r_i\}\}_{i=1}^{\mu}$.
6. Return 0 if the PCS evaluation proofs in $\boldsymbol{\pi}_{\boldsymbol{r}}$ do not verify for the commitments $((\hat{C}_{M,i})_{i=0}^{2}, \{C_{\sigma,\langle i \rangle}\}_{i=0}^{2}, \{C_{s,i}\}_{i=1}^{3}, C_a)$, the label lbl_1, and the claimed evaluations in $\boldsymbol{\pi}_{\mathsf{zero}}$ for the inputs $\boldsymbol{r}$, $\{(\boldsymbol{r}, b), (b, \boldsymbol{r})\}_{b\in\{0,1\}}$. Return 0 if $(\boldsymbol{\pi}_{v,x,b}, \boldsymbol{\pi}_{v,b,x})_{b=0}^{1}$ do not verify for the labels $\{\mathsf{lbl}_{2,i}\}_{i=1}^{4}$ and the inputs specified at step 6.e.2. Return 0 if $\boldsymbol{\pi}_{v,s}$ does not verify for the claimed evaluation $\tilde{v}(1, 1, \ldots, 1, 0) = 1$ and the label lbl_3.

If none of the previous checks failed, return 1.

We note that, at step 6 of Prove, the virtual polynomial $F[\boldsymbol{X}]$ has maximal degree $\max(2d+1, 9)$ since $Q_2[\boldsymbol{X}]$ has degree ≤ 9 in X_μ and $f[\boldsymbol{X}]$ has degree $\leq 2d+1$ in X_μ, where d is the degree of $\boldsymbol{G}$.

At step 6.c.1, the prover runs in linear time in 2^μ by using the algorithm of [18, Section 3] (see also [53] and [54, Section 3.2]). Since $eq_{\boldsymbol{\gamma}}[X_1, \ldots, X_\mu]$, $\{\bar{M}_i[X_1, \ldots, X_\mu]\}_{i=0}^{2}$, $\tilde{v} \in \mathbb{Z}_p[X_1, \ldots, X_{\mu+1}]$, $\{S_i[X_1, \ldots, X_\mu]\}_{i=1}^{3}$ and the partial permutation polynomials (15) are multilinear polynomials in $(X_1, \ldots, X_{\mu+1})$,

the virtual $F[\boldsymbol{X}]$ of step 6 can be written[3] as a composition $F[X_1, \ldots, X_\mu] = g(h_1, \ldots, h_c)$ of a c-variate function g of total degree d with multilinear polynomials $\{h_i[X_1, \ldots, X_\mu]\}_{i=1}^c$. When $c \approx d$, the round polynomials $\{\theta_i[X]\}_{i=1}^\mu$ of the sumcheck protocol are computable in time $O(2^\mu \cdot d \cdot \log^2 d)$.

We note that the verifier can compute $I_{\boldsymbol{r}} = I(r_1, \ldots, r_\mu)$ itself at step 5 using $O(n \cdot \mu)$ field operations since $I[\boldsymbol{X}]$ is multilinear and can be expressed as $I[\boldsymbol{X}] = \sum_{i=1}^n \mathbb{x}_i \cdot eq_{(\langle i \rangle, 0, \ldots, 0)}[X_1, \ldots, X_\mu]$.

On the SRS and Proof Sizes. The above instantiation is not optimized in terms of space. In order to enable custom gates of degree d, the common reference string srs must contain $O(d \cdot 2^\mu) = O(d \cdot |\mathcal{C}|)$ group elements (since $\mathcal{U}_{\bar{d},\ell}$ contains $O(\bar{d} \cdot 2^\mu)$ elements, where $\bar{d} = \max(2d+1, 9)$). It is possible to optimize the scheme and reduce the SRS size to $O(d + |\mathcal{C}|)$ group elements by committing to all multivariate polynomials as multilinear polynomials.[4] The linearization of multivariate polynomials then allows using the batch evaluation protocol of [18, Section 3.8] so as to reduce the proof size. We did not include these optimizations in the description in order to keep it as simple as possible.

5.2 Security

We first give a proof of knowledge-soundness. In a second step, we will describe a zero-knowledge simulator and argue that the knowledge extractor still works (without rewinding) when the adversary can observe many simulated proofs.

Knowledge-Soundness. Knowledge-soundness was proven in [18] for the underlying PIOP. Here, we adapt the proof to the instantiation from KZG/PST-based commitments since we aim at a non-rewinding extractor (in the AGM) that will also be used in the proof of Theorem 6. In particular, since our instantiation uses both multivariate and univariate commitments sharing the same SRS, we need to account for the fact that the algebraic representations of univariate commitments can depend on SRS components related to multivariate ones. In Supplementary Material B, we show that the batch evaluation protocol of [12] still provides the suitable extractability properties when KZG commitments are used on a larger SRS allowing commitments to multivariate polynomials.

Theorem 4. *In the AGM+ROM model and under the $(d\ell, d\ell)$-DLOG assumption, the above instantiation of HyperPlonk provides knowledge-soundness with straight-line extractability.* (The proof is given in Supplementary Material D.1.)

[3] This remains true after having added $R_v[X_1 + X_{\mu+1}] \cdot X_\mu(X_\mu - 1)$ to $\tilde{v}$.

[4] Following ideas from [13,42], one can commit to a univariate $f[X] = \sum_{i=0}^{D-1} f_i \cdot X^i$ of degree $D - 1 = 2^d - 1$ by considering the bits $i_1 \ldots i_d \in \{0,1\}^d$ of each $i \in [0, D-1]$ and committing to the multilinear $F[X_1, \ldots, X_d] = \sum_{i=0}^{D-1} f_i \cdot X_1^{i_1} \ldots X_d^{i_d}$. Univariate evaluations $f(z) = y$ are then proven as $F(z, z^2, z^4, \ldots, z^{2^{d-1}}) = y$. The same idea extends to linearize multivariate polynomials by increasing the number of variables.

Zero-Knowledge. The proof of Theorem 5 provides a trapdoor-less ZK simulator which will be used in the proof of simulation-extractability. The simulator proceeds by sampling a "fake witness" consisting of a mutlilinear polynomial $M[X_1, \ldots, X_{\mu+2}]$ that induces a polynomial $F[\boldsymbol{X}] = f[\boldsymbol{X}]+\zeta_1 \cdot Q_1[\boldsymbol{X}]+\zeta_2 \cdot Q_2[\boldsymbol{X}]$ for which $F[\boldsymbol{x}] = 0$ for all $\boldsymbol{x} \in B_\mu$ at step 6 of the prover. Since the wiring identity $M(\boldsymbol{x}, \langle i \rangle) = M(\hat{\sigma}(\boldsymbol{x}, \langle i \rangle))$ is not satisfied for all $\boldsymbol{x} \in B_\mu$, the corresponding polynomial $\tilde{v}[X_1, \ldots, X_{\mu+1}]$ (defined at step 3) does not satisfy the condition $\tilde{v}(1, \ldots, 1, 0) = 1$ when we apply Lemma 1. As a loophole, we can simulate a fake PCS evaluation proof that $\tilde{v}(1, \ldots, 1, 0) = 1$. To make sure that no information leaks about the internal wires of the circuit, the simulator also simulates PCS proofs of random evaluations (instead of the real evaluations) for the partial polynomials $\{\bar{M}_i[X_1, \ldots, X_\mu]\}_{i=0}^{2}$ and $\{\bar{v}[X_1, \ldots, X_\mu, b], \bar{v}[b, X_1, \ldots, X_\mu]\}_{b=0}^{1}$ on inputs outside the Boolean hypercube. For this reason, the PCS evaluation proofs at steps 6.e.2-6.e.4 are generated using the PCS of Section 4. Thanks to the masking technique of (4), these evaluations can be proven statistically indistinguishable from real evaluations. The proof relies on the fact that $R_v[X_1 + X_{\mu+1}]$ is evaluated on four distinct inputs when $\bar{v}[X_1, \ldots, X_{\mu+1}]$ is evaluated on the inputs $\{(r_1, \ldots, r_\mu, b), (b, r_1, \ldots, r_\mu)\}_{b=0}^{1}$ at step 6.e.3.

Theorem 5. *The above HyperPlonk protocol provides statistical zero-knowledge in the ROM.* (The proof is given in Supplementary Material D.2.)

Simulation-Extractability. The only obstacle that prevents from applying the same analysis as in the proof of Theorem 4 is that the simulator $\mathcal{S}$ creates simulated proofs of false statements by programming random oracles, which is the only case where the outputs of random oracles are determined before their inputs (and the algebraic representations thereof). However, the only random oracle that is ever programmed is H_{PCS}, which is used in the polynomial commitment of Section 4.1 (or its batch version, where the security proof does not program any other hash function than H_{PCS}). In particular, evaluation proofs of univariate commitments are faithfully generated by $\mathcal{S}$. Therefore we only need to worry about proofs for incorrect evaluations of multivariate polynomials.

In the treatment of multivariate evaluation proofs, we need to rely either on the simulation-extractability of the PCS from Section 4 or the knowledge-soundness of the one from Section 3.2.

Theorem 6. *In the AGM+ROM, the above instantiation of HyperPlonk provides (straight-line) simulation-extractability under the $(d\ell, d\ell)$-DLOG assumption.* (The proof is given in Supplementary Material D.3.)

References

1. B. Abdolmaleki, N. Glaes, S. Ramacher, and D. Slamanig. Universally composable NIZKs: Circuit-succinct, non-malleable and crs-updatable. Cryptology ePrint Archive Report 2023/097, 2023.

2. B. Abdolmaleki, S. Ramacher, and D. Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In *ACM-CCS*, 2020.
3. O. Amine, K. Baghery, Z. Pindado, and C. Ràfols. Simulation extractable versions of Groth's zk-SNARK revisited. *Int. Journal of Information Security*, 2023.
4. S. Atapoor and K. Baghery. Simulation extractability in Groth's zk-SNARK. In *ESORICS*, 2019.
5. K. Baghery, M. Kohlweiss, J. Siim, and M. Volkhov. Another look at extraction and randomization of Groth's zk-SNARK. In *FC*, 2021.
6. K. Baghery, Z. Pindado, and C. Ràfols. Simulation extractable versions of Groth's zk-SNARK revisited. In *CANS*, 2020.
7. S. Bayer and J. Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Eurocrypt*, 2012.
8. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Fast Reed-Solomon interactive oracle proofs of proximity. In *ICALP*, 2018.
9. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable zero knowledge with no trusted setup. In *Crypto*, 2019.
10. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. Ward. Aurora: Transparent succinct arguments for R1CS. In *Eurocrypt*, 2019.
11. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *TCC*, 2016B.
12. D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Halo infinite: Recursive zk-SNARKs from any additive polynomial commitment scheme. In *Crypto*, 2021.
13. J. Bootle, A. Chiesa, Y. Hu, and M. Orrù. Gemini: Elastic SNARKs for diverse environments. In *Eurocrypt*, 2022.
14. S. Bowe and A. Gabizon. Making Groth's zk-SNARK simulation extractable in the random oracle model. Cryptology ePrint Archive Report 2018/187.
15. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE S&P*, 2018.
16. M. Campanelli, N. Gailly, R. Gennaro, M. Jovanovic, P. Mihali, and J. Thaler. Testudo: Linear time prover SNARKs with constant size proofs and square root size universal setup. In *Latincrypt*, 2023.
17. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, 2001.
18. B. Chen, B. Bünz, D. Boneh, and Z. Zhang. Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In *Eurocrypt*, 2023.
19. A. Chiesa, Y. Hu, M. Maller, P. Mishra, P. Vesely, and N. Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. In *Eurocrypt*, 2020.
20. I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *Crypto*, 1991.
21. Q. Dao and P. Grubbs. Spartan and Bulletproofs are simulation-extractable (for free!). In *Eurocrypt*, 2023.
22. A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero-knowledge. In *Crypto*, 2001.
23. Y. Dodis, K. Haralambiev, A. Lopez-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In *Asiacrypt*, 2010.
24. A. Faonio, D. Fiore, M. Kohlweiss, L. Russo, and M. Zajac. From polynomial IOP and commitments to non-malleable zkSNARKs. In *TCC*, 2023.
25. S. Faust, M. Kohlweiss, G. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In *Indocrypt*, 2012.

26. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto*, 1986.
27. N. Fleischhacker, M. Hall-Andersen, M. Simkin, and B. Wagner. Jackpot: Non-interactive aggregatable lotteries. Cryptology ePrint Archive Report 2023/1570.
28. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *Crypto*, 2018.
29. G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In *Eurocrypt*, 2020.
30. G. Gabizon, Z. Williamson, and O. Ciobotaru. PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953,2019.
31. C. Ganesh, H. Khoshakhlagh, M. Kohlweiss, A. Nitulescu, and M. Zajac. What makes Fiat-Shamir zkSNARKs (Updatable SRS) simulation extractable? In *SCN*, 2022.
32. C. Ganesh, Y. Kondi, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Witness-succinct universally-composable SNARKs. In *Eurocrypt*, 2023.
33. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-Shamir Bulletproofs are non-malleable (in the random oracle model). Cryptology ePrint Archive Report 2023/147.
34. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat–Shamir Bulletproofs are Non-Malleable (in the Algebraic Group Model). In *Eurocrypt*, 2022.
35. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Eurocrypt*, 2013.
36. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Asiacrypt*, 2006.
37. J. Groth. On the size of pairing-based non-interactive arguments. In *Eurocrypt*, 2016.
38. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In *Crypto*, 2017.
39. A. Kate, G. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and applications. In *Asiacrypt*, 2010.
40. J. Katz and N. Nang. Efficiency improvements for signature schemes with tight security reductions. In *ACM-CCS*, 2003.
41. M. Kohlweiss, M. Pancholi, and A. Takahashi. How to compile polynomial IOP into simulation-extractable SNARKs: a modular approach. In *TCC*, 2023.
42. T. Kohrita and P. Towa. Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. Cryptology ePrint Archive Report 2023/917, 2023.
43. A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. Papamanthou, R. Pass, a. shelat, and E. Shi. C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive Report 2015/1093.
44. H. Lipmaa. A unified framework for non-universal SNARKs. In *PKC*, 2022.
45. C. Lund, L. Fortnow, H. Karlo, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4), 1992.
46. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updateable structured reference strings. In *ACM-CCS*, 2019.
47. C. Papamanthou, E. Shi, and R. Tamassia. Signatures of correct computation. In *TCC*, 2013.

48. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. of Cryptology*, 2000.
49. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, 1999.
50. C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Crypto*, 1989.
51. S. Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *Crypto*, 2020.
52. S. Setty and J. Lee. Quarks: Quadruple-efficient transparent zkSNARKs. Cryptology ePrint Archive 2020/1275.
53. J. Thaler. Time-optimal interactive proofs for circuit evaluation. In *Crypto*, 2013.
54. T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song. Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation. In *Crypto*, 2019.
55. J. Zhang, T. Xie, Y. Zhang, and D. Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *IEEE Symposium on Security and Privacy*, 2020.
56. Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. A zero-knowledge version of vSQL. Cryptology ePrint Archive Report 2017/1146.

A Deferred Material for the PCS of Sections 3 and 4

A.1 Proof of Theorem 1

Proof. We prove that, if an algebraic adversary $\mathcal{A}$ can break the (extended) knowledge-soundness property, we can solve the $(d\ell, d\ell)$-DLOG problem by computing $\alpha \in \mathbb{Z}_p$ from a problem instance

$$\mathsf{inst} = \big(g, \{g^{(\alpha^i)}\}_{i=1}^{d\ell}, \hat{g}, \{\hat{g}^{(\alpha^i)}\}_{i=1}^{d\ell}\big)$$

The reduction $\mathcal{B}$ first chooses random vectors $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_\ell, \rho_r) \xleftarrow{R} \mathbb{Z}_p^{\ell+1}$, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_\ell, \theta_r) \xleftarrow{R} \mathbb{Z}_p^{\ell+1}$ and implicitly sets $\alpha_i = \rho_i \cdot \alpha + \theta_i$ for each $i \in [\ell]$ and $\alpha_r = \rho_r \cdot \alpha + \theta_r$. Note that $\mathcal{B}$ can properly simulate the CRS srs from inst and $\{(\rho_i, \theta_i)\}_{i=1}^{\ell}$, (ρ_r, θ_r).

When $\mathcal{A}$ outputs a commitment C and a pair $(y, \boldsymbol{z}) \in \mathbb{Z}_p \times \mathbb{Z}_p^{\mu}$, for some $\mu \leq \ell$, with a claim that C commits to an μ-variate polynomial $f[X_1, \ldots, X_\mu] \in \mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\mu]$ such that $y = f(\boldsymbol{z})$, it must also output a representation of C as $C = \prod_{\mathcal{I} \in \mathcal{W}_{d,\ell}} g_{\mathcal{I}}^{f_{\mathcal{I}}} \cdot g_r^r$ for known coefficients $(r, \{f_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{d,\ell}})$ in $\mathbb{Z}_p$. Let the polynomials

$$f[X_1, \ldots, X_\ell] = \sum_{\mathcal{I}=(i_1,\ldots,i_\ell) \in \mathcal{W}_{d,\ell}} f_{\mathcal{I}} \cdot \prod_{j=1}^{\ell} X_j^{i_j}.$$

$$F[X_1, \ldots, X_\ell, X_r] = f[X_1, \ldots, X_\ell] + r \cdot X_r$$

for which we have $C = g^{F(\alpha_1,\ldots,\alpha_\ell,\alpha_r)} = g^{f(\alpha_1,\ldots,\alpha_\ell)} \cdot g_r^r$.

If $f[z_1, \ldots, z_\mu, X_{\mu+1}, \ldots, X_\ell]$ is constantly y for any assignment of the variables $(X_{\mu+1}, \ldots, X_\ell)$, then $\mathcal{B}$ succeeds as a knowledge extractor since (f, r) is a valid witness. Moreover, this implies that the difference

$$f[z_1, \ldots, z_\mu, X_{\mu+1}, \ldots, X_\ell] - f[z_1, \ldots, z_\mu, 0, \ldots, 0] \tag{20}$$

is the zero polynomial. In turn, this implies that, for a random $\boldsymbol{z}' \in \mathbb{Z}_p^{\ell-\mu}$ (and even any arbitrary $\boldsymbol{z}' \in \mathbb{Z}_p^{\ell-\mu}$), the extended input $(\boldsymbol{z} \mid \boldsymbol{z}') \in \mathbb{Z}_p^{\ell}$ is a zero of

$$f[X_1, \ldots, X_\ell] - f[X_1, \ldots, X_\mu, \mathbf{0}^{\ell-\mu}].$$

So, if $\boldsymbol{z} \in \mathbb{Z}_p^{\mu}$ is a random evaluation input chosen independently of f, the Schwartz-Zippel lemma implies $f[X_1, \ldots, X_\ell] = f[X_1, \ldots, X_\mu, \mathbf{0}^{\ell-\mu}]$ with overwhelming probability (in which case the extracted $f[X_1, \ldots, X_\ell]$ does not depend on the variables $(X_{\mu+1}, \ldots, X_\ell)$) since, otherwise, the difference (20) would cancel with probability $d\ell/p$.

We now assume that $f[z_1, \ldots, z_\mu, X_{\mu+1}, \ldots, X_\ell] - y$ is a non-zero polynomial. Together with its fake proof $\boldsymbol{\pi} = (\{\pi_{1,i}\}_{i=1}^{\mu}, \pi_2)$, $\mathcal{A}$ also outputs algebraic representations of $\{\pi_{1,i}\}_{i=1}^{\mu}$ and π_2, which reveal polynomials

$$\begin{aligned} A_i[X_1, \ldots, X_\ell, X_r] &= a_i[X_1, \ldots, X_\ell] + \bar{a}_i \cdot X_r, \qquad \forall i \in [\mu] \\ B[X_1, \ldots, X_\ell, X_r] &= b[X_1, \ldots, X_\ell] + \bar{b} \cdot X_r \end{aligned}$$

where $a_i[X_1, \ldots, X_\ell], b_i[X_1, \ldots, X_\ell] \in \mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\ell]$, $\bar{a}_1, \ldots, \bar{a}_\mu, \bar{b} \in \mathbb{Z}_p$, such that $\pi_2 = g^{B(\alpha_1, \ldots, \alpha_\ell, \alpha_r)}$ and $\pi_{1,i} = g^{A_i(\alpha_1, \ldots, \alpha_\ell, \alpha_r)}$ for each $i \in [\mu]$. The verification equation (5) then implies

$$F(\alpha_1, \ldots, \alpha_\ell, \alpha_r) - y = \sum_{i=1}^{\mu} A_i(\alpha_1, \ldots, \alpha_\ell, \alpha_r) \cdot (\alpha_i - z_i) + B(\alpha_1, \ldots, \alpha_\ell, \alpha_r) \cdot \alpha_r.$$

This means that the $(\ell+1)$-variate polynomial

$$\begin{aligned} T[X_1, \ldots, X_\ell, X_r] \triangleq\ & F[X_1, \ldots, X_\ell, X_r] - y \\ & - \sum_{i=1}^{\mu} A_i[X_1, \ldots, X_\ell, X_r] \cdot (X_i - z_i) - B[X_1, \ldots, X_\ell, X_r] \cdot X_r \end{aligned} \tag{21}$$

vanishes on $(\alpha_1, \ldots, \alpha_\ell, \alpha_r) \in \mathbb{Z}_p^{\ell+1}$. This polynomial is not identically zero since

$$T[z_1, \ldots, z_\mu, X_{\mu+1}, \ldots, X_\ell, 0] = f[z_1, \ldots, z_\mu, X_{\mu+1}, \ldots, X_\ell] - y$$

is non-zero by hypothesis.

Now, let the univariate polynomial

$$L[X] \triangleq T[\rho_1 X + \theta_1, \ldots, \rho_\ell X + \theta_\ell, \rho_r X + \theta_r]$$

If it is not identically zero, then $\alpha \in \mathbb{Z}_p$ is computable as a root of $L[X]$ since $L(\alpha) = T(\alpha_1, \ldots, \alpha_\ell, \alpha_r) = 0$.

We now consider the event zero_L that $L[X]$ is identically zero although $T[X_1, \ldots, X_{\ell+1}]$ is not and argue that zero_L occurs with negligible probability. If $L[X]$ is identically zero, then $(\theta_1, \ldots, \theta_\ell, \theta_r) \in \mathbb{Z}_p^{\ell+1}$ is also a vanishing point of $T[X_1, \ldots, X_\ell, X_r]$ since $L(0) = T(\theta_1, \ldots, \theta_\ell, \theta_r)$. However, conditionally on the adversary's view, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_\ell, \theta_r)$ is uniformly distributed in $\mathbb{Z}_p^{\ell+1}$ since the CRS srs only depends on $\alpha_i = \rho_i \cdot \alpha + \theta_i$ for $i \in [\ell]$ and $\alpha_r = \rho_r \cdot \alpha + \theta_r$.[5] Although $\mathcal{A}$ implicitly generates a polynomial $T[X_1, \ldots, X_\ell, X_r]$

[5] Note that $\{g_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{d,\ell}}$ do not reveal any more information than $\{\hat{g}_i\}_{i=1}^{\ell}$ since they are uniquely determined by $\{\alpha_i\}_{i=1}^{\ell}$.

such that $T(\alpha_1, \ldots, \alpha_\ell, \alpha_r) = 0$, this polynomial does not depend on $\boldsymbol{\theta}$ and the probability to have $T(\theta_1, \ldots, \theta_\ell, \theta_r) = 0$ is the same as if we were first choosing the secret exponents $(\alpha_1, \ldots, \alpha_\ell, \alpha_r) \xleftarrow{R} \mathbb{Z}_p^{\ell+1}$ before letting $\mathcal{A}$ choose the polynomial $T[X_1, \ldots, X_\ell, X_r]$, at which point we would sample $(\theta_1, \ldots, \theta_\ell, \theta_r) \xleftarrow{R} \mathbb{Z}_p^{\ell+1}$ and define $\rho_r = (\alpha_r - \theta_r)/\alpha$ and $\rho_i = (\alpha_i - \theta_i)/\alpha$ for each $i \in [\ell]$. Since $T[X_1, \ldots, X_\ell, X_r]$ has total degree $\leq d\ell + 1$, we have

$$\Pr[\mathsf{zero}_L] \leq \Pr_{(\theta_1, \ldots, \theta_\ell, \theta_r)}[T(\theta_1, \ldots, \theta_\ell, \theta_r) = 0] \leq (d\ell + 1)/p$$

(by the Schwartz-Zippel lemma) which is negligible as claimed.

To conclude, we observe that the proof carries over when $\mathcal{A}$ obtains honestly generated commitments and proofs from an oracle. While the algebraic representation of $\mathcal{A}$'s commitment and proof may depend on the group elements obtained from the oracle, the extractor knows a representation of these w.r.t. the generators contained in srs. It can thus always infer a representation of each group element produced by $\mathcal{A}$ in terms of the components of srs. □

A.2 Batching Proofs for a Common Input in the PCS of Section 3.1

Using standard techniques, the scheme allows aggregating multiple proofs for k_1 distinct commitments $(C_j)_{j\in[k_1]}$ in $\mathbb{G}$ and k_1 outputs $(f_j(\boldsymbol{z}))_{j\in[k_1]}$ on a common input $\boldsymbol{z} \in \mathbb{Z}_p^{\mu'}$. To this end, we need to include a random oracle $H_\delta : \{0,1\}^* \to \mathbb{Z}_p$ in the public parameters.

$\mathsf{Eval}_{\mathsf{srs}}\big((C_j)_{j\in[k_1]}, \boldsymbol{z}, (y_j)_{j\in[k_1]}, \mathsf{aux}\big)$: On input of commitments $(C_j)_{j\in[k_1]}$, witnesses $\mathsf{aux} = \big((r_j, f_j)_{j\in[k_1]},\big)$, an input $\boldsymbol{z} = (z_1, \ldots, z_{\mu'}) \in \mathbb{Z}_p^{\mu'}$ and outputs $\{y_j = f_j(\boldsymbol{z})\}_{j\in[k_1]}$, return $\perp$ if $y_j \neq f_j(\boldsymbol{z})$ for some $j \in [k_1]$. Otherwise, do the following:

1. For each $j \in [k_1]$, generate a proof

$$\boldsymbol{\pi}_j = \big((\pi_{j,1,i})_{i=1}^{\mu}, \pi_{j,2}\big) \in \mathbb{G}^{\mu'+1}$$

   that $f_j(\boldsymbol{z}) = y_j$ as in Section 3.1.
2. Compute

$$\delta = H_\delta\big((C_j)_{j\in[k_1]}, \boldsymbol{z}, (y_j)_{j\in[k_1]}\big) \in \mathbb{Z}_p. \tag{22}$$

3. Compute $\pi_{1,i} = \prod_{j=1}^{k_1} \pi_{j,1,i}^{\delta^{j-1}}$ for each $i \in [k]$ and $\pi_2 = \prod_{j=1}^{k_1} \pi_{j,2}^{\delta^{j-1}}$.
4. Re-randomize proofs by computing $\tilde{\pi}_{1,i} = \pi_{1,i} \cdot g^{s'_i \cdot \alpha_r}$ for each $i \in [\mu']$ and $\tilde{\pi}_2 = \pi_2 \cdot g^{-\sum_{i=1}^{\mu'} s'_i \cdot (\alpha_i - z_i)}$ where $s'_1, \ldots, s'_{\mu'} \xleftarrow{R} \mathbb{Z}_p$.

Return the proof

$$\boldsymbol{\pi} = \big((\tilde{\pi}_{1,i})_{i=1}^{\mu'}, \tilde{\pi}_2\big) \in \mathbb{G}^{\mu'+1}. \tag{23}$$

$\mathsf{Verify}_{\mathsf{srs}}\big((C_j)_{j\in[k_1]}, (y_j)_{j\in[k_1]}, z, \boldsymbol{\pi}\big)$: On input of $(C_j)_{j\in[k_1]}$, claimed evaluations $\{y_j = f_j(\boldsymbol{z})\}_{j\in[k_1]}$, an input $\boldsymbol{z} \in \mathbb{Z}_p^{\mu'}$, and a candidate proof $\boldsymbol{\pi}$, return 0 if $\boldsymbol{\pi}$ does not parse as in (23). Otherwise, compute δ as per (22) and return 1 if

$$e\Big(\big(\prod_{j=1}^{k_1} C_j \cdot g^{-y_j}\big)^{\delta^{j-1}}, \hat{g}\Big) = \prod_{i=1}^{\mu'} e(\pi_{i,1}, \hat{g}_i \cdot \hat{g}^{-z_i}) \cdot e(\pi_2, \hat{g}_r) \tag{24}$$

and 0 otherwise.

The zero-knowledge simulator of Section 3.1 immediately extends to the above batch version. We now adapt its proof of knowledge-soundness.

Theorem 7. *In the AGM+ROM under the $(d\ell, d\ell)$-DLOG assumption, the batch evaluation protocol provides extended knowledge-soundness as an argument of knowledge of $f_1, \ldots, f_{k_1} \in \mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\ell]$ such that, for all $(x_{\mu'+1}, \ldots, x_\ell) \in \mathbb{Z}_p^{\ell-\mu'}$, we have $f_j(z_1, \ldots, z_{\mu'}, x_{\mu'+1}, \ldots, x_\ell) = y_j$ for all $j \in [k_1]$.*

Proof. The proof proceeds identically to that of Theorem 1 and we only outline the changes.

When the adversary $\mathcal{A}$ outputs a verifying proof $\boldsymbol{\pi}$ for commitments $\{C_j\}_{j=1}^{k_1}$, a common input $\boldsymbol{z} \in \mathbb{Z}_p^{\mu'}$, and claimed evaluations $\{y_j\}_{j=1}^{k_1}$, $\mathcal{B}$ can compute α as follows. Since $\mathcal{A}$ is algebraic, it must also output algebraic representations of each C_j as $C_j = \prod_{\mathcal{I}\in\mathcal{W}_{d,\ell}} g_{\mathcal{I}}^{f_{j,\mathcal{I}}} \cdot g_r^{r_j}$ for known coefficients $(r_j, \{f_{j,\mathcal{I}}\}_{\mathcal{I}\in\mathcal{W}_{d,\ell}})$ in $\mathbb{Z}_p$, which define polynomials $f_j[X_1, \ldots, X_\ell] \in \mathbb{Z}_p^{(\leq d)}[X_1, \ldots, X_\ell]$ and

$$F_j[X_1, \ldots, X_\ell, X_r] = f_j[X_1, \ldots, X_\ell] + r_j \cdot X_r$$

such that $C_j = g^{F_j(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$ for each $j \in [k_1]$.

If $f_j[\boldsymbol{z}, X_{\mu'+1}, \ldots, X_\ell] - y_j$ is the zero polynomial for each $j \in [k_1]$, then $\mathcal{B}$ obtained valid witnesses. In this case, if $\boldsymbol{z} \in \mathbb{Z}_p^{\mu'}$ is a random input, we can apply the same argument as in the proof of Lemma 1 to argue that each f_j only depends on the variables $(X_1, \ldots, X_{\mu'})$. We now show that, if one of the polynomials $\{f_j[\boldsymbol{z}, X_{\mu'+1}, \ldots, X_\ell] - y_j\}_{j=1}^{k_1}$ is non-zero, then $\mathcal{B}$ can solve its given $(d\ell, d\ell)$-DLOG instance.

Together with its proof $\boldsymbol{\pi} = (\{\pi_{1,i}\}_{i=1}^{\mu'}, \pi_2)$, $\mathcal{A}$ also outputs representations of $\{\pi_{1,i}\}_{i=1}^{\mu'}$ and π_2, which define polynomials

$$\{A_i[X_1, \ldots, X_\ell, X_r]\}_{i\in[\mu']}, \qquad B[X_1, \ldots, X_\ell, X_r]$$

such that $\pi_{1,i} = g^{A_i(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$ for each $i \in [\mu']$ and $\pi_2 = g^{B(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$.

The verification equation (24) implies

$$\sum_{j=1}^{k_1} \delta^{j-1} \cdot (F_j(\alpha_1, \ldots, \alpha_\ell, \alpha_r) - y_j)$$
$$= \sum_{i=1}^{\mu'} A_i(\alpha_1, \ldots, \alpha_\ell, \alpha_r) \cdot (\alpha_i - z_i) + B(\alpha_1, \ldots, \alpha_\ell, \alpha_r) \cdot \alpha_r,$$

which means that the $(\ell+1)$-variate polynomial

$$T[X_1,\ldots,X_\ell,X_r] \triangleq \sum_{j=1}^{k_1} \delta^{j-1}\cdot\big(F_j[X_1,\ldots,X_\ell]-y_j\big) - \sum_{i=1}^{\mu'} A_i[X_1,\ldots,X_\ell,X_r]\cdot(X_i-z_i) + B[X_1,\ldots,X_\ell,X_r]\cdot X_r$$

has a zero in $(\alpha_1,\ldots,\alpha_\ell,\alpha_r)\in\mathbb{Z}_p^{\ell+1}$ but it is not identically zero w.h.p. if there exists $j\in[k_1]$ such that $F_j[z_1,\ldots,z_{\mu'},X_{\mu'+1},\ldots,X_\ell,0]-y_j$ is non-zero. Indeed, if $T[X_1,\ldots,X_\ell,X_r]$ was zero, so would be

$$T[\boldsymbol{z},X_{\mu'+1},\ldots,X_\ell,0] \;=\; \sum_{j=1}^{k_1}\delta^{j-1}\cdot(F_j[z_1,\ldots,z_{\mu'},X_{\mu'+1},\ldots,X_\ell,0]-y_j).$$

By the Schwartz-Zippel lemma, this can only happen with probability $< k/p$ since $\delta = H_\delta\big((C_j)_{j\in[k_1]},\boldsymbol{z},(y_j)_{j\in[k_1]}\big)$ is uniformly distributed in $\mathbb{Z}_p$ and chosen after the $(\ell+1)$-variate polynomials $\{F_j[X_1,\ldots,X_\ell,X_r]\}_{j=1}^{k_1}$ and the claimed evaluations $\{y_j\}_{j=1}^{k_1}$.

The rest of the proof is identical to the proof of Theorem 1 and allows the reduction to find α by factoring the univariate polynomial

$$L[X] \triangleq T[\rho_1 X+\theta_1,\ldots,\rho_\ell X+\theta_\ell,\rho_r X+\theta_r]$$

which is non-zero with probability $\geq 1-(d\ell+1)/p$. □

A.3 Proof of Theorem 2

Proof. The proof is identical to the proof of Theorem 1 with the difference that, when $\mathcal{A}$ outputs a commitment $\hat{C}$ and a pair $(y,\boldsymbol{z})\in\mathbb{Z}_p\times\mathbb{Z}_p^{\mu'}$ (where μ' may be different from μ) with a statement that $\hat{C}$ commits to an μ'-variate polynomial $f[X_1,\ldots,X_{\mu'}]$ such that $y=f(\boldsymbol{z})$, it must also output a representation of $\hat{C}$ with respect to the group elements $\big(\{\hat{g}_{\mathcal{I}}\}_{\mathcal{I}\in\mathcal{W}_{1,\ell}},\{\hat{g}^{(\alpha_1+\alpha_{\mu+1})^i)\cdot\alpha_\mu\cdot(\alpha_\mu-1)}\}_{i=0}^{3}\big)$. This representation defines a a polynomial

$$f[X_1,\ldots,X_\ell] = f'[X_1,\ldots,X_\ell] + R[X_1+X_{\mu+1}]\cdot X_\mu\cdot(X_\mu-1),$$

where $f'\in\mathbb{Z}_p^{(\leq 1)}[X_1,\ldots,X_\ell]$ is multilinear and where $R[X]$ is univariate of degree ≤ 3, such that $\hat{C}=\hat{g}^{f(\alpha_1,\ldots,\alpha_\ell)+r\cdot\alpha_r}$. Then, if the extracted $(r,f[X_1,\ldots,X_\ell])$ does not form a valid witness (i.e., such that $f[z_1,\ldots,z_{\mu'},X_{\mu'+1},\ldots,X_\ell]-y$ is the zero-polynomial), $\mathcal{B}$ can break the assumption in the same way as in the proof of Theorem 1. □

A.4 Batching Proofs for a Common Input for the Simulation-Extractable PCS of Section 4

The construction of Section 4.1 readily extends to compress proofs when multiple evaluations $\{y_j\}_{j=1}^k$ of polynomials $\{f_j[X_1,\ldots,X_\ell]\}_{j=1}^k$ have to be proven for a common input $\boldsymbol{z}\in\mathbb{Z}_p^\mu$. The idea is again to use Fiat-Shamir to prove knowledge of a batched evaluation proof. Namely, the prover first computes $(\{\pi_{1,i}\}_{i=1}^{\mu'},\pi_2)$ as in Section A.2 but, instead of revealing it in the clear, it only reveals $\{\pi_{1,i}\}_{i=1}^{\mu'}$ and proves knowledge of $\pi_2\in\mathbb{G}$ satisfying

$$e\Big(g,\big(\prod_{j=1}^{k}\hat{C}_j\cdot\hat{g}^{-y_j}\big)^{\delta^{j-1}}\Big)\cdot\prod_{i=1}^{\mu'}e(\pi_{i,1},\hat{g}_i\cdot\hat{g}^{-z_i})^{-1}=e(\pi_2,\hat{g}_r)$$

where $\delta=H_\delta\big((\hat{C}_j)_{j\in[k]},\boldsymbol{z},(y_j)_{j\in[k]}\big)$.

The proof of simulation-extractability is almost identical to that of Theorem 3. The only differences are that equation (10) becomes

$$\begin{aligned}&R(\alpha_1,\ldots,\alpha_\ell,\alpha_r)=S(\alpha_1,\ldots,\alpha_\ell,\alpha_r)\cdot\alpha_r\\&-c\cdot\Big(\sum_{j=1}^{k}\delta^{j-1}(F_j(\alpha_1,\ldots,\alpha_\ell,\alpha_r)-y_j)-\sum_{i=1}^{\mu'}A_i(\alpha_1,\ldots,\alpha_\ell,\alpha_r)\cdot(\alpha_i-z_i)\Big),\end{aligned}$$

while (11) becomes

$$\begin{aligned}&T[X_1,\ldots,X_\ell,X_r]\triangleq R[X_1,\ldots,X_\ell,X_r]-S[X_1,\ldots,X_\ell,X_r]\cdot X_r\\&+c\cdot\Big(\sum_{j=1}^{k}\delta^{j-1}\cdot(F_j[X_1,\ldots,X_\ell,X_r]-y_j)-\sum_{i=1}^{\mu'}A_i[X_1,\ldots,X_\ell,X_r]\cdot(X_i-z_i)\Big),\end{aligned}$$

and which is is a non-zero polynomial with overwhelming probability $>1-k/p$ if there exists $j\in[k]$ such that $f_j[\boldsymbol{z},X_{\mu+1},\ldots,X_\ell]-y_j$ is non-zero.

B Knowledge-Soundness of KZG Commitments with an Extended SRS

In the AGM, we prove the knowledge soundness of the original univariate KZG commitment scheme when the SRS is extended to enable commitments to multivariate polynomials as in 3.2. We only do it for the deterministic version of KZG since it suffices for its application to the sumcheck protocol, as considered in HyperPlonk [18].

B.1 The Case of Individual Proofs

In this extended-SRS setting, the KZG commitment scheme can be spelled out as follows.

CRS-Gen$(1^\lambda, 1^{\bar{d}}, 1^\ell)$**:** On input of a security parameter λ, a number of variables ℓ and a degree $\bar{d}$ such that $(\bar{d}+1)^\ell \in \mathsf{poly}(\lambda)$, generate the SRS as follows:

1. Choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^{l(\lambda)}$, for some function $l : \mathbb{N} \to \mathbb{N}$, and $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$.
2. Pick $\alpha_1, \ldots, \alpha_\ell, \alpha_r \xleftarrow{R} \mathbb{Z}_p$. Compute $g_r = g^{\alpha_r}$ and $\hat{g}_r = \hat{g}^{\alpha_r}$.
3. For each $\mathcal{I} = (i_1, \ldots, i_\ell) \in \mathcal{U}_{\bar{d},\ell}$, compute $g_{\mathcal{I}} = g^{\prod_{j=1}^{\ell} \alpha_j^{i_j}}$.
4. For each $\mathcal{I} = (i_1, \ldots, i_\ell) \in \mathcal{W}_{1,\ell}$, compute $\hat{g}_{\mathcal{I}} = \hat{g}^{\prod_{j=1}^{\ell} \alpha_j^{i_j}}$. Compute $\big\{\hat{g}^{(\alpha_1+\alpha_{\mu+1})^i \cdot \alpha_\mu \cdot (\alpha_\mu - 1)}\big\}_{i=0}^{3}$ for an arbitrary index $\mu \in [2, \ell-1]$.

The public parameters are

$$\mathsf{srs} = \Big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g_r, \hat{g}_r, \{g_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{U}_{\bar{d},\ell}} \\ \{\hat{g}_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{1,\ell}}, \big\{\hat{g}^{(\alpha_1+\alpha_{\mu+1})^i \cdot \alpha_\mu \cdot (\alpha_\mu - 1)}\big\}_{i=0}^{3}\Big).$$

Com$_{\mathsf{srs}}(f)$**:** To commit to $f[X] \in \mathbb{Z}_p^{(\leq \bar{d})}[X]$, compute $C = g^{f(\alpha_1)}$ using $\{g^{(\alpha_1^i)}\}_{i \in [0,\bar{d}]}$ which are contained in $\{g_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{\bar{d},\ell}}$. Then, output $(\mathsf{aux}, f) = (\emptyset, f)$.

Eval$_{\mathsf{srs}}\big(C, z, y, \mathsf{aux}\big)$**:** given a commitment C, a witness $\mathsf{aux} = (\emptyset, f)$, an input $z \in \mathbb{Z}_p$ and an output $y = f(z) \in \mathbb{Z}_p$, return $\perp$ if $y \neq f(z)$. Otherwise, compute $q[X] = (f[X] - y)/(X - z) = \sum_{i=0}^{\bar{d}-1} q_i \cdot X^i$ and the proof

$$\pi = g^{q(\alpha_1)}$$

Return $\pi \in \mathbb{G}$.

Verify$_{\mathsf{srs}}\big(C, y, z, \pi\big)$**:** Given $C \in \mathbb{G}$, an input $z \in \mathbb{Z}_p$, a claimed evaluation $y \in \mathbb{Z}_p$, and a candidate proof $\pi \in \mathbb{G}$, return 1 if

$$e\big(C \cdot g^{-y}, \hat{g}\big) = e(\pi, \hat{g}^{\alpha_1} \cdot \hat{g}^{-z}). \tag{25}$$

and 0 otherwise.

Theorem 8. *Under the $(\bar{d}\ell, \bar{d}\ell)$-DLOG assumption in the AGM, the scheme provides extended knowledge soundness as an argument that C commits to a polynomial $F[X_1, \ldots, X_\ell, X_r]$ such that $F[z, X_2, \ldots, X_\ell, X_r] = y$ for all assignments of $(X_2, \ldots, X_\ell, X_r)$. Moreover, for a random input z, the extractor outputs $F[X]$ such that $F(z) = y$ and $C = g^{F(\alpha_1)}$ with overwhelming probability.*

Proof. We show that, if an algebraic adversary $\mathcal{A}$ can break knowledge-soundness, we can build an algorithm solving the $(\bar{d}\ell, \bar{d}\ell)$-DLOG problem by computing $\alpha \in \mathbb{Z}_p$ from

$$\mathsf{inst} = \big(g, \{g^{(\alpha^i)}\}_{i=1}^{\bar{d}\ell}, \{\hat{g}^{(\alpha^i)}\}_{i=1}^{\bar{d}\ell}, \hat{g}^{\alpha}\big)$$

The reduction $\mathcal{B}$ prepares the SRS as in the proof of Theorem 1. Namely, it first chooses random vectors $(\rho_1, \ldots, \rho_\ell, \rho_r) \xleftarrow{R} \mathbb{Z}_p^\ell$, $(\theta_1, \ldots, \theta_\ell, \theta_r) \xleftarrow{R} \mathbb{Z}_p^\ell$ and implicitly sets $\alpha_i = \rho_i \cdot \alpha + \theta_i$ for each $i \in [\ell] \cup \{r\}$. Note that $\mathcal{B}$ can properly simulate the CRS pp from inst.

When $\mathcal{A}$ outputs a commitment C and a pair $(y, z) \in \mathbb{Z}_p^2$ with a claim that C commits to $F[X_1, \ldots, X_\ell, X_r]$ such that $F[z, X_2, \ldots, X_\ell, X_r] - y$ is the zero polynomial, it must also output a representation of C as $C = \prod_{\mathcal{I} \in \mathcal{U}_{\bar{d},\ell}} g_{\mathcal{I}}^{f_{\mathcal{I}}} \cdot g_r^r$ for known coefficients $(r, \{f_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{\bar{d},\ell}})$ in $\mathbb{Z}_p$. Let the polynomial

$$f[X_1, \ldots, X_\ell] = \sum_{\mathcal{I}=(i_1,\ldots,i_\ell) \in \mathcal{U}_{\bar{d},\ell}} f_{\mathcal{I}} \cdot \prod_{j=1}^{\ell} X_j^{i_j}.$$

and let

$$F[X_1, \ldots, X_{\ell+1}] = f[X_1, \ldots, X_\ell] + r \cdot X_r$$

for which we have $C = g^{F(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$. If $F[z, X_2, \ldots, X_\ell, X_r] - y$ is the zero polynomial, then $\mathcal{B}$ is done since (F, r) is a valid witness. We henceforth assume that $F[z, X_2, \ldots, X_\ell, X_r] - y$ is non-zero.

Together with its fake proof π, $\mathcal{A}$ also outputs an algebraic representation of π, which defines a polynomial

$$A[X_1, \ldots, X_\ell, X_r] = a[X_1, \ldots, X_\ell] + \bar{a} \cdot X_r,$$

such that $\pi = g^{A(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$, where $a[X_1, \ldots, X_\ell] \in \mathbb{Z}_p^{(\leq \bar{d})}[X_1, \ldots, X_\ell]$ and $\bar{a} \in \mathbb{Z}_p$. The verification equation (25) implies

$$F(\alpha_1, \ldots, \alpha_\ell, \alpha_r) - y = A(\alpha_1, \ldots, \alpha_\ell, \alpha_r) \cdot (\alpha_1 - z)$$

which means that $(\alpha_1, \ldots, \alpha_\ell, \alpha_r) \in \mathbb{Z}_p^{\ell+1}$ is a zero of the $(\ell+1)$-variate polynomial

$$\begin{aligned} T[X_1, \ldots, X_\ell, X_r] \triangleq\ & F[X_1, \ldots, X_\ell, X_r] - y \\ & - A[X_1, \ldots, X_\ell, X_r] \cdot (X_1 - z) \end{aligned} \tag{26}$$

However, $T[X_1, \ldots, X_\ell, X_r]$ cannot identically zero since we have

$$T[z, X_2, \ldots, X_\ell, X_r] = F[z, X_2, \ldots, X_\ell, X_r] - y,$$

which is non-zero by hypothesis.

Now, let the univariate

$$t[X] \triangleq T[\rho_1 \cdot X + \theta_1, \rho_2 \cdot X + \theta_2, \ldots, \rho_\ell X + \theta_\ell, \rho_r X + \theta_r]$$

of which $\alpha_1 = \alpha$ is a root. We note that $t[X]$ is not identically zero w.h.p. as this would imply $t(0) = T(\theta_1, \ldots, \theta_\ell, \theta_r) = 0$ which is only possible with negligible probability by the same argument as in the proof of Theorem 1. This allows computing α by factoring $t[X_1]$.

To conclude the proof, we remark that, if $F[z, X_2, \ldots, X_\ell, X_r] - y$ is the zero polynomial, we have $F[z, 0, \ldots, 0, 0] = y$. In particular, for a random tuple $(z_2, \ldots, z_\ell, z_r) \in \mathbb{Z}_p^\ell$, we have $F[z, z_2, \ldots, z_\ell, z_r] - F[z, 0, \ldots, 0, 0]$. If z is a random evaluation input, this is only possible with negligible probability if $F[X_1, X_2, \ldots, X_\ell, X_r] \neq F[X_1, 0, \ldots, 0, 0]$. □

B.2 Batch Evaluations of KZG with an Extended SRS

We now prove the knowledge-soundness of the batch evaluation protocol of Boneh *et al.* [12, Section 4.1] when KZG commitments are used over an extended SRS for the multivariate PCS of Section 3.2.

The SRS of the scheme is the same as in Section B except that it includes the description of an additional hash function $H_{\mathsf{batch}} : \{0,1\}^* \to \mathbb{Z}_p$ modeled as a random oracle.

$\mathsf{Eval}_{\mathsf{srs}}\big(\{C_i\}_{i=1}^k, \{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k, \mathsf{aux}\big)$**:** given commitments $\{C_i\}_{i=1}^k$ to polynomials $\{f_i[X_1]\}_{i=1}^k$ with corresponding vectors of inputs $\{\boldsymbol{\Omega}_i = (\omega_{i,1}, \ldots, \omega_{i,k_i})\}_{i=1}^k$ and outputs $\{\boldsymbol{t}_i = (t_{i,1}, \ldots, t_{i,k_i})\}_{i=1}^k$ and witnesses $\mathsf{aux} = (\emptyset, \{f_i[X_1]\}_{i=1}^k)$, return $\perp$ if there exists $i \in [k]$ such that $\boldsymbol{\Omega}_i \in \mathbb{Z}_p^{k_i}$ contains an element $\omega_{i,j}$ such that $f_i(\omega_{i,j}) \neq t_{i,j}$. Otherwise,

1. For each $i \in [k]$, let $t_i[X_1]$ the unique polynomial of degree k_i such that $t_i(\omega_{i,j}) = t_{i,j}$ and compute $\tilde{C}_i = C_i \cdot g^{-t_i(\alpha_1)}$.
2. Let the sets $\{\Omega_i = \{\omega_{i,1}, \ldots, \omega_{i,k_i}\}\}_{i=1}^k$. Let $\Omega = \bigcup_{i=1}^k \Omega_i$ and $\bar{\Omega}_i = \Omega \setminus \Omega_i$ for each $i \in [k]$. Compute the polynomials $z[X_1] = \prod_{\omega \in \Omega}(X_1 - \omega)$ and $z_i[X_1] = \prod_{\omega \in \bar{\Omega}_i}(X_1 - \omega)$ for each $i \in [k]$.
3. Compute $\epsilon = H_{\mathsf{batch}}(\{C_i\}_{i=1}^k, \{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k) \in \mathbb{Z}_p$.
4. Compute $q[X_1] = \sum_{i=1}^k \epsilon^{i-1} \cdot z_i[X_1] \cdot f_i[X_1]/z[X_1]$, which is a polynomial of degree $\leq \bar{d}$ since $z_i[X_1]$ divides $z[X_1]$. Compute a commitment

$$C_q = g^{q(\alpha_1)}$$

   using $\{g^{(\alpha_1^i)}\}_{i \in [0,d]}$ which are contained in $\{g_{\mathcal{I}}\}_{\mathcal{I} \in \mathcal{W}_{d,\ell}}$.
5. Compute $r = H_{\mathsf{batch}}(\{C_i\}_{i=1}^k, \{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k, C_q) \in \mathbb{Z}_p$.
6. Compute

$$g[X_1] = \sum_{i=1}^k \epsilon^{i-1} \cdot z_i(r) \cdot f_i[X_1] - z(r) \cdot q[X_1]$$

   which is the committed polynomial that underlies

$$C_g = C_q^{-z(r)} \cdot \prod_{i=1}^k \tilde{C}_i^{\,\epsilon^{i-1} \cdot z_i(r)}. \tag{27}$$

   Generate a proof $\pi_g \in \mathbb{G}$ that C_g evaluates to 0 the input r.

Return $\boldsymbol{\pi} = (C_q, \pi_g) \in \mathbb{G}^2$.

$\mathsf{Verify}_{\mathsf{srs}}\big(\{C_i\}_{i=1}^k, \{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k, \boldsymbol{\pi}\big)$**:** Given commitments $\{C_i\}_{i=1}^k$, claimed evaluation input/output vectors $\{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k$, and a candidate $\boldsymbol{\pi} = (C_q, \pi_g) \in \mathbb{G}^2$,

1. For each $i \in [k]$, compute $\tilde{C}_i = C_i \cdot g^{-t_i(\alpha_1)}$ using $\{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k$ as in step 1 of Eval.

2. Let $z[X_1] = \prod_{\omega\in\Omega}(X_1-\omega)$ and $z_i[X_1] = \prod_{\omega\in\bar{\Omega}_i}(X_1-\omega)$ for each $i\in[k]$ with $\Omega = \bigcup_{i=1}^k \Omega_i$ and $\bar{\Omega}_i = \Omega\setminus\Omega_i$ for each $i\in[k]$. Compute

$$\epsilon = H_{\mathsf{batch}}(\{C_i\}_{i=1}^k, \{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k) \in \mathbb{Z}_p$$
$$r = H_{\mathsf{batch}}(\{C_i\}_{i=1}^k, \{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k, C_q) \in \mathbb{Z}_p.$$

and then

$$C_g = C_q^{-z(r)} \cdot \prod_{i=1}^k \tilde{C}_i^{\,\epsilon^{i-1}\cdot z_i(r)}.$$

3. Return 1 if

$$e\big(C_g, \hat{g}\big) = e(\pi_g, \hat{g}^{\alpha_1}\cdot\hat{g}^{-r}). \tag{28}$$

and 0 otherwise.

The proof of knowledge soundness proceeds analogously to the one of Theorem 8 and adapts ideas from [12, Theorem 5]. The main difference is that, in order to use it in HyperPlonk while preserving straight-line extraction in the security proof, we need to make sure that the reduction prepares the SRS in the same way as in the security proof of the multivariate PCS scheme.

Theorem 9. *In the AGM+ROM and under the $(\bar{d}\ell, \bar{d}\ell)$-DLOG assumption, the batch evaluation protocol is an extended knowledge-sound argument of knowledge of polynomials $\{F_i \in \mathbb{Z}_p^{(\leq\bar{d})}[X_1,\ldots,X_\ell,X_r]\}_{i=1}^k$ such that $C_i = g^{F_i(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$ and $F_i(\omega_{i,j},0,\ldots,0) = t_{i,j}$ for all $i\in[k]$ and $j\in[k_i]$.*

Proof. Assuming that an algebraic adversary $\mathcal{A}$ can break knowledge-soundness, we give a reduction $\mathcal{B}$ solving the $(d\ell, d\ell)$-DLOG problem by computing $\alpha\in\mathbb{Z}_p$ from $\mathsf{inst} = \big(g, \{g^{(\alpha^i)}\}_{i=1}^{\bar{d}\ell}, \hat{g}, \{\hat{g}^{(\alpha^i)}\}_{i=1}^{\bar{d}\ell}\big)$.

Algorithm $\mathcal{B}$ sets up the structured reference string by choosing vectors $(\rho_1,\ldots,\rho_\ell,\rho_r) \xleftarrow{R} \mathbb{Z}_p^{\ell+1}$, $(\theta_1,\ldots,\theta_\ell,\theta_r) \xleftarrow{R} \mathbb{Z}_p^{\ell+1}$ and implicitly setting $\alpha_i = \rho_i\cdot\alpha+\theta_i$ for each i and $\alpha_r = \rho_r\cdot\alpha+\theta_r$ before simulating pp from inst.

When $\mathcal{A}$ halts, it outputs commitment $\{C\}_{i=1}^k$ and vectors $\{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k$ together with a claim that each C_i commits to some $f_i[X_1]\in\mathbb{Z}_p^{(\leq\bar{d})}[X_1]$ such that $f_i(\omega_{i,j}) = t_{i,j}$ for each $j\in[k_i]$, where $\boldsymbol{\Omega}_i = (\omega_{i,1},\ldots,\omega_{i,k_i})$ and $\boldsymbol{t}_i = (t_{i,1},\ldots,t_{i,k_i})$. It also outputs a representation of each commitment C_i w.r.t. generators $(\{g_{\mathcal{I}}\}_{\mathcal{I}\in\mathcal{W}_{\bar{d},\ell}}, g_r)$. These representations define multivariate polynomials $f_i[X_1,\ldots,X_\ell]\in\mathbb{Z}_p^{(\leq\bar{d})}[X_1,\ldots,X_\ell]$ and

$$F_i[X_1,\ldots,X_\ell,X_r] = f_i[X_1,\ldots,X_\ell] + r_i\cdot X_r,$$

for some known $r_i\in\mathbb{Z}_p$, such that we have $C_i = g^{F_i(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$.

Together with its fake proof $\boldsymbol{\pi} = (C_q,\pi_g)\in\mathbb{G}^2$, $\mathcal{A}$ also outputs an algebraic representation of π_g, which defines a polynomial

$$A[X_1,\ldots,X_\ell,X_r] = a[X_1,\ldots,X_\ell] + \bar{a}\cdot X_r,$$

such that $\pi_g = g^{A(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$, where $a[X_1,\ldots,X_\ell] \in \mathbb{Z}_p^{(\leq \bar{d})}[X_1,\ldots,X_\ell]$ and $\bar{a} \in \mathbb{Z}_p$. It similarly provides an algebraic representation of C_q, which defines a polynomial

$$Q[X_1,\ldots,X_\ell,X_r] = q[X_1,\ldots,X_\ell] + \bar{q}\cdot X_r,$$

such that $C_q = g^{Q(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$, where $q[X_1,\ldots,X_\ell] \in \mathbb{Z}_p^{(\leq \bar{d})}[X_1,\ldots,X_\ell]$ and $\bar{q} \in \mathbb{Z}_p$.

The verification equation (28) implies

$$\begin{aligned} -z(r)\cdot Q(\alpha_1,\ldots,\alpha_\ell,\alpha_r) + \sum_{i=1}^{k} \epsilon^{i-1}\cdot z_i(r)\cdot (F_i(\alpha_1,\ldots,\alpha_\ell,\alpha_r) - t_i(\alpha_1)) \\ = A(\alpha_1,\ldots,\alpha_\ell,\alpha_r)\cdot(\alpha_1 - r). \end{aligned}$$

This implies that $(\alpha_1,\ldots,\alpha_\ell,\alpha_r) \in \mathbb{Z}_p^{\ell+1}$ is a zero of

$$\begin{aligned} T[X_1,\ldots,X_\ell,X_r] \triangleq\ & -z(r)\cdot Q[X_1,\ldots,X_\ell,X_r] \\ & + \sum_{i=1}^{k} \epsilon^{i-1}\cdot z_i(r)\cdot (F_i[X_1,\ldots,X_\ell,X_r] - t_i[X_1]) \\ & - A[X_1,\ldots,X_\ell,X_r]\cdot(X_1 - r) \end{aligned} \tag{29}$$

If $T[X_1,\ldots,X_\ell,X_r]$ is a non-zero polynomial, the sought-after $\alpha \in \mathbb{Z}_p$ is computable as a root of the univariate polynomial

$$L[X] \triangleq T[\rho_1\cdot X + \theta_1,\ldots,\rho_\ell\cdot X + \theta_\ell, \rho_r\cdot X + \theta_r]$$

Indeed, by the same argument as in Theorem 1, if $T[X_1,\ldots,X_\ell,X_r]$ is not identically zero, neither is $L[X]$ except with negligible probability.

We now assume that $T[X_1,\ldots,X_r]$ is identically zero. In particular, we have $T(r,0,\ldots,0) = 0$ and thus

$$-z(r)\cdot Q[r,0,\ldots,0,0] + \sum_{i=1}^{k} \epsilon^{i-1}\cdot z_i(r)\cdot \big(F_i[r,0,\ldots,0,0] - t_i(r)\big) = 0 \tag{30}$$

Now, if

$$z[X_1]\cdot Q[X_1,0,\ldots,0,0] \neq \sum_{i=1}^{k} \epsilon^{i-1}\cdot z_i[X_1]\cdot (F_i[X_1,0,\ldots,0] - t_i[X_1]),$$

the Schwartz-Zippel lemma implies that (30) can only hold with probability d/p since $r = H_{\mathsf{batch}}(\{C_i\}_{i=1}^k, \{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k, C_q)$ is defined after the sets $\{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k$ (which define $z[X]$, $\{z_i[X], t_i[X]\}_{i=1}^k$), the commitments $\{C_i\}_{i=1}^k$ (which determine $\{F_i[X_1,\ldots,X_\ell,X_r]\}_{i=1}^k$ via their algebraic representation) and C_q (which

determines $Q[X_1, X_1, \ldots, X_\ell, X_r]$).[6]

We now assume

$$z[X_1] \cdot Q[X_1, 0, \ldots, 0, 0] = \sum_{i=1}^{k} \epsilon^{i-1} \cdot z_i[X_1] \cdot (F_i[X_1, 0, \ldots, 0] - t_i[X_1]), \quad (31)$$

which implies that the vanishing polynomial $z[X_1] = \prod_{\omega \in \Omega}(X_1 - \omega)$ divides

$$\sum_{i=1}^{k} \epsilon^{i-1} \cdot z_i[X_1] \cdot (F_i[X_1, 0, \ldots, 0] - t_i[X_1]),$$

where $z_i[X_1] = \prod_{\omega \in \Omega \setminus \Omega_i}(X_1 - \omega)$ for each $i \in [k]$. However, if there exists $i \in [k]$ such that $F_i[X_1, 0, \ldots, 0] - t_i[X_1]$ does not cancel on Ω_i, it implies that $z[X_1]$ does not divide $z_i[X_1] \cdot (F_i[X_1, 0, \ldots, 0] - t_i[X_1])$. Then, by [30, Claim 4.6], the equality (31) can only hold with negligible probability k/p over the choice of $\epsilon = H_{\mathsf{batch}}(\{C_i\}_{i=1}^k, \{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k)$, which is chosen uniformly after $\{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k$ and the commitments $\{C_i\}_{i=1}^k$.

This shows that, if there exists an index $i \in [k]$ such that the difference $F_i[X_1, 0, \ldots, 0] - t_i[X_1]$ does not cancel on the set Ω_i, then $T[X_1, \ldots, X_\ell, X_r]$ is non-zero with overwhelming probability, thus allowing $\mathcal{B}$ to compute α w.h.p. by factoring $L[X]$.

Finally, if $F_i[X_1, 0, \ldots, 0] - t_i[X_1]$ vanishes on Ω_i for all $i \in [k]$, then we have $F_i(\omega_{i,j}, 0, \ldots, 0) = t_{i,j}$ for each $i \in [k]$ and $j \in [k_i]$, meaning that $\{F_i\}_{i=1}^k$ are valid witnesses. □

C More Efficient Simulation-Extractable KZG Commitments in the Univariate Case

When restricting the scheme of Section 4 to univariate polynomials, we immediately obtain a simulation-extractable PCS that can be used to build simulation-extractable SNARKs from univariate PIOPs. In this section, we provide a more efficient construction in the univariate case. In particular, it can be used to obtain a new simulation-extractable instantiation of Plonk.

Our randomized version of KZG uses the same commitment algorithm as the one obtained from the homomorphic-to-hiding transformation of Boneh *et al.* [12, Appendix B.2]. However, its evaluation proofs are very different.

A commitment to a polynomial $f[X] = \sum_{i=0}^{d} f_i \cdot X^i$ is obtained by choosing $\gamma \xleftarrow{R} \mathbb{Z}_p$ and computing

$$C = g^\gamma \cdot \prod_{i=1}^{d+1} g_i^{f_{i-1}},$$

[6] We also assume that the hash value $r = H_{\mathsf{batch}}(\{C_i\}_{i=1}^k, \{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k, C_q)$ is defined after $\epsilon = H_{\mathsf{batch}}(\{C_i\}_{i=1}^k, \{\boldsymbol{\Omega}_i, \boldsymbol{t}_i\}_{i=1}^k)$ but this can be enforced by having the former hash query trigger the latter.

which we interpret (as also done in [12]) as a deterministic KZG commitment $C = g^{F(\alpha)}$ to $F[X] = \gamma + X \cdot f[X]$. We use a different evaluation protocol in order to prove that $y = f(z)$ for a given input $z \in \mathbb{Z}_p$. Instead of masking $f[X]$ with a random committed polynomial $s[X]$, we prove knowledge of $\gamma \in \mathbb{Z}_p$ such that $C \cdot g^{-\gamma}$ is a commitment to a polynomial $F_0 \triangleq F[X] - \gamma = X \cdot f[X]$ for which $F_0(z) = z \cdot y$.

We also need to prove that $F_0[X]$ has no degree-0 monomial (i.e., $F_0(0) = 0$), which is necessary to convince the verifier that C is really a deterministic KZG commitment to a polynomial of the form $F[X] = \gamma + X \cdot f[X]$. To this end, we use the batch evaluation protocol of [39, Section 3.4] and prove that $F_0[X] - y \cdot X$ is divisible by $X \cdot (X - z)$ (note that $r[X] = y \cdot X$ is the unique degree-1 polynomial such that $r(z) = z \cdot y$ and $r(0) = 0$). In the evaluation protocol, the prover thus generates a proof of knowledge of $(\gamma, \pi) \in \mathbb{Z}_p \times \mathbb{G}$ such that

$$e(C \cdot g^{-\gamma} \cdot g_1^{-y}, \hat{g}) = e(\pi, \hat{g}_2 \cdot \hat{g}_1^{-z})$$

which can be done using a standard Σ-protocol. The resulting NIZK proof consists of one element of $\mathbb{G}$ and two elements of $\mathbb{Z}_p$. In comparison, the technique of Boneh *et al.* [12, Appendix B] requires evaluation proofs consisting of 3 elements of $\mathbb{G}$ and 3 elements of $\mathbb{Z}_p$.

C.1 Description

CRS-Gen$(1^\lambda, 1^d)$**:** On input of a security parameter λ and the maximal degree of committed polynomials $d \in \text{poly}(\lambda)$, do the following:

1. Choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^{l(\lambda)}$, for some function $l : \mathbb{N} \to \mathbb{N}$, and $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$.
2. Pick a random $\alpha \xleftarrow{R} \mathbb{Z}_p$ and compute $g_1, \ldots, g_{d+1} \in \mathbb{G}$, where $g_i = g^{(\alpha^i)}$ for each $i \in [d+1]$ and $\hat{g}_1 = \hat{g}^\alpha$, $\hat{g}_2 = \hat{g}^{(\alpha^2)}$.
3. Choose a hash function $H : \{0,1\}^* \to \mathbb{Z}_p$ modeled as a random oracle.

The public parameters are defined to be

$$\mathsf{srs} = \Big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \{g_i\}_{i \in [d+1]}, \hat{g}, \hat{g}_1, \hat{g}_2, H \Big)$$

Com$_{\mathsf{srs}}(f)$**:** To commit to a polynomial $f[X] = \sum_{j=0}^{d} f_j \cdot X^j \in \mathbb{Z}_p^{(\leq d)}[X]$, choose a random $\gamma \xleftarrow{R} \mathbb{Z}_p$ and compute

$$C = g^\gamma \cdot \prod_{j=1}^{d+1} g_j^{f_{j-1}}$$

Return $C \in \mathbb{G}$ and the opening information $\mathsf{aux} = (\gamma, f) \in \mathbb{Z}_p \times \mathbb{Z}_p^{(\leq d)}[X]$.

Eval$_{\mathsf{srs}}\big(C, z, y, \mathsf{aux}, \mathsf{lbl}\big)$**:** given a commitment C, a witness $\mathsf{aux} = (\gamma, \boldsymbol{f})$, an input $z \in \mathbb{Z}_p^*$ and an output $y = f(z) \in \mathbb{Z}_p$, return $\perp$ if $y \neq f(z)$. Otherwise,

1. Let the polynomials $F[X] = X \cdot f[X] + \gamma$ and $F_0[X] = F[X] - \gamma$. Compute a batch proof π that $F_0(z) = z \cdot y$ and $F_0(0) = 0$, which is obtained by computing $q[X] = \sum_{i=0}^{d-1} q_i \cdot X^i$ such that

$$\frac{F_0[X] - y \cdot X}{X \cdot (X - z)} = q(X)$$

and then

$$\pi = \prod_{i=0}^{d-1} g_i^{q_i} = g^{q(\alpha)}$$

Note that $\pi \in \mathbb{G}$ satisfies

$$e(C \cdot g_1^{-y}, \hat{g}) = e(g, \hat{g})^{\gamma} \cdot e(\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) \tag{32}$$

2. Generate a NIZK proof of knowledge of $(\gamma, \pi) \in \mathbb{Z}_p \times \mathbb{G}$ satisfying (32). Namely,
   a. Choose $r_\gamma \xleftarrow{R} \mathbb{Z}_p$, $R_\pi \xleftarrow{R} \mathbb{G}$ and compute
   $$R = e(g, \hat{g})^{r_\gamma} \cdot e(R_\pi, \hat{g}_2 \cdot \hat{g}_1^{-z})$$
   b. Compute a challenge $c = H(\mathsf{lbl}, C, y, z, R) \in \mathbb{Z}_p$.
   c. Compute the response
   $$s_\gamma = r_\gamma + c \cdot \gamma$$
   $$S_\pi = R_\pi \cdot \pi^c$$

Return the proof

$$\boldsymbol{\pi} = (c, s_\gamma, S_\pi) \in \mathbb{Z}_p^2 \times \mathbb{G}. \tag{33}$$

$\mathsf{Verify}_{\mathsf{srs}}(C, y, z, \boldsymbol{\pi})$: Given a commitment $C \in \mathbb{G}$, an input $z \in \mathbb{Z}_p^*$, a claimed evaluation $y \in \mathbb{Z}_p$, and a candidate proof $\boldsymbol{\pi}$, return 0 if the latter does not parse properly as in (33). Otherwise,

1. Compute
$$R = e(g, \hat{g})^{s_\gamma} \cdot e(S_\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) \cdot e(C \cdot g_1^{-y}, \hat{g})^{-c}. \tag{34}$$
2. Return 1 if $c = H(\mathsf{lbl}, C, y, z, R) \in \mathbb{Z}_p$ and 0 otherwise.

Correctness. The correctness of (34) follows from a standard application of Schnorr-like proofs [50] of homomorphism pre-images. In details the verifier computes

$$\begin{aligned} & e(g, \hat{g})^{s_\gamma} \cdot e(S_\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) \cdot e(C \cdot g_1^{-y}, \hat{g})^{-c} \\ & \quad = e(g, \hat{g})^{r_\gamma + c \cdot \gamma} \cdot e(R_\pi \cdot \pi^c, \hat{g}_2 \cdot \hat{g}_1^{-z}) \cdot \Big(e(g, \hat{g})^{\gamma} \cdot e(\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) \Big)^{-c} \\ & \quad = e(g, \hat{g})^{r_\gamma} \cdot e(R_\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) = R \end{aligned}$$

where the first equality stems from (32).

Efficiency. We note that the prover does not have to compute any pairing as it can implicitly define $R_\pi = g^{t_\pi}$, for a randomly chosen $t_\pi \xleftarrow{R} \mathbb{Z}_p$, and compute

$$R = e(g, \hat{g})^{r_\gamma} \cdot e(g, \hat{g}_2)^{t_\pi} \cdot e(g, \hat{g}_1)^{-z \cdot t_\pi}$$

at step 2.a (note that $e(g, \hat{g}_2)$ and $e(g, \hat{g}_1)$ can be pre-computed) and $S_\pi = g^{t_\pi} \cdot \pi^c$ at step 2.c. The verifier only needs to compute 3 exponentiations (regardless of the degree of the polynomial) and 2 pairings since $e(g, \hat{g})$ can be pre-computed.

Compared to the original randomized KZG commitment of [39, Section 3.3] (which is recalled in Supplementary Material C.3), we can reduce the size of the SRS and the number of exponentiations of the Com and Eval algorithms by a factor 2. Although it does not change the asymptotic complexity, it is significant since the SRS size of Plonk (which is roughly the degree of committed polynomials) is linear in the maximal size of circuits, where the number of gates can be as large as 2^{25} in some applications.

C.2 Security Proofs

The zero-knowledge property holds in the random oracle model and can be proven by relying on the HVZK property of the underlying Σ-protocol. We omit the straightforward proof.

Theorem 10. *The construction provides statistical zero-knowledge in the ROM.*

We now prove that the evaluation protocol of our modified KZG scheme provides simulation-extractability. By adapting ideas from [29], our proof can exploit the features of the algebraic group model to perform straight-line extraction (i.e., without rewinding) in the proof of knowledge. As done in [29] for Schnorr signatures [50], we thus obtain a tighter security proof by avoiding the use of the forking lemma [48].

Theorem 11. *Under the $(d+1, 2)$-DLOG assumption, the scheme provides simulation-extractability in the algebraic group model and in the random oracle model.*

Proof. In the AGM+ROM model, we show that, unless the $(d+1, 2)$-DLOG assumption is false, there exists an extractor that can extract a witness from any adversarially-generated proof $\boldsymbol{\pi}$ and statement (C, z, y). Specifically, we give an algorithm $\mathcal{B}$ that can either extract a witness $(\gamma, \boldsymbol{f}) \in \mathbb{Z}_p \times \mathbb{Z}_p^{(\leq d)}[X]$ with $y = f(z)$ or solve a $(d+1, 2)$-DLOG instance by computing $\alpha \in \mathbb{Z}_p^*$ from

$$\left(g, \{g_i = g^{(\alpha^i)}\}_{i=1}^{d+1}, \hat{g}, \hat{g}_1 = \hat{g}^\alpha, \hat{g}_2 = \hat{g}^{(\alpha^2)}\right).$$

The given problem instance $\{(g, g_1, \ldots, g_{d+1}), \hat{g}, \hat{g}_1, \hat{g}_2\}$ is used to define the CRS pp. Our reduction/extractor $\mathcal{B}$ then interacts with $\mathcal{A}$ as follows.

Queries: When $\mathcal{A}$ makes a random oracle query, $\mathcal{B}$ returns the previously defined value if it exists. Otherwise, it returns a random element in the appropriate

range.

At any time, $\mathcal{A}$ can also choose a commitment $C \in \mathbb{G}$, a label lbl, and a pair $(y, z) \in \mathbb{Z}_p^2$ and ask for a simulated proof that C commits to some polynomial $f \in \mathbb{Z}_p^{(\leq d)}[X]$ such that $f(z) = y$. Then, $\mathcal{B}$ simulates a proof using the standard simulation technique of Fiat-Shamir-like proofs.[7] Namely, it runs the HVZK simulator of the Σ-protocol, which samples $c, s_\gamma, t_0, \ldots, t_{d+1} \xleftarrow{R} \mathbb{Z}_p$, and computes

$$S_\pi = \prod_{i=0}^{d+1} g^{t_i}$$

$$R = e(g, \hat{g})^{s_\gamma} \cdot e(S_\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) \cdot \left(\frac{e(C, \hat{g})}{e(g_1, \hat{g})^y} \right)^{-c}$$

before programming $c = H(\mathsf{lbl}, C, y, z, R)$. If H was already defined for the input $(\mathsf{lbl}, C, y, z, R)$, then $\mathcal{B}$ aborts. However, since R is uniformly distributed over $\mathbb{G}_T$, this can only happen with probability $(Q_H + Q_S)/p$ if Q_H (resp. Q_S) denotes the number of random oracle (resp. simulation) queries. If the simulator does not fail, the proof $\boldsymbol{\pi} = (c, s_\gamma, S_\pi)$ has the same distribution as a proof that would be generated using the real witnesses (note that valid witnesses always exist since $\mathbb{G}$ is cyclic and g is a generator). Consequently, the simulation is perfect, unless a collision occurs on random oracles in one of the simulation queries. If Q_S (reps. Q_H) denotes the number of queries made by $\mathcal{A}$ to the simulator (resp. to the random oracle), this happens with probability at most $(Q_S + Q_H) \cdot Q_H / p$.

Importantly, when $\mathcal{A}$ queries a hash value $H(\mathsf{lbl}, C, y, z, R)$, it must provide an algebraic representation $\{w_i\}_{i=0}^{d+1}$ of C as $C = \prod_{i=0}^{d+1} g_i^{w_i}$ as well as an algebraic representation $\{(\alpha_j, \beta_j, \delta_j)\}_{j=0}^{d+1}$ of R as

$$R = \prod_{j=0}^{d+1} e(g_j, \hat{g})^{\alpha_j} \cdot \prod_{j=0}^{d+1} e(g_j, \hat{g}_1)^{\beta_j} \cdot \prod_{j=0}^{d+1} e(g_j, \hat{g}_2)^{\delta_j} \tag{35}$$

where we define $g_0 = g$. We note that, although the representation provided by $\mathcal{A}$ can also depend on factors $\{e(S_\pi^{(i)}, \hat{g}_b)\}_{b \in [0,2]}$ for which $S_\pi^{(i)}$ were part of earlier simulated proofs, the simulator always computes $S_\pi^{(i)}$ by sampling its algebraic representation w.r.t. $(g, g_1, \ldots, g_{d+1})$, thus allowing $\mathcal{B}$ to infer a representation that only depends on $\{e(g_i, \hat{g}_b)\}_{i \in [0,d+1], b \in [0,2]}$.

Output: When $\mathcal{A}$ halts, it outputs a statement $(C, y, z) \in \mathbb{G} \times \mathbb{Z}_p^2$, a label lbl, together with a proof $\boldsymbol{\pi} = \big(c, s_\gamma, S_\pi\big) \in \mathbb{Z}_p^2 \times \mathbb{G}$ and representations $\{w_i\}_{i=0}^{d+1}$, $\{\psi_i\}_{i=0}^{d+1}$ such that $C = \prod_{i=0}^{d+1} g_i^{w_i}$, $S_\pi = \prod_{i=0}^{d+1} g_i^{\psi_i}$. If $\sum_{i=0}^{d} w_{i+1} \cdot z^i = y$, then $\mathcal{B}$ is done since it can simply output a valid witness $(f[X], \gamma)$, where $f[X] = \sum_{i=0}^{d} w_{i+1} \cdot X^i$ and $\gamma = w_0$. From here on, we thus assume that $y \neq \sum_{i=0}^{d} w_i \cdot z^i$.

[7] In the AGM, $\mathcal{A}$ must also provide an algebraic representation of C w.r.t. previously observed elements of $\mathbb{G}$. However, we do not use it for the simulation.

Since $\boldsymbol{\pi}$ is a valid proof, we must have $c = H(\mathsf{lbl}, C, y, z, R)$, where

$$R = e(g, \hat{g})^{s_\gamma} \cdot e(S_\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) \cdot \left(\frac{e(C, \hat{g})}{e(g_1, \hat{g})^y} \right)^{-c} \tag{36}$$

If $\mathcal{A}$ did not query H on $(\mathsf{lbl}, C, y, z, R)$, then $\mathcal{B}$ fails. However, in this case, verification would only succeed with probability $1/p$. We henceforth assume that $\mathcal{A}$ queried H on the input (C, y, z, R), where R is obtained as per (36).

We now distinguish two cases: (i) $(\mathsf{lbl}, C, y, z, c)$ was recycled from simulation query for which the output was (c, s'_γ, S'_π); (ii) $(\mathsf{lbl}, C, y, z, c)$ is a fresh tuple.

In case (i), $\mathcal{A}$'s winning conditions imply $(s'_\gamma, S'_\pi) \neq (s_\gamma, S_\pi)$. So, unless a collision occurs on H (which occurs with negligible probability $(Q_H + Q_S)^2/p$), we must have

$$\begin{aligned} R &= e(g, \hat{g})^{s'_\gamma} \cdot e(S'_\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) \cdot \left(\frac{e(C, \hat{g})}{e(g_1, \hat{g})^y} \right)^{-c} \\ &= e(g, \hat{g})^{s_\gamma} \cdot e(S_\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) \cdot \left(\frac{e(C, \hat{g})}{e(g_1, \hat{g})^y} \right)^{-c} \end{aligned}$$

so that

$$e(g, \hat{g})^{s_\gamma - s'_\gamma} \cdot e\big(S_\pi / S'_\pi,\ \hat{g}_2 \cdot \hat{g}_1^{-z}\big) = 1_{\mathbb{G}_T}$$

Since $\mathcal{B}$ knows a representation $(t_0, \ldots, t_{d+1})$ of $S'_\pi = \prod_{i=0}^{d+1} g_i^{t_i}$ (which was chosen in a simulation query) and the representation $\{\psi_i\}_{i=0}^{d+1}$ of $S_\pi = \prod_{i=0}^{d+1} g_i^{\psi_i}$ (which was revealed by $\mathcal{A}$), it obtains a non-zero polynomial

$$Q[X] = (s_\gamma - s'_\gamma) + \big(X^2 - z \cdot X\big) \cdot \Big(\sum_{i=0}^{d+1} (\psi_i - t_i) \cdot X^i \Big)$$

of which $\alpha \in \mathbb{Z}_p$ is a root.

In case (ii), $\mathcal{B}$ recalls the algebraic representation of R (35) that must have been supplied by $\mathcal{A}$ when the hash query $H(C, y, z, R)$ was made. From the corresponding $\{(\alpha_j, \beta_j, \delta_j)\}_{j=0}^{d+1}$, $\mathcal{B}$ can compute a triple $(\bar{c}, \bar{s}_\gamma, \bar{S}_\pi) \in \mathbb{Z}_p^2 \times \mathbb{G}$ such that

$$R = e(g, \hat{g})^{\bar{s}_\gamma} \cdot e(\bar{S}_\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) \cdot e(C \cdot g_1^{-y}, \hat{g})^{-\bar{c}} \tag{37}$$

This triple can be obtained via a representation $\{\theta_j\}_{j=0}^{d+1}$ of $\bar{S}_\pi = \prod_{j=0}^{d+1} g_j^{\theta_j}$ w.r.t. the generators $(g, g_1, \ldots, g_{d+1})$. Such a representation must satisfy

$$\begin{aligned}
R &= e(g,\hat{g})^{\bar{s}_\gamma} \cdot e\big(\prod_{j=0}^{d+1} g_j^{\theta_j}, \hat{g}_2 \cdot \hat{g}_1^{-z}\big) \cdot e\big(\prod_{j=0}^{d+1} g_j^{w_j} \cdot g_1^{-y}, \hat{g}\big)^{-\bar{c}} \\
&= e(g,\hat{g})^{\bar{s}_\gamma} \cdot e\big(\prod_{j=0}^{d+1} g_j^{\theta_j}, \hat{g}_2\big) \cdot e\big(\prod_{j=0}^{d+1} g_j^{-z\cdot\theta_j}, \hat{g}_1\big) \cdot e\big(\prod_{j=0}^{d+1} g_j^{w_j} \cdot g_1^{-y}, \hat{g}\big)^{-\bar{c}} \qquad (38) \\
&= e(g,\hat{g})^{\bar{s}_\gamma} \cdot e\big(g_1^{-z\theta_0} \cdot \prod_{j=2}^{d+1} g_j^{-z\cdot\theta_{j-1}+\theta_{j-2}}, \hat{g}\big) \\
&\qquad \cdot e(g_{d+1},\hat{g}_1)^{-z\cdot\theta_{d+1}+\theta_d} \cdot e(g_{d+1},\hat{g}_2)^{\theta_{d+1}} \cdot e\big(\prod_{j=0}^{d+1} g_j^{w_j} \cdot g_1^{-y}, \hat{g}\big)^{-\bar{c}} \\
&= e(g,\hat{g})^{\bar{s}_\gamma - c\cdot w_0} \cdot e\big(g_1^{-z\cdot\theta_0 - \bar{c}\cdot(w_1-y)} \cdot \prod_{j=2}^{d+1} g_j^{-z\cdot\theta_{j-1}+\theta_{j-2}-\bar{c}\cdot w_j}, \hat{g}\big) \\
&\qquad \cdot e(g_{d+1},\hat{g}_1)^{-z\cdot\theta_{d+1}+\theta_d} \cdot e(g_{d+1},\hat{g}_2)^{\theta_{d+1}}
\end{aligned}$$

while (35) can be written

$$\begin{aligned}
R &= e\big(\prod_{i=0}^{d+1} g_i^{\alpha_i}, \hat{g}\big) \cdot e\big(\prod_{i=0}^{d+1} g_i^{\beta_i}, \hat{g}_1\big) \cdot e\big(\prod_{i=0}^{d+1} g_i^{\delta_i}, \hat{g}_2\big) \\
&= e\big(g^{\alpha_0} \cdot g_1^{\alpha_1+\beta_0} \cdot \prod_{i=2}^{d+1} g_i^{\alpha_i+\beta_{i-1}+\delta_{i-2}}, \hat{g}\big) \cdot e(g_{d+1},\hat{g}_1)^{\beta_{d+1}+\delta_d} \cdot e(g_{d+1},\hat{g}_2)^{\delta_{d+1}}
\end{aligned} \qquad (39)$$

By identifying the representations of R in (38) and (39), we see that $\mathcal{B}$ can obtain exactly one representation $(\theta_0, \ldots, \theta_{d+1}) \in \mathbb{Z}_p^{d+2}$ of S_π and the corresponding $(s_\gamma, c) \in \mathbb{Z}_p^2$ by solving the linear system

$$\begin{bmatrix}
1 & -w_0 & & & & & \\
 & y-w_1 & -z & & & & \\
 & -w_2 & 1 & -z & & & \\
 & -w_3 & & 1 & -z & & \\
 & \vdots & & & \ddots & \ddots & \\
 & -w_{d+1} & & & & 1 & -z \\
 & & & & & & 1
\end{bmatrix} \cdot \begin{bmatrix} \bar{s}_\gamma \\ \bar{c} \\ \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \\ \theta_{d+1} \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \alpha_1+\beta_0 \\ \alpha_2+\beta_1+\delta_0 \\ \vdots \\ \alpha_{d+1}+\beta_d+\delta_{d-1} \\ \beta_{d+1}+\delta_d \\ \delta_{d+1} \end{bmatrix},$$

which is possible since the determinant is $y - \sum_{i=0}^{d} w_{i+1} \cdot z^i \neq 0$. This allows $\mathcal{B}$ to compute $(\bar{c}, \bar{s}_\gamma, \bar{S}_\pi) \in \mathbb{Z}_p^2 \times \mathbb{G}$ such that

$$e(g, \hat{g})^{\bar{s}_\gamma} \cdot e(\bar{S}_\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) \cdot \left(\frac{e(C, \hat{g})}{e(g_1, \hat{g})^y} \right)^{-\bar{c}} = e(g, \hat{g})^{s_\gamma} \cdot e(S_\pi, \hat{g}_2 \cdot \hat{g}_1^{-z}) \cdot \left(\frac{e(C, \hat{g})}{e(g_1, \hat{g})^y} \right)^{-c}. \quad (40)$$

Moreover, with overwhelming probability $1 - 1/p$, we have $\bar{c} \neq c$ since $c = H(\mathsf{lbl}, C, y, z, R)$ is uniformly distributed over $\mathbb{Z}_p$ and chosen after R, which uniquely determines $\bar{c}$ if $y \neq \sum_{i=0}^{d} w_{i+1} \cdot z^i$. This implies that $\gamma \triangleq (s_\gamma - \bar{s}_\gamma)/(\bar{c} - c)$ and $\pi \triangleq (S_\pi \cdot \bar{S}_\pi^{-1})^{1/(\bar{c}-c)}$ are valid witnesses for the evaluation relation (32).

Now, we observe that $\mathcal{B}$ can compute a representation $\{\phi_i\}_{i=0}^{d+1}$ of the extracted $\pi = \prod_{i=0}^{d+1} g_i^{\phi_i}$ (recall that $\mathcal{A}$ revealed $\{\psi_i\}_{i=0}^{d+1}$ such that $S_\pi = \prod_{i=0}^{d+1} g_i^{\psi_i}$ and $\mathcal{B}$ previously computed $\{\theta_i\}_{i=1}^{d+1}$ such that $\bar{S}_\pi = \prod_{i=0}^{d+1} g_i^{\theta_i}$). Since $\mathcal{A}$ also revealed $\{w_i\}_{i=0}^{d+1}$ such that $C = \prod_{i=0}^{d+1} g_i^{w_i}$, we observe from (32) that $\alpha \in \mathbb{Z}_p$ is a root of

$$\begin{aligned} P[X] &= -(\sum_{i=0}^{d+1} w_i \cdot X^i) + y \cdot X + \gamma + (\sum_{i=0}^{d+1} \phi_i \cdot X^i) \cdot (X^2 - z \cdot X) \\ &= (\gamma - w_0) + (-w_1 - z \cdot \phi_0 + y) \cdot X + \sum_{i=2}^{d+1} (-w_i + \phi_{i-2} - z \cdot \phi_{i-1}) \cdot X^i \\ &\quad + (\phi_d - z \cdot \phi_{d+1}) \cdot X^{d+2} + \phi_{d+1} \cdot X^{d+3} \end{aligned}$$

so that $\mathcal{B}$ can compute $\alpha \in \mathbb{Z}_p$ by factoring $P[X]$ as long as it is a non-zero polynomial. We claim that, if $y \neq \sum_{i=0}^{d} w_{i+1} \cdot z^i$, then $P[X]$ is indeed non-zero. Indeed, if $P[X]$ was identically zero, then we would have $\gamma = w_0$ and $(\phi_0, \phi_1, \ldots, \phi_{d+1})$ would satisfy the overdetermined linear system

$$\underbrace{\begin{bmatrix} -z & & & & \\ 1 & -z & & & \\ & 1 & -z & & \\ & & \ddots & \ddots & \\ & & & 1 & -z \\ & & & & 1 \end{bmatrix}}_{\in \mathbb{Z}_p^{(d+3) \times (d+2)}} \cdot \begin{bmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{d+1} \end{bmatrix} = \begin{bmatrix} w_1 - y \\ w_2 \\ \vdots \\ w_d \\ w_{d+1} \\ 0 \\ 0 \end{bmatrix}, \quad (41)$$

which is impossible if $y \neq \sum_{i=0}^{d} w_{i+1} \cdot z^i$ since $(1, z, z^2, \ldots, z^{d+1}, z^{d+2})$ is in the left-kernel of the matrix in (41). □

C.3 Comparison With Other Randomized KZG Commitments

As pointed out by Kohlweiss *et al.* [41], randomized versions of KZG commitments are necessary to compile certain polynomial IOPs into simulation-

extractable SNARKs. They suggested to use the perfectly hiding KZG commitment of Kate *et al.* [39] in order to obtain a simulation-extractable variant of Plonk using their framework.

The Original Randomized KZG. In the original randomized KZG commitment [39, Section 3.3], the structured reference string contains group elements $(\{g_i = g^{(\alpha^i)}\}_{i=0}^{d}, \{\hat{g}_i = \hat{g}^{(\alpha^i)}\}_{i=0,1})$ and $(\{h_i = h^{(\alpha^i)}\}_{i=0}^{d}, \{\hat{h}_i = \hat{h}^{(\alpha^i)}\}_{i=0,1})$, where $g, h \in \mathbb{G}$ and $\hat{g}, \hat{h} \in \hat{\mathbb{G}}$ are generators. A perfectly hiding commitment to a univariate $f[X] = \sum_{i=0}^{d} f_i \cdot X^i$ is obtained by choosing a random degree-d polynomial $t[X] = \sum_{i=0}^{d} t_i \cdot X^i$ and computing

$$C = g^{f(\alpha)} \cdot h^{t(\alpha)} = \prod_{i=0}^{d} g_i^{f_i} \cdot \prod_{i=0}^{d} h_i^{t_i}$$

In order to prove that the committed $f[X]$ satisfies $y = f(z)$ for a given input $z \in \mathbb{Z}_p$, the committer can compute the degree-$(d-1)$ polynomials $q_f[X] = (f(X) - y)/(X - z)$, $q_t[X] = (t[X] - t(z))/(X - z)$, which can be used to compute

$$\pi = g^{q_f(\alpha)} \cdot h^{q_t(\alpha)}$$

from the group elements $\{g^{(\alpha^i)}\}_{i=0}^{d+1}$, $\{h^{(\alpha^i)}\}_{i=0}^{d+1}$ contained in the CRS. The evaluation proof consists of a pair $(\pi, t(z)) \in \mathbb{G} \times \mathbb{Z}_p$ and is verified by testing the equality

$$e\big(C \cdot g^{-y} \cdot h^{-t(z)}, \hat{g}\big) = e(\pi, \hat{g}_1 \cdot \hat{g}^{-z}). \tag{42}$$

This construction is perfectly hiding and still provides knowledge-soundness under the $(d, 1)$-DLOG assumption in the algebraic group model. It can also be adapted to prove degree bounds as described in [42]. Unfortunately, it remains malleable. For example, an adversary can observe a commitment C to $f[X]$ and, without knowing anything about f, it can compute $C \cdot g^{\omega}$ for an arbitrary $\omega \in \mathbb{Z}_p$ of its choice. Then, a proof $(\pi, t(z))$ that $y = f(z)$ is also a proof that $f[X] + \omega$ evaluates to $y + \omega$ for the input z.

Despite its malleability, Kohlweiss *et al.* [41] showed that the above variant of KZG commitments can still be used[8] in order to compile Plonk [30] into a simulation-extractable SNARK using their framework. However, this comes at the cost of doubling the size of the common reference string (which is already very large since the parameter d is the maximal size of proven arithmetic circuits in Plonk). Our construction of Section C.1 can achieve the same result without increasing the SRS size nor the number of exponentiations at the committer/prover.

In [19], it was pointed out that the degree of $t[X]$ only needs to be as large as the number of polynomial evaluations obtained by the verifier, which is small for

[8] The reason is that it provides "quasi-unique proofs", which means that it is computationally hard to find two distinct proofs for the same evaluation $y = f(x)$ and the same commitment C.

several SNARKs, including Marlin [19]. Still, the number of exponentiations and the SRS size both depend on the maximal number of polynomial evaluations, which is specific to the compiled PIOP. When the number of proven evaluations of any given commitment is not known ahead of time, the batch evaluation protocol of [12, Section 4.1] can be used to prove multiple evaluations at once by only revealing one evaluation of a linear combination of the individual polynomials. However, this require to prove all these evaluations at once.

In contrast, our scheme does not require to increase the SRS size based on a pre-determined maximal number of polynomial evaluations. The SRS can thus be set up without knowing in advance which specific PIOP will be used while supporting proofs for individual evaluations. Only the maximal circuit size should be known ahead of time. As another advantage, its evaluation protocol is zero-knowledge and immediately provides a simple trapdoor-less simulator.

The BDFG Variant. Boneh, Drake, Fisch and Gabizon [12, Appendix B] described a generic compiler that turns any (not necessarily hiding) homomorphic PCS scheme into a hiding PCS with a zero-knowledge evaluation protocol. Their compiler proceeds in two steps. In a first step, it builds a hiding PCS (where the evaluation protocol is not zero-knowledge) from an additively homomorphic one. In a second step, it turns a hiding PCS into a PCS scheme where the evaluation protocol is zero-knowledge.

The BDFG compiler is designed to work for any additively homomorphic PCS. When it is applied to deterministic KZG commitments, it intuitively proceeds as follows.

The prover commits to $f[X] = \sum_{i=0}^{d+1} f_i \cdot X^i$ by choosing a random $r \xleftarrow{R} \mathbb{Z}_p$ and computing

$$C = g^r \cdot \prod_{j=1}^{d+1} g_j^{f_{j-1}}$$

as a deterministic commitment to $F[X] = r + X \cdot f[X]$.

In the evaluation protocol, the prover first computes a commitment

$$C_\alpha = g^{\alpha_r} \cdot \prod_{j=1}^{d+1} g_j^{\alpha_{j-1}}$$

to a random polynomial $\alpha[X]$ with the randomness $\alpha_r \xleftarrow{R} \mathbb{Z}_p$ and sends both $y_\alpha = \alpha(z)$ and C_α to the verifier. The verifier then homomorphically computes $C_s = C_\alpha \cdot C^c$, which is a randomized KZG commitment to $s[X] = \alpha[X] + c \cdot f[X]$. Next, the prover and the verifier run a basic (non-zero-knowledge) evaluation protocol showing that $s[X]$ evaluates to $s(z) = y_\alpha + c \cdot y$.[9] The basic evaluation protocol for $s[X]$ proceeds by having the prover send a non-hiding KZG commitment $\bar{C}_s = \prod_{j=0}^{d} g_j^{s_j}$ to $s[X]$ and prove that the committed polynomial

[9] This basic evaluation protocol does not have to be ZK since it is applied to $s[X]$, where a one-time masking polynomial $\alpha[X]$ hides $f[X]$.

evaluates to $y_\alpha + c \cdot y$. However, the prover has to convince the verifier that $\bar{C}_s$ is consistent with the homomorphically evaluated $C_s = g^{s_r} \cdot \prod_{j=1}^{d} g_j^{s_{j-1}}$. To this end, it sends $s_r = \alpha_r + c \cdot r$ of $C_s = g^{s_r}$ and proves that $\bar{C}_s$ and $C'_s \triangleq C_s \cdot g^{-s_r}$ are deterministic KZG commitments to $s[X]$ and $X \cdot s[X]$, respectively. This last step is achieved by showing that they evaluate to $s(\rho)$ and $\rho \cdot s(\rho)$ for a random $\rho \in \mathbb{Z}_p$. The Schwartz-Zippel lemma ensures that this happens with negligible probability if $\bar{C}_s$ is not consistent with C'_s.

It is possible to optimize the protocol use the batched evaluation protocol of [12, Appendix C.2] which allows proving multiple evaluations for multiple polynomials using a single group element. This batched evaluation protocol can be used to simultaneously prove 3 evaluations for the polynomials $\bar{s}[X]$ and $s[X] - s_r$ and the evaluation points (z, ρ). Still, even in this space-optimized variant, each proof consists of 3 elements of $\mathbb{G}$ and 4 elements of $\mathbb{Z}_p$.

The Kohrita-Towa/Zhang *et al.* Variant. In [42, Section 3.5.3], Kohrita and Towa considered a randomized variant of KZG which is obtained by restricting the scheme of Zhang *et al.* [56] to univariate polynomials. As in our scheme, the committer's randomness consists of a constant number of field elements. Their construction has randomizable proofs and is thus not simulation-extractable. Their scheme has slightly shorter proofs (made of 2 elements of $\mathbb{G}$) than our univariate PCS, but requires 3 pairing evaluations to verify. In the scheme of Section C.1, evaluation proofs only cost 2 pairings to verify and consist of one element of $\mathbb{G}$ and 2 scalars in $\mathbb{Z}_p$.

D Deferred Proofs for the Instantiation of HyperPlonk

D.1 Proof of Theorem 4

Proof. We describe an algorithm $\mathcal{B}$ that interacts with a possibly cheating prover $\mathcal{A}$ and either extracts a witness (using the standard extractor enabled by the AGM) or breaks the knowledge-soundness of the underlying (univariate or multivariate) polynomial commitment schemes in case knowledge extraction fails.

At the outset of the game, $\mathcal{B}$ prepares the SRS as in the PCS scheme of Section 4.

When the adversary $\mathcal{A}$ halts, it outputs a proof $\boldsymbol{\pi} = \left(\hat{C}_M, \hat{C}_v, \boldsymbol{\pi}_{\mathsf{zero}}\right)$, where

$$\boldsymbol{\pi}_{\mathsf{zero}} = \big(C_a, a_{\boldsymbol{r}}, y_a, \{m_{i,\boldsymbol{r}}, \sigma_{i,\boldsymbol{r}}\}_{i=0}^{2}, \{s_{i,\boldsymbol{r}}\}_{i=1}^{3}, \{v_{\boldsymbol{r},b}, v_{b,\boldsymbol{r}}\}_{b\in\{0,1\}}, \{C_{\theta,i}, \bar{\theta}_{i,0}, \bar{\theta}_{i,1}\}_{i\in[\mu]}, \boldsymbol{\pi}_{\mathsf{batch}}, \boldsymbol{\pi}_{\boldsymbol{r}}\big)$$

for a statement $(\mathbb{i}, \mathbb{x})$ of its choice, where $\mathbb{i} = \mathcal{C}$ uniquely determines circuit-dependent verifier parameters vp.[10] Since $\mathcal{A}$ is algebraic, it also outputs repre-

[10] We note that vp is generated by $\mathcal{B}$ since, in the knowledge-soundness experiment, it is computed by the verifier in the pre-processing phase. So, we may assume that $\mathcal{B}$ knows an opening of all commitments contained in vp.

sentations of $\{\hat{C}_{M,i}\}_{i=0}^{2}$ and $\hat{C}_v$ w.r.t. the group elements

$$\left(\hat{g}_r, \{\hat{g}_{\mathcal{I}}\}_{\mathcal{I}\in\mathcal{W}_{1,\ell+1}}, \left\{\hat{g}^{(\alpha_1+\alpha_{\mu+1})^i\cdot\alpha_\mu\cdot(\alpha_\mu-1)}\right\}_{i=0}^{3}\right)$$

as well as a representation of C_a w.r.t. $\{g_{\mathcal{I}}\}_{\mathcal{I}\in\mathcal{W}_{\bar{d},\ell}}$. From these algebraic representations, the knowledge extractor $\mathcal{B}$ can compute the $(\ell+1)$-variate polynomials

$$F_{M,i}[X_1,\ldots,X_\ell,X_r] = \underbrace{M_i[X_1,\ldots,X_\ell] + R_{M,i}[X_1+X_{\mu+1}]\cdot X_\mu\cdot(X_\mu-1)}_{\triangleq\ \bar{M}_i[X_1,\ldots,X_\ell]} + r_{M,i}\cdot X_r$$

$$F_v[X_1,\ldots,X_\ell,X_r] = \underbrace{V[X_1,\ldots,X_\ell] + R_v[X_1+X_{\mu+1}]\cdot X_\mu\cdot(X_\mu-1)}_{\triangleq\ \bar{V}[X_1,\ldots,X_\ell]} + r_v\cdot X_r \tag{43}$$

such that

$$\hat{C}_{M,i} = \hat{g}^{F_{M,i}(\alpha_1,\ldots,\alpha_\ell,\alpha_r)} \qquad \forall i\in[0,2],$$
$$\hat{C}_v = \hat{g}^{F_v(\alpha_1,\ldots,\alpha_\ell,\alpha_r)},$$

for ℓ-variate $M_0, M_1, M_2, V \in \mathbb{Z}_p^{(\leq 1)}[X_1,\ldots,X_\ell]$ and univariate $R_{M,0}, R_{M,1}$, $R_{M,2}, R_v \in \mathbb{Z}_p^{(\leq 3)}[X]$ (note that the degree bounds are enforced by the highest powers of each α_i that are available in the exponent in srs). Let also

$$A[X_1,\ldots,X_\ell,X_r] = a[X_1,\ldots,X_\ell] + r_a\cdot X_r$$

such that $C_a = g^{a(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$ and $a[X_1,\ldots,X_\ell]\in\mathbb{Z}_p^{(\leq\bar{d})}[X_1,\ldots,X_\ell]$. Let also the polynomials $\{\Theta_i[X_1,\ldots,X_\ell,X_r]\}_{i=1}^{\mu}$ that $\mathcal{A}$ also reveals via the algebraic representation of $C_{\theta,i}$ which satisfy $C_{\theta,i} = g^{\Theta_i(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$.

Unless $\mathcal{A}$ was able to break the knowledge-soundness of the batch evaluation proofs of the univariate PCS, the following equalities hold

$$\Theta_i(0,0,\ldots,0,0) = \bar{\theta}_{i,0} \qquad \Theta_i(1,0,\ldots,0,0) = \bar{\theta}_{i,1} \qquad \forall i\in[\mu],$$

$$\Theta_i(r_i,0,\ldots,0,0) = \bar{\theta}_{i-1,0} + \bar{\theta}_{i-1,1} \qquad \forall i\in[2,\mu],$$

where r_i is computed as in (19). Moreover, unless $\mathcal{A}$ was able to break the knowledge-soundness of the multivariate PCS,[11] we have

$$\bar{M}_i(r_1,\ldots,r_\mu,x_\ell) = m_{i,\boldsymbol{r}} \qquad \forall i\in[0,2],\ \forall x_\ell\in\mathbb{Z}_p.$$

and

$$s_\sigma(r_1,\ldots,r_\mu,\langle i\rangle) = \sigma_{i,\boldsymbol{r}} \qquad S_i(r_1,\ldots,r_\mu) = s_{i,\boldsymbol{r}}.$$

[11] The reduction from the security of univariate and multivariate PCS is straightforward since they use the same SRS structure and their security proof simulates them in the same way.

Since $(r_1, \ldots, r_\mu)$ was a random input, the polynomials $\{\bar{M}_i\}_{i=0}^2$ must be μ-variate (i.e., independent of the variable X_ℓ) with overwhelming probability by the same argument as in the proof of Theorem 1.

Under the assumption that the multivariate PCS is knowledge-sound, we also have

$$\begin{aligned} \bar{V}(r_1, \ldots, r_\mu, b) &= v_{\boldsymbol{r},b} && \forall b \in \{0,1\} \\ \bar{V}(b, r_1, \ldots, r_\mu) &= v_{b,\boldsymbol{r}} && \forall b \in \{0,1\} \\ \bar{V}(1, \ldots, 1, 0) &= 1, \end{aligned} \tag{44}$$

and

$$a[r_1, \ldots, r_\mu, \boldsymbol{x}] = a_{\boldsymbol{r}} \qquad \forall \boldsymbol{x} \in \mathbb{Z}_p. \tag{45}$$

Note that (45) implies $a[r_1, \ldots, r_\mu, X_{\mu+1}] - a[r_1, \ldots, r_\mu, 0]$ for a random input $\boldsymbol{r} = (r_1, \ldots, r_\mu)$. By Schwartz-Zippel, this only happens with negligible probability if $a[X_1, \ldots, X_\mu, X_{\mu+1}] - a[X_1, \ldots, X_\mu, 0]$ is not identically zero. Then, (45) implies that $a[X_1, \ldots, X_\ell]$ only depends on $(X_1, \ldots, X_\mu)$ with overwhelming probability.

Let the polynomial $\tilde{v}[X_1, \ldots, X_{\mu+1}] = V[X_1, \ldots, X_{\mu+1}]$ defined in (43), which is multilinear and agrees with $\bar{V}[X_1, \ldots, X_{\mu+1}]$ on $B_\mu \times \mathbb{Z}_p$. For the variables $\boldsymbol{X} = (X_1, \ldots, X_\mu)$, if we now define the μ-variate polynomials

$$\begin{aligned} Q_1[\boldsymbol{X}] &= \bar{v}[1, \boldsymbol{X}] - \bar{v}[\boldsymbol{X}, 0] \cdot \bar{v}[\boldsymbol{X}, 1] \\ Q_2[\boldsymbol{X}] &= \prod_{i=0}^{2} \big(\bar{M}_i[\boldsymbol{X}] + \beta_2 \cdot s_{\mathrm{id}}[\boldsymbol{X}, \langle i \rangle] + \beta_1\big) \\ &\qquad - \bar{v}[0, \boldsymbol{X}] \cdot \prod_{i=0}^{2} \big(\bar{M}_i[\boldsymbol{X}] + \beta_2 \cdot s_{\sigma}[\boldsymbol{X}, \langle i \rangle] + \beta_1\big) \end{aligned} \tag{46}$$

$$\begin{aligned} f[\boldsymbol{X}] &= S_1[\boldsymbol{X}] \cdot \big(\bar{M}_0[\boldsymbol{X}] + \bar{M}_1[\boldsymbol{X}]\big) \\ &\quad + S_2[\boldsymbol{X}] \cdot \big(\bar{M}_0[\boldsymbol{X}] \cdot \bar{M}_1[\boldsymbol{X}]\big) \\ &\quad + S_3[\boldsymbol{X}] \cdot \boldsymbol{G}\big(\bar{M}_0[\boldsymbol{X}], \bar{M}_1[\boldsymbol{X}]\big) - \bar{M}_2[\boldsymbol{X}] + I[\boldsymbol{X}], \end{aligned} \tag{47}$$

and

$$F[\boldsymbol{X}] = f[\boldsymbol{X}] + \zeta_1 \cdot Q_1[\boldsymbol{X}] + \zeta_2 \cdot Q_2[\boldsymbol{X}]$$

the last verification check ensures that

$$\Theta_1(r_1, 0, \ldots, 0) = F(r_1, \ldots, r_\mu) \cdot eq_{(\gamma_1, \ldots, \gamma_\mu)}(r_1, \ldots, r_\mu) + \xi \cdot a(r_1, \ldots, r_\mu) \tag{48}$$

Since r_1 is chosen uniformly after $(r_j)_{j=2}^{\mu}$, (48) implies

$$\Theta_1[X, 0, \ldots, 0] = F[X, r_2, \ldots, r_\mu] \cdot eq_{(\gamma_1, \ldots, \gamma_\mu)}[X, r_2, \ldots, r_\mu] + \xi \cdot a[X, r_2, \ldots, r_\mu]$$

with overwhelming probability. By inductively applying the Schwartz-Zippel lemma, the soundness analysis of the sumcheck protocol then yields

$$\sum_{\boldsymbol{x} \in B_\mu} \big(F(\boldsymbol{x}) \cdot eq_{\boldsymbol{\gamma}}(\boldsymbol{x}) + \xi \cdot a(\boldsymbol{x})\big) = \xi \cdot y_a \tag{49}$$

with overwhelming probability $\geq 1 - \mu \cdot d/p$ over the random choice of $(r_i)_{i\in[\mu]}$. In details, since r_2 was chosen uniformly after $(r_i)_{i\geq 3}$, the condition

$$\begin{aligned}\Theta_2(r_2, 0, \ldots, 0) = \theta_2 \triangleq \bar{\theta}_{1,0} + \bar{\theta}_{1,1} &= \Theta_1(0, 0, \ldots, 0) + \Theta_1(1, 0, \ldots, 0) \\ &= F[0, r_2, \ldots, r_\mu] \cdot eq_{\boldsymbol{\gamma}}(0, r_2, \ldots, r_\mu) + \xi \cdot a(0, r_2, \ldots, r_\mu) \\ &\quad + F[1, r_2, \ldots, r_\mu] \cdot eq_{\boldsymbol{\gamma}}(1, r_2, \ldots, r_\mu) + \xi \cdot a(1, r_2, \ldots, r_\mu)\end{aligned}$$

implies

$$\begin{aligned}\Theta_2[X, 0, \ldots, 0] = F[0, X, r_3, \ldots, r_\mu] \cdot eq_{\boldsymbol{\gamma}}[0, X, r_3, \ldots, r_\mu] + \xi \cdot a[0, X, r_3, \ldots, r_\mu] \\ + F[1, X, r_3, \ldots, r_\mu] \cdot eq_{\boldsymbol{\gamma}}[1, X, r_3, \ldots, r_\mu] + \xi \cdot a[1, X, r_3, \ldots, r_\mu]\end{aligned}$$

except with probability d/p over the choice of r_2. By induction, since each r_i is chosen uniformly after $(r_j)_{j>i}$, the equality

$$\begin{aligned}\Theta_i(r_i, 0, \ldots, 0) = \theta_i \triangleq \bar{\theta}_{i-1,0} + \bar{\theta}_{i-1,1} &= \theta_{i-1}(0, 0, \ldots, 0) + \theta_{i-1}(1, 0, \ldots, 0) \\ = \sum_{\boldsymbol{b}\in B_{i-2}} &\big(F(\boldsymbol{b}, 0, r_i, \ldots, r_\mu) \cdot eq_{\boldsymbol{\gamma}}(\boldsymbol{b}, 0, r_i, \ldots, r_\mu) + \xi \cdot a(\boldsymbol{b}, 0, r_i, \ldots, r_\mu)\big) \\ + \sum_{\boldsymbol{b}\in B_{i-2}} &\big(F(\boldsymbol{b}, 1, r_i, \ldots, r_\mu) \cdot eq_{\boldsymbol{\gamma}}(\boldsymbol{b}, 1, r_i, \ldots, r_\mu) + \xi \cdot a(\boldsymbol{b}, 1, r_i, \ldots, r_\mu)\big) \\ = \sum_{\boldsymbol{b}\in B_{i-1}} &\big(F(\boldsymbol{b}, r_i, \ldots, r_\mu) \cdot eq_{\boldsymbol{\gamma}}(\boldsymbol{b}, r_i, \ldots, r_\mu) + \xi \cdot a(\boldsymbol{b}, r_i, \ldots, r_\mu)\big) \qquad (50)\end{aligned}$$

implies w.h.p. the polynomial identity

$$\begin{aligned}&\Theta_i[X, 0, \ldots, 0] \\ &= \sum_{\boldsymbol{b}\in B_{i-1}} \big(F[\boldsymbol{b}, X, r_{i+1}, \ldots, r_\mu] \cdot eq_{\boldsymbol{\gamma}}[\boldsymbol{b}, X, r_{i+1}, \ldots, r_\mu] + \xi \cdot a[\boldsymbol{b}, X, r_{i+1}, \ldots, r_\mu]\big) \qquad (51)\end{aligned}$$

Indeed, if we call E_i the event that (50) holds when (51) does not, we have $\Pr[E_i] \leq d/p$. Together with the condition

$$\xi \cdot y_a = \bar{\theta}_{\mu,0} + \bar{\theta}_{\mu,1} = \theta_\mu(0, 0, \ldots, 0) + \theta_\mu(1, 0, \ldots, 0)$$

(which is enforced by step 4 of Verify) and since (51) implies in particular $\Theta_\mu[X, 0, \ldots, 0] = \sum_{\boldsymbol{b}\in B_{\mu-1}} \big(f[\boldsymbol{b}, X] \cdot eq_{\boldsymbol{\gamma}}[\boldsymbol{b}, X] + \xi \cdot a[\boldsymbol{b}, X]\big)$, this eventually implies the equality (49) with soundness error $\leq \mu \cdot d/p$. More precisely, let F_i the event that the i-th polynomial identity (51) does not hold but the first $i-1$ ones do (i.e., $F_i = E_i \wedge \bigwedge_{j=1}^{i-1}(\neg E_j)$). The probability $\Pr[F]$ over the random choice of $(r_1, \ldots, r_\mu)$ that (49) is not satisfied although the sumcheck verifications succeed can be bounded as $\Pr[F] \leq \sum_{i=1}^{\mu} \Pr[F_i] \leq \mu \cdot d/p$.

Given that the random oracle output

$$\xi = H_\xi(\mathbb{x}, \mathsf{vp}, \{\hat{C}_{M,i}\}_{i=0}^{2}, \hat{C}_v, C_a, y_a, (\zeta_1, \zeta_2), (\beta_1, \beta_2), \boldsymbol{\gamma}) \in \mathbb{Z}_p^*$$

was defined after y_a and the polynomials $F[\boldsymbol{X}]$, $eq_{\boldsymbol{\gamma}}[\boldsymbol{X}]$ and $a[\boldsymbol{X}]$, it comes that (49) implies $\sum_{\boldsymbol{x}\in B_\mu} F(\boldsymbol{x}) \cdot eq_{\boldsymbol{\gamma}}(\boldsymbol{x}) = 0$ and $y_a = \sum_{\boldsymbol{x}\in B_\mu} a(\boldsymbol{x})$ unless

$$\xi = \big(\sum_{\boldsymbol{x}\in B_\mu} F(\boldsymbol{x}) \cdot eq_{\boldsymbol{\gamma}}(\boldsymbol{x})\big) / \big(y_a - \sum_{\boldsymbol{x}\in B_\mu} a(\boldsymbol{x})\big),$$

which is only possible with probability $1/(p-1)$.

Then, by applying Schwartz-Zippel again as in the proof [18, Theorem 3.2], the equality $\sum_{\boldsymbol{x}\in B_\mu} F(\boldsymbol{x}) \cdot eq_{\boldsymbol{\gamma}}(\boldsymbol{x}) = 0$ in turn implies that $F(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in B_\mu$ with all but negligible probability.

Since $(\zeta_1, \zeta_2) = H_\zeta(\mathbb{x}, \mathsf{vp}, \{\hat{C}_{M,i}\}_{i=0}^2, \hat{C}_v) \in \mathbb{Z}_p^2$ was chosen after $\{F_{M,i}\}_{i=0}^2$ and F_V (which are defined via the algebraic representations of $(\{\hat{C}_{M,i}\}_{i=0}^2, \hat{C}_v)$ and uniquely determine $Q_1[\boldsymbol{X}]$, $Q_2[\boldsymbol{X}]$ and $f[\boldsymbol{x}]$ in (46)), the condition

$$F(\boldsymbol{x}) = 0 \qquad \forall \boldsymbol{x} \in B_\mu$$

implies (with overwhelming probability over the choice of (ζ_1, ζ_2)) that the following equalities hold for all $\boldsymbol{x} \in B_\mu$:

$$\begin{aligned}
\tilde{v}[1, X_1, \ldots, X_\mu] &= \tilde{v}[X_1, \ldots, X_\mu, 0] \cdot \tilde{v}[X_1, \ldots, X_\mu, 1] \\
\tilde{v}[0, X_1, \ldots, X_\mu] &= \frac{\prod_{i=0}^2 \left(\bar{M}_i[X_1, \ldots, X_\mu] + \beta_2 \cdot s_{\mathrm{id}}[X_1, \ldots, X_\mu, \langle i \rangle] + \beta_1\right)}{\prod_{i=0}^2 \left(\bar{M}_i[X_1, \ldots, X_\mu] + \beta_2 \cdot s_\sigma[X_1, \ldots, X_\mu, \langle i \rangle] + \beta_1\right)} \\
f[X_1, \ldots, X_\mu] &= 0
\end{aligned}$$

Since $\tilde{v}[X_1, \ldots, X_{\mu+1}]$ is a multilinear polynomial and thanks to the additional condition $\tilde{v}(1, \ldots, 1, 0) = 1$, Lemma 1 then implies that

$$\prod_{\boldsymbol{x}\in B_\mu} \frac{\prod_{i=0}^2 \left(\bar{M}_i(\boldsymbol{x}) + \beta_2 \cdot s_{\mathrm{id}}(\boldsymbol{x}, \langle i \rangle) + \beta_1\right)}{\prod_{i=0}^2 \left(\bar{M}_i(\boldsymbol{x}) + \beta_2 \cdot s_\sigma(\boldsymbol{x}, \langle i \rangle) + \beta_1\right)} = 1.$$

Let the multilinear $M[X_1, \ldots, X_{\mu+2}]$ such that $M[\boldsymbol{x}, \langle i \rangle] = \bar{M}_i(\boldsymbol{x})$ for all $\boldsymbol{x} \in B_\mu$ and $i \in [0, 2]$. Since $s_\sigma(\boldsymbol{x}, \langle i \rangle) = s_{\mathrm{id}}(\hat{\sigma}(\boldsymbol{x}, \langle i \rangle))$ by the definition of the permutation polynomial (14), this implies $M(\boldsymbol{x}, \langle i \rangle) = M(\hat{\sigma}(\boldsymbol{x}, \langle i \rangle))$ for all $\boldsymbol{x} \in B_\mu$ with overwhelming probability over the random choice of (β_1, β_2) by [30, Claim A.1]. Together with the condition $f(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in B_\mu$, this implies that the multilinear polynomial $M[X_1, \ldots, X_{\mu+2}]$ is a valid witness satisfying both the gate identity and the wiring identity. □

D.2 Proof of Theorem 5

Proof. We first describe a simulator, which is slightly different from the one of [18, Appendix A]. Then, we prove that its output distribution is statistically indistinguishable from that of a real prover.

On input of a statement (consisting of an input $\mathbb{x}$ and the description of a circuit $\mathcal{C}[G]$) and a witness polynomial $M_W[X_1, \ldots, X_{\mu+2}]$, the adversary's oracle

in the zero-knowledge experiment returns $\perp$ if $M_W[X_1, \ldots, X_{\mu+2}]$ is not a valid witness. Otherwise, the runs the zero-knowledge simulator $\mathcal{S}$, which proceeds as follows to simulate a proof without using the witness polynomial M_W.

The simulator $\mathcal{S}$: Given $\mathbb{x}$ and $\mathbb{i} = \mathcal{C}[G]$, compute the input polynomial I, the selector polynomials S_1, S_2, S_3 and the wiring polynomials $s_\sigma(\cdot, \langle i \rangle)$ for each $i \in [0, 2]$. Then, do the following:

0. Choose a random multilinear $M[X_1, \ldots, X_{\mu+2}]$ that satisfies the gate identity (13) (but not the wiring identity). Namely, choose $M[X_1, \ldots, X_{\mu+2}]$ such that the virtual polynomial $f[\boldsymbol{X}]$ of (13) vanishes over B_μ. To this end, choose $\hat{M}(\boldsymbol{x}, 0, 0), \hat{M}(\boldsymbol{x}, 0, 1) \xleftarrow{R} \mathbb{Z}_p$ uniformly for each $\boldsymbol{x} \in B_\mu$ before setting

$$\begin{aligned}\hat{M}(\boldsymbol{x}, 1, 0) = S_1(\boldsymbol{x}) \cdot \big(\hat{M}(\boldsymbol{x}, 0, 0) + \hat{M}(\boldsymbol{x}, 0, 1)\big) \\ + S_2(\boldsymbol{x}) \cdot \big(\hat{M}(\boldsymbol{x}, 0, 0) \cdot \hat{M}(\boldsymbol{x}, 0, 1)\big) \\ + S_3(\boldsymbol{x}) \cdot \boldsymbol{G}\big(\hat{M}(\boldsymbol{x}, 0, 0), \hat{M}(\boldsymbol{x}, 0, 1)\big) + I(\boldsymbol{x})\end{aligned}$$

   and $\hat{M}(\boldsymbol{x}, 1, 1) = 0$ for each $\boldsymbol{x} \in B_\mu$. Then, compute $M[X_1, \ldots, X_{\mu+2}]$ as the multilinear extension of $\hat{M}[X_1, \ldots, X_{\mu+2}]$. By construction, $M[X_1, \ldots, X_{\mu+2}]$ induces a virtual polynomial $f[X_1, \ldots, X_\mu]$ (as defined in (13)) such that $\sum_{\boldsymbol{x} \in B_\mu} f(\boldsymbol{x}) = 0$. However, it does not satisfy $M(\boldsymbol{x}, \langle i \rangle) = M(\hat{\sigma}(\boldsymbol{x}, \langle i \rangle))$ for all $\boldsymbol{x} \in B_\mu$ w.h.p.
1. For each $i \in [0, 2]$, let $\bar{M}_i[\boldsymbol{X}] = M[\boldsymbol{X}, \langle i \rangle]$. Compute the multilinear polynomial $\tilde{v} \in \mathbb{Z}_p^{(\leq 1)}[X_1, \ldots, X_{\mu+1}]$ such that

$$\tilde{v}(0, \boldsymbol{x}) = \prod_{i=0}^{2} \frac{\bar{M}_i(\boldsymbol{x}) + \beta_2 \cdot s_{\mathrm{id}}(\boldsymbol{x}, \langle i \rangle) + \beta_1}{\bar{M}_i(\boldsymbol{x}) + \beta_2 \cdot s_\sigma(\boldsymbol{x}, \langle i \rangle) + \beta_1}$$

   for all $\boldsymbol{x} \in B_\mu$, as in step 3 of Prove. Note that we have

$$\prod_{\boldsymbol{x} \in B_\mu} \tilde{v}(0, \boldsymbol{x}) = \tilde{v}(1, \ldots, 1, 0)$$

   by Lemma 1 (but $\tilde{v}(1, \ldots, 1, 0) \neq 1$ w.h.p.). Compute $\{\hat{C}_{M,i}\}_{i=0}^2$ and $\hat{C}_v$ as in the real Prove algorithm and then hash values $\boldsymbol{\gamma} = H_\gamma(\mathbb{x}, \mathsf{vp}, \{\hat{C}_{M,i}\}_{i=0}^2, \hat{C}_v) \in \mathbb{Z}_p^\mu$ and $(\zeta_1, \zeta_2) = H_\zeta(\mathbb{x}, \mathsf{vp}, \{\hat{C}_{M,i}\}_{i=0}^2, \hat{C}_v) \in \mathbb{Z}_p^2$. Let the polynomials

$$\begin{aligned}Q_1[\boldsymbol{X}] &= \bar{v}[1, \boldsymbol{X}] - \bar{v}[\boldsymbol{X}, 0] \cdot \bar{v}[\boldsymbol{X}, 1] \\ Q_2[\boldsymbol{X}] &= \prod_{i=0}^{2} \big(\bar{M}_i[\boldsymbol{X}] + \beta_2 \cdot s_{\mathrm{id}}[\boldsymbol{X}, \langle i \rangle] + \beta_1\big) \\ &\quad - \bar{v}[0, \boldsymbol{X}] \cdot \prod_{i=0}^{2} \big(\bar{M}_i[\boldsymbol{X}] + \beta_2 \cdot s_\sigma[\boldsymbol{X}, \langle i \rangle] + \beta_1\big) \\ f[\boldsymbol{X}] &= S_1[\boldsymbol{X}] \cdot \big(\bar{M}_0[\boldsymbol{X}] + \bar{M}_1[\boldsymbol{X}]\big) \\ &\quad + S_2[\boldsymbol{X}] \cdot \big(\bar{M}_0[\boldsymbol{X}] \cdot \bar{M}_1[\boldsymbol{X}]\big) \\ &\quad + S_3[\boldsymbol{X}] \cdot \boldsymbol{G}\big(\bar{M}_0[\boldsymbol{X}], \bar{M}_1[\boldsymbol{X}]\big) - \bar{M}_2[\boldsymbol{X}] + I[\boldsymbol{X}],\end{aligned} \tag{52}$$

where each $\bar{M}_i[X_1, \ldots, X_\mu]$ is obtained from $M[X_1, \ldots, X_\ell]$ by adding a multiple of $X_\mu \cdot (X_\mu - 1)$ as in the real Prove algorithm. Note that, by design,

$$F[\boldsymbol{X}] \triangleq f[\boldsymbol{X}] + \zeta_1 \cdot Q_1[\boldsymbol{X}] + \zeta_2 \cdot Q_2[\boldsymbol{X}]. \tag{53}$$

vanishes over B_μ and we thus have $\sum_{\boldsymbol{x} \in B_\mu} F(\boldsymbol{x}) \cdot eq_{\boldsymbol{\gamma}}(\boldsymbol{x}) = 0$.

2. Run step 1 of the real prover, by computing C_a as a commitment to a polynomial $a[X_1, \ldots, X_\mu] = a_0 + \sum_{i=0}^{\mu} a_i[X_i]$, where $a_i[X_i] = \sum_{j=1}^{\bar{d}'} a_{i,j} \cdot X^j$ is a random univariate polynomial.
3. Compute $\xi = H_\xi(\mathbb{x}, \mathsf{vp}, \{\hat{C}_{M,i}\}_{i=0}^2, \hat{C}_v, \hat{C}_a, y_a, (\zeta_1, \zeta_2), (\beta_1, \beta_2), \boldsymbol{\gamma}) \in \mathbb{Z}_p^*$. Define

$$F_{zk}[X_1, \ldots, X_\mu] \triangleq F[X_1, \ldots, X_\mu] \cdot eq_{\boldsymbol{\gamma}}[X_1, \ldots, X_\mu] + \xi \cdot a[X_1, \ldots, X_\mu]$$

and prove that $\sum_{\boldsymbol{x} \in B_\mu} F_{zk}(\boldsymbol{x}) = y_a$ by faithfully running step 6.c-6.d of the real prover where

$$\theta_i[X] = \sum_{\boldsymbol{x} \in B_{i-1}} F_{zk}[\boldsymbol{b}, X, r_{i+1}, \ldots, r_\mu]$$

at step $i \in [\mu]$ of the sumcheck protocol of step 6.c.
4. At step 6.e.1, generate a real proof $\boldsymbol{\pi}_{C,a}$ for the polynomials committed in $\{C_{s,i}\}_{i=1}^3$, $\{C_{\sigma,\langle i \rangle}\}_{i=0}^2$, and C_a. At step 6.e.2, generate a simulated PCS proof $\boldsymbol{\pi}_M$ for random evaluations instead of the real evaluations of $\{\bar{M}_i[\boldsymbol{X}]\}_{i=0}^2$. At step 6.e.3, simulate $(\boldsymbol{\pi}_{v,x,b}, \boldsymbol{\pi}_{v,b,x})_{b=0}^1$ for random evaluations instead of the real evaluations of $\{\bar{v}[\boldsymbol{X}, b], \bar{v}[b, \boldsymbol{X}]\}_{b=0}^1$.
5. Simulate a PCS evaluation proof $\boldsymbol{\pi}_{v,s}$ (for a label lbl_3 containing the entire transcript so far) that $\hat{C}_v$ opens to 1 for the input $(1, \ldots, 1, 0) \in \mathbb{Z}_p^{\mu+1}$ using the trapdoor-less simulator of the PCS from the proof of Theorem 3.

The simulator does not use the zero-knowledge sumcheck simulator of [54, Theorem 3] since it knows a "witness" $F[\boldsymbol{X}]$ satisfying $\sum_{\boldsymbol{x} \in B_\mu} F(\boldsymbol{x}) \cdot eq_{\boldsymbol{\gamma}}(\boldsymbol{x}) = 0$. Instead, it relies on the perfect ZK simulation of the underlying PCS in order to generate a fake proof that $\bar{v}(1, 1, \ldots, 1, 0) = 1$ and fake proofs that $\{\bar{M}_i[\boldsymbol{X}]\}_{i=0}^2$ and $\tilde{v}[X_1, \ldots, X_{\mu+1}]$ evaluate to random outputs. We also note that the simulator is algebraic since the underlying PCS evaluation protocol has itself an algebraic simulator in the proof of Theorem 3.

We now prove that the simulator is indistinguishable from a real prover for any true statement $(\mathbb{i}, \mathbb{x})$ (i.e., such that there exists a witness polynomial $M_W[X_1, \ldots, X_{\mu+2}]$ satisfies both the gate identity and the wiring identity). To this end, we consider a sequence of hybrid experiments.

Exp_0: In this experiment, the distinguisher interacts with an oracle that runs the above zero-knowledge simulator $\mathcal{S}$ at each query.

Exp_1: This experiment is identical to Exp_0 except that: (i) At step 2 of the simulator, C_a is computed as a commitment to the zero polynomial instead of the real polynomial $a[\boldsymbol{X}]$ used at step 3; (ii) At step 4, the batch evaluation

proof $\boldsymbol{\pi}_{C,a}$ is simulated using the ZK simulator of the PCS in Section 3.1 (and using the trapdoor α_r). Since the latter has perfectly zero-knowledge evaluations, Exp_1 is perfectly indistinguishable from Exp_0.

Exp_2: This experiment is like Exp_1 with the difference that the simulator replaces $M[X_1, \ldots, X_{\mu+2}]$ (and its partial polynomials $\{\bar{M}_i[\boldsymbol{X}]\}_{i=0}^2$) by the real witness polynomial $M_W[X_1, \ldots, X_{\mu+2}]$ at step 1. By relying on the zero-knowledge property of the sumcheck protocol from [54, Theorem 3], Lemma 2 shows that Exp_2 is statistically indistinguishable from Exp_1.

Exp_3: This experiment is identical to Exp_2 but: (i) At step 2 of the simulator, C_a is computed as in a real proof; (ii) At step 4, $\boldsymbol{\pi}_{C,a}$ is now generated as a real proof instead of a simulated one. Since the PCS of Section 3.1 has perfectly ZK evaluations, Exp_3 is perfectly indistinguishable from Exp_2.

Exp_4: This experiment is like Exp_3 with the difference that, at steps 6.e.2-6.e.3, the simulated PCS proofs are generated for the real evaluations of $\{\bar{M}_i[\boldsymbol{X}]\}_{i=0}^2$ and $\tilde{v}[X_1, \ldots, X_{\mu+1}]$. Lemma 3 shows that Exp_4 is statistically indistinguishable from Exp_3.

Exp_5: In this experiment, at step 6.e.2, we now generate $\boldsymbol{\pi}_M$ as a real proof for the polynomials $\{\bar{M}_i[\boldsymbol{X}]\}_{i=0}^2$, $\{C_{s,i}\}_{i=1}^3$, $\{C_{\sigma,\langle i\rangle}\}_{i=0}^2$, and C_a. Also, at step 6.e.3, $(\boldsymbol{\pi}_{v,x,b}, \boldsymbol{\pi}_{v,b,x})_{b=0}^1$ are now generated as real evaluation proofs. By the perfect zero-knowledge property of the PCS, this modification does not alter the distribution of proofs and Exp_5 is indistinguishable from Exp_4.

Exp_6: This is the same as Exp_5 except that step 5 of the modified simulator (which corresponds to step 6.e.4 of Prove) generates $\boldsymbol{\pi}_{v,s}$ as a real PCS proof that $\tilde{v}(1, 1, \ldots, 1, 0) = 1$ (which is possible since $M_W[X_1, \ldots, X_{\mu+2}]$ is a valid witness). By the zero-knowledge property of the PCS evaluation protocol, Exp_6 is indistinguishable from Exp_5.

In Exp_6, the distinguisher is interacting with the real prover at each query. By combining the above, we find that the output distribution of the simulator $\mathcal{S}$ is statistically indistinguishable from that of the real prover, as claimed. □

Lemma 2. Exp_2 *and* Exp_1 *are statistically indistinguishable.*

Proof. The proof follows from the same arguments as in [54, Theorem 3]. We note that replacing the fake polynomial $M[X_1, \ldots, X_{\mu+2}]$ (which satisfies the gate identity but not the wiring identity) used by $\mathcal{S}$ by the real witness polynomial $M_W[X_1, \ldots, X_{\mu+2}]$ has the effect of trading the sumcheck witness polynomial $F[X_1, \ldots, X_\mu] \cdot eq_{\boldsymbol{\gamma}}[X_1, \ldots, X_\mu]$ by another witness polynomial whose evaluations over B_μ also sum to 0. We now argue that the proof distributions are identical by exploiting the ZK property of the sumcheck protocol of [54].

In both experiments, evaluation proofs of secret polynomials $\{\bar{M}_i[\boldsymbol{X}]\}_{i=0}^2$ and $\{\tilde{v}[\boldsymbol{X}, b], \tilde{v}[b, \boldsymbol{X}]\}_{b\in\{0,1\}}$ outside the Boolean hypercube are simulated for random evaluations. At step 5 of Verify, the simulated random evaluations define

$\theta_1(r_1) = \theta_{1,\boldsymbol{r}} = \tilde{F} + \xi \cdot a(r_1, \ldots, r_\mu)$ for a random

$$\tilde{F} = \Big[\Big(s_{1,\boldsymbol{r}} \cdot (m_{0,\boldsymbol{r}} + m_{1,\boldsymbol{r}}) + s_{2,\boldsymbol{r}} \cdot (m_{0,\boldsymbol{r}} \cdot m_{1,\boldsymbol{r}}) + s_{3,\boldsymbol{r}} \cdot \boldsymbol{G}(m_{0,\boldsymbol{r}}, m_{1,\boldsymbol{r}}) - m_{2,\boldsymbol{r}} + I_{\boldsymbol{r}}\Big) + \zeta_1 \cdot \Big(v_{1,\boldsymbol{r}} - v_{\boldsymbol{r},0} \cdot v_{\boldsymbol{r},1}\Big) + \zeta_2 \cdot \Big(\prod_{i=0}^{2}(m_{i,\boldsymbol{r}} + \beta_2 \cdot s_{\mathrm{id}}(\boldsymbol{r}, \langle i \rangle) + \beta_1) - v_{0,\boldsymbol{r}} \cdot \prod_{i=0}^{2}(m_{i,\boldsymbol{r}} + \beta_2 \cdot \sigma_{i,\boldsymbol{r}} + \beta_1)\Big)\Big] \cdot eq_{(\gamma_1,\ldots,\gamma_\mu)}(r_1, \ldots, r_\mu)$$

whose distribution does not depend on $M[X_1, \ldots, X_{\mu+2}]$ or $M_W[X_1, \ldots, X_{\mu+2}]$. We now consider $\mathcal{A}$'s global view in Exp_1 and Exp_2 and, in particular, the revealed evaluations of the round polynomials $\{\theta_i[X]\}_{i=1}^{\mu}$.

For a given transcript, the verifier obtains

$$\sum_{\boldsymbol{x} \in B_\mu} F_{zk}(\boldsymbol{x}) = \xi \cdot y_a = \theta_\mu(0) + \theta_\mu(1) \tag{54}$$

where $F_{zk}[\boldsymbol{X}]$ is the virtual polynomial defined at step 6.b of the prover. Let an arbitrary polynomial $F[\boldsymbol{X}]$ such that $F(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in B_\mu$ and satisfying the condition $F(r_1, \ldots, r_\mu) \cdot eq_{\boldsymbol{\gamma}}(r_1, \ldots, r_\mu) = \tilde{F}$. We claim that there is a compatible masking polynomial $a[\boldsymbol{X}] = a_0 + \sum_{i=1}^{\mu}\sum_{j=1}^{\bar{d}'} a_{i,j} \cdot X_i^j$ such that

$$F_{zk}[\boldsymbol{X}] = F[\boldsymbol{X}] \cdot eq_{\boldsymbol{\gamma}}[\boldsymbol{X}] + \xi \cdot a[\boldsymbol{X}].$$

Indeed, the non-redundant information that $\mathcal{A}$ obtains about the polynomials $\{\theta_i[X]\}_{i=1}^{\mu}$ consists of $(\theta_1(0), \theta_1(1), \theta_1(\alpha_1), \theta_1(r_1))$ and $\{(\theta_i(0), \theta_i(1), \theta_i(\alpha_1))\}_{i=2}^{\mu}$ since $\theta_i(r_i) = \theta_{i-1}(0) + \theta_{i-1}(1)$ for each $i \in [2, \mu]$ (revealing $y_a = \sum_{\boldsymbol{x} \in B_\mu} a(\boldsymbol{x})$ does not impose any additional constraint on the coefficients of $a[\boldsymbol{X}]$ due to the second equality in (54)). So, even if $\bar{d} > 3$, it suffices to choose $\bar{d}' = 3$ and set $a[\boldsymbol{X}] = a_0 + \sum_{i=1}^{\mu} a_i(X_i)$ for univariate degree-3 polynomials of the form $a_i[X_i] = \sum_{j=1}^{3} a_{i,j} \cdot X_i^j$ for $i \in [\mu]$: Since $a[\boldsymbol{X}]$ has $3\mu + 1$ coefficients, there exists an admissible $a[\boldsymbol{X}]$ explaining $\mathcal{A}$'s view. For each candidate $F[\boldsymbol{X}]$, the corresponding masking polynomial $a[\boldsymbol{X}]$ is determined by the equalities

$$\theta_1(x) = F(x, r_2, \ldots, r_\mu) \cdot eq_{\boldsymbol{\gamma}}(x, r_2, \ldots, r_\mu) + \xi \cdot a(x, r_2, \ldots, r_\mu) \qquad \forall x \in [0, \bar{d}' - 1] \cup \{r_1\}$$

and

$$\theta_i(x) = \sum_{\boldsymbol{b} \in B_{i-1}} \big(F(\boldsymbol{b}, x, r_{i+1}, \ldots, r_\mu) \cdot eq_{\boldsymbol{\gamma}}(\boldsymbol{b}, x, r_{i+1}, \ldots, r_\mu) + \xi \cdot a(\boldsymbol{b}, x, r_{i+1}, \ldots, r_\mu)\big) \qquad \forall x \in [0, \bar{d}' - 1], i \in [2, \mu]$$

which yield a linear system that can be solved for $(a_0, \{a_{i,j}\}_{i \in [\mu], j \in [\bar{d}']})$ whenever $r_1 \notin \{0, \ldots, \bar{d}' - 1\}$ (which occurs with overwhelming probability $1 - \bar{d}'/p$).

Finally, we note that the batch evaluation proof $\boldsymbol{\pi}_{C,a}$ do not impose any constraint on the coefficients of $\boldsymbol{a}[\boldsymbol{X}]$ in Exp_2 nor Exp_1 since they are generated by the ZK simulator of the PCS in Section 3.2 in both experiments. Consequently, the two sumcheck witnesses $F[X_1, \ldots, X_\mu] \cdot eq_{\boldsymbol{\gamma}}[X_1, \ldots, X_\mu]$ induced by $M_W[X_1, \ldots, X_{\mu+2}]$ and $M[X_1, \ldots, X_{\mu+2}]$ produce the same output distributions. □

Lemma 3. Exp_4 *and* Exp_3 *are statistically indistinguishable.*

Proof. The difference between Exp_4 and Exp_3 is that, in the latter, PCS evaluations outside the hypercube B_μ are simulated for random outputs whereas Exp_4 simulates them for the real outputs. In the case of the witness polynomials, the two distributions are identical (except with negligible probability) since evaluating the masked partial polynomials $\{\bar{M}_i[X_1, \ldots, X_\mu]\}_{i=0}^{2}$ outside B_μ results in random and independent outputs (due the the masking randomness R_i of $\hat{C}_{M,i}$ in Exp_4) and only one evaluation of each $\bar{M}_i$ is given out.

Similarly, the evaluations of the polynomial $\tilde{v}[X_1, \ldots, X_{\mu+1}]$ are masked by a polynomial $R[X_1 + X_{\mu+1}] \cdot X_\mu \cdot (X_\mu - 1)$, where $R[X]$ has degree 3. Since the only revealed evaluations outside B_μ are for the inputs $\{(r_1, \ldots, r_\mu, b)\}_{b=0}^{1}$ and $\{(b, r_1, \ldots, r_\mu)\}_{b=0}^{1}$, the mask

$$R[X_1 + X_{\mu+1}] \cdot X_\mu \cdot (X_\mu - 1)$$

takes on 4 independent values since $r_{\mu-1}, r_\mu \in \mathbb{Z}_p \setminus \{0, 1\}$ with overwhelming probability $1 - 4/p$ and the sum $X_1 + X_{\mu+1}$ takes on 4 distinct values (namely, $\{r_1, r_1 + 1, r_\mu, r_\mu + 1\}$) for the evaluation inputs, thus allowing $R[X_1 + X_{\mu+1}]$ to act as a 4-wise independent function.

It follows that the output distributions of the simulators of Exp_4 and Exp_3 are indistinguishable as long as $r_{\mu-1} \notin \{0, 1\}$ and $r_\mu \notin \{0, 1\}$ at each query (which is the case with overwhelming probability $\geq 1 - 4Q/p$ if Q is the number of queries to the simulator). □

D.3 Proof of Theorem 6

Proof. We give a reduction $\mathcal{B}$ that turns any PPT adversary $\mathcal{A}$ with non-negligible advantage in the simulation-extractability experiment into an adversary that breaks either the simulation-extractable PCS of Section 4 or the extended knowledge-soundness of the PCS in Section 3.2. Algorithm $\mathcal{B}$ sets up the CRS by re-using the SRS of the PCS scheme it obtains from its PCS challenger. Note that the schemes of sections 3.2 and 4 make use of almost identically distributed reference strings (the only difference is that the latter uses an additional random oracle).

Queries: At any time, $\mathcal{A}$ can also choose a statement that includes an input $\mathbb{x}$ and the description of a circuit $\mathcal{C}[G]$ and request $\mathcal{B}$ to simulate a proof. To simulate a proof, $\mathcal{B}$ runs the simulator $\mathcal{S}$ from the proof of Theorem 5. We note that this simulator works even for false statements (where no valid witness

polynomial satisfies the gate identity *and* the wiring identity) since one can always sample a multilinear polynomial that only satisfies the gate identity. We also recall that the simulator is algebraic since the underlying PCS evaluation protocol has itself an algebraic simulator in the proof of Theorem 3.

Output: When $\mathcal{A}$ outputs a proof of its own $\boldsymbol{\pi} = \big(\{\hat{C}_{M,i}\}_{i=0}^{2}, \hat{C}_v, \boldsymbol{\pi}_{\mathsf{zero}}\big)$ for a statement $(\mathbb{i}, \mathbb{x}) = (\mathcal{C}, \mathbb{x})$ of its choice, $\mathcal{B}$ parses

$$\boldsymbol{\pi}_{\mathsf{zero}} = \big(C_a, a_{\boldsymbol{r}}, y_a, \{m_{i,\boldsymbol{r}}, \sigma_{i,\boldsymbol{r}}\}_{i=0}^{2}, \{s_{i,\boldsymbol{r}}\}_{i=1}^{3}, \{v_{\boldsymbol{r},b}, v_{b,\boldsymbol{r}}\}_{b\in\{0,1\}}, \{C_{\theta,i}, \bar{\theta}_{i,0}, \bar{\theta}_{i,1}\}_{i\in[\mu]}, \boldsymbol{\pi}_{\mathsf{batch}}, \boldsymbol{\pi}_{\boldsymbol{r}}\big)$$

and

$$\boldsymbol{\pi}_{\boldsymbol{r}} = \Big(\boldsymbol{\pi}_{C,a}, (\boldsymbol{\pi}_M, (\boldsymbol{\pi}_{v,x,b}, \boldsymbol{\pi}_{v,b,x})_{b=0}^{1}, \boldsymbol{\pi}_{v,s}\Big).$$

Since $\mathcal{A}$ is algebraic, it outputs a representation of each commitment with respect to the group elements that have been observed during the experiment. Since the simulator $\mathcal{S}$ is algebraic, $\mathcal{B}$ can use its internal representation of simulated proof elements to infer a representation of all commitments w.r.t. the group elements contained in srs. From these representations, it can compute the $(\ell+1)$-variate polynomials (43) such that $\{\hat{C}_{M,i} = \hat{g}^{F_{M,i}(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}\}_{i=0}^{2}$ and $\hat{C}_v = \hat{g}^{F_v(\alpha_1,\ldots,\alpha_\ell,\alpha_r)}$. From these polynomials, it obtains $\{\bar{M}_i[X_1,\ldots,X_\mu]\}_{i=0}^{2}$, $\tilde{v}[X_1,\ldots,X_{\mu+1}]$ as in the proof of Theorem 4.

If all claimed evaluations of extracted polynomials are correct, then $\mathcal{B}$ can obtain a valid witness polynomial $M[X_1,\ldots,X_{\mu+2}]$ exactly as in the proof of Theorem 4 since the same arguments carry over (in particular, the outputs of H_ξ, H_ζ, H_β, and H_γ are always determined after their inputs). We now assume that at least one of the claimed polynomial evaluations is incorrect. In the case of univariate polynomials, the analysis in the proof of Theorem 9 carries over as well due to the algebraic nature of $\mathcal{S}$ and the fact that $\mathcal{S}$ never programs H_{batch}. Then, if $\boldsymbol{\pi}_{\mathsf{batch}}$ convincingly proves an incorrect univariate PCS evaluation, we can immediately rely on the extended knowledge soundness of the batch evaluation protocol of Section B.2 and the result of Theorem 9.

We are left with the treatment of incorrect evaluations of multivariate polynomials and now distinguish several cases.

• If $(\mathbb{x}, \mathcal{C}, (\hat{C}_{M,i})_{i=0}^{2})$ is a fresh tuple that was never involved in a proof generated by $\mathcal{S}$, the same holds for $(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^{2})$ unless we have a collision with two circuits $\mathcal{C}$, $\mathcal{C}^{(\tau)}$ leading to the same digest vp (which would break the $(d\ell, d\ell)$-DLOG assumption by defeating the security of the deterministic commitment). Then, we know that none of the hash values involved in $\boldsymbol{\pi}$ was programmed by $\mathcal{S}$ since $(\mathbb{x}, \mathsf{vp}, (\hat{C}_{M,i})_{i=0}^{2})$ are included in the inputs of all random oracles. This implies that all random oracle outputs are chosen *after* the algebraic representations of group elements contained in the inputs. This allows $\mathcal{B}$ to apply exactly the same analysis as in the proof of Theorem 4 since it can extract a representation of all proof components w.r.t. the group elements contained in pp (thanks to the algebraic nature of $\mathcal{S}$). In this case, we just need to rely on the extended knowledge-soundness of underlying polynomial commitments.

• If $(\mathbb{x}, \mathcal{C}, (\hat{C}_{M,i})_{i=0}^2)$ was involved in some simulation query $\tau \in [Q_S]$ resulting in a proof $\boldsymbol{\pi}^{(\tau)} = (\{\hat{C}_{M,i}\}_{i=0}^2, \hat{C}_v^{(\tau)}, \boldsymbol{\pi}_{\mathsf{zero}}^{(\tau)})$, it must have been for different pair $(\hat{C}_v^{(\tau)}, \boldsymbol{\pi}_{\mathsf{zero}}^{(\tau)}) \neq (\hat{C}_v, \boldsymbol{\pi}_{\mathsf{zero}})$. Moreover, with overwhelming probability $\geq 1 - Q_S^2/p$ (where Q_S is the number of queries to $\mathcal{S}$), there was only one simulation query involving $(\mathbb{x}, \mathcal{C}, (\hat{C}_{M,i})_{i=0}^2)$ due to the randomness of commitments $(\hat{C}_{M,i})_{i=0}^2$ generated by $\mathcal{S}$. We then distinguish two cases:

- If $\hat{C}_v^{(\tau)} = \hat{C}_v$, we must have $\boldsymbol{\pi}_{\mathsf{zero}}^{(\tau)} \neq \boldsymbol{\pi}_{\mathsf{zero}}$. Then, we further distinguish two sub-cases
  a. If $\boldsymbol{r} = (r_1, \ldots, r_\mu) \neq (r_1^{(\tau)}, \ldots, r_\mu^{(\tau)}) = \boldsymbol{r}^{(\tau)}$, we have $\mathsf{lbl}_1 \neq \mathsf{lbl}_1^{(\tau)}$, $\mathsf{lbl}_{2,j}^{(\tau)} \neq \mathsf{lbl}_{2,j}$ for each $j \in [4]$, and $\mathsf{lbl}_3 \neq \mathsf{lbl}_3^{(\tau)}$ since $(r_1, \ldots, r_\mu)$ is included in all labels at step 6.e of the prover. Hence, if one of the claimed polynomial evaluations
  $$(a_{\boldsymbol{r}}, \{m_{i,\boldsymbol{r}}, \sigma_{i,\boldsymbol{r}}\}_{i=0}^2, \{s_{i,\boldsymbol{r}}\}_{i=1}^3, \{v_{\boldsymbol{r},b}, v_{b,\boldsymbol{r}}\}_{b\in\{0,1\}}, 1)$$
  is not the correct evaluations of multivariate polynomials at step 6 of the verifier, then $\mathcal{B}$ can use $\mathcal{A}$ to break either the simulation-extractability of the batched multivariate PCS (if one of the evaluations $\{m_{i,\boldsymbol{r}}\}_{i=0}^2$ is not consistent with $\bar{M}_i[\boldsymbol{X}]$ or one of $\{v_{\boldsymbol{r},b}, v_{b,\boldsymbol{r}}\}_{b\in\{0,1\}}$ is not consistent with $\tilde{v}[X_1, \ldots, X_{\mu+1}]$ or $\tilde{v}(1, 1, \ldots, 1, 0) \neq 1$) or the extended knowledge soundness of the PCS in Section 3.2 (in the event that $(a_{\boldsymbol{r}}, \{s_{i,\boldsymbol{r}}\}_{i=1}^3)$ is not a correct evaluation of $a[\boldsymbol{X}]$ or $\{S_i[\boldsymbol{X}]\}_{i=1}^3$, respectively).
  b. If $\boldsymbol{r} = (r_1, \ldots, r_\mu) = (r_1^{(\tau)}, \ldots, r_\mu^{(\tau)}) = \boldsymbol{r}^{(\tau)}$, then we must have $\boldsymbol{\gamma} = \boldsymbol{\gamma}^{(\tau)}$, $(C_a, y_a) = (C_a^{(\tau)}, y_a^{(\tau)})$, and
  $$(C_{\theta,j}, \theta_j(0), \theta_j(1)) = (C_{\theta,j}^{(\tau)}, \theta_j(0)^{(\tau)}, \theta_j(1)^{(\tau)}) \qquad \forall j \in [\mu]$$
  unless a collision on H occurs (which happens with probability $\leq Q_H^2/p$ in the ROM). Since $\boldsymbol{\pi}_{\mathsf{zero}}^{(\tau)} \neq \boldsymbol{\pi}_{\mathsf{zero}}$, we have either $(\boldsymbol{\pi}_{\mathsf{batch}}, \boldsymbol{\pi}_{\boldsymbol{r}}) \neq (\boldsymbol{\pi}_{\mathsf{batch}}^{(\tau)}, \boldsymbol{\pi}_{\boldsymbol{r}}^{(\tau)})$ or at least one of the following inequalities
  $$\exists i \in [0,2] : m_{i,\boldsymbol{r}} \neq m_{i,\boldsymbol{r}}^{(\tau)} \tag{55}$$
  $$\exists i \in [0,2] : \sigma_{i,\boldsymbol{r}} \neq \sigma_{i,\boldsymbol{r}}^{(\tau)} \tag{56}$$
  $$\exists i \in [3] : s_{i,\boldsymbol{r}} \neq s_{i,\boldsymbol{r}}^{(\tau)} \tag{57}$$
  $$\exists b \in \{0,1\} : v_{\boldsymbol{r},b} \neq v_{\boldsymbol{r},b}^{(\tau)} \tag{58}$$
  $$\exists b \in \{0,1\} : v_{b,\boldsymbol{r}} \neq v_{b,\boldsymbol{r}}^{(\tau)} \tag{59}$$
  We then have two more cases to distinguish.
  - One of the inequalities (55)-(59) holds: Since
  $$(\mathbb{x}, \mathsf{vp}, (C_{M,i})_{i=0}^2, \hat{C}_v) = (\mathbb{x}^{(\tau)}, \mathsf{vp}^{(\tau)}, (C_{M,i}^{(\tau)})_{i=0}^2, \hat{C}_v^{(\tau)})$$

and $\boldsymbol{r} = \boldsymbol{r}^{(\tau)}$, any of the inequalities (55)-(59) implies that $\mathcal{A}$ was able to produce a batch PCS proof for a distinct polynomial evaluation than the one of the τ-th simulation query $(\mathbb{x}^{(\tau)}, \mathsf{vp}^{(\tau)})$ and for the same commitment (note that the polynomials $\{S_i[\boldsymbol{X}]\}_{i=1}^3$, $\{s_\sigma[\boldsymbol{X}, \langle i \rangle]\}_{i=0}^2$ of the forgery must be identical to those of the τ-th query since they are uniquely determined by $(\mathbb{x}^{(\tau)}, \mathsf{vp}^{(\tau)})$). In the cases of inequalities (56), (57), $\mathcal{B}$ can break the standard knowledge-soundness of the PCS in Section 3.2 since $\{s_{i,\boldsymbol{r}}^{(\tau)}\}_{i=1}^3$, $\{\sigma_{i,\boldsymbol{r}}^{(\tau)}\}_{i=0}^2$ were real polynomial evaluations at the τ-th query. In the cases (55), (58) and (59), $\mathcal{B}$ can break the simulation-extractability of the scheme in Section 4 by proving a different fake polynomial evaluation.

- If none of the inequalities (55)-(59) holds, it must be that $(\boldsymbol{\pi}_{\mathsf{batch}}, \boldsymbol{\pi}_{\boldsymbol{r}}) \neq (\boldsymbol{\pi}_{\mathsf{batch}}^{(\tau)}, \boldsymbol{\pi}_{\boldsymbol{r}}^{(\tau)})$. If $\boldsymbol{\pi}_{\mathsf{batch}} \neq \boldsymbol{\pi}_{\mathsf{batch}}^{(\tau)}$, then $\mathcal{A}$ was able to prove a fake polynomial evaluation $m_{i,\boldsymbol{r}}$ (recall that $\mathcal{S}$ simulates evaluation proofs for random evaluations, so that we must have $\bar{M}_i[\boldsymbol{r}, \langle i \rangle] \neq m_{i,\boldsymbol{r}}$ with overwhelming probability) for a different label lbl_1 than the label $\mathsf{lbl}_1^{(\tau)}$ of the τ-th query (recall that $\boldsymbol{\pi}_{\mathsf{batch}}$ is included in lbl_1). If $\boldsymbol{\pi}_{\boldsymbol{r}} \neq \boldsymbol{\pi}_{\boldsymbol{r}}^{(\tau)}$, $\mathcal{A}$ managed to produce a different PCS evaluation proof for the same inputs and outputs. If $\boldsymbol{\pi}_{C,a} \neq \boldsymbol{\pi}_{C,a}^{(\tau)}$, it means that $\boldsymbol{\pi}_M$ proves fake evaluations $\{m_{i,\boldsymbol{r}}\}_{i=0}^2$ of $\{\bar{M}_{i,\boldsymbol{r}}\}_{i=0}^2$ for a different label lbl_1 than the label $\mathsf{lbl}_1^{(\tau)}$ of the τ-th query since $\boldsymbol{\pi}_{C,a}$ is included in lbl_1. If one of $(\boldsymbol{\pi}_M, (\boldsymbol{\pi}_{v,x,b}, \boldsymbol{\pi}_{v,b,x})_{b=0}^1, \boldsymbol{\pi}_{v,s})$ differs from the corresponding proof element of the τ-th query, $\mathcal{A}$ obtained a different proof for a simulated fake polynomial evaluation among $\{m_{i,\boldsymbol{r}}\}_{i=0}^2$, $\{v_{\boldsymbol{r},b}, v_{b,\boldsymbol{r}}\}_{b\in\{0,1\}}$ and $\tilde{v}(1, 1, \ldots, 1, 0) = 1$. Again, this contradicts the simulation-extractability of the PCS in Section 4.

- If $\hat{C}_v^{(\tau)} \neq \hat{C}_v$, we must have $\mathsf{lbl}_1 \neq \mathsf{lbl}_1^{(\tau)}$ since $\hat{C}_v$ is included in lbl_1 at step 6.e.2 of the prover (and step 4 of $\mathcal{S}$).
  a. If $m_{i,\boldsymbol{r}} = m_{i,\boldsymbol{r}}^{(\tau)}$ for each $i \in [0, 2]$, then $\boldsymbol{\pi}_M$ proves fake evaluations $\{m_{i,\boldsymbol{r}}\}_{i=0}^2$ of $\{\bar{M}_{i,\boldsymbol{r}}\}_{i=0}^2$ for a different label lbl_1 than the label $\mathsf{lbl}_1^{(\tau)}$ of the τ-th query.
  b. If there exists $i \in [0, 2]$ such that $m_{i,\boldsymbol{r}} \neq m_{i,\boldsymbol{r}}^{(\tau)}$, with overwhelming probability $1 - 1/p$, the proven $m_{i,\boldsymbol{r}}$ must also be an incorrect evaluation of the corresponding $\bar{M}_i[\boldsymbol{X}]$ since $\mathcal{S}$ outputs random evaluations of each $\{\bar{M}_i[\boldsymbol{X}]\}_{i=0}^2$, which reveal no information on the actual masking randomness $\{R_i\}_{i=0}^2$ (recall that $\{\hat{C}_{M,i}\}_{i=0}^2$ are involved in only one query w.h.p.).

  In both cases, the reduction $\mathcal{B}$ can break the simulation-extractability of the PCS in Section 4.

E Proof Randomization in PST Commitments

The polynomial commitment scheme of Zhang *et al.* [56] has publicly randomizable evaluation proofs. One of source of this randomizability is the probabilistic

generation of evaluation proofs and the fact that the underlying randomness $\{s_i\}_{i=1}^{\mu}$ can be publicly modified. It is tempting to believe that unique proofs can be obtained by de-randomizing the prover. Unfortunately, it is not sufficient.

In order to obtain randomized PST commitments to multivariate polynomials with a deterministic prover, an alternative approach is to adapt the technique of randomized univariate KZG commitment [39, Section 3.3]. The idea is to have the prover choose a masking polynomial with the same degree and number of variables as the committed polynomial. A randomized commitment is a group element of the form

$$C = g^{f(\alpha_1,\ldots,\alpha_\ell)+\gamma\cdot h(\alpha_1,\ldots,\alpha_\ell)},$$

where $h \xleftarrow{R} U(\mathbb{Z}_p^{(\leq d)}[X_1,\ldots,X_\ell])$ is a random masking polynomial and where the SRS contains

$$\big(g, \{(g^{\prod_{j=1}^{\ell}\alpha_j^{i_j}}, g^{\gamma\cdot\prod_{j=1}^{\ell}\alpha_j^{i_j}})\}_{(i_1,\ldots,i_\ell)\in\mathcal{W}_{d,\ell}}, \hat{g}, \{\hat{g}^{\alpha^i}\}_{i=1}^{\ell}\big)$$

for a random $\gamma \in \mathbb{Z}_p$. The verification equation remains the same as in the deterministic PST construction and is of the form

$$e(C\cdot g^{-y}, \hat{g}) = \prod_{i=1}^{\ell} e(\pi_i, \hat{g}^{\alpha^i}\cdot\hat{g}^{-z_i})$$

Unfortunately, this construction does not provide the weak uniqueness property defined in [41]. A proof $(\pi_1,\ldots,\pi_\ell) \in \mathbb{G}^\ell$ can be turned into a modified proof $(\pi_1', \pi_2', \pi_3, \ldots, \pi_\ell)$ of the same evaluation $(\boldsymbol{z} = (z_1,\ldots,z_\ell), y)$ by setting, e.g., $\pi_1' = \pi_1\cdot(g^{\alpha^2}\cdot g^{-z_2})$ and $\pi_2' = \pi_2\cdot(g^{-\alpha}\cdot g^{z_1})$. In the original deterministic PST commitments, proofs can be randomized in the same way. For this reason, these constructions cannot be used in (a multivariate PIOP extension of) the framework of Kohlweiss *et al.* [41], which requires a weak uniqueness property on behalf of the underlying PCS evaluation proofs.