Sumcheck-based zkSNARKs are Non-Malleable

Antonio Faonio[1] and Luigi Russo[2]

[1] EURECOM, Sophia Antipolis, France faonio@eurecom.fr
[2] TU Wien, Austria luigi.russo@tuwien.ac.at

Abstract. Simulation extractability ensures that any adversary who produces a valid proof must possess a corresponding witness, even after seeing simulated proofs for potentially false statements. This property is vital for preventing malleability attacks and is therefore essential for securely deploying zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) in distributed systems. While prior work, particularly the frameworks by Faonio *et al.* (CCS'24, TCC'23) and Kohlweiss *et al.* (TCC'23), has established simulation extractability for a wide class of pairing-based zkSNARKs using the KZG univariate polynomial commitment scheme (Kate *et al.*, Asiacrypt'10), we initiate a systematic study of simulation extractability for zkSNARKs based on the celebrated sumcheck protocol and the PST multivariate polynomial commitment scheme (Papamanthou *et al.*, TCC'13).
PST cannot be simulation extractable, due to its linear homomorphism, however, we show that it satisfies a refined notion of *controlled malleability* similar to the notion of Chase *et al.* (EUROCRYPT'12), which informally captures that linear homomorphism is essentially the only admissible malleability. We demonstrate that our notion of controlled malleability suffices to ensure security within the widely adopted design paradigm of compiling polynomial interactive oracle proofs into zkSNARKs, covering several state-of-the-art schemes such as HyperPlonk (EUROCRYPT'23), Spartan (CRYPTO'20) and Libra (CRYPTO'19).

1 Introduction

Zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) [54] are powerful cryptographic primitives that enable scalable, privacy-preserving computation. At their core, zk-SNARKs allow a prover to convince a verifier that a statement is true without revealing any additional information, while also being succinct and easy to verify. These properties make zkSNARKs an attractive solution for many applications, such as voting systems, digital currencies, and secure multi-party computation.

While the security of zkSNARKs is well understood in the context of a single, isolated prover, the situation becomes more complex when considering an attacker with access to multiple proofs. In this setting, a malicious prover may attempt to exploit the information contained in the proofs to create a new proof for a different statement, without actually possessing the corresponding witness. This attack scenario, known as a *malleability attack*, was first identified by Sahai [58], who introduced the concept of simulation soundness as a way to address this issue, and was later extended to simulation extractability (SE) by De Santis *et al.* [22].

The risks of malleability are not merely hypothetical. For example, malleability attacks have historically involved over three hundred thousand Bitcoins [23]. More recently, these vulnerabilities have surfaced in state-of-the-art SNARKs such as Nova [44,55], which is already deployed in production environments (e.g., Lurk [52]). Such vulnerabilities are critical as they allow attackers to redirect funds or corrupt state transitions in decentralized ledgers. For this reason, there has recently been a growing interest in the study of SE for zkSNARKs.

In this work, we build on the line of research initiated by Faonio *et al.* [25,26] and Kohlweiss *et al.* [40], which develops general frameworks for achieving SE in a broad class of zkSNARKs.

These frameworks apply to zkSNARKs constructed via the now-standard compilation pipeline: transforming polynomial interactive oracle proofs (PIOPs) [8] into non-interactive arguments using the Fiat–Shamir transform [29], together with polynomial commitment schemes [39].

The aforementioned works focus on compiling univariate PIOPs into zkSNARKs using KZG [39] as the underlying polynomial commitment scheme. An alternative approach considers multivariate PIOPs, which rely on multivariate polynomial commitment schemes for encoding proof messages. In the univariate setting, KZG remains the standard choice for pairing-based zkSNARKs due to its succinctness and efficiency. In contrast, the multivariate setting has inspired a variety of alternative commitment schemes, each with different trade-offs. Notable examples include Orion+ [18], Gemini [11], Zeromorph [41], and Dory [45]. In this work, we focus on the PST scheme [56] for two main reasons: it provides the shortest proof size among multivariate polynomial commitment schemes, with the exception of the very recent Mercury [24] and Samaritan [34]; and it remains the simplest and one of the earliest constructions in this category.

Moreover, when compiling multivariate PIOPs into zkSNARKs, we focus on the subclass of sumcheck-based PIOPs, as defined in [18]. This class employs the classical sumcheck protocol [51] as a core component of the proof system. Although sumcheck-based PIOPs typically yield larger proof sizes compared to their univariate counterparts, they offer a significant advantage in prover efficiency. Specifically, by avoiding Fast Fourier Transform (FFT), these constructions reduce the prover's running time from quasi-linear (in the degree of the oracle polynomials) to linear. This trade-off has made the sumcheck protocol a central building block in several recent and influential zkSNARK constructions such as [11,18,59,63,64].

1.1 Our Contributions

In this work, we initiate a systematic study of simulation extractability for the class of zkSNARKs based on the sumcheck protocol, by addressing both theoretical and practical aspects of the compilation pipeline and its building blocks. Our main technical contributions are described hereafter.

Controlled malleability of PST and KZG. We provide a formal analysis of the controlled malleability of the PST commitment scheme by precisely characterizing the allowable malleabilities. This builds on and revisits known results [26] on the non-malleability of KZG.

Compiler to simulation-extractable zkSNARKs. We present a generic compiler that transforms sumcheck-based PIOPs into simulation-extractable zkSNARKs. Our construction proceeds in three steps:

1. **Zero-Knowledge.** Starting from a sumcheck-based PIOP that may not be zero knowledge, we transform it into a zero knowledge PIOP. Our transformation improves on previous work [64] by reducing the degrees of certain masking polynomials.
2. **Query minimization.** We then apply known optimizations [18], which we formally analyze and slightly improve, to reduce the query complexity. We show that these optimizations are not only useful for efficiency, but also provide a structure for the resulting PIOP that allows us to apply the controlled malleability of PST and, consequently, achieve simulation extractability for the zkSNARK.
3. **zkSNARK compilation.** Finally, we apply the, by now standard, PIOP-to-zkSNARK transform together with non-hiding PST. As noted by [18], deriving zkSNARKs from non hiding PST evaluation proofs is not straightforward. We identify and prove the minimal zero-knowledge

properties of non-hiding PST evaluation proofs that are required for the construction. Chen *et al.* [18] proposed using the technique of [10] to construct a PST-HyperPlonk $\underline{\text{zk}}$SNARK. We show that this extra step is unnecessary, requiring only minor additional masking at the PIOP level. The resulting PST-HyperPlonk zkSNARK is strictly more efficient, using only about half the number of group elements compared to the construction of [18].

A graphical summary of our results is presented in Fig. 1, and we elaborate on each contribution from a technical perspective in Section 1.3.

Similarly to previous generic frameworks [25,26,40], we establish our results in the Algebraic Group Model (AGM) [30]. Removing the AGM remains a significant challenge. First, a formal proof of knowledge soundness for PST in the standard model was only newly established [6]. Second, recent results demonstrate that security proofs for optimized SNARKs face fundamental barriers. Specifically, Lipmaa *et al.* [50] showed that the *linearization trick* [26,31] fails to satisfy special soundness in the standard model. Furthermore, recent work by Lipmaa [48] demonstrates that both *linearized* and *batched* KZG proofs (the latter of which we employ in our construction) do not achieve *trapdoorless zero knowledge*, which is required by the framework of Faust *et al.* [28] to prove simulation extractability.

1.2 Related Work

Several recent works [1,14,21,25,26,32,33,40,48] have established simulation extractability for a range of zkSNARKs including Basilisk [57], Bulletproofs [12], Jolt [2], Lasso [61], Lunar [13], Marlin [19], PLONK [31], Sonic [53], and Spartan [21]. These contributions can be grouped into two main categories. The first group includes works such as [14,21,32,33,48] that analyze the simulation extractability of (simple variants of) specific schemes. The second group includes general frameworks, such as those proposed in [25,26] and [40], which apply to broad classes of zkSNARKs, particularly those built using polynomial interactive oracle proofs and polynomial commitments. This second line provides modular security guarantees that do not require a bespoke security proof for every new (optimization or) protocol variant. Our work follows this second paradigm, extending it to the multivariate and sumcheck-based setting.

A closely related work is that of Libert [46], which investigates the simulation extractability of HyperPlonk instantiated with PST-based commitments. Libert follows a different approach than ours. Rather than relying on the standard PST evaluation proofs, he constructs a new multivariate polynomial commitment scheme that retains PST-style commitments but replaces the evaluation proof mechanism with a Σ-protocol proving knowledge of a PST evaluation proof. Since this Σ-protocol satisfies weak unique response and trapdoorless zero-knowledge, simulation extractability can be established via the general result of [33].

Additionally, Libert showed that PST evaluation proofs are not unique [47, Appendix E]. This observation imposes inherent limitations on what can be proven when using unmodified PST proofs. In particular, it implies that sumcheck-based zkSNARKs compiled via Fiat–Shamir using vanilla PST commitments can at best achieve *weak* simulation extractability: i.e., adversaries cannot produce a valid proof for a new statement that wasn't previously simulated. This *weak* notion suffices in many real-world applications [35,36,42], as it accounts for schemes that can still effectively resist relevant and practical classes of attacks [4,9], and in fact captures (a flavor of) UC-security of zkSNARKs [3].

1.3 Technical Overview

On the controlled malleability of PST (and KZG). Our first contribution is a detailed analysis of the PST polynomial commitment scheme. Due to its linear homomorphism, PST inherently lacks simulation extractability, and thus cannot satisfy standard non-malleability. However, we show that it does satisfy a weaker yet meaningful security notion, akin to the *controlled malleability* of Chase *et al.* [16,17].

More precisely, *controlled malleability* is defined with respect to a class of *admissible* transformations $\mathcal{T}$, where each $T \in \mathcal{T}$ maps an instance–witness pair (or several such pairs) of $\mathcal{R}$ to another valid pair in $\mathcal{R}$. For example, in proofs of knowledge of valid openings for homomorphic polynomial commitments, one may consider linear transformations that take two commitments as input and produce their linear combination together with the corresponding witness polynomials. A non-interactive proof system is *malleable* with respect to $\mathcal{T}$ if there exists an algorithm that, given verifying proof(s) for some source instance(s), applies $T \in \mathcal{T}$ to the proof(s) and outputs a new verifying proof for the target instance without requiring additional witness information. Moreover, a proof system is said to be *controlled malleable* with respect to $\mathcal{T}$ if these are the only transformations that can be performed, even by a malicious prover. More precisely, this is formalized by requiring the existence of a knowledge extractor that either extracts the witness for the forged instance $\tilde{\mathbb{x}}$, or else identifies simulation queries on instances $\mathbb{x}_1, \ldots, \mathbb{x}_t$ together with a transformation $T \in \mathcal{T}$ such that the forged instance is $\tilde{\mathbb{x}} = T(\mathbb{x}_1, \ldots, \mathbb{x}_t)$. Continuing with the example above of homomorphic polynomial commitments, the extractor should either recover a polynomial $p \in \mathbb{F}[\boldsymbol{X}]$, or else identify commitments $\mathsf{c}_1, \ldots, \mathsf{c}_t$ in simulation queries together with a polynomial $f \in \mathbb{F}[\boldsymbol{X}, Z_1, \ldots, Z_t]$, linear in the variables Z_i, such that the forged commitment opens to $\tilde{f}(\boldsymbol{X}) := f(\boldsymbol{X}, p_1(\boldsymbol{X}), \ldots, p_t(\boldsymbol{X}))$, under the assumption that each c_i opens to p_i for $i \in [t]$.

Recall that a polynomial commitment scheme allows to prove that a commitment c to a (possibly multivariate) polynomial $p \in \mathbb{F}[\boldsymbol{X}]$ evaluates to y on an evaluation point $\boldsymbol{x}$. Ideally, we would like to show that for PST polynomial commitment scheme, beyond linear transformations, the scheme does not permit any other form of proof reuse. For instance, a proof at evaluation point $\boldsymbol{x}$ should not be usable by the adversary to derive a valid proof at a different evaluation point $\boldsymbol{x}'$. While we are unable to establish this exact statement, we can prove a useful relaxation of it.

To capture the exact boundaries of this malleability, we rely on the policy-based simulation extractability framework of Faonio *et al.* [25], which allows us to incorporate additional constraints into the security model. In particular, our result holds under the assumption that the evaluation point $\boldsymbol{x}$ in the forgery of the adversary is chosen uniformly at random, and that the adversary's view does not include simulation queries encoding obviously contradictory information (such as a commitment c being shown to evaluate to both y and y' at the same point $\boldsymbol{x}$). These requirements are necessary as otherwise we have attacks [25].

Also notice, we follow the approach of Faonio *et al.* [25] rather than that of Kohlweiss *et al.* [40], as the latter framework requires proof uniqueness, a property that PST does not satisfy, as shown in [47]. We state our first result below (see Theorem 3 in Section 5 for the formal version).

Informal Theorem 1. *The PST polynomialcommitment scheme is policy-based controlled malleable in the AGM with respect to the class of linear transformations assuming that (1) the simulation oracle queries do not imply contradictory statements about the evaluated commitments and (2) the evaluation point in the forged proof is defined applying the random oracle to the commitment in the forged proof.*

In essence, the proof of the theorem proceeds by reducing the controlled malleability of PST over μ-variate polynomials to that of PST over $(\mu - 1)$-variate polynomials, with the base case given by the controlled malleability of KZG. This introduces the next technical challenges we encountered. Unfortunately, the result of Faonio *et al.* [26] is not sufficient for our base case for two reasons: (1) they establish only (policy-based) simulation extractability, whereas we need to prove (policy-based) controlled malleability; and (2) crucially, their policy is not strong enough for our setting. In particular, KZG evaluation proofs are themselves KZG commitments to quotient polynomials, which an adversary could query to the simulation oracle, thereby obtaining a simulation proof of a proof. To address this, Faonio *et al.* introduce a metric called the *nesting level*, which bounds the number of proof-of-proofs that the adversary can query. However, this metric becomes unbounded in our applications to sumcheck-based PIOP, whereas in their setting it remains small (for instance, equal to two in optimized PLONK). To address this issue, we provide a refined analysis for KZG, showing that certain recursive proof queries can be safely ignored. In particular, we demonstrate that proof-of-proof queries made *after* the adversary has (implicitly[3]) committed to the forgery do not need to be considered in the security reduction. We informally state our second result (see Theorem 2 in Section 5 for the formal version).

Informal Theorem 2. *The KZG polynomial commitment scheme is policy-based controlled malleable in the AGM with respect to the class of linear transformations assuming that (1) the simulation oracle queries do not imply contradictory statements about the evaluated commitments and (2) the evaluation point in the forged proof is defined applying the random oracle to the commitment in the forged proof. Moreover, the policy considered is strictly more general than [26].*

On the zero-knowledge of PST. Typically, (zk)SNARKs use non-hiding polynomial commitments since the zero-knowledge property can be added more efficiently at the information theoretic layer of the PIOP, if necessary. One of the technical challenges we encountered is that *vanilla* PST evaluation proofs over non-hiding commitments are not zero-knowledge. On the one hand, we could consider a modified version of PST evaluation proofs that do achieve zero-knowledge[4]. On the other hand, our goal is to remain as close as possible to the most efficient implementation of PST evaluation proofs. We observe that if the witness is sufficiently randomized via a carefully crafted masking algorithm msk (see Algorithm 1), one can prove a weaker form of zero-knowledge, called *distributional zero-knowledge*, with respect to that masking.

From sumcheck-based PIOPs to non-malleable zkSNARKs. Modern zkSNARKs are usually designed in two steps. First, one builds a protocol in an idealized model that ensures information-theoretic security. Then, this protocol is made practical by replacing the idealized components with a concrete cryptographic primitive. For this paper, we take as a starting point a *polynomial interactive oracle proof* (PIOP), where the prover commits to low-degree multivariate polynomials through an oracle and the verifier queries these oracles. To move beyond this idealized setting, the oracle is instantiated with a *polynomial commitment scheme*, which provides both commitments and proofs of evaluation. Since the resulting protocol remains interactive, the Fiat–Shamir transformation is applied to obtain a non-interactive zkSNARK. This, by now standard, compilation paradigm underlies many of the most efficient SNARK constructions developed in recent years, and we refer to it as the $\mathsf{PC+FS}$ compiler.

[3] More precisely, we rely on observability in the random oracle model to pinpoint the moment at which the forgery becomes fully defined.

[4] In particular, using the technique employed in the attack against uniqueness by Libert [47], one can alter the evaluation proofs so that they become zero knowledge.

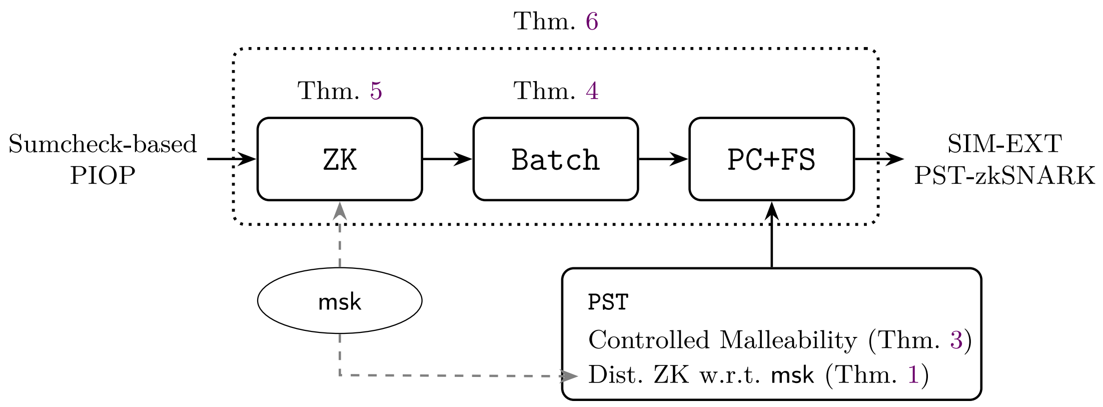

Fig. 1. A diagram summarizing our contributions.

Our third contribution is an analysis of the `PC+FS` compiler for multivariate PIOPs: instead of analyzing the `PC+FS` compiler alone, our approach is to analyze the pipeline that first adds zero-knowledge and optimizations and then the `PC+FS` compilation; moreover, unlike previous works [25,26,40], we focus specifically on the subclass of sumcheck-based multivariate PIOPs and on PST commitment scheme, a setting that has not been systematically studied before in the context of simulation extractability.[5] Sumcheck-based multivariate PIOPs [18] are PIOPs that rely on one or more invocations of the classic sumcheck protocol [51]. The sumcheck protocol itself is a special case of a multivariate PIOP, where the verifier issues random challenges and the oracle polynomial is queried at a single evaluation point determined by these challenges.

We improve the zero-knowledge compiler of Xie *et al.* [64] by showing that for the class of high-degree sumcheck protocols in [18], it suffices to mask the round polynomials with constant-degree masking polynomials instead of high-degree ones. This follows because these round polynomials are sent *as oracles* by the sumcheck-based PIOP; since they are only evaluated twice and committed using non-hiding PST, masking of degree 3 is sufficient. We refer to this as the `ZK` compiler (see Theorem 5).

A key observation in [18] is that batching techniques allow reducing the number of oracle queries to just two. Intuitively, while running k sumcheck protocols independently would require k evaluations of multivariate polynomials (each corresponding to a separate sumcheck instance), batching makes it possible to replace them with a single evaluation of a *virtual* batched polynomial. Conveniently, when the polynomial commitment scheme is homomorphic, these virtual polynomials can be queried directly by the compiled zkSNARK. We adopt and formalize the optimizations introduced in [18], treating them as a separate compiler that transforms one sumcheck-based PIOP into another (see Theorem 4). Along the way, we refine and extend the optimizations. We call the resulting pipeline, which combines zero-knowledge, these optimizations with the PIOP-to-zkSNARK compilation, the `ZK+Batch+PC+FS` compiler. Interestingly, these optimizations induce a useful structural property in the optimized sumcheck-based PIOP, that we leverage to prove simulation extractability for a broad class of such protocols. In particular, it suffices that every multivariate polynomial

[5] While existing literature [14,21] has addressed simulation extractability for (variants of) specific protocols such as Spartan and Lasso, these results are primarily *ad-hoc* works that, furthermore, rely on the framework of [28], which fundamentally requires *unique response* to establish security: however, as noted, this requirement is inherently incompatible with the linear-homomorphic properties of PST.

sent by the prover is included in at least one sumcheck protocol. This ensures that the batching step involves all multivariate polynomials in a random evaluation point chosen at the end of the protocol, which allows us to extract them by invoking the controlled-malleability of PST.

Finally, we highlight that the masking algorithm serves a dual purpose: it is required both for the zero-knowledge compiler and for achieving distributional zero-knowledge for PST (see Fig. 1). Thus our compiler applies a tailored masking to the PIOP, aligned with the leakage profile of the polynomial commitment scheme CS used in the compilation. This enables us to compile directly using a non-hiding version of CS and its non-zero-knowledge evaluation proofs, rather than relying on a hiding version of CS combined with zero-knowledge evaluation proofs (as can be achieved via transformations like the one in [10]). While we show how to instantiate this recipe using PST, we believe that this methodology paves the way for future instantiations with other non-ZK commitment schemes, most notably FRI [7]. In fact our work shows that once the leakage behavior of a given CS is characterized, the PIOP-to-zkSNARK can be instantiated with CS in its original and most efficient form, without further modification. Thus, it is an interesting open question whether FRI satisfies distributional zero-knowledge with respect to an efficient masking algorithm.

We state our third result in the following informal theorem, with the corresponding formal version given in Theorem 6 in Section 7.

Informal Theorem 3. *Let IP be a sumcheck-based PIOP in which every multivariate oracle is queried in at least one invocation of the sumcheck protocol. The zkSNARK obtained through the* `ZK+Batch+PC+FS` *compilation of IP using the PST polynomial commitment scheme is simulation-extractable in the AGM.*

We emphasize that the structural requirement we identify is not merely a theoretical artifact but is natural to meet in practice (including state-of-the-art schemes such as Libra [64], HyperPlonk [18], Spartan [59], and SuperSpartan [60]) and can be easily checked by protocol designers of future schemes.

Paper outline. In Section 2, we introduce the necessary preliminaries, including background on polynomial commitment schemes and zkSNARKs. In Section 3, we present our definition of the policy-based controlled-malleability experiment. In Section 4, we show that the PST commitment scheme achieves a (weak) form of zero-knowledge. We prove that KZG and PST achieve controlled-malleability in Section 5. In Section 6, we formally introduce multivariate sumcheck-based PIOPs. Finally, in Section 7 we present our compiler to zkSNARKs, and we show under which conditions the compiled scheme is simulation-extractable.

2 Preliminaries

A function f is negligible in λ (we write $f \in \mathsf{negl}(\lambda)$) if it approaches zero faster than the reciprocal of any polynomial. For an integer $n \geq 1$, we use $[n]$ to denote the set $\{1, 2, \dots, n\}$. Vectors and matrices are denoted in boldface. Calligraphic letters denote sets, while set sizes are written as $|\mathcal{X}|$. Lists are represented as ordered tuples, e.g. $L \coloneqq (L_i)_{i \in [n]}$ is a shortcut for the list of n elements $(L_1, \dots, L_n)$. Vectors differ from lists in that all their elements have the same type. We use $\mathcal{F}_{\boldsymbol{d},\mu}$ (resp. $\mathcal{F}_{d,\mu}$) to denote the set of multivariate polynomials in $\mathbb{F}[X_1, \dots, X_\mu]$, where the degree in the i-th variable is at most d_i (resp. d).

Asymmetric Bilinear groups. An asymmetric bilinear group is a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q, the elements P_1, P_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, $e \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently-computable non-degenerate bilinear map, and there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Let GGen be some probabilistic polynomial-time (PPT) algorithm which on input 1^λ, where λ is the security parameter, returns a description $\mathsf{pp}_\mathbb{G}$ of a bilinear group. We use implicit notation for elements of $\mathbb{G}_i$ ($i \in \{1, 2, T\}$): $[a]_i := aP_i$ where $P_T := e(P_1, P_2)$. Each element of $\mathbb{G}_i$ can be written as $[a]_i$ for some $a \in \mathbb{Z}_q$, but recovering a from $[a]_i$ is hard (discrete log problem). For $a, b \in \mathbb{Z}_q$, we distinguish between $[ab]_i$ (whose log is ab), $[a]_i \cdot b$ (scalar multiplication of $[a]_i$ by b), and $[a]_1 \cdot [b]_2 = [ab]_T$ (the pairing $e([a]_1, [b]_2)$). Implicit notation is not used for variables; for instance, $\mathsf{c} = [a]_1$ denotes a variable name for the group element with logarithm a.

```
$\mathbf{Exp}^{(n,d)\text{-OMSDH}}_{\mathsf{GGen},\mathcal{A}}(\lambda)$
$\mathcal{Q}_x \leftarrow \emptyset$
$\mathsf{pp}_\mathbb{G} \leftarrow\$ \mathsf{GGen}(1^\lambda)$
$s \leftarrow\$ \mathbb{F}_q$
$(x^*, \mathsf{y}^*) \leftarrow \mathcal{A}^{\mathcal{O}_s}(\mathsf{pp}_\mathbb{G}, \left[1, s, \ldots, s^d\right]_1, [1, s]_2)$
return $x^* \notin \mathcal{Q}_x \wedge \mathsf{y}^* = \left[\frac{1}{s - x^*}\right]_1$
```

```
Oracle $\mathcal{O}_s(x, i)$
if $i > n$ :
    return $\perp$
$\mathcal{Q}_x \leftarrow \mathcal{Q}_x \cup \{x\}$
let $z \leftarrow \left[(s - x)^{-i}\right]_1$
return $z$
```

Fig. 2. The OMSDH experiment.

Definition 1 ((n, d)-OMSDH, [26]). *Consider the experiment in Fig. 2. The n-one-more d-strong DH assumption holds for a bilinear group generator* GGen *if for every PPT adversary, making at most n oracle queries, the following advantage is negligible in λ:*

$$\mathbf{Adv}^{(n,d)\text{-OMSDH}}_{\mathsf{GGen},\mathcal{A}}(\lambda) := \Pr\left[\mathbf{Exp}^{(n,d)\text{-OMSDH}}_{\mathsf{GGen},\mathcal{A}}(\lambda) = 1\right]$$

With the lingo of [5], OMSDH is a special case of an adaptive Uber Assumption for Rational Fractions. When the set of points $\mathcal{Q}_x$ is fixed before the experiment starts, the assumption falls back to an Uber Assumption for Rational Fractions and Flexible Target, as defined in [5], that is reducible to discrete log in the AGM [26].

Definition 2 (Algebraic algorithm, [30]). *An algorithm $\mathcal{A}$ is* algebraic *if for all group elements* z *that $\mathcal{A}$ outputs (either as returned by $\mathcal{A}$ or by invoking an oracle), it additionally provides the representation of* z *relative to all previously received group elements. That is, if* elems *is the list of group elements that $\mathcal{A}$ has received so far, then $\mathcal{A}$ must also provide a vector $\boldsymbol{r}$ such that* $\mathsf{z} = \langle \boldsymbol{r}, \mathsf{elems} \rangle$.

Since in our work we focus on algebraic adversaries receiving as input a structured reference string that is a commitment key for a μ-variate polynomial commitment scheme with degree bounds $\boldsymbol{d}$, we parse the first $\prod_{i \in [\mu]}(d_i + 1)$ coefficients of $\boldsymbol{r}$ as an encoding of a μ-variate polynomial $f(\boldsymbol{X})$.

Outputs of the adversary and Random Oracle transcripts. We model adversaries' outputs and oracle queries as tuples (s, aux), where s is the primary output and aux contains auxiliary information. For example, for algebraic adversaries, aux includes the algebraic representations of group elements encoded in s. This syntax applies to all adversary queries. For instance, when querying a random oracle, the adversary sends a tuple (s, aux), and the random oracle responds with a value, denoted as $\mathsf{RO}(s)$, which depends only on s and not on aux. Note that aux may be empty or contain data required by a security game. More concretely, the auxiliary inputs to the random oracle help define security games where the adversary must commit to specific values before obtaining a challenge generated by the random oracle. Moreover, we introduce notation to assert that outputs of the random oracle *depend* on a list of (auxiliary or not) elements $x_1, \ldots, x_n$ and are computed *after* such elements are declared by the adversary.

Definition 3. *Let $x_1, \ldots, x_n$ be strings, $n \in \mathbb{N}$ and let a be an element in the image of a RO. We write $(x_1, \ldots x_n) \to_{\mathsf{RO}} a$ if there is a list of random oracle queries $(s_1, \mathsf{aux}_1), \ldots, (s_k, \mathsf{aux}_k)$ for an integer $k \geq 1$ such that:*

1. *$\forall i \in [k-1] : \mathsf{RO}(s_i)$ is a substring of s_{i+1} and $\mathsf{RO}(s_k) = a$,*
2. *$\forall j \in [n], \exists i \in [k] : x_j$ is a substring of s_i or x_j is contained in aux_i.*

Also, for $\boldsymbol{a} = (a_1, \ldots, a_\mu)$, we write $(x_1, \ldots x_n) \to_{\mathsf{RO}} \boldsymbol{a}$ to indicate that for any $i \in [\mu]$ we have that $(x_1, \ldots x_n) \to_{\mathsf{RO}} a_i$.

This helps capturing cases where $\boldsymbol{a}$ is derived by concatenating the round random challenges of a multi-round protocol (e.g., a sumcheck [51]).

2.1 Non-Interactive Zero-Knowledge Arguments

We define a PT relation $\mathcal{R}$ verifying triple $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ as in [38]. We say that $\mathbb{w}$ is a witness to the instance $\mathbb{x}$ being in the relation defined by the parameterspp when $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ (we sometimes write $\mathcal{R}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) = 1$). For example, pp could be the description of a bilinear group or contain a commitment key or a common reference string. We generalize the notion of relations to a more fine-grained notion of *indexed relations*, where we treat $(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ as equivalent to $(\mathsf{pp}, (\mathbb{i}, \mathbb{x}), \mathbb{w}) \in \mathcal{R}$. For simplicity, we sometimes omit the cryptographic parameters pp from the notation and assume them implicitly. A non-interactive proof system Π for a relation $\mathcal{R}$ (and group generator GGen) is a tuple of algorithms $(\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify})$ where:

$\mathsf{KGen}(\mathsf{pp}_{\mathbb{G}}) \to \mathsf{srs}$ is a PPT algorithm that takes as input the group parameters $\mathsf{pp}_{\mathbb{G}} \leftarrow\$ \mathsf{GGen}(1^\lambda)$ and outputs $\mathsf{srs} := (\mathsf{ek}, \mathsf{vk}, \mathsf{pp})$, where ek is the evaluation key, vk is the verification key, and pp are the parameters for $\mathcal{R}$.

$\mathsf{Prove}(\mathsf{ek}, \mathbb{x}, \mathbb{w}) \to \pi$ takes as input an evaluation key ek, a statement $\mathbb{x}$, and a witness $\mathbb{w}$ s.t. $\mathcal{R}(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ holds, and returns a proof.

$\mathsf{Verify}(\mathsf{vk}, \mathbb{x}, \pi) \to b$ takes as input a verification key, a statement $\mathbb{x}$, and either accepts $(b = 1)$ or rejects $(b = 0)$ the proof π.

If the running time of Verify is $\mathsf{poly}(\lambda + |\mathbb{x}| + \log |\mathbb{w}|)$ and the proof size is $\mathsf{poly}(\lambda + \log |\mathbb{w}|)$, we say that Π is succinct. Basic notions for a non-interactive proof systems are completeness, knowledge soundness and zero-knowledge. Informally, knowledge soundness means that any PPT prover producing a valid proof must know the corresponding witness. We omit the formal definition of this property as it is implied by simulation extractability that we present in the next section.

Zero-Knowledge in the SRS and RO model. Following the syntax of [28], the zero-knowledge simulator $\mathcal{S}$ of a NIZK is a stateful PPT algorithm that can operate in three modes:

- $(\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$ generates the parameters and the simulation trapdoor (if necessary)
- $(\pi, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathbb{x})$ simulates the proof for a statement $\mathbb{x}$
- $(a, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s)$ answers random oracle queries

The state $\mathsf{st}_{\mathcal{S}}$ is updated after each operation. Similarly to [28], we define the following wrappers.

Definition 4 (Wrappers for NIZK Simulator). *The following oracles are stateful and share their state* $\mathsf{st} = (\mathsf{st}_{\mathcal{S}}, \mathsf{coms}, \mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathsf{srs}}, \mathcal{Q}_{\mathsf{aux}})$ *where* $\mathsf{st}_{\mathcal{S}}$ *is initially set to be the empty string, and* $\mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}$ *and* $\mathcal{Q}_{\mathsf{aux}}$ *are initially set to be the empty sets.*

- $\mathcal{S}_1(\mathbb{x}, \mathsf{aux})$ *returns the first output of* $\mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathbb{x}, \mathsf{aux})$.[6]
- $\mathcal{S}'_1(\mathbb{x}, \mathbb{w})$ *first checks* $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ *where* pp *is part of* srs *and then runs (and returns the output of)* $\mathcal{S}_1(\mathbb{x})$.
- $\mathcal{S}_2(s, \mathsf{aux})$ *first checks if the query s is already present in* $\mathcal{Q}_{\mathsf{RO}}$ *and in case answers accordingly, otherwise it returns the first output a of* $\mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s)$. *The oracle updates the set* $\mathcal{Q}_{\mathsf{RO}}$ *by adding the tuple* (s, aux, a) *to the set. In the case of* non-programmable *random oracle model,* $\mathcal{S}$ *is notified of the RO query but cannot control the answer a.*

Almost all the oracles in our definitions can take auxiliary information as additional input. As explained in [25], this auxiliary information can be used in a rather liberal form. For example, in the definition above, the auxiliary information for $\mathcal{S}_2$ could contain the algebraic representations of the group elements in s (when we restrict to algebraic adversaries) or other information the security experiments might need.

Definition 5 (Zero-Knowledge w.r.t. $\mathbb{A}$ in the ROM). *A* NIZK *is zero-knowledge with respect to the class of adversaries* $\mathbb{A}$ *if there exists a PPT simulator* $\mathcal{S}$ *such that for all adversaries* $\mathcal{A} \in \mathbb{A}$:

$$\Pr\left[\begin{array}{l}\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GGen}(1^\lambda)\\ \mathsf{srs} \leftarrow \mathsf{KGen}(\mathsf{pp}_{\mathbb{G}})\\ \mathcal{A}^{\mathsf{Prove}(\mathsf{ek},\cdot,\cdot),\mathsf{RO}}(\mathsf{srs}) = 1\end{array}\right] \approx \Pr\left[\begin{array}{l}\mathsf{pp}_{\mathbb{G}} \leftarrow \mathsf{GGen}(1^\lambda)\\ (\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})\\ \mathcal{A}^{\mathcal{S}'_1,\mathcal{S}_2}(\mathsf{srs}) = 1\end{array}\right]$$

Zero-knowledge is a property that is only guaranteed for *true* statements, hence the above definition uses $\mathcal{S}'_1$ as a proof simulation oracle.

2.2 Commit-and-Prove Arguments

Commitment Schemes. A commitment scheme CS with message space $\mathcal{M}$ (and group parameters GGen) is a tuple of algorithms $\mathsf{CS} := (\mathsf{KGen}, \mathsf{Com}, \mathsf{VerCom})$ where :

$\mathsf{KGen}(\mathsf{pp}_{\mathbb{G}}, \mathsf{aux}) \rightarrow \mathsf{ck}$ takes as input $\mathsf{pp}_{\mathbb{G}} \leftarrow\!\$\, \mathsf{GGen}(1^\lambda)$, and any additional required parameters aux, and outputs a commitment key ck.

$\mathsf{Com}(\mathsf{ck}, m) \rightarrow (\mathsf{c}, o)$ takes as input the commitment key ck, and a message $m \in \mathcal{M}$, and outputs a commitment c and an opening o.

$\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, m, o) \rightarrow b$ takes as input the commitment key ck, a commitment c, a message m and an opening o, and outputs a bit.

[6] More often, simulators need only the first three inputs; abusing notation, we assume that such simulators simply ignore the auxiliary input aux.

Besides correctness, a scheme CS can satisfy two more properties.

Definition 6 (Binding). *A commitment scheme CS is (computationally) binding if no PPT adversary can find, unless with negligible probability, a commitment c, two messages $m \neq m'$ and two openings o, o':*

$$\mathsf{VerCom}(\mathsf{ck}, c, m, o) = \mathsf{VerCom}(\mathsf{ck}, c, m', o') = 1$$

Definition 7 (Hiding). *A commitment scheme CS is (statistically) hiding if $\forall m, m'$, $\forall \mathsf{ck}$:*

$$\{c : (c, o) \leftarrow \mathsf{Com}(\mathsf{ck}, m)\} \approx \{c' : (c', o') \leftarrow \mathsf{Com}(\mathsf{ck}, m')\}$$

CP-SNARKs. Commit-and-Prove SNARKs, or simply CP-SNARKs, are proof systems whose relations verify predicates over commitments [15]. We refer to a CP-SNARK for a relation $\mathcal{R}$ and a commitment scheme CS as a tuple of algorithms $\Pi := (\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify})$ where $\mathsf{KGen}(\mathsf{ck}) \rightarrow \mathsf{srs}$ is an algorithm that takes as input a commitment key ck for CS and outputs $\mathsf{srs} := (\mathsf{ek}, \mathsf{vk}, \mathsf{pp})$[7]; ek is the evaluation key, vk is the verification key, and pp are the parameters for the relation $\mathcal{R}$ (which include the commitment key ck). Also, if we consider the key generation algorithm KGen' that, upon group parameters $\mathsf{pp}_{\mathbb{G}}$, runs $\mathsf{ck} \leftarrow\!\!\$\ \mathsf{CS}.\mathsf{KGen}(\mathsf{pp}_{\mathbb{G}})$ and $\mathsf{srs} \leftarrow\!\!\$\ \Pi.\mathsf{KGen}(\mathsf{ck})$, and outputs srs; then the tuple $(\mathsf{KGen}', \mathsf{Prove}, \mathsf{Verify})$ defines a SNARK that we identify with $\Pi[\mathsf{CS}] := (\mathsf{KGen}', \mathsf{Prove}, \mathsf{Verify})$.

2.3 The PST multivariate Polynomial Commitment Scheme

We describe a non-hiding version of the multi-variate polynomial commitment by Papamanthou, Shi and Tamassia [56] (PST for short). Specifically, PST is a multi-variate polynomial commitment scheme with message space $\mathcal{F}_{\boldsymbol{d},\mu}$, defined over a bilinear group, that consists of the following algorithms:

- $\mathsf{KGen}(\mathsf{pp}_{\mathbb{G}}, \boldsymbol{d}, \mu)$ on input the group parameters $\mathsf{pp}_{\mathbb{G}}$, it samples $\beta_i \leftarrow\!\!\$\ \mathbb{F}_q$ for any $i \in [\mu]$, and it outputs the tuple $\mathsf{ck} := (\mathsf{ek}, \mathsf{vk})$, where $\mathsf{ek} := \left(\left[\prod_{i\in[\mu]} \beta_i^{j_i}\right]_1\right)_{\boldsymbol{j}\in[\boldsymbol{d}]}$, $\mathsf{vk} := ([\beta_i]_2)_{i\in[\mu]}$, and $[\boldsymbol{d}]$ is the set $[0, d_1] \times \cdots \times [0, d_\mu]$.
- $\mathsf{Com}(\mathsf{ck}, f(\boldsymbol{X}))$ outputs the commitment $\mathsf{c} := [f(\boldsymbol{\beta})]_1$.
- $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, f(\boldsymbol{X}))$ outputs 1 iff $\mathsf{c} = [f(\boldsymbol{\beta})]_1$.

We notice that for the special case $\mu = 1$ we recover the KZG scheme [39]. To emphasize the different settings, we define $\mathsf{CS}_{\mathsf{PST},\mu}$ to be the polynomial commitment scheme above where KGen fixes the last parameter to μ, thus $\mathsf{CS}_{\mathtt{KZG}} := \mathsf{CS}_{\mathsf{PST},1}$.

The PST commitment scheme allows for *evaluation proofs* which, in the framework of [15], is a CP-SNARK $\Pi_{\mathtt{evl}}$ for $\mathcal{R}_{\mathtt{evl}}$ where:

$$\mathcal{R}_{\mathtt{evl}}(\mathsf{ck}, (\mathsf{c}, \boldsymbol{x}, y), f) = 1 \text{ iff } f(\boldsymbol{x}) = y \wedge \mathsf{c} = [f(\boldsymbol{\beta})]_1 .$$

We describe such a CP-SNARK below:

- $\Pi_{\mathtt{evl}}.\mathsf{Prove}(\mathsf{ek}, \mathbb{x} = (\mathsf{c}, \boldsymbol{x}, y), \mathbb{w} = f)$ outputs $(\pi_i)_{i\in[\mu]}$ such that $\pi_i := [\pi_i(\boldsymbol{\beta})]_1$, where $(\pi_i(X))_{i\in[\mu]}$ are the quotient polynomials such that $\sum_{i\in[\mu]} \pi_i(\boldsymbol{X})(X_i - x_i) \equiv f(\boldsymbol{X}) - y$.
- $\Pi_{\mathtt{evl}}.\mathsf{Verify}(\mathsf{vk}, \mathbb{x} = (\mathsf{c}, \boldsymbol{x}, y), \pi = (\pi_i)_{i\in[\mu]})$ outputs 1 iff $e(\mathsf{c} - [y]_1, [1]_2) = \sum_{i\in[\mu]} e(\pi_i, [\beta_i - x_i]_2)$.

[7] Often, such an algorithm simply and deterministically (re)-parses ck as $(\mathsf{ek}, \mathsf{vk})$, in this case we can omit the algorithm from the description of the proof system.

Since there are multiple ways to compute the polynomials $(\pi_i)_{i\in[\mu]}$, we consider the prover that computes $\pi_1, \dots, \pi_\mu$ itereatively, starting from μ down to 1, using the Euclidean division for polynomials.

```
Prover $\mathit{\Pi}_{\mathtt{evl}}[\mathsf{CS}_{\mathsf{PST},\mu}].\mathsf{Prove}(\mathsf{ek}, \mathbb{x} = (\mathsf{c}, \boldsymbol{x}, y), \mathbb{w} = f)$:
  – Let $R_{\mu+1}(\boldsymbol{X}) \leftarrow f(\boldsymbol{X}) - y$
  – For $i = \mu, \dots, 1$:
      • Compute $\pi_i(\boldsymbol{X})$ such that $R_{i+1}(\boldsymbol{X}) = \pi_i(\boldsymbol{X})(X_i - x_i) + R_i$ and $\mathtt{deg}_{X_i}(R_i) = 0$.
  – Output $[\pi_1(\boldsymbol{\beta})]_1, \dots, [\pi_\mu(\boldsymbol{\beta})]_1$.
```

Notice that, when c is fixed the proof elements for two different proofs can be correlated, e.g. a proof (π_1, π_2) for $(\mathsf{c}, (0,0), 0)$ and a proof (π_1', π_2') for $(\mathsf{c}, (1,0), 0)$ have that $\pi_2 = \pi_2'$.

Generalized Evaluation for Polynomial Commitment. While the relation $\mathcal{R}_{\mathtt{evl}}$ provides a natural and self-contained abstraction for reasoning about polynomial evaluation proofs, it is somewhat limited in scope. In particular, it captures a "stand-alone" notion of evaluation: a single committed polynomial evaluated at a single point. However, in the context of concrete proof systems, evaluation proofs often arise in more structured and interdependent forms. For instance, protocols may involve multiple committed polynomials, evaluated at shared or distinct points, and the statement to be proved may assert that a specific linear combination of these evaluations equals a claimed value. To properly capture such scenarios, we introduce a *generalized evaluation relation*, $\mathcal{R}_{\mathtt{geval}}$, an indexed relation that extends $\mathcal{R}_{\mathtt{evl}}$ by incorporating:

- a list of commitments $(\mathsf{c}_j)_{j\in[m]}$, each to a polynomial f_j,
- two public evaluation points $\boldsymbol{x} \in \mathbb{F}^\mu$, $\boldsymbol{z} \in \mathbb{F}^k$ for some k,
- a list of polynomials $(a_j)_{j\in[m]}$ that serve as index and are used as *coefficient polynomials* in a linear combination over the (committed) polynomials $(f_j)_{j\in[m]}$.

Formally, $\mathcal{R}_{\mathtt{geval}}$ is defined as follows.

$$(\mathbb{i} = (a_j)_{j\in[m]}, \mathbb{x} = ((\mathsf{c}_j)_{j\in[m]}, \boldsymbol{z}, \boldsymbol{x}, y), \mathbb{w} = (f_j)_{j\in[m]}) \in \mathcal{R}_{\mathtt{geval}} \iff \begin{cases} \forall j : \mathsf{c}_j = [f_j(\boldsymbol{\beta})]_1, \\ \sum_j a_j(\boldsymbol{z}, x_1) f_j(\boldsymbol{x}) = y \end{cases}.$$

This formulation may appear more elaborate at first glance, but it reflects the structure required in many constructions, especially when applying the PIOP-to-zkSNARK compilation pipeline in Section 7. In fact, the indexer polynomials allow to express different forms of batching under one single hat. One may ask why the indexer polynomials also depend on the variable x_1. While this dependence is not required for our purposes, Faonio *et al.* [26] showed that allowing the indexer polynomial to depend on the evaluation variable x_1 (in their univariate setting) enables the analysis of the so-called *linearization trick* [53,31], where proving the evaluation of an arbitrary combination of committed polynomials reduces to a single evaluation proof at the point x_1. We keep this dependence since it may be useful in future extensions and strengthens the generality of our work. The CP-SNARK $\mathit{\Pi}_{\mathtt{geval}}$ for $\mathcal{R}_{\mathtt{geval}}$, that we simply call generalized PST, works as follows:

- The prover of $\mathit{\Pi}_{\mathtt{geval}}$ runs $\mathit{\Pi}_{\mathtt{evl}}$ on instance $\mathbb{x}' := (\sum_j a_j(\boldsymbol{z}, x_1)\mathsf{c}_j, \boldsymbol{x}, y)$ and witness $\mathbb{w}' := \sum_j a_j(\boldsymbol{z}, x_1) f_j$ and outputs the PST proof;
- The verifier of $\mathit{\Pi}_{\mathtt{geval}}$ runs the verifier of $\mathit{\Pi}_{\mathtt{evl}}$ recomputing the input $\mathbb{x}'$ from $\mathbb{i}$ and $\mathbb{x}$, as the prover does, and the PST proof.

We call $\mathit{\Pi}_{\mathsf{PST},\mu} := \mathit{\Pi}_{\mathtt{geval}}[\mathsf{CS}_{\mathsf{PST},\mu}]$ and $\mathit{\Pi}_{\mathsf{KZG}} := \mathit{\Pi}_{\mathtt{geval}}[\mathsf{CS}_{\mathsf{KZG}}]$.

```
$\mathbf{Exp}^{(\Phi,\mathcal{T})\text{-cm}}_{\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda)$
$\mathsf{pp}_{\mathbb{G}} \leftarrow\!\$\ \mathsf{GGen}(1^\lambda)$
$(\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}}), \mathsf{coms} \leftarrow\!\$\ \mathbb{G}_1^n$
$(\mathbb{x}, \pi, \mathsf{aux}_{\mathcal{E}}, \mathsf{aux}_{\Phi}) \leftarrow \mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}(\mathsf{srs}, \mathsf{coms})$
$\mathsf{view} \leftarrow (\mathsf{srs}, \mathsf{coms}, \mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}})$
$(\mathbb{w}, T = (T_x, T_w), (\mathbb{x}_i)_{i\in[k]}) \leftarrow \mathcal{E}(\mathsf{view}, \mathsf{st}_{\mathcal{S}}, \mathsf{aux}_{\mathcal{E}})$
$b_R \leftarrow (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}$
$b_T \leftarrow (\exists i : \mathbb{x}_i \notin \mathcal{Q}_{\mathsf{sim}} \vee T_x(\mathsf{pp}, (\mathbb{x}_i)_i) \neq \mathbb{x} \vee T \notin \mathcal{T})$
$b_\Phi \leftarrow \Phi((\mathbb{x}, \pi), \mathsf{view}, \mathsf{aux}_\Phi)$
return $\mathsf{Verify}^{\mathcal{S}_2}(\mathsf{srs}, \mathbb{x}, \pi) \wedge b_\Phi \wedge b_R \wedge b_T$
```

```
$\mathcal{S}_1(\mathbb{x}, \mathsf{aux})$
$\pi, \mathsf{st}_{\mathcal{S}} \leftarrow \mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathbb{x}, \mathsf{aux})$
$\mathcal{Q}_{\mathsf{sim}} \leftarrow \mathcal{Q}_{\mathsf{sim}} \cup \{(\mathbb{x}, \mathsf{aux}, \pi)\}$
return $\pi$

$\mathcal{S}_2(s, \mathsf{aux})$
if $\nexists \mathsf{aux}, a : (s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$ :
  $a, \mathsf{st}_{\mathcal{S}} \leftarrow \mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s, \mathsf{aux})$
  $\mathcal{Q}_{\mathsf{RO}} \leftarrow \mathcal{Q}_{\mathsf{RO}} \cup \{(s, \mathsf{aux}, a)\}$
return $a$
```

Fig. 3. The Φ-simulation $\mathcal{T}$-controlled-malleability security experiment.

3 Policy-Based Controlled-Malleability in the AGM

We recall the definitional framework of [25]. In their framework a policy is a pair $\Phi := (\Phi_0, \Phi_1)$ of PPT algorithms. The Φ-simulation extractability experiment defined by [25] executes the policy algorithm Φ_0 which generates public information pp_Φ. We slightly simplify their framework, since our work focuses on algebraic adversaries, by defining a policy as a single PPT algorithm Φ and assuming that, instead of Φ_0, it simply outputs n uniformly random group elements in $\mathbb{G}_1$ for a parameter n that is polynomial in the security paramater. The adversary is given as input the SRS and these random group elements. After receiving a forgery from the adversary, the security experiment runs the policy Φ. The policy Φ is a predicate that decides whether the attack is legitimate, e.g., it is not a trivial one such as returning a proof received by the simulation oracle.

We extend the framework to the setting of controlled-malleability [16]. This framework assumes that the extractor is capable of either extracting a witness or providing a valid explanation for the forged proof. Such an explanation must describe the forged proof as the result of applying a transformation to a set of simulated proofs. Moreover, the framework mandates that this transformation belongs to a predefined set of benign transformations, such as the set of homomorphic operations supported by the scheme. Looking ahead, we can consider a simulation-extractability game in which the adversary has oracle access to the PST evaluation proof simulator but must produce a forgery for the generalized evaluation proof. This is why we slightly change the notion of admissible transformation from [16] to allow instances from different relations.

Definition 8 (Admissible Transformation and Allowable Set). *We say that a k-ary transfomation $T = (T_x, T_w)$ is* admissible *for a source relation $\mathcal{R}_S$ and a target relation $\mathcal{R}$ if:*

$$\forall i \in [k] : (\mathsf{pp}, \mathbb{x}_i, \mathbb{w}_i) \in \mathcal{R}_S \Rightarrow \Big(T_x(\mathsf{pp}, (\mathbb{x}_i)_{i\in[k]}), T_w((\mathbb{w}_i)_{i\in[k]})\Big) \in \mathcal{R}$$

We say that a set of transformation $\mathcal{T}$ is allowable *for $\mathcal{R}_S$ and $\mathcal{R}$ if every $T \in \mathcal{T}$ is admissible for $\mathcal{R}_S$ and $\mathcal{R}$, membership in $\mathcal{T}$ is efficiently decidable, and computing T_x and T_w takes polynomial time.*

We now define the notion of controlled malleability in the AGM. The experiment in Fig. 3 models an adversary with oracle access to the simulator and requires it to output a forgery. We provide $\mathcal{A}$

with additional input in the form of a list of $\mathbb{G}_1$ elements, which we call *simulated commitments*: this models the concept of obliviously sampled elements in the AGM [49] that, in the context of KZG/PST, can be interpreted as commitments for which even the experiment does not know a valid opening. The extractor's goal is to output either a valid witness $\mathbb{w}$ or a transformation $T = (T_x, T_w)$ together with a list of queries to the simulation oracle. The component T_x explains the input of the forgery as $\mathbb{x} = T_x(\mathbb{x}_1, \dots, \mathbb{x}_k)$. Hence, by the definition of admissible transformation, if we had valid witnesses $\mathbb{w}_i$ for $\mathbb{x}_i$ for all $i \in [k]$, then $T(\mathbb{w}_1, \dots, \mathbb{w}_k)$ would yield a valid witness for the forgery.

Definition 9 ($\varPhi$-Simulation $\mathcal{T}$-controlled-malleability). *Let $\mathcal{T}$ be a set of allowable transformations for source relation $\mathcal{R}_S$ and target relation $\mathcal{R}$, and consider $\mathbf{Exp}^{(\varPhi,\mathcal{T})\text{-}\mathtt{cm}}$ defined in Fig. 3 for some policy $\varPhi$. A NIZK $\varPi$ for a relation $\mathcal{R}$ and simulator $\mathcal{S}$ for instances in $\mathcal{R}_S$ is $\varPhi$-simulation $\mathcal{T}$-controlled-malleable (briefly, $(\varPhi, \mathcal{T})$-CM) in the AGM (and in the SRS model) if for every PT algebraic adversary $\mathcal{A}$ there exists an efficient extractor $\mathcal{E}$ such that the following advantage is negligible in λ:*

$$\mathbf{Adv}^{(\varPhi,\mathcal{T})\text{-}\mathtt{cm}}_{\varPi,\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) \coloneqq \Pr\left[\mathbf{Exp}^{(\varPhi,\mathcal{T})\text{-}\mathtt{cm}}_{\varPi,\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) = 1\right]$$

We say that a CP-SNARK $\varPi$ is $(\varPhi, \mathcal{T})$-CM for the relation $\mathcal{R}$ and a commitment scheme CS if the NIZK $\varPi[\mathsf{CS}]$ is $(\varPhi, \mathcal{T})$-CM for the relation $\mathcal{R}$. We say that $\varPi$ is $\varPhi$-simulation-extractable (briefly, $\varPhi$-SE) if $\varPi$ is $(\varPhi, \emptyset)$-CM.

4 Distributional Zero-Knowledge for PST

As noted in Section 1, PST evaluation proofs are not zero-knowledge. In this section we establish a weaker form of zero-knowledge, where valid instances (together with their witnesses) are assumed to be masked using masking polynomials.

Masking Polynomials. We import the necessary definition of masking algorithms and their properties.

Definition 10 (Masking, [18]). *A randomized algorithm msk is a (t, C, μ)-masking if:*

1. *For every polynomial $f \in \mathcal{F}_{d,\mu}$, the masked polynomial $f^* \leftarrow \mathsf{msk}(f, t, C)$ satisfies $f^*(x) = f(x)$ for all $x \in \{0,1\}^\mu$.*
2. *For every such f and any sequence of queries $q = (q_1, \dots, q_t)$ accepted by C, the tuple $(f^*(q_1), \dots, f^*(q_t))$ is uniformly random in $\mathbb{F}^t$.*

We consider the class of checkers $\mathcal{C} = \{C_1, \dots, C_\mu\}$ where, for any ℓ, C_ℓ accepts a list of queries $(q_1, \dots, q_t)$ and $q_i = (b_{i,1}, \dots, b_{i,\mu}) \in \mathbb{F}^\mu$, if and only if for all i the ℓ-th coordinate satisfies $b_{i,\ell} \notin \{0, 1, b_{1,\ell}, \dots, b_{i-1,\ell}\}$.

Lemma 1 (Masking Lemma, [18, Lemma A.1]). *For every $\mu, t \in \mathbb{N}$ and $\ell \in [\mu]$, the masking Algorithm 1 is a (t, C_ℓ, μ)-masking. Moreover, for any $f \in \mathcal{F}_{d,\mu}$ and $\ell \in [\mu]$, the masked polynomial $f^* \leftarrow \mathsf{msk}(f, t, \ell)$ has total degree $\mathsf{deg}(f^*) = \max(d,\ t+1,\ 2)$.*

Notice that the lemma above was proved for the masking algorithm that omits the boxed operation in Lines 2 and 3 of Algorithm 1. However, it is clear that adding additional masking with independent randomness can only enhance the privacy's property of the masking algorithm (or at least it cannot worsen such properties).

Algorithm 1 $\mathsf{msk}(f, t, \ell)$:

```
1: Sample $r_\ell(X_\ell) \leftarrow_r \mathbb{F}_{<t}[X_\ell]$
2: $\boxed{\text{Sample } r_i \leftarrow\!\$\ \mathbb{F} \text{ for all } i \neq \ell}$
3: return $f + \boxed{\sum_{i \neq \ell} Z(X_i) \cdot r_i} + Z(X_\ell) \cdot r_i(X_\ell)$ where $Z(X) := X(X-1)$.
```

Wrapper adversary $\mathcal{B}_{\mathsf{msk}}[\mathcal{A}]$:

- At initialization, $\mathcal{B}_{\mathsf{msk}}$ receives the SRS for the μ-variate $\Pi_{\mathtt{evl}}$ with degree bounds $\boldsymbol{d} = (d_j)_{j \in [\mu]}$. It runs $\mathcal{A}$ with the same SRS.
- When queried by $\mathcal{A}$, $\mathcal{B}_{\mathsf{msk}}$ can respond to three types of queries:
  - $(\mathtt{sample}, i, f_i, t_i, C_i)$: If p_i is not defined yet and $C_i \in \mathcal{C}$, then it samples $p_i \leftarrow\!\$\ \mathsf{msk}(f_i, t_i, C_i)$ else output $\perp$. If $\exists j : \mathtt{deg}_j(p_i) > d_j$ then return $\perp$. Else set $\mathcal{Q}_i \leftarrow \{\boldsymbol{\beta}\}$, it sends $\mathsf{c}_i = [p_i(\boldsymbol{\beta})]_1$ to $\mathcal{A}$
  - $(\mathtt{value}, i, \boldsymbol{x})$: If p_i is defined and $|\mathcal{Q}_i| < t_i$ and $C_i(\mathcal{Q}_i \cup \{\boldsymbol{x}\}) = 1$, it updates $\mathcal{Q}_i \leftarrow \mathcal{Q}_i \cup \{\boldsymbol{x}\}$, it outputs the value $p_i(\boldsymbol{x})$.
  - $(\mathtt{proof}, (i, \boldsymbol{x}, y))$: It parses $\boldsymbol{x} = (x_j)_j$, asserts $p_i(\boldsymbol{x}) = y$, $\forall j : x_j \notin \{0, 1\}$, and that the i-th polynomial was not already queried for simulation, and if so, it calls the simulation oracle with input $(\mathsf{c}_i, \boldsymbol{x}, y)$, receiving proof π that forwards to $\mathcal{A}$, otherwise it returns $\perp$ to $\mathcal{A}$.
- When $\mathcal{A}$ outputs its decision bit, $\mathcal{B}_{\mathsf{msk}}$ returns the same bit.

Fig. 4. The wrapper adversary for the distributional zero-knowledge definition.

Distributional Zero-Knowledge. We notice that the PST's evaluation proof $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathtt{PST},\mu}]$ defined in Section 2.3 is not zero-knowledge. The reason is that, while, for any multivariate polynomial f and evaluation points $\boldsymbol{x}$, there is a unique set of polynomials $\pi_i(X)$, as computed by the prover[8] defined in Section 2.3, such that $f(X) - f(\boldsymbol{x}) = \sum_i \pi_i(X)(X - x_i)$, there are many possible assignments of the group elements π_i such that the pairing equation $e(\mathsf{c} - [f(\boldsymbol{x})]_1, [1]_2) = \sum_i e(\pi_i, [\beta_i - x_i]_2)$ holds. In particular, the simulator should be able to find the set of group elements $\{[\pi_i(\boldsymbol{\beta})]_1\}_i$ given only the commitment c and the trapdoor $\boldsymbol{\beta}$. For a μ-variate polynomial with bounded degrees $\boldsymbol{d}$ and whose coefficients are sampled uniformly at random, this task is information-theoretically impossible for any simulator.

Definition 11 (Distributional Zero-Knowledge for $\Pi_{\mathtt{evl}}$). *Let $\mu \in \mathbb{N}$, $\mathcal{C}$ be a class of checkers, and let msk be a (t, C, μ)-masking algorithm for any checker $C \in \mathcal{C}$ and any t, consider the wrapper adversary $\mathcal{B}_{\mathsf{msk}}[\mathcal{A}]$ against zero-knowledge for $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathtt{PST},\mu}]$ with a parameterized algorithm $\mathcal{A}$ described in Fig. 4. Let $\mathbb{B}_{\mathsf{msk}} = \{\mathcal{B}_{\mathsf{msk}}[\mathcal{A}] : \mathcal{A}\}$.*

We say that $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathtt{PST},\mu}]$ is distributional zero-knowledge w.r.t. a masking algorithm msk with checkers $\mathcal{C}$ if (1) it is zero-knowledge with respect to the class of adversaries $\mathbb{B}_{\mathsf{msk}}$ and (2) the zero-knowledge simulator is correct, namely, let $\mathcal{S}$ be the zero-knowledge simulator w.r.t. the class of adversaries $\mathbb{B}_\mu$ then, and for any $\mathsf{pp}_{\mathbb{G}}$, any $\mathsf{srs}, \mathsf{st}_{\mathcal{S}} \leftarrow \mathcal{S}_0(\mathsf{pp}_{\mathbb{G}})$, for any $(\mathbb{x}, \mathbb{w})$ s.t. $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathtt{evl}}$:

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, \mathbb{x}, \mathcal{S}(\mathsf{st}_{\mathcal{S}}, 1, \mathbb{x})) = 0\right] \leq \mathsf{negl}(\lambda).$$

[8] Note that there are multiple ways to compute such polynomials if we do not follow the prover's specification; in particular, as shown in [46], PST proofs (even for non-hiding commitments) are not unique.

Remark 1. Item (2) in the definition above is mainly a technical point. In standard zero-knowledge, the simulator is correct as defined in (2). However, in distributional zero-knowledge, this does not hold because the simulator can behave arbitrarily on (true) instances where the checker outputs 0.

Theorem 1. *For any degree bounds $\boldsymbol{d}$ and for any $\mu \in \mathbb{N}$, the SNARK $\Pi_{\mathsf{evl}}[\mathsf{CS}_{\mathsf{PST},\mu}]$ is distributional zero-knowledge w.r.t. the masking Algorithm 1 with checkers $\mathcal{C}$. In particular, the statistical distance between the ideal and real worlds is $O(q_{\mathsf{sim}}\mu/|\mathbb{F}|)$ where q_{sim} is the number of simulation queries made by the adversary.*

Proof. For any μ, let $\mathcal{S}^{(\mu)}$ be the following (recursively-defined) stateful zero-knowledge simulator for $\Pi_{\mathsf{evl}}[\mathsf{CS}_{\mathsf{PST},\mu}]$.

<u>Simulator $\mathcal{S}^{(\mu)}((\beta_j)_{j\in[\mu]}, (\mathsf{c}, \boldsymbol{x}, y))$</u>

- At initialization the simulator defines an empty map with signature $\mathbb{G}_1 \times \mathbb{F} \to \mathbb{G}_1$.
- If $\mu = 1$ the simulator returns $(\mathsf{c} - [y]_1)(\beta_1 - x_1)^{-1}$;
- Else it checks whether $(\mathsf{c} - [y]_1, x_\mu)$ is defined in the map: if it is not defined, it samples $\mathsf{r} \leftarrow\$ \mathbb{G}_1$ uniformly at random and maps $(\mathsf{c} - [y]_1, x_\mu)$ to r; otherwise, it retrieves the associated r.
  The simulator computes[9] $\pi_\mu \leftarrow (\mathsf{c} - [y]_1 - \mathsf{r})(\beta_\mu - x_\mu)^{-1}$, and recursively calls the simulator for $(\mu - 1)$-variate PST, namely:
  $$(\pi_1, \dots, \pi_{\mu-1}) \leftarrow \mathcal{S}^{(\mu-1)}((\beta_j)_{j\in[\mu-1]}, (\mathsf{r}, \boldsymbol{x}_{:\mu-1}, 0)).$$
- It returns $(\pi_j)_{j\in[\mu]}$.

We first show that the simulator is correct, namely that for any $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ the simulator can produce a valid proof. We can verify correctnes by induction. First, $\mathcal{S}^{(1)}$ is the zero-knowledge simulator of KZG's evaluation proofs, which covers the base case. Assume $\mathcal{S}^{(\mu-1)}$ is correct, then:

$$e(\mathsf{r}, [1]_2) = \sum_{j\in[\mu-1]} e(\pi_j, [\beta_j - x_j]_2)$$

and using the definition of π_μ and the equation above:

$$\begin{aligned}\sum_{j\in[\mu]} e(\pi_j, [\beta_j - x_j]_2) &= e(\mathsf{r}, [1]_2) + e((\mathsf{c} - [y]_1 - \mathsf{r})(\beta_\mu - x_\mu)^{-1}, [\beta_\mu - x_\mu]_2) \\ &= e(\mathsf{r} + \mathsf{c} - [y]_1 - \mathsf{r}, [1]_2),\end{aligned}$$

which concludes the proof of correctness.

We prove (1) of Definition 11 by induction on μ. We prove that the statistical distance between the ideal and real world is $O(q_{\mathsf{sim}}\mu/|\mathbb{F}|)$ where $q_{\mathsf{sim}} = \mathsf{poly}(\lambda)$ is the number of simulation oracle queries made by the adversary. For the base of our induction argument, because KZG is perfect zero-knowledge, we have that it is also distributional zero-knowledge.

[9] We notice small caveat to avoid division by zero, this caveat applies also to KZG's simulator: the simulator and accordingly the prover must set the proof element π_μ directly to $[1]_1$ when $\exists j \in [\mu] : x_j = \beta_j$. The choice is arbitrary, in fact, when $x_\mu = \beta_\mu$ the verifier computes the pairing $e(\pi_\mu, [0]_2)$, which is equal to the element $[0]_T$ independently of π_μ, as part of its verification procedure.

For $\mu > 1$, let $\mathcal{A}$ be an adversary that breaks the distributional zero-knowledge of $\Pi_{\mathtt{evl},\mu}$ and that makes at most q_{sim} queries to the simulation oracle.

We define a sequence of seven hybrids where $\mathbf{H}_0$ is the real experiment and $\mathbf{H}_6$ is the ideal experiment, and we let $\varepsilon_i := \Pr[\mathbf{H}_i = 1]$. Specifically, let $\mathbf{H}_0$ be the real-world zero-knowledge distribution for $\mathcal{B}_{\mathsf{msk}}[\mathcal{A}]$, namely, the game where $\mathcal{B}_{\mathsf{msk}}[\mathcal{A}]$ has oracle access to the proving algorithm.

Let $\mathbf{H}_1$ be the same as the previous hybrid but where the oracle queries are handled differently. In particular, for any query to $(\mathtt{proof}, (i, \boldsymbol{x}, y))$ from $\mathcal{A}$, if $\boldsymbol{x}$ was not previously queried on a $\mathtt{value}$ query to the polynomial $i \in [n]$ then simply returns $\perp$. Otherwise the hybrid proceeds as in $\mathbf{H}_0$.

Lemma 2. *Let $q_{\mathsf{sim}} = \mathsf{poly}(\lambda)$ be the number of simulation oracle queries made by $\mathcal{A}$, then $|\varepsilon_1 - \varepsilon_0| \leq O(q_{\mathsf{sim}}/|\mathbb{F}|)$.*

Proof. Similarly to the analysis in [20], the two hybrids diverge if there exists an index j such that the j-th simulation query $(\mathtt{proof}, i, \boldsymbol{x}, y)$ from $\mathcal{A}$ is the first (and only) query where $p_i(\boldsymbol{x}) = y$ and $C_{\ell_i}(\mathcal{Q}_i) = 1$ but $(\mathtt{value}, i, \boldsymbol{x})$ was not previously queried. By Lemma 1, the value $p_i(\boldsymbol{x})$ is uniformly random, thus $p_i(\boldsymbol{x}) = y$ with probability at most $1/(|\mathbb{F}| - j)$. The lemma above follows applying the union bound over all the simulation queries and noticing that, when $q_{\mathsf{sim}} = \mathsf{poly}(\lambda)$ then $q_{\mathsf{sim}} < |\mathbb{F}|/2$, and $q_{\mathsf{sim}}/(|\mathbb{F}| - q_{\mathsf{sim}}) = O(q_{\mathsf{sim}}/|\mathbb{F}|)$.

Let $\mathbf{H}_2$ be the same as the previous hybrid but where the oracle queries are handled differently. More in detail, we parse the polynomial $p_i(\boldsymbol{X}) = \sum X_\mu^j p_{i,j}(\boldsymbol{X})$ for any i where $p_{i,j} \in \mathbb{F}[X_1, \ldots, X_{\mu-1}]$, and where we recall that $(p_i)_{i\in[n]}$ are sampled by the wrapper-adversary $\mathcal{B}_{\mathsf{msk}}$, as described in Fig. 4. At oracle call from $\mathcal{A}$ with message $(\mathtt{proof}, i, \boldsymbol{x}, y)$, the hybrid computes the quotient polynomial $\mathsf{q}(X)$ in $\mathbb{G}_1[X]$ such that:

$$\sum [p_{i,j}(\boldsymbol{\beta})]_1 X^j = \mathsf{q}(X)(X - x_\mu) + \sum [p_{i,j}(\boldsymbol{\beta})]_1 x_\mu^j$$

Notice the hybrid can compute the polynomial division with some of the polynomials having coefficients in $\mathbb{G}_1$ because the divisor polynomial $(X - x_\mu)$ has coefficients in the field $\mathbb{F}$, and all the inversions needed are computed in the field. The hybrid, with the knowledge of the trapdoor β_μ, sets $\pi_\mu \leftarrow \mathsf{q}(\beta_\mu)$. Then, the hybrid computes $i_1, \ldots, \pi_{\mu-1}$ by running:

$$\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathsf{PST},\mu-1}].\mathsf{Prove}(\mathsf{ek}, (\textstyle\sum [p_{i,j}(\boldsymbol{\beta})]_1 x_\mu^j - y, \boldsymbol{x}_{:\mu-1}, 0), \sum p_{i,j}(\boldsymbol{X}) x_\mu^j - y)$$

where, the latter algorithm is the $(\mu-1)$-variate PST's prover for evaluation proofs. Notice that the two hybrids compute, in two equivalent but syntattically different ways, the same group element π_μ, thus the difference between the hybrids is only syntactical. This implies that $\varepsilon_1 = \varepsilon_2$.

Let $\mathbf{H}_3$ be the same as the previous hybrid but where instead of using the $(\mu - 1)$-variate prover we use the simulator $\mathcal{S}^{(\mu-1)}$. By induction hypothesis we have that $O(q_{\mathsf{sim}}(\mu - 1)/|\mathbb{F}|)$ is the statistical distance for the distributional zero-knowledge for $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathsf{PST},\mu-1}]$. It is easy to show that $|\varepsilon_3 - \varepsilon_2| \leq O(q_{\mathsf{sim}}(\mu - 1)/|\mathbb{F}|)$, the proof proceeds by a straight-forward reduction and is therefore omitted.

Let $\mathbf{H}_4$ be the same as the previous hybrid but it computes the component π_μ of the proofs differently. Specifically, at each oracle invocation with message $(\mathtt{proof}, i, \boldsymbol{x}, y)$ it computes:

$$\pi_\mu = \left(\mathsf{c}_i - [y]_1 - \sum_j [p_{i,j}(\boldsymbol{\beta})]_1 x_\mu^j\right)(\beta_\mu - x_\mu)^{-1}$$

Also for these two hybrids the difference is only syntactical as they compute the same value π_μ in two different ways. I particular, in $\mathbf{H}_3$ we compute π_μ as $\mathsf{q}(\beta_\mu)$ where:

$$\mathsf{q}(X) = \left(\sum_j [p_{i,j}(\boldsymbol{\beta})]_1 X^j - y - \sum_j [p_{i,j}(\boldsymbol{\beta})]_1 x_\mu^j \right) (X - x_\mu)^{-1}$$

and it is easy to check that $\mathsf{q}(\beta_\mu) = (\mathsf{c}_i - [y]_1 - \sum [p_{i,j}(\boldsymbol{\beta})]_1 x_\mu^j)(\beta_\mu - x_\mu)^{-1}$.

Let $\mathbf{H}_5$ be the same as the previous hybrid but at each oracle call with message $(\texttt{proof}, i, \boldsymbol{x}, y)$, if $C_{\ell_i}(\mathcal{Q}_i) = 1$ and $|\mathcal{Q}_i| \le t_i$, it computes:

$$\pi_\mu = (\mathsf{c}_i - [y]_1 - \mathsf{r})(\beta_\mu - x_\mu)^{-1}$$

and similarly computes $(\pi_j)_{j \in [\mu-1]} \leftarrow \mathcal{S}_{\mu-1}(\boldsymbol{\beta}, (\mathsf{r}, \boldsymbol{x}_{:\mu-1}, 0))$ for a random value r that we associate with the tuple (c_i, x_μ) (as the simulator does).

Lemma 3. $\epsilon_5 = \epsilon_4$.

Proof. Recall that q_{sim} is the number of simulation oracle queries made by $\mathcal{A}$. Also we assume w.l.g. that $\mathcal{A}$'s oracle queries are all distinct. We define sub-hybrid $\mathbf{G}_i$ to be equivalent to $\mathbf{H}_5$ for the first i queries to the simulation oracle, namely queries of the form $(\texttt{proof}, \cdot, \cdot, \cdot)$, and the sub-hybrid behaves as $\mathbf{H}_4$ for the remaining simulation oracle queries. Obviously, $\mathbf{G}_0$ is equivalent to $\mathbf{H}_4$ while $\mathbf{G}_{q_{\mathsf{sim}}}$ is equivalent to $\mathbf{H}_5$. Let ε'_i be the probability of $\mathbf{G}_i$ returning 1. We show that for any $i \in [q_{\mathsf{sim}}]$, $\varepsilon'_i = \varepsilon'_{i+1}$ which implies that $\varepsilon_4 = \varepsilon_5$.

Notice that we can focus on the indeces where the simulation query does not output $\perp$, otherwise the two consecutive hybrids are trivially equivalent. Thus w.l.g. we can assume that the i-th query to the simulation oracle involves the i-th polynomial and any of the queries outputs $\perp$. Let us focus then on the step from $\mathbf{G}_i$ and $\mathbf{G}_{i+1}$. Let the query be $(\texttt{proof}, i, \boldsymbol{x}, y)$ and assume, by the modification in $\mathbf{H}_1$, that $p_i(\boldsymbol{x}) = y$ and $(\texttt{value}, i, \boldsymbol{x})$ was previously queried as polynomial query and thus $\boldsymbol{x} \in \mathcal{Q}_i$. We can parse $\boldsymbol{x} = (x_j)_j$. First notice that if $x_\mu = \beta_\mu$ then the simulator and the prover behave exactly the same by setting π_μ to $[1]_1$. Thus we can additionally assume $x_\mu \neq \beta_\mu$.

Let $\boldsymbol{x}^* = (\beta_1, \ldots, \beta_{\mu-1}, x_\mu)$, recall that the difference between $\mathbf{G}_i$ and $\mathbf{G}_{i+1}$ is the value $p_i(\boldsymbol{x}^*)$ is substituted with an uniformly random element.

Thus, consider the list of points $(\boldsymbol{x}'_j)_j := (\boldsymbol{\beta}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-2}, \boldsymbol{x}^*)$ where $\mathcal{Q}_i = \{\boldsymbol{\beta}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-2}\}$. We need to show that the vector

$$\boldsymbol{p} = (p_i(\boldsymbol{x}'_j))_j$$

is uniformly random distribued over $\mathbb{F}^{t_i}$. Notice that we can write the vector $\boldsymbol{p} = \boldsymbol{f} + (\boldsymbol{r}, 0) + (\boldsymbol{0}, s)$ where:

$$\begin{aligned}
f_j &:= f_i(\boldsymbol{x}'_j) + \sum_{k \neq \ell_i} Z_k(x'_{j,k}) \cdot r_k(x'_{j,k}) && \text{if } j \le t_i - 1 \\
f_{t_i} &:= f_i(\boldsymbol{x}'_{t-1}) + \sum_{k \neq \mu-1} Z_k(x'_{t-1,\ell_i}) \cdot r_k(x'_{j,\ell_i}) && \text{else} \\
r_j &:= Z_{\ell_i}(x'_{j,\ell_i}) \cdot r_{\ell_i}(x'_{j,\ell_i}) \\
s &:= Z_{\mu-1}(x'_{j,\mu-1}) \cdot r_{\mu-1}
\end{aligned}$$

Notice that the vector $\boldsymbol{r}$ and the value s are independent random variables, since $\boldsymbol{r}$ depends only on the polynomial r_{ℓ_i} and s depends only on value $r_{\mu-1}$ (see Algorithm 1). We show that $\boldsymbol{r}$ is uniformly random distributed. First notice that $Z_{\ell_i}(x'_{j,\ell_i})$ is always non-zero since the checker C_{ℓ_i} accepted the list of points. Moreover, $\boldsymbol{r}$ consists of $t_i - 1$ evaluations on different points of a t_i-degree random polynomials, thus, by interpolation, the vector $\boldsymbol{r}$ is uniformly random. Noticing that $Z_{\mu-1}(x'_{j,\mu-1})$ is non-zero, with the same argument, we can see that s is uniformly random, and thus $\boldsymbol{p}$ is uniformly random.

Let $\mathbf{H}_6$ be equivalent to $\mathbf{H}_5$ but where we remove the check on the simulation queries introduced in $\mathbf{H}_1$. Since we roll-back the modification we can easily assert that $|\varepsilon_6 - \varepsilon_5| \leq O(q_{\mathsf{sim}}/|\mathbb{F}|)$. Finally, it is straight forward to check that $\mathbf{H}_6$ is equivalent to the ideal world for the distributional zero-knowledge experiment. Summing up the differences we have $|\varepsilon_6 - \varepsilon_0| \leq O(q_{\mathsf{sim}}/|\mathbb{F}|) + O(q_{\mathsf{sim}}(\mu - 1)/|\mathbb{F}|) = O(q_{\mathsf{sim}}\mu/|\mathbb{F}|)$.

5 Controlled-Malleability of PST and KZG

In Definition 9, the adversary must produce a forgery for $\mathcal{R}$ in the presence of a simulator for $\mathcal{R}_S$. In this way, the definition *decouples* the zero-knowledge simulator from the NIZK scheme. Of course, our ultimate goal is to prove security when the simulator is exactly the one guaranteed by the zero-knowledge property of the NIZK scheme (so that $\mathcal{R}_S = \mathcal{R}$). Nevertheless, in our setting it is much easier to analyze the simulation extractability of $\Pi_{\mathsf{PST},\mu}$ (the generalized PST) in the presence of the zero-knowledge simulator of $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathsf{PST},\mu}]$ (the single PST evaluation scheme). The motivation is that the generalized PST belongs to a broader class of CP-SNARKs, which we call PST-based CP-SNARKs, where the prover, the verifier, and most importantly the simulator can all be seen as algorithms that internally invoke the PST evaluation scheme one or multiple times. Informally:

– A CP-SNARK $\Pi[\mathsf{CS}_{\mathsf{PST}}]$ for $\mathcal{R}$ is called PST-based if both its prover and its verifier internally invoke $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathsf{PST}}]$. One can view this as a *reduction of knowledge* [43] from $\mathcal{R}$ to $\mathcal{R}^k_{\mathtt{evl}}$ for some $k \in \mathbb{N}$.
– A CP-SNARK $\Pi[\mathsf{CS}_{\mathsf{PST}}]$ is called *algebraic* if, for any instance $\mathbb{x}$, the commitments in the derived instances for $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathsf{PST}}]$ are linear combinations of the group elements in the original instance $\mathbb{x}$.
– A CP-SNARK $\Pi[\mathsf{CS}_{\mathsf{PST}}]$ is called *simulation-friendly* if the simulator only makes simple queries of the form $(\mathsf{c}, \boldsymbol{x}, y)$, where c is a commitment from the original instance $\mathbb{x}$, to the distributional zero-knowledge simulator of $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathsf{PST}}]$.

For example, Π_{PST} is a PST-based CP-SNARK, since its prover and verifier simply invoke $\Pi_{\mathtt{evl}}$. It is algebraic because the call is on $\mathbb{x}' = \sum_j a_j(\boldsymbol{z})\mathsf{c}_j$, and it is straightforward to show that it is simulation-friendly.

Definition 12 (PST-based CP-SNARK). *We say that Π is PST-based if Π is a CP-SNARK for some relation $\mathcal{R}$ and for the commitment scheme $\mathsf{CS}_{\mathsf{PST},\mu}$ the verifier (resp. the prover) is a two step algorithm. Namely, we can parse the verifier (resp. the prover) as a subrutine V' (resp. the subrutine P') and the algorithm that:*

– *Parse* $\pi = \pi_0, (\pi_j)_{j\in[k]}$
– $(\mathbb{x}_j)_{j\in[k]} \leftarrow \mathsf{V}(\mathsf{vk}, \mathbb{x}, \pi_0)$ *(resp. the prover computes* $(\mathbb{x}_j)_{j\in[k]}, (\mathbb{w}_j)_{j\in[k]} \leftarrow \mathsf{P}(\mathsf{ek}, \mathbb{x}, \mathbb{w})$*).*
– *Return* $\forall j \in [1, k] : \Pi_{\mathtt{evl}}.\mathsf{V}(\mathsf{vk}, \mathbb{x}_j, \pi_j)$ *(resp. the prover computes* $\forall j \in [1, k] : \Pi_{\mathtt{evl}}.\mathsf{P}(\mathsf{ek}, \mathbb{x}_j, \mathbb{w}_j)$*).*

We say that a PST-based CP-SNARK is algebraic *if for any j, the PST-commitment $\mathbb{x}_j.\mathsf{c}$ can be derived as a linear combination of the group elements in $\mathbb{x}$. We say that a PST-based CP-SNARK is* simulation friendly *if there exists a PT algorithm $\bar{\mathcal{S}}$ such that the following procedure defines a (distributional) zero-knowledge simulator for Π:*

- *Let $(\mathsf{c}_j)_j$ the commitments in the instance $\mathbb{x}$.*
- *Run $\pi_0, (\boldsymbol{b}_i, \boldsymbol{x}_i, \boldsymbol{y}_i)_i, \mathsf{st}_{\mathcal{S}} \leftarrow \bar{\mathcal{S}}(\mathbb{x}, \mathsf{st}_{\mathcal{S}})$*
- *For any i and any j query $(\mathsf{c}_j, \boldsymbol{x}_i, y_{i,j})$ to $\mathcal{S}^{(\mu)}$ and receives $\pi_{i,j}$.*
- *Set*[10] $\pi_i \leftarrow \sum_j b_{i,j}\pi_{i,j}$
- *Output $(\pi_j)_{j\in[0,k]}$.*

The lemma below summarizes our strategy. It shows that, in order to prove controlled malleability (resp. simulation extractability) for any PST-based algebraic and simulation-friendly CP-SNARK $\Pi[\mathsf{CS}_{\mathtt{PST},\mu}]$, it suffices to analyze a controlled-malleability (resp. simulation-extractability) game where the adversary has oracle access to the simulator of $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathtt{PST},\mu}]$. The resulting scheme then achieves controlled malleability (resp. simulation extractability) with its own simulator, up to minor adjustments needed to align the type checking of transformations and policies.

Lemma 4. *Let $\mathcal{S}^{(\mu)}$ be a simulator for $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathtt{PST},\mu}]$. For any PST-based Π for $\mathcal{R}$ that is algebraic and simulation-friendly, any extractor $\mathcal{E}$, any policy Φ, any set of transformations $\mathcal{T}$ from $\mathcal{R}_{\mathtt{evl}}$ to $\mathcal{R}$, and any adversary $\mathcal{A}$, there exist an allowable set of transformations $\mathcal{T}'$, an extractor $\mathcal{E}'$, an adversary $\mathcal{B}$ with black-box access to $\mathcal{A}$, and a policy Φ' such that $\Phi'((\mathbb{x},\pi),\mathsf{view},\mathsf{aux}_\Phi) := \Phi((\mathbb{x},\pi),\mathsf{view}_{\mathcal{A}},\mathsf{aux}_\Phi)$ where $\mathsf{view}_{\mathcal{A}}$ is the view provided to $\mathcal{A}$ by $\mathcal{B}$, and*

$$\mathbf{Adv}^{(\Phi,\mathcal{T}')\text{-}\mathtt{cm}}_{\Pi[\mathsf{CS}_{\mathtt{PST},\mu}],\mathcal{A},\mathcal{S},\mathcal{E}'}(\lambda) \leq \mathbf{Adv}^{(\Phi',\mathcal{T})\text{-}\mathtt{cm}}_{\Pi[\mathsf{CS}_{\mathtt{PST},\mu}],\mathcal{B},\mathcal{S}^{(\mu)},\mathcal{E}}(\lambda),$$

where $\mathcal{S}$ is a zero-knowledge simulator for Π. Moreover, if $\mathcal{T} = \emptyset$ then $\mathcal{T}' = \emptyset$ and $\mathcal{E} = \mathcal{E}'$.

Proof. We define $\mathcal{T}'$ as the largest allowable set containing transformations (T'_x, T'_w) such that there exist $k, s, l \in \mathbb{N}$ with $k = s + l$, group elements $(\bar{\mathsf{c}}_j)_{j\in[s+1,k]}$ a k-ary transformation $(T_x, T_w) \in \mathcal{T}$, vectors $\boldsymbol{x}_j \in \mathbb{F}^\mu$ and values $y_i \in \mathbb{F}$ for $j \in [k]$ that are efficiently computable from the input of T_x, such that $T_x((\mathbb{x}_j)_{j\in[k]})$ finds the $\mathbb{G}_1$-elements in $(\mathbb{x}_j)_{j\in[k]}$, let them be $(\bar{\mathsf{c}}_j)_{j\in[s]}$, then applies a $T_x((\bar{\mathsf{c}}_j, \boldsymbol{x}_i, y_i)_{i\in[k]})$.

Roughly speaking, $\mathcal{T}'$ is the set of transformations that apply the transformation in $\mathcal{T}$ to the PST commitments in the instances. We constraint $\mathcal{T}'$ to be an allowable set, thus by definition, the transformation T'_w in $\mathcal{T}'$ can define polynomials $(\bar{c}_j(\boldsymbol{X}))j \in [s]$ and $[\bar{c}_j(\boldsymbol{\beta})]_1 = \bar{\mathsf{c}}_j$ from the witness $(\mathbb{w}_j)_{j\in[k']}$ and we have $\bar{c}_j(\boldsymbol{x}_j) = y_j$ for any j.

Now we can define the reduction $\mathcal{B}$. Let q be the number of oracle queries of $\mathcal{A}$ and let $n' = n + q\cdot k$ where k is the number of PST evaluation proof performed by a call of the Π prover.

```
Reduction $\mathcal{B}(\mathsf{ck}, \mathsf{coms}_B)$:
 – Run $\mathcal{A}(\mathsf{ck}, \mathsf{coms}_A)$.
 – On the $i'$-th query to the simulation oracle with message $\mathbb{x}$ from $\mathcal{A}$run the zero-knowledge
   simulator of $\Pi$, namely:
     • Run $\pi_0, (\boldsymbol{b}_i, \boldsymbol{x}_i, \boldsymbol{y}_i)_i \leftarrow \bar{\mathcal{S}}(\mathbb{x}, (\mathsf{c}_{i',j})_j, \mathsf{st}_{\mathcal{S}})$.
     • For any $i$ and any $j$ query $(\mathsf{c}_j, \boldsymbol{x}_i, y_{i,j})$ to $\mathcal{S}^{(\mu)}$ and receives $\pi_{i,j}$.
```

[10] Whose instances are $(\tilde{\mathsf{c}}_i, \boldsymbol{x}_i, y_i)$ with $\tilde{\mathsf{c}}_i \leftarrow \boldsymbol{b}_i^T(\mathsf{c}_j)_j$ and $y_i \leftarrow \boldsymbol{b}_i^T\boldsymbol{y}_i$.

  - Set $\pi_i \leftarrow \sum_j b_{i,j}\pi_{i,j}$.
  - Return $(\pi_j)_{j\in[0,k]}$ to $\mathcal{A}$.
- Eventually, return the same forgery as $\mathcal{A}$ does.

We define the extractor $\mathcal{E}'$ that runs $(\mathbb{w}, T, (\mathbb{x}_i)_{i\in[k']}) \leftarrow \mathcal{E}(\mathsf{view}, \mathsf{st}_{\mathcal{S}}, \mathsf{aux}_{\mathcal{E}})$ and returns $(\mathbb{w}, T', (\mathbb{x}'_i)_{i\in\mathcal{I}})$ where $T' = (T'_x, T'_w)$ and the $(\mathbb{x}'_i)_i$ are the queries of $\mathcal{A}$, as follow:

- Define $\mathcal{I}$ as the set of indexes, with $i' \in \mathcal{I}$ iff there exists index i and $\mathbb{x}_i$ was queried by $\mathcal{B}$ at the i'-th the simulation oracle query of $\mathcal{A}$.
- The transformation $T'_x((\mathbb{x}'_i)_{i\in\mathcal{I}})$ has hard-coded, in an efficiently decodable manner, the values $\boldsymbol{x}_i, \boldsymbol{y}_i$ derived from the proofs and outputs $T_x((\mathbb{x}_i)_{i\in[k']})$ where $(\mathbb{x}_i)_i$ can be computed using the hardcoded values and the inputs $\mathbb{x}'_i$.

The associated transformation T'_w has hard-coded the same values of T'_x and can derive the witnesses $(c_j)_j$ polynomials associated with the commitments in the tuple of instances $(\mathbb{x}'_i)_i$ and outputs $T_w((c_j)_j)$.

It is not hard to see that if $\mathcal{A}$ wins the $\varPhi$-simulation $\mathcal{T}'$-controlled-malleability experiment then the forgery is also a valid forgery for $\mathcal{B}$. In fact, by construction, $\mathcal{B}$ passes the constraints of $\varPhi'$, moreover, either $\mathbb{w}$ is not a valid witness, and this would be the same in both experiments, or $T' \notin \mathcal{T}'$ but $T \in \mathcal{T}$, however this cannot happen by construction of T'. □

Simplified Adversaries and Algebraic Consistency. We define an algebraic adversary for a PST-based CP-SNARK as *simplified* if the commitments in its simulation oracle queries consist only of elements from either the set of simulated commitments or the set of simulated proofs that we denote with coms and proofs respectively.[11]

Although this class of adversaries is strictly more restrictive than those considered in [26], it is sufficient for our goal of proving that multivariate PIOP-based zkSNARKs are simulation extractable.

Faonio *et al.* [25] identify a necessary property for extractability in the presence of a simulation oracle for any KZG-based SNARK. This stems from a generalization of a simple attack: given a commitment c, an adversary with two simulated KZG evaluation proofs π_1, π_2 for the same evaluation point x but for two different evaluation values y_1 and y_2, can exploit KZG's homomorphism to forge for the statement $((\alpha+\beta)\mathsf{c}, x, \alpha y_1 + \beta y_2)$ by using the evaluation proof $\alpha\pi_1 + \beta\pi_2$. This attack can be generalized whenever the adversary can leverage *algebraic inconsistencies* provided by simulated proofs, as we explain hereafter. Let $\boldsymbol{A} \in \mathbb{F}[\boldsymbol{X}]^{m\times n}$, and let $\boldsymbol{b} \in \mathbb{F}[\boldsymbol{X}]^m$. We have that $(\boldsymbol{A}\|\boldsymbol{b})$ describes a linear system of multivariate polynomial equations that admits a solution if there exists a vector $\boldsymbol{z} \in (\mathbb{F}[\boldsymbol{X}])^n$ such that $\boldsymbol{A}\cdot\boldsymbol{z} = \boldsymbol{b}$.

Definition 13 (Algebraic Consistency, simplified). *Let $\mathcal{A}$ be a simplified adversary that has oracle access to a simulator $\mathcal{S}^{(\mu)}$ for $\varPi_{\mathsf{evl}}[\mathsf{CS}_{\mathsf{PST},\mu}]$ and receives as input a commitment key for $\mathsf{CS}_{\mathsf{PST},\mu}$ and a list of $\mathbb{G}_1$-elements $\mathsf{coms} = (\mathsf{c}_i)_{i\in[n]}$, for some n. We say that the view of $\mathcal{A}$ after interacting with $\mathcal{S}^{(\mu)}$ is* algebraic consistent *if the linear system of multivariate polynomial equations, that we describe next, admits a solution.*

[11] In fact, in Section 5.2, we further restrict our attention to an even simpler class of adversaries whose queries include only elements from coms. However, for technical reasons, particularly in the context of KZG evaluation proofs, we require the more general notion defined above.

Let $\mathsf{proofs} := (\pi_k)_k$ *be the list of simulated proofs, where* $\pi_k = (\mathsf{q}_{k,j})_j$*. We assign to each* c_k *in* view *a formal variable (defining a polynomial)* Z_k*; similarly we assign to the simulated proofs formal variables (defining polynomials)* $Q_{k,j} \in \mathbb{F}_{\leq d}[\boldsymbol{X}]$*. For each simulation query* $(\mathsf{g}, \boldsymbol{x}, y)$ *we define a new equation:*

$$\begin{aligned} Z_j - y &= \textstyle\sum_i Q_{k,i} \cdot (X_i - x_i) && \textit{if } \mathsf{g} = \mathsf{c}_j \\ Q_{k',j} - y &= \textstyle\sum_i Q_{k,i} \cdot (X_i - x_i) && \textit{if } \mathsf{g} = \mathsf{q}_{k',j} \end{aligned}$$

Roughly, the idea is that a view is algebraic consistent if, in a symbolic world, we can assign polynomials to the simulated commitments in a way that is coherent with all the claims, expressed as polynomial equations, generated by the simulator.

Definition 14 (Simplified Policy for PST-based CP-SNARK). *We say that a policy* Φ *is a* simplified policy *for a PST-based CP-SNARK for* $\mathcal{R}$ *if* Φ *returns true then (i) the view is algebraic consistent and (ii) let* $\mathcal{X}$ *be the set of all the* $\mathbb{G}_1$*-group elements in the instances queried to the simulation oracle, then* $\mathcal{X} \subseteq \mathsf{coms} \cup \mathsf{proofs}$*.*

5.1 Controlled-Malleability of KZG

Faonio *et al.* [26] showed a policy-based simulation-extractability of KZG when the evaluation point in the forgery never appears in any of the previous simulation query. However, for our reduction to PST, we need to handle forgeries that reuse the same evaluation point of previous simulation oracle queries.

We consider the admissible transformations $\mathcal{T}_{\mathtt{LH}}$ with source relation $\mathcal{R}_{\mathtt{evl}}$ and target relation $\mathcal{R}_{\mathtt{geval}}$ for the controlled-malleability of KZG/PST (cf. Eq. (1)). Briefly, with each transformation $T \in \mathcal{T}_{\mathtt{LH}}$ we associate a set of multivariate polynomials $f_j(\boldsymbol{X}, \boldsymbol{Y})$, where the variables $\boldsymbol{Y}$ correspond to the simulated commitments observed by the adversary. The component T_x plays an additional role: it ensures that malleability only applies when these simulated commitments were involved in simulation queries at the evaluation point $\boldsymbol{x}(= \boldsymbol{x}_1)$.

We additionally assume there exists an efficient procedure that upon input $T \in \mathcal{T}_{\mathtt{LH}}$ outputs the values $\boldsymbol{z}, (f_j)_{j \in [m]}$. Note the slight abuse of notation in Eq. (1): for a polynomial p, we define $p(\mathsf{ck}, \boldsymbol{Y}) = [p(\boldsymbol{\beta}, \boldsymbol{Y})]_1$.

$$T \in \mathcal{T}_{\mathtt{LH}} \iff \begin{cases} \exists (a_j)_{j \in [m]}, \boldsymbol{z}, (f_j)_{j \in [m]} \text{ s.t. } f_j \in \mathbb{F}[\boldsymbol{X}, Y_1, \ldots, Y_k] : \\ \forall (\mathbb{x}_i)_{i \in [k]} = (\mathsf{c}_i, \boldsymbol{x}_i, y_i)_{i \in [k]} : \\ \quad \textbf{if } \exists i, j : \boldsymbol{x}_j \neq \boldsymbol{x}_i \Rightarrow T_x(\mathsf{ck}, (\mathbb{x}_i)_{i \in [k]}) \to \bot \\ \quad \textbf{else } T_x(\mathsf{ck}, (\mathbb{x}_j)_{j \in [k]}) \to ((\bar{\mathsf{c}}_j)_{j \in [m]}, \boldsymbol{z}, \boldsymbol{x}_1, y) \text{ s.t.} \\ \quad\quad \forall j \in [m] : \bar{\mathsf{c}}_j \leftarrow f_j(\mathsf{ck}, (\mathsf{c}_i)) \\ \quad\quad y \leftarrow \sum_i a_i(\boldsymbol{z}, x_1) f_i(\boldsymbol{x}, (y_j)_{j \in [k]}), \\ \quad (f_j(\boldsymbol{X}, c_1, \ldots, c_k))_{j \in [m]} = T_w((c_i(\boldsymbol{X}))_{i \in [k]}), \end{cases} \tag{1}$$

We show that the set of transformations $\mathcal{T}_{\mathtt{LH}}$ is allowable. In fact, given $\forall i \in [k] : (\mathsf{c}_i, \boldsymbol{x}, y_i), c_i \in \mathcal{R}_{\mathtt{evl}}$ we have that for any j the witness $\bar{c}_j(\boldsymbol{X}) := f_j(\boldsymbol{X}, (c_i(\boldsymbol{X}))_{i \in [m]}) \in \mathbb{F}[\boldsymbol{X}]$ and $\bar{\mathsf{c}}_i = f_j(\mathsf{ck}, (\mathsf{c}_i)_i) =$

$f_j(\mathsf{ck}, ([c_i(\boldsymbol{\beta})]_1)) = [f_j(\boldsymbol{\beta}, (c_i(\boldsymbol{\beta})]_1 = [\bar{c}_j(\boldsymbol{\beta})]_1$. Moreover:

$$\begin{aligned}\sum_j a_j(\boldsymbol{z}, x_1) \cdot \bar{c}_j(\boldsymbol{x}) &= \sum_j a_j(\boldsymbol{z}, x_1) \cdot f_j(\boldsymbol{x}, (c_i(\boldsymbol{x}))_{i \in [k]}) \\ &= \sum_j a_j(\boldsymbol{z}, x_1) \cdot f_j(\boldsymbol{x}, (y_i)_{i \in [k]}) \\ &= y\end{aligned}$$

Nesting Level. To prove our result on KZG, we first recall the notion of *nesting level* introduced in [26]. Informally, it captures a minimal upper bound on the degrees of the denominators of rational functions associated with simulated proofs. The term "nesting level" reflects the recursive structure that can arise in composable proof systems such as CP-SNARKs. In particular, since KZG commitments and KZG proofs belong to the same algebraic domain, it is possible for an instance of a CP-SNARK to contain commitments that are themselves (simulated) proofs. When a proof is computed over such an instance, it effectively becomes a *proof of a proof*, resulting in a form of nesting. The nesting level thus measures how deeply such recursive structure can occurin the simulation.

In our context, to reduce the controlled malleability of PST to that of KZG, it is crucial to ensure that the nesting level does not increase. This restriction is not merely technical: the soundness of our reduction relies on an inductive argument over the number of variables. If the nesting level were allowed to increase, we would not be able to apply the inductive hypothesis, and the reduction would no longer go through.

Looking ahead, the reduction involves a sequence of simulation oracle queries. While some of these queries may increase the overall maximum nesting level, as originally formulated by [26], we observe that such increases cannot be effectively exploited by the adversary.

To make this precise, we slightly modify the original definition of nesting level. Specifically, we define the *maximum nesting level* by considering only simulated proofs that are generated *before* the forgery instance is fixed. This change reflects that queries made after the forgery point may aid the reduction but cannot affect the adversary's ability to forge, and is key to making the reduction go through: although the original definition would not suffice, our refined version correctly captures the adversary's limitations while preserving the structure needed for the reduction.

Definition 15 (Maximum Nesting Level for KZG, revisited). *Let $\mathcal{S}^{(1)}$ be the zero-knowledge simulator of KZG. Let* view *be a view of an adversary $\mathcal{A}$ interacting with the simulation-extractability game with Π_{KZG}. Let $\mathcal{Q} \subset \mathcal{Q}_{\mathsf{sim}}$, where the latter set contains all the simulation oracle queries to $\mathcal{S}^{(1)}$ issued by $\mathcal{A}$. Let $\nu_{\mathsf{g},x,\mathcal{Q}}$ for $\mathsf{g} \in \mathbb{G}_1$ and $x \in \mathbb{F}$ be such that:*

- *For any $\mathsf{c} \in \mathsf{coms}$ and for any $x \in \mathbb{F}$, $\nu_{\mathsf{c},x,\mathcal{Q}} = 0$.*
- *For any π such that $((\mathsf{c}, x, y), \pi) \in \mathcal{Q}$, we have $\nu_{\pi,x,\mathcal{Q}} = \nu_{\mathsf{c},x,\mathcal{Q}} + 1$.*

We define $\bar{\nu}_{\mathcal{Q}}$ be the maximum nesting level *of $\mathcal{Q}$ as:*

$$\bar{\nu}_{\mathcal{Q}} := \max_{\mathsf{g} \in \mathsf{proofs} \cup \mathsf{coms}} \sum_{x \in \mathbb{F}} \nu_{\mathsf{g},x,\mathcal{Q}}.$$

Let $\tilde{x}$ be the evaluation point in the forgery of the adversary, and consider the set $\tilde{\mathcal{Q}}$ containing all the queries to the simulation oracle made by $\mathcal{A}$ before the random oracle query with output $\tilde{x}$. We define the maximum nesting level of $\mathcal{A}$'s forgery to be $\bar{\nu} := \bar{\nu}_{\tilde{\mathcal{Q}}}$.

Finally, we present our simplified policy for KZG, which relies on two key technical notions. First, to characterize the class of index polynomials for which we can prove controlled malleability, we introduce the notion of *ν-admissibility.* Intuitively, this condition ensures a form of structured (and possibly high degree ν) independence among the polynomials involved. Second, following the approach of prior work, we define when a forgery point is considered *valid.* We refer to this condition as the *generalized hash-check.* In particular, it requires that the evaluation points in the forgery are sampled, using the random oracle, *after* a virtual representation of the commitments (in the forgery) have been fixed.

Definition 16. *Let $\boldsymbol{a} := (a_j)_{j\in[m]}$ be a list of polynomials in $\mathbb{F}[X]$ and $\nu \in \mathbb{N}$. We say that $\boldsymbol{a}$ are ν-independent polynomials if $\forall(\alpha_j)_j \in \mathbb{F}_{\leq\nu}[X]$: either $\sum_j \alpha_j A_j(X) \neq 0$ or $\forall j : \alpha_j = 0$. Moreover, let $\boldsymbol{a} := (a_j)_{j\in[m]}$ be m polynomials in $\mathbb{F}[\boldsymbol{Z}, X]$. We say that $\boldsymbol{a}$ is ν-admissible if either $\forall j : a_j \in \mathbb{F}[\boldsymbol{Z}]$ (namely, $\mathtt{deg}_X(a_j) = 0$) and they are linearly independent or $\forall j : a_j \in \mathbb{F}[X]$ and they are ν-independent.*

Definition 17 (Generalized hash-check policy for KZG). *For any $\nu \in \mathbb{N}$, and any index $\mathbb{i} = (a_j)_{j\in[m]}$ of $\mathcal{R}_{\mathtt{geval}}$, let $\Phi^{\nu}_{\mathsf{gHC},\mathbb{i}}$ be a simplified policy that, upon input the forgery $(\mathbb{i}^*, \mathbb{x}^*, \pi^*)$, the view view and the auxiliary output aux_{Φ}, parses $\mathbb{x}^* = ((\mathsf{c}^*_i)_i, \boldsymbol{z}^*, x^*, y^*)$, and returns 1 if and only if:*

- *The maximum nesting level of the $\mathcal{A}$'s forgery is s.t. $\bar{\nu} \leq \nu$ and $\mathbb{i} = \mathbb{i}^*$,*
- *Either (1) $\forall i : a_i \in \mathbb{F}[\boldsymbol{Z}]$ and $(\mathsf{c}^*_i)_i \rightarrow_{\mathsf{RO}} \boldsymbol{z}^*$ and $(\sum_i a_i(\boldsymbol{z})\mathsf{c}^*_i) \rightarrow_{\mathsf{RO}} x^*$ or (2) $\forall i : a_i \in \mathbb{F}[X]$ and $(\mathsf{c}^*_i)_i \rightarrow_{\mathsf{RO}} x^*$, we call this condition the generalized hash check.*

We define the family of policies $\boldsymbol{\Phi}^{\nu}_{\mathsf{gHC}} := \{\Phi^{\nu}_{\mathsf{gHC},\mathbb{i}} : \mathbb{i}$ are ν-admissible$\}$.

Theorem 2 (Controlled-Malleability of KZG). *For any $\nu \in \mathbb{N}$, the scheme $\Pi_{\mathtt{KZG}}$ is $\boldsymbol{\Phi}^{\nu}_{\mathsf{gHC}}$-simulation $\mathcal{T}_{\mathtt{LH}}$-controlled-malleable in the AGM under the OMSDH assumption.*

Proof Ideas. The proof of the theorem closely follows the proof of policy-based simulation extractability in [26]. The main difference is that we must define an extractor that additionally extracts a transformation whenever the forged point $\bar{x}$ lies in the set $\mathcal{Q}_x$ of points queried by the adversary. Using the AGM, we can interpret the representation of the forged commitment provided by the adversary as such a transformation.

The first part of the proof shows that whenever the extracted transformation is invalid, we can construct a new point $x^* \notin \mathcal{Q}_x$ for which we obtain a valid simulation-extractability forgery. In particular, when the transformation is invalid, we can use the simulated material to construct a proof π for the instance $(\bar{\mathsf{c}}, \bar{x}, y')$, where the commitment and evaluation point coincide with those of the forgery, but y' is the correct evaluation obtained by applying the extracted transformation and the claims of the simulated statements. It then follows that, for any arbitrary point x^*, the forged proof $(\bar{\pi} - \pi)/(\bar{x} - x^*)$ constitutes a valid proof that the commitment $\mathsf{c}^* = \bar{\pi} - \pi$ evaluates to $(\bar{y} - y')/(\bar{x} - x^*)$ at the point x^*. Thus, we reduce controlled malleability for $\boldsymbol{\Phi}^{\nu}_{\mathsf{gHC}}$ of KZG to its simulation extractability for the same policy.

What remains is to show that simulation extractability holds under this refined policy, which can be proved with only a minor modification of the original proof.

Proof. The proof follows closely the proof of [27, Theorem 1, pag 16]. For completeness, we give the details.

By the definition of algebraic adversary (cf. Definition 2) for each group element output, $\mathcal{A}$ additionally attaches a representation $(f, \boldsymbol{r})$ of such a group element with respect to all the elements seen during the experiment (included elements in coms and the simulated proofs proofs). In particular, we assume that for each query $(\mathbb{x}, \mathsf{aux})$ to the oracle $\mathcal{S}_1$, we can parse the value aux as $((f_i, \boldsymbol{r}_i)_i, \mathsf{aux}')$, where $(f_i, \boldsymbol{r}_i)$ is a valid representation for the commitment $\mathbb{x}.\mathsf{c}_i$ with respect to the basis $([\beta^i]_1)_i, \mathsf{coms}, \mathsf{proofs}$, i.e., $\mathbb{x}.\mathsf{c}_i = [f_i(\beta)]_1 + \boldsymbol{r}_i^T(\mathsf{coms}\|\mathsf{proofs})$.

Let $((a_j)_{j\in[m]}, (\tilde{\mathsf{c}}_j)_{j\in[m]}, \tilde{\boldsymbol{z}}, \tilde{x}, y)$ be the forgery of $\mathcal{A}$, and let $(\tilde{f}_i, \tilde{\boldsymbol{r}})$ be the algebraic representation of $\tilde{\mathsf{c}}_i$ for any i. We can parse $\tilde{\boldsymbol{r}}_i = \tilde{\boldsymbol{c}}_i \| \tilde{\boldsymbol{o}}_i$, where $\tilde{\boldsymbol{r}}_i^T \cdot (\mathsf{coms}\|\mathsf{proofs}) = \tilde{\boldsymbol{c}}_i^T \cdot \mathsf{coms} + \tilde{\boldsymbol{o}}_i^T \cdot \mathsf{proofs}$. We define our extractor $\mathcal{E}(\mathsf{view}, \mathsf{st}_{\mathcal{S}}, \mathsf{aux}_{\mathcal{E}})$ that:

1. If $\tilde{x} \in \mathcal{Q}_x$, then from $(\tilde{f}_i, \tilde{\mathbf{c}}_i, \tilde{\boldsymbol{o}}_i)_{i\in[m]}$ and from $\mathcal{Q}_{\mathsf{sim}}$ it defines a transformation T, by defining the set of polynomials $\tilde{f}_i(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{P}) := f_i(\boldsymbol{X}) + \sum_j \tilde{c}_{i,j} \cdot Y_j + \tilde{o}_{i,j} \cdot P_j$, for any i. For any i, j, it finds the query $\mathbb{x}_{i,j} = (\mathsf{c}_j, \tilde{x}, y_{i,j})$ for $\tilde{c}_{i,j} \neq 0$ and the query $\mathbb{x}'_{i,j} = (\pi_j, \tilde{x}, y'_{i,j})$ for $\tilde{p}_{i,j} \neq 0$ in the set $\mathcal{Q}_{\mathsf{sim}}$. The transformation maps $(\mathbb{x}_{i,j})_{i,j}, (\mathbb{x}'_{i,j})_{i,j}$ to:
   - For any i, $\tilde{\mathsf{c}}_i = \left[\tilde{f}_i(\beta)\right]_1 + \sum_j \tilde{c}_{i,j} \cdot \mathbb{x}_{i,j}.\mathsf{c} + \sum_j \tilde{o}_{i,j} \cdot \mathbb{x}'_{i,j}.\mathsf{c}$.
   - $\tilde{y} = \sum_i a_i(\tilde{\boldsymbol{z}}, \tilde{x}) \cdot (f_i(\tilde{x}) + \sum_j \tilde{c}_{i,j} y_{i,j} + \sum_j \tilde{o}_{i,j} y'_{i,j})$.

   It returns $(\mathbb{w} = \perp, T, (\mathbb{x}_{i,j}, \mathbb{x}'_{i,j})_{i,j})$.
2. Else it returns $(\mathbb{w} = (\tilde{f}_j)_{j\in[m]}, \perp, \perp)$.

We let $\mathbf{H}_0$ be the $\mathbf{Exp}_{\mathcal{A},\mathcal{S},\mathcal{E}}^{\Phi^\nu_{\mathtt{gHC,i}}, \mathcal{T}_{\mathtt{LH\text{-}cm}}}$ experiment, and we denote by $\epsilon_i := \Pr[\mathbf{H}_i = 1]$.

Hybrid $\mathbf{H}_1$. We set $\mathbf{H}_1$ to be the simulation-extractability game $\mathbf{Exp}_{\mathcal{A}',\mathcal{S},\mathcal{E}}^{\Phi^\nu_{\mathtt{gHC,i}}\text{-}se}$ with the *alternative* adversary $\mathcal{A}'$ defined below:

1. The alternative adversary runs $\mathcal{A}$ forwarding all its queries until $\mathcal{A}$ sends its forgery. Let $\bar{\mathbb{x}} := ((\bar{\mathsf{c}}_j)_{j\in[m]}, \bar{\boldsymbol{z}}, \bar{x}, \bar{y}), \bar{\pi}$ be its forgery, and let $\bar{\mathsf{c}} := \sum_k a_k(\bar{\boldsymbol{z}}, \bar{x})\bar{\mathsf{c}}_k$. Let $\mathcal{Q}_x$ be the set of points x for which the adversary queried the simulation oracle.
2. If $\bar{x} \in \mathcal{Q}_x$, namely, when the adversary made a simulation query with evaluation point set to $\bar{x}$:
   (a) If from the algebraic representations of $(\bar{\mathsf{c}}_j)_{j\in[m]}$ and the simulation oracle query we can define a valid transformation T (see the extractor, Item 1) then it aborts returning $\perp_E$.
   (b) Else, it queries $\mathbb{x}' := (\bar{\mathsf{c}}, \bar{x}, y')$, obtaining π, to the simulation oracle for a value y' that preserves the algebraic consistency. (Notice that $\mathcal{A}$ is allowed to make queries with the commitments in the instances that are in $\mathsf{coms} \cup \mathsf{proofs}$. However, given the algebraic representation $(\bar{f}, \bar{\boldsymbol{c}}, \bar{\boldsymbol{o}})$ of $\bar{\mathsf{c}}$, we can emulate the query on $\mathbb{x}$ by querying on $\tilde{x}$ all the commitments in coms (resp. proofs in proofs) such that the coefficients in $\bar{\boldsymbol{c}}$ (resp. in $\bar{\boldsymbol{o}}$) are not zero.)
3. Else it outputs $\bar{\mathbb{x}}, \bar{\pi}$.
4. It computes the forgery $\mathbb{x}^* := (\mathsf{c}^*, x^*, y^*), \pi^*$ (this forgery can easily be parsed as a forgery for $\Pi_{\mathtt{geval}}$), where:

$$\mathsf{c}^* \leftarrow (\bar{\pi} - \pi) \qquad \pi^* \leftarrow \frac{\bar{\pi} - \pi}{\bar{x} - x^*} \qquad y^* \leftarrow \frac{\bar{y} - y'}{\bar{x} - x^*}$$

the forgery point $x^* \leftarrow \mathsf{RO}(s)$, and s is a string never queried to the RO by $\mathcal{A}$ and containing c^* as substring, which yields $\mathsf{c}^* \rightarrow_{\mathsf{RO}} x^*$.

We show that, unless it occurs the bad event that $x^* \in \mathcal{Q}_x$, we have $\mathbf{H}_0 = 1 \iff \mathbf{H}_1 = 1$. We proceed by a case analysis.

- If $\tilde{x} \notin \mathcal{Q}_x$ then the forgery of $\mathcal{A}'$ and the forgery of $\mathcal{A}'$ are the same. Moreover, by definition of $\mathcal{E}$, the extractor outputs a witness in both experiments, so the transformation case in the controlled-malleability cannot be triggered.
- If $\tilde{x} \in \mathcal{Q}_x$ and $\mathcal{A}'$ outputs $\perp_E$, then by definition the extractor $\mathcal{E}$ in the controlled malleability experiment would output a valid transformation, therefore $\mathbf{H}_0 = 0$. Moreover, since the adversary $\mathcal{A}'$ fails to output a forgery, we have $\mathbf{H}_1 = 0$.
- If $\tilde{x} \in \mathcal{Q}_x$ and $\mathcal{A}'$ does not output $\perp_E$, then the forgery of the adversary $\mathcal{A}'$ is valid whenever the forgery of $\mathcal{A}$ is valid. First, we notice that, by the verification equation of KZG, $(\bar{\pi} - \pi)(s - \bar{x}) = [y' - \bar{y}]_1$. Thus:

$$\pi^*(s - x^*) = \tfrac{\bar{\pi}-\pi}{\bar{x}-x^*}(s - x^* + \bar{x} - \bar{x}) = \tfrac{\bar{\pi}-\pi}{\bar{x}-x^*}(-x^* + \bar{x}) + \tfrac{\bar{\pi}-\pi}{x^*-\bar{x}}(s - \bar{x}) = \mathsf{c}^* - [y^*]_1 \tag{2}$$

Finally, the probability of the bad event that $x^* \in \mathcal{Q}_x$ is at most $\frac{Q_{\mathsf{RO}}+1}{q}$, where Q_{RO} is the number of queries of $\mathcal{A}$ to the random oracle. We have that $\epsilon_1 \leq \epsilon_0 + \frac{Q_{\mathsf{RO}}+1}{q}$

The proof of [26] continues defining a series of hybrid to show that their adversary can be simplified, in substance turning an arbitrary adversary to the class of simplified adversary that we consider as in Definition 14. Because of our simplification, their sequence of hybrid can be avoided and we can focus on the next hybrid (which follows the stragegy of their fifth hybrid).

Hybrid $\mathbf{H}_2$. Let $((\tilde{\mathsf{c}}_j)_{j\in[m]}, \tilde{\boldsymbol{z}}, \tilde{x}, \tilde{y})$ be the forgery of $\mathcal{A}'$. Let $\tilde{c}_i(X, \boldsymbol{Y})$ be the rational function such that $\tilde{c}_i(\beta, \mathsf{coms}) = \tilde{\mathsf{c}}_i$ for any i, these rational functions can be defined from the algebraic representations by plugging the rational functions that each simulated proof defines as described in the definition of algebraic consistency in Definition 13. This hybrid is equal to $\mathbf{H}_1$ but it returns 0 if $\sum_i a(\tilde{\boldsymbol{z}}, \tilde{x})\tilde{c}_i(\tilde{x}, \mathsf{coms}) \neq y^*$.

Lemma 5. $\epsilon_2 \leq \epsilon_1 + \epsilon_{\mathsf{OMSDH}}$

The proof of this lemma is almost identical to [27, Lemma 3], thus we give just a proof sketch.

Proof. We show how to generate a forgery to the OMSDH assumption when $\sum_i a(\tilde{\boldsymbol{z}}, \tilde{x})\tilde{c}_i(\tilde{x}, \mathsf{coms}) \neq y^*$. The reduction gives the adversary the same SRS generated by the OMSDH challenger and samples value $r_i \leftarrow\$ \mathbb{F}$ such that $\mathsf{coms} = [\boldsymbol{r}]_1$. The oracle $\mathcal{O}_s$ allows the reduction to compute the proofs for any statement $\mathbb{x} := (\mathsf{c}, x, y)$, where c is in $\mathsf{coms} \cup \mathsf{proofs}$. In fact, for $\mathsf{c} = \mathsf{c}_i$ for some i, then we can query $\mathcal{O}_s$ on $(x, 1)$ to receive the element $\mathsf{p}_{1,x} := [\beta - x]_1^{-1}$, and set $\pi = (r_i - y) \cdot \mathsf{p}_{1,x}$. Otherwise, if $\mathsf{c} \in \mathsf{proofs}$, we need to query $\mathcal{O}_s$ on $(x, \nu_{\mathsf{c},x} + 1)$ to retrieve $\mathsf{p}_{\nu_{\mathsf{c}}+1,x} := [\beta - x]_1^{\nu_{\mathsf{c},x}+1}$ where we recall that $\nu_{\mathsf{c},x}$ is the nesting level for the commitment c on point x, and derive the proof π from the algebraic representation of c, from y and $\mathsf{p}_{\mathsf{c},x}$. In fact, as shown in [27, pag 17, Lemma 2], let rational function $\pi_j(X, Y_k)$ such that $\pi_j = [\pi_j(\beta, r_k)]_1$ for all $r \in \mathbb{F}$, $\pi(X, r)$ belongs to the span below:

$$\forall j : \pi_j(X, r) \in \mathsf{Span}(\left\{ \tfrac{1}{(X-x')^j} : j \leq \max_{\mathsf{g}} \nu_{\mathsf{g},x'}, x' \in \mathcal{Q}_x \right\} \cup \left\{ \tfrac{1}{(X-x)^{\nu_{\mathsf{c},x}+1,x}} \right\}) \tag{3}$$

Let $v(X) := \sum_i a_i(\tilde{\boldsymbol{z}}, \tilde{x})(\tilde{c}_i(x, \mathsf{coms})) - \tilde{y}$, and let $q(X), r$ be such that $v(X) = q(x) + r(X - \tilde{x})$. The reduction submits the forgery $(\tilde{x}, \mathsf{y})$, where $\mathsf{y} := r^{-1}(\tilde{\pi} - [q(\beta)]_1)$. This is a valid forgery because y is equal to $[(\beta - \tilde{x})^{-1}]_1$ and $\tilde{x}$ was never queried to $\mathcal{O}_s$ by the change introduced in $\mathbf{H}_1$.

Hybrid $\mathbf{H}_3$. Let $\mathbf{H}_3$ return 0 if $\exists i : \mathsf{deg}_{Y_i}(\tilde{c}(X, \boldsymbol{Y})) > 0$.

Lemma 6. $\epsilon_6 \leq \epsilon_5 + \epsilon_{\mathsf{DLOG}} + (\max_i \mathsf{deg}(a_i) + \bar{\nu})/|\mathbb{F}|$

This lemma follows almost exactly the same argument as in [27, Lemma 3, pag 19] with the crucial difference that, when applying the notion of maximum nesting level, we can indeed use our revisited notion that considers the maximum nesting level computed over all the proofs queried until the point $\tilde{x}$ was generated as output of the random oracle.

Proof. Let $\mathcal{Q}_j = \mathsf{coms} \cup \{\pi_1, \ldots, \pi_{j-1}\}$, Eq. (3) implies that, for any j, there exist polynomials $p_{j,0}$ and $p_{j,1}$ and an index $k \in [n]$ such that we can write the rational functions $\pi_j(X, Y_k)$ associated with the proof $\pi_j = \pi_j(\beta, \mathsf{c}_k)$ as:

$$\pi_j(X, Y_k) = \frac{p_{j,0}(X) + Y_k p_{j,1}(X)}{\prod_{x \in \mathcal{Q}_x}(X - x)^{\nu_{x,j}}}$$

where $\nu_{x,j} = \max_{\mathsf{g} \in \mathcal{Q}_j} \nu_{\mathsf{g},x}$ for any $x \in \mathcal{Q}_x$. Clearly, the degree of the denominator in the equation above is bounded by the nesting level of the set $\mathcal{Q}_j$.

Thus, there are rational functions $m_0, \ldots, m_n$, and an index j^*, such that we can define the polynomial $\hat{c}(\boldsymbol{Z}, X, \boldsymbol{Y}) = m_0(\boldsymbol{Z}, X) + \sum m_j(\boldsymbol{Z}, X) Y_j$ where:

$$m_0(\boldsymbol{Z}, X) := f(X) + \sum_i a_i(\boldsymbol{Z}, X) \underbrace{\left(\sum_{j \leq j^*} o_j \cdot \frac{p_{j,0}(X)}{\prod_{x \in \mathcal{Q}_x}(X - x)^{\nu_{x,j}}} \right)}_{m_{0,i}(X)}$$

$$m_j(\boldsymbol{Z}, X) := \sum_i a_i(\boldsymbol{Z}, X) \underbrace{\left(c_j + \sum_{k \in \mathcal{P}_j} o_k \cdot \frac{p_{k,1}(X)}{\prod_{x \in \mathcal{Q}_x}(X - x)^{\nu_{x,k}}} \right)}_{m_{j,i}(X)}, \ \forall j > 0$$

Where, for any j, the set $\mathcal{P}_j$ is the set of indexes such that $k \in \mathcal{P}_j$ iff the variable Y_j appears in the polynomial π_k and $k \leq j^*$. Clearly, $\hat{c}(\tilde{\boldsymbol{z}}, \tilde{x}, \boldsymbol{Y}) = \hat{c}(\tilde{x}, \boldsymbol{Y})$. Moreover, the index j^* is the index of the last simulation oracle query before the value $\tilde{x}$ is defined as an output of the random oracle (see Definition 15).

Let E_i be the distinguishing event that $\exists j : c_j \neq 0 \vee o_j \neq 0$. We show that:

$$\Pr[E_i] \leq \epsilon_{\mathsf{DLOG}} + (\bar{\nu} + \max_i \mathsf{deg}(a_i))/|\mathbb{F}|.$$

We first notice that there must exist indexes j' and i' such that $m_{j',i'}(X) \neq 0$. We show that also $m_{j'}(\boldsymbol{Z}, X) \neq 0$.

In fact, when $\forall i : a_i \in \mathbb{F}[\boldsymbol{Z}]$ then they are linearly independent, thus the polynomial $m_{j'}(\boldsymbol{Z}, X) \neq 0$. On the other hand, when $\forall i : a_i \in \mathbb{F}[X]$, then they are ν-independent for $\nu \geq \bar{\nu}$, where the latter is the maximum nesting level of the forgery. Notice that by the definition of the index j^* and by our definition of the maximum nesting level, we have that for any i the degree of the polynomial $\prod_{x \in \mathcal{Q}_x}(X - x)^{\nu_{x,j^*}} \cdot m_{j',i}(X)$ is smaller or equal to $\bar{\nu}$, the maximum nesting level and thus $m_{j'} \neq 0$.

Since $\tilde{c}_i(\tilde{x}, \mathsf{coms}) = [\tilde{y}]_1$ by the check introduced in the hybrid $\mathbf{H}_2$, and $\tilde{c}_i(\tilde{x}, \mathsf{coms}) = \hat{c}_i(\tilde{\boldsymbol{z}}, \tilde{x}, \mathsf{coms})$ by definition, we can reduce to DLOG as follows. The reduction generates the SRS and simulates using the trapdoor β, while the commitments coms are uniformly random group elements from the challenger. We set the coefficients $\mu_j \leftarrow m_j(\tilde{\boldsymbol{z}}, \tilde{x})$ and $\hat{y} = \tilde{y} - m_0(\tilde{\boldsymbol{z}}, \tilde{x})$. Notice that $\sum_j \mu_j \mathsf{c}_j = [\hat{y}]_1$, which implies a non-trivial DLOG relation (from which we can break the representation problem and therefore DLOG) if at least one coefficient μ_j is non-zero.

Given that the *generalized hash check* is satisfied, $\tilde{\mathsf{c}} \rightarrow_{\mathsf{RO}} \tilde{x}$ and $(\tilde{\mathsf{c}}_j)_{j\in[m]} \rightarrow_{\mathsf{RO}} \tilde{\boldsymbol{z}}$, which implies that $\tilde{\mathsf{c}}$ is a function of the coefficients $(c_j, o_j)_j$ and the polynomials $(a_j)_j$ that are fixed before $\tilde{x}$ and $\tilde{\boldsymbol{z}}$ are computed. By Schwartz-Zippel, we derive that the coefficient $\mu_{j'} = m_{j'}(\tilde{\boldsymbol{z}}, \tilde{x})$ is 0 only with negligible probability $\bar{\nu} + \max_i \mathsf{deg}(a_i)/|\mathbb{F}|$. $\square$

Hybrid $\mathbf{H}_4$. Let $\mathbf{H}_4$ return 0 if $\exists i, j : \mathsf{deg}_{Y_i}(\tilde{c}_j(X, \boldsymbol{Y})) > 0$.

Lemma 7. $\epsilon_4 \leq \epsilon_3 + \epsilon_{\mathsf{DLOG}} + (\max_i \mathsf{deg}(a_i) + \bar{\nu})/|\mathbb{F}|$

The proof of the lemma above proceeds similarly to that of the previous lemma, thus we simply sketch the proof. We notice that $\hat{c}(\boldsymbol{Z}, X) = \sum_i a_i(\boldsymbol{Z}, X) \cdot \tilde{c}_i(X, \mathsf{coms})$, because of the previous hybrid and by definition of $\hat{c}$. Therefore, if there exist indexes i' and j' such that the distinghuishing event holds, by the independence properties of $(a_j)_{j\in[m]}$, we have that $\sum_i a_i(\boldsymbol{Z}, X) \cdot \tilde{c}_i(X, \boldsymbol{Y}) \neq 0$. Thus, we can find a non-trivial relation of the random group elements coms unless, once assigned the random variables $\tilde{\boldsymbol{z}}, \tilde{x}$, the polynomials vanish, which happens, by Schwartz-Zippel lemma, only with negligible probability.

Finally, we have that $\epsilon_4 = 0$. In fact, assume $\mathbf{H}_4 = 1$, then the forgery proof verifies, and we have $\tilde{\mathsf{c}}_j = [\tilde{c}_j(\beta)]_1$, by the change of hybrid $\mathbf{H}_4$ and $\sum_i a_i(\tilde{\boldsymbol{z}}, \tilde{x}) \cdot \tilde{c}_j(\tilde{x}) = \tilde{y}$ by the change in $\mathbf{H}_3$, thus the extractor extracts a valid witness which implies $\mathbf{H}_4 = 0$, which is a contradiction. $\square$

5.2 Controlled-Malleability of PST

We now focus on the controlled malleability of PST evaluation proofs. We define a PST policy that parallels the one for KZG but imposes even stricter limitations on the types of simulation oracle queries the simplified adversary may perform. Crucially, our proof technique removes the restriction on the maximum nesting level, which is essential for establishing the simulation-extractability of sumcheck-based zkSNARKs (see Section 7).

Definition 18 (Generalized hash-check policy for PST). *For any index $\mathbb{i} = (a_j)_{j\in[m]}$ of $\mathcal{R}_{\mathtt{geval}}$, let $\Phi^{\mathtt{PST}}_{\mathtt{gHC},\mathbb{i}}$ be a simplified policy that, upon input the forgery $(\mathbb{i}^*, \mathbb{x}^*, \pi^*)$ and $\mathbb{i} = \mathbb{i}^*$, the view* view *and the auxiliary output aux_Φ, parses $\mathbb{x}^* = ((\mathsf{c}^*_i)_i, \boldsymbol{z}^*, x^*, y^*)$, and returns 1 if and only if:*

- *The set $\mathcal{X}$ of the commitments in the simulation queries (see Definition 14) is such that $\mathcal{X} \subset$* coms *and,*
- *if either (1) $\forall i : a_i \in \mathbb{F}[\boldsymbol{Z}]$ and $(\mathsf{c}^*_i)_i \rightarrow_{\mathsf{RO}} \boldsymbol{z}^*$ and $(\sum_i a_i(\boldsymbol{z})\mathsf{c}^*_i) \rightarrow_{\mathsf{RO}} x^*$ or (2) $\forall i : a_i \in \mathbb{F}[X]$ and $(\mathsf{c}^*_i)_i \rightarrow_{\mathsf{RO}} x^*$, we call this condition the generalized hash check.*

We define the family of policies $\boldsymbol{\Phi}^{\mathtt{PST}}_{\mathtt{gHC}} := \{\Phi^{\mathtt{PST}}_{\mathtt{gHC},\mathbb{i}} : \mathbb{i}$ are linearly independent$\}$.

We are now ready to state our main result on the PST scheme.

Theorem 3 (Controlled-Malleability of PST). *For all $\mu \in \mathbb{N}$, the scheme $\Pi_{\mathtt{PST},\mu}$ is $\boldsymbol{\Phi}^{\mathtt{PST}}_{\mathtt{gHC}}$-simulation $\mathcal{T}_{\mathtt{LH}}$-controlled-malleable in the AGM under the OMSDH assumption.*

Proof Ideas. The proof of the theorem uses induction on the number of variables in the common reference string.

The base case holds by the controlled malleability of KZG, so we now focus on the induction step. In this step, we construct a reduction $\mathcal{B}$ for $\Pi_{\mathsf{PST},\mu-1}$ that internally makes use of an adversary against $\Pi_{\mathsf{PST},\mu}$. Observe that $\mathcal{B}$ can easily extend a $(\mu-1)$-variate common reference string for PST to a μ-variate one by sampling the last variable β_μ directly in the exponent.

What remains is to adapt simulation queries from the μ-variate setting to the $(\mu-1)$-variate one, and to show how to convert a forged evaluation proof against a μ-variate polynomial commitment into a forged evaluation proof for the $(\mu-1)$-variate case. For the first task, we note that given β_μ in clear, the reduction can fully simulate the proof by randomly sampling $\mu-1$ of the quotient polynomials (and performing the necessary bookkeeping when multiple queries are made on the same commitment by $\mathcal{A}$). Thus, at this stage, $\mathcal{B}$ can simulate for $\mathcal{A}$ without invoking the simulator $\mathcal{S}^{(\mu-1)}$.

As for the forgery, the key idea is that under the AGM the commitment $\tilde{\mathsf{c}}$ can be associated with a multivariate polynomial $p(\boldsymbol{X},\boldsymbol{Z},\boldsymbol{Q})$, where $\boldsymbol{X}=(X_1,\ldots,X_\mu)$ are the variable corresponding to the SRS, $\boldsymbol{Z}$ are the variables corresponding to the simulated commitments coms, and $\boldsymbol{Q}$ are the variables corresponding to the simulated proofs. By exploiting the simulation strategy of $\mathcal{B}$, we can simplify p and rewrite it as a polynomial in $\boldsymbol{X}$ and an extended set of simulated commitments $\boldsymbol{Z}$. This allows us to define our $(\mu-1)$-variate forgery by reinterpreting $\tilde{\mathsf{c}}$ as a $(\mu-1)$-variate commitment to $p(\boldsymbol{X}_{:\mu-1},\beta_\mu,\boldsymbol{Z})$.

Notice that the forged proof consists of quotient polynomials $\pi_1,\ldots,\pi_\mu$, extracted under the AGM, such that:

$$p(\boldsymbol{X},\boldsymbol{Z})=\sum_{i\in[\mu]}\pi_i(\boldsymbol{X},\boldsymbol{Z})(X_i-\tilde{x}_i)+\tilde{y}.$$

Using the same idea as for $\tilde{\mathsf{c}}$, we can reinterpret all these quotient polynomials by assigning $X_\mu\leftarrow\beta_\mu$. However, we still have μ quotient polynomials, whereas the forged proof should contain only $\mu-1$. We remove the last quotient π_μ from the verification equation by computing an evaluation proof of π_μ using the simulation oracle and absorbing it using the homomorphic property of PST. In details, the reduction $\mathcal{B}$ invokes the simulation oracle to generate an evaluation proof $(\pi'_j)_{j\in[\mu-1]}$ for the quotient π_μ at the point $\tilde{\boldsymbol{x}}_{:\mu-1}$ evaluating at y_π, the latter value y_π is carefully chosen by $\mathcal{B}$ to ensure the algebraic consistency of the view. Thanks to the homomorphic property of PST we can finally define the forged proof $(\hat{\pi}_j)_{j\in[\mu-1]}$ for $\tilde{\mathsf{c}}$ as:

$$\begin{aligned}&p(\boldsymbol{X}_{:\mu-1},\beta_\mu,\boldsymbol{Z})=\\&\sum_{i\in[\mu-1]}\pi_i(\boldsymbol{X}_{:\mu-1},\beta_\mu,\boldsymbol{Z})(X_i-\tilde{x}_i)+\pi_\mu(\boldsymbol{X}_{:\mu-1},\beta_\mu,\boldsymbol{Z})(\beta_\mu-\tilde{x}_\mu)+\tilde{y}=\\&\sum_{i\in[\mu-1]}\underbrace{(\pi_i(\boldsymbol{X}_{:\mu-1},\beta_\mu,\boldsymbol{Z})+\pi'_i(\boldsymbol{X},\boldsymbol{Z})(\beta_\mu-\tilde{x}_\mu))}_{\hat{\pi}_i(\boldsymbol{X}_{:\mu-1},\boldsymbol{Z})}(X_i-\tilde{x}_i)+(\tilde{y}+(\beta_\mu-\tilde{x}_\mu)y_\pi)\end{aligned}$$

Notice that although $\mathcal{B}$ invokes its simulation oracle, all simulation queries are issued only *after* the forged commitment has been fixed. At this point we can leverage the refinement on the nesting level that we introduced and analyzed for KZG. In particular, when the reduction targets the controlled malleability of KZG, the nesting level of $\mathcal{B}$ remains zero regardless of how many simulation queries $\mathcal{A}$ makes. The last step, and also the most technical part of the proof, is to show that $p(\tilde{\boldsymbol{x}},(y_i)_i)=\tilde{y}$

where the y_i are defined by looking at the simulation queries of the form $(\mathsf{c}_i, \tilde{\boldsymbol{x}}, y_i)$ made by $\mathcal{A}$. To prove this claim we need to use multiple time the one-more SDH assumption starting from the induction hypothesis, which states that:

$$p(\tilde{\boldsymbol{x}}_{:\mu-1}, \beta_\mu, (y_i)_i) = \tilde{y} + (\beta_\mu - \tilde{x}_\mu) y_\pi.$$

Roughly, we show how to replace β_μ with a formal variable X_μ in the above equation.

Proof. The proof of the theorem proceeds by induction on the number of variables of the committed polynomials. The inductive step is proved in Lemma 8 while the base case can be reduced to Theorem 2.

Lemma 8. *For all $\mu \in \mathbb{N}$ and $\mu > 1$ and for any $\boldsymbol{d} \in \mathbb{N}^\mu$, there exists an extactor $\mathcal{E}$, for any simplified algebraic adversary $\mathcal{A}$ there exist adversaries $\mathcal{B}$ and $\mathcal{B}^*$ and a constant c such that:*

$$\mathbf{Adv}^{(\Phi_{\mathtt{gHC},\mathbb{i}}, \mathcal{T}_{\mathtt{LH}})\text{-}\mathtt{cm}}_{\Pi_{\mathtt{PST},\mu}, \mathcal{A}, \mathcal{S}^{(\mu)}, \mathcal{E}}(\lambda) \leq \mathbf{Adv}^{(\Phi, \mathcal{T}_{\mathtt{LH}})\text{-}\mathtt{cm}}_{\Pi_{\mathtt{PST},\mu-1}, \mathcal{B}, \mathcal{S}^{(\mu-1)}, \mathcal{E}}(\lambda) + c \cdot \mu \cdot \mathbf{Adv}^{(1,d_\mu)\text{-}\mathtt{OMSDH}}_{\mathsf{GGen}, \mathcal{B}^*}(\lambda) + \frac{d^*}{|\mathbb{F}|}$$

where if $\mu > 2$ then $\Phi := \Phi^{\mathtt{PST}}_{\mathtt{gHC},\mathbb{i}}$ and otherwise $\Phi := \Phi^{\nu}_{\mathtt{gHC},\mathbb{i}}$ for $\nu = 1$, and $d^* = \sum d_i$.

Proof. We first describe the extractor. Let $((a_j)_{j\in[m]}, (\tilde{\mathsf{c}}_j)_{j\in[m]}, \tilde{\boldsymbol{z}}, \tilde{\boldsymbol{x}}, y)$ be the forgery of the adversary, and let $(\tilde{f}_i, \tilde{\boldsymbol{r}})$ be the algebraic representation of $\tilde{\mathsf{c}}_i$ for any i. We can parse $\tilde{\boldsymbol{r}}_i = \tilde{\boldsymbol{c}}_i \| \tilde{\boldsymbol{o}}_i$ where $\tilde{\boldsymbol{r}}_i^T \cdot (\mathsf{coms} \| \mathsf{proofs}) = \tilde{\boldsymbol{c}}_i^T \cdot \mathsf{coms} + \tilde{\boldsymbol{o}}_i^T \cdot \mathsf{proofs}$. We define our extractor $\mathcal{E}(\mathsf{view}, \mathsf{st}_\mathcal{S}, \mathsf{aux}_\mathcal{E})$ that:

1. If $\tilde{\boldsymbol{x}} \in \mathcal{Q}_x$ then from $(\tilde{f}_i, \tilde{\mathbf{c}}_i, \tilde{\boldsymbol{o}}_i)_{i\in[m]}$ and from $\mathcal{Q}_{\mathsf{sim}}$, it defines a transformation T, by defining the set of polynomials $\tilde{f}_i(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{P}) = f_i(\boldsymbol{X}) + \sum_j \tilde{c}_{i,j} \cdot Y_j + \tilde{o}_{i,j} \cdot P_j$ for any i. For any i, j, it finds the query $\mathbb{x}_{i,j} = (\mathsf{c}_j, \tilde{\boldsymbol{x}}, y_{i,j})$ for $\tilde{c}_{i,j} \neq 0$ in the set $\mathcal{Q}_{\mathsf{sim}}$. The transformation maps $(\mathbb{x}_{i,j})_{i,j}$ to:
   - For any i, $\tilde{\mathsf{c}}_i = \left[\tilde{f}_i(\beta)\right]_1 + \sum_j \tilde{c}_{i,j} \cdot \mathbb{x}_{i,j}.\mathsf{c}$.
   - $\tilde{y} = \sum_i a_i(\tilde{\boldsymbol{z}}, \tilde{\boldsymbol{x}}) \cdot (f_i(\tilde{\boldsymbol{x}}) + \sum_j \tilde{c}_{i,j} y_{i,j})$.

   It returns $(\mathbb{w} = \bot, T, (\mathbb{x}_{i,j})_{i,j})$.
2. Else it returns $(\mathbb{w} = (\tilde{f}_j)_{j\in[m]}, \bot, \bot)$.

For any μ, we let $\mathsf{ck}^{(\mu)}$ be the commitment key of $\mathsf{CS}_{\mathtt{PST},\mu}$. We describe the reduction below.

<u>Reduction $\mathcal{B}(\mathsf{ck}^{(\mu-1)}, \mathsf{coms})$ with oracle access to $\mathcal{S}^{(\mu-1)}(\mathsf{st}_\mathcal{S}, \cdot)$:</u>

1. It samples β_μ and generates the SRS $\mathsf{ck}^{(\mu)}$ for $\mathsf{CS}_{\mathtt{PST},\mu}$ from the input $\mathsf{ck}^{(\mu-1)}$ and β_μ. Specifically, let $\mathsf{ck}^{(\mu-1)} = (\mathsf{ek}^{(\mu-1)}, \mathsf{vk}^{(\mu-1)})$, set $\mathsf{ek}^{(\mu)} \leftarrow (\beta_\mu^j)_{j\in[d_\mu]} \otimes \mathsf{ek}^{(\mu-1)}$ and $\mathsf{vk}^{(\mu)} \leftarrow (\mathsf{vk}^{(\mu-1)}, [\beta_\mu]_2)$ and it runs the adversary $\mathcal{A}$ on input $\mathsf{ck}^{(\mu)}$ answering its oracle queries.
2. It keeps a (lazy sampled) random map G mapping elements $\mathbb{G}_1 \times \mathbb{F}^*$ to values in $\mathbb{F}$.
3. At the i-th simulation oracle query $(\mathsf{c}, \boldsymbol{x}, y)$ it computes $r_{i,j} \leftarrow G(\mathsf{c} - [y]_1, x_{j+1}, \ldots, x_\mu)$ for any j. It defines the polynomial $R_i(\boldsymbol{X}) \leftarrow (\sum_j r_{i,j}(X_j - x_j))(X_\mu - x_\mu)$ and let $\mathsf{r}_i = [R_i(\boldsymbol{\beta})]_1$. Then it computes $\pi_{i,\mu} \leftarrow (\mathsf{c} - y - \mathsf{r}_i)/(\beta_\mu - x_\mu)$ and computes $\pi_{i,j} \leftarrow [r_{i,j}(\beta_\mu - x_\mu)]_1$ for $j \in [\mu - 1]$.
4. At forgery $((\tilde{\mathsf{c}}_j)_{j\in[m]}, \tilde{\boldsymbol{z}}, \tilde{\boldsymbol{x}}, \tilde{y}), (\tilde{\pi}_j)_{j\in[\mu]}$ from $\mathcal{A}$, it computes the algebraic representation $\tilde{\pi}_\mu(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{Q})$ of $\tilde{\pi}_\mu$, it can simplify the representation by assigning the variables $\boldsymbol{Q}$ associated with the simulated proof materials to the polynomials in $\mathbb{F}[\boldsymbol{X}, \boldsymbol{C}]$ defined by the reduction. Namely, if the i-th simulation is on $(\mathsf{c}_k, \boldsymbol{x}, y)$ for some k, it assigns $Q_{i,\mu} \leftarrow (Z_k - y - R_i(\boldsymbol{X}))/(X_\mu - x_\mu)$ and $Q_{i,j} \leftarrow r_{i,j} \cdot (X_\mu - x_\mu)$ for $j \in [1, \mu - 1]$.
   Let $\pi_\mu(\boldsymbol{X}_{:\mu-1}, \beta_\mu, \boldsymbol{Z}) = p(\boldsymbol{X}_{:\mu-1}, \beta_\mu) + \sum c_i \cdot Z_i$ for some p and $(c_i)_i$:

- For all $c_i \neq 0$, it sets $y_i \leftarrow y'$ if there exists y' such that $(\mathsf{c}_i, \tilde{\boldsymbol{x}}, y') \in \mathcal{Q}_{\mathsf{sim}}$ or $y_i \leftarrow 0$. Then it queries the simulator on $(\mathsf{c}_i, \tilde{\boldsymbol{x}}_{:\mu-1}, y_i)$ obtaining proof elements $(\pi_{Z_i,j})_{j\in[\mu]}$.
- It computes proof material for $p(\boldsymbol{X}_{:\mu-1}, \beta_\mu)$, namely it computes $y_p = p(\tilde{\boldsymbol{x}}_{:\mu-1}, \beta_\mu)$ and $\pi_{p,j}(\boldsymbol{X})$ such that $p(\boldsymbol{X}_{:\mu-1}, \beta_\mu) - y_p = \sum_j \pi_{p,j}(\boldsymbol{X})(X_j - \tilde{x}_j)$ and let $\pi_{f,j} = [\pi_{f,j}(\boldsymbol{\beta})]_1$.
- It computes proof materials for $(\tilde{\pi}_\mu, \tilde{\boldsymbol{x}}_{\mu-1}, y_p)$:

$$\pi'_j \leftarrow \pi_{p,j} + \sum_i c_i \cdot \pi_{Z_i,j}, \forall j$$
$$y_\pi \leftarrow y_p + \sum_i c_i \cdot y_i$$

It computes a new forged proof for $(\tilde{\mathsf{c}}_j)_{j\in[m]}$ as an $(\mu-1)$-variate polynomial commitment, using the homomorphic properties of PST. Namely, it computes:

$$\hat{\pi}_j \leftarrow \tilde{\pi}_j + (\beta_\mu - x_\mu)\pi'_j$$
$$\hat{y} \leftarrow \tilde{y} + (\beta_\mu - x_\mu) y_\pi.$$

and sets the forgery $((\tilde{\mathsf{c}}_j)_{j\in[m]}, \tilde{\boldsymbol{z}}, \tilde{\boldsymbol{x}}_{:\mu-1}, \hat{y})$ with proof $(\hat{\pi}_j)_{j\in[\mu-1]}$.

We notice that the simulated proofs provided by $\mathcal{B}$ to $\mathcal{A}$ are equivalently distributed to the simulated proofs generated by $\mathcal{S}^{(\mu)}$, in particular, the values $r_{i,j}$ are uniformly distributed and thus the proof material $\pi_{i,j} = [r_{i,j}(\beta_\mu - x_\mu)]_1$; moreover, by following the map G, we make sure that when querying the simulator with two queries $(\mathsf{c}, \boldsymbol{x}, y)$ and $(\mathsf{c}, \boldsymbol{x}', y)$ the proof material for the indexes relative to the common suffix of $\boldsymbol{x}$ and $\boldsymbol{x}'$ are the same.

We show that the reduction $\mathcal{B}$ outputs a forgery that verifies whenever $\mathcal{A}$ does. Briefly, if $\mathcal{A}$ sends a valid proof for the instance $((\tilde{\mathsf{c}}_j)_{j\in[m]}, \tilde{\boldsymbol{z}}, \tilde{\boldsymbol{x}}, \tilde{y})$ and with witness $(c_i(X_1, \dots, X_\mu))$ for $\mathcal{R}_{\mathtt{geval}}$ then the reduction computes a valid proof for the instance $((\tilde{\mathsf{c}}_j)_{j\in[m]}, \tilde{\boldsymbol{z}}, \tilde{\boldsymbol{x}}_{:\mu-1}, \hat{y})$ with witness $(c(X_1, \dots, X_{\mu-1}, \beta_\mu))_i$ for $\mathcal{R}_{\mathtt{geval}}$. Notice that the commitments remain identical.

Moreover, the reduction uses the homomorphic properties of the commitment scheme and knowledge of the trapdoor β_μ to create a proof for the commitments $(\tilde{\mathsf{c}}_j)_{j\in[m]}$ interpreted as commitments to $(\mu-1)$-variate polynomials. In particular, we have that, thanks to the algebraic representation and the simulated proofs obtained, $\tilde{\pi}_\mu$ is such that:

$$[\tilde{\pi}_\mu]_T = \sum_{j\in[\mu-1]} e(\pi'_j, [\beta_j - x_j]_2) + [y_\pi]_T.$$

Let $\tilde{\mathsf{c}} = \sum_j a_j(\tilde{\boldsymbol{z}}, \tilde{x}_1)\tilde{\mathsf{c}}_j$, plugging the R.H.S. of the equation above in the verification equation for $\mathcal{B}$, namely the verification for $\Pi_{\mathtt{PST},\mu-1}$:

$$\begin{aligned} e(\tilde{\mathsf{c}}, [1]_2) &= \sum_{j\in[\mu-1]} e(\tilde{\pi}_j, [\beta_j - x_j]_2) + (\beta_\mu - x_\mu)\left(\sum_{j\in[\mu-1]} e(\pi'_j, [\beta_j - x_j]_2) + [y_\pi]_T\right) + [y]_T \\ &= \sum_{j\in[\mu-1]} e(\tilde{\pi}_j + (\beta_\mu - x_\mu)\pi'_j, [\beta_j - x_j]_2) + [y + (\beta_\mu - x_\mu)y_\pi]_T. \end{aligned}$$

We show that $\mathcal{B}$ satisfies the policy . It is straightforward to notice that for $\mu > 2$ then the $\mathcal{B}$ satisfies the policy, since the forgery of $\mathcal{A}$ and the forgery of $\mathcal{B}$ are syntattically the same. For the case $\mu = 2$, we additionally need to analyse the maximum nesting level of $\mathcal{B}$'s forgery. In particular, we notice that $\mathcal{B}$ does not query its own simulation oracle until $\mathcal{A}$'s forgery is defined. Therefore the set $\tilde{\mathcal{Q}}$ that contains all the queries to the simulation oracle made by $\mathcal{B}$ before the random oracle query with output $\tilde{x}$ (as defined in Definition 15) is empty and in particular $\bar{\nu}_{\tilde{\mathcal{Q}}} = 0 \leq \nu = 1$.

We now need to show that the canonical algebraic-model extractor that considers as witness the algebraic representations of the group elements $(\tilde{c}_i)_{i \in [m]}$ is indeed a valid extractor for the controlled-malleability experiment with adversary $\mathcal{A}$ against $\Pi_{\mathtt{PST},\mu}$.

Let $(\mathbb{w}, T, (\mathbb{x}_j)_{j \in [k]})$ the output of the extractor $\mathcal{E}$ of the $\mathcal{T}_{\mathtt{LH}}$-controlled-malleability experiment with adversary $\mathcal{B}$ against $\Pi_{\mathtt{PST},\mu-1}$. Without lose of generality we can parse the output of $\mathcal{E}$ as a tuple of polynomials $(\hat{c}_j)_{j \in [m]}$ in $\mathbb{F}[\boldsymbol{X}, \boldsymbol{Y}]$ where $|\boldsymbol{Y}| = |\mathcal{Q}_{\mathsf{sim}}|$ and if $\mathsf{deg}_{Y_i}(\hat{c}_j)$ is non-zero then the i-th query to the simulator is in the list instances $(\mathbb{x}_j)_{j \in [k]}$ output by the extractor. Notice when all the coefficients for the variables $\boldsymbol{Y}$ are zero in the polynomials $(\hat{c}_j)_{j \in [m]}$ then we can regard them as a witness for $\mathcal{R}_{\mathtt{geval}}$, otherwise we can regard them as a transformation in $\mathcal{T}_{\mathtt{LH}}$.

Let $\hat{c}(\boldsymbol{X}_{:\mu-1}, \boldsymbol{Y}) = \sum_j a_j(\tilde{\boldsymbol{z}}, \tilde{x}_1)\hat{c}_j(\boldsymbol{X}_{:\mu-1}, \boldsymbol{Y})$. By the guarantees of the security experiment we have $\hat{c}(\tilde{\boldsymbol{x}}_{:\mu-1}, \boldsymbol{y}) = \hat{y}$ and $\boldsymbol{y} = (y_i)_i$ as defined in the reduction in Item 4 (since those are the only simulation queries made by $\mathcal{B}$).

Let $\tilde{c}(\boldsymbol{X}, \boldsymbol{Y})$ be the algebraic representation of $\tilde{\mathsf{c}}$ where, similarly to Item 4, we assign the variable $\boldsymbol{Q}$ accordingly to the simulation queries. We can show that:

$$\tilde{c}(\boldsymbol{X}_{:\mu-1}, \beta_\mu, \boldsymbol{Y}) = \hat{c}(\boldsymbol{X}_{:\mu-1}, \boldsymbol{Y}) \tag{4}$$

Assume the contrary, thus the equation above does not hold but $\tilde{c}(\boldsymbol{\beta}, \mathsf{coms}) = \hat{c}(\boldsymbol{\beta}, \mathsf{coms})$. We have two possible cases, either $\tilde{c}(\boldsymbol{\beta}, \boldsymbol{Y}) \neq \hat{c}(\boldsymbol{\beta}, \boldsymbol{Y})$ or $\tilde{c}(\boldsymbol{X}_{:\mu-1}, \beta_\mu, \mathsf{coms}) \neq \hat{c}(\boldsymbol{X}_{:\mu-1}, \beta_\mu, \mathsf{coms})$. In the first case we consider the reduction to DLOG that defines $\mathsf{coms} = \boldsymbol{r}\,[a]_1$ for a random challenge $[a]_1$ and runs the controlled-malleability experiment with $\mathcal{A}$. The reduction can answer the simulation oracle queries thanks to the knowledge of β_μ. Eventually, it can run the extractor $\mathcal{E}_{\mu-1}$ which returns $\hat{c}$ and, thanks to the algebraic representation given by $\mathcal{A}$ it can define $\tilde{c}$. Because $\tilde{c}(\boldsymbol{\beta}, \mathsf{coms}) = \hat{c}(\boldsymbol{\beta}, \mathsf{coms})$, the non-zero polynomial $\tilde{c}(\boldsymbol{\beta}, \boldsymbol{Y}) - \hat{c}(\boldsymbol{\beta}, \boldsymbol{Y})$ must vanish in $r_i \cdot a$ for some i. Thus it can find the DLOG of $[a]_1$. The other case it is very similar, however now we need to reduce to the SDH assumption. In particular, we can consider the loose reduction that sample a random index i^* obtain challenge $(\left[\beta_{i^*}^k\right]_1)_{k \in [d_{i^*}]}$ and set ck by sampling the remaining β_j for $j \neq i^*$.

Notice that by the controlled-malleability of $\Pi_{\mathtt{PST},\mu-1}$ we have that $\hat{c}(\boldsymbol{X}_{:\mu-1}, \boldsymbol{Y}) = \hat{f}(\boldsymbol{X}_{:\mu-1}) + \sum \hat{c}_i Y_i$ for a polynomial $\hat{f}$ and coefficient $\hat{c}_i$ such that $(\mathsf{c}_i, \tilde{\boldsymbol{x}}_{:\mu-1}, y_i)$ was queried by the reduction to the simulation oracle. Thus by Eq. (4), we have

$$\hat{c}(\boldsymbol{X}_{:\mu-1}, \boldsymbol{Y}) = \tilde{f}(\boldsymbol{X}_{:\mu-1}, \beta_\mu) + \textstyle\sum_i \hat{c}_i Y_i,$$

where $\tilde{c}(\boldsymbol{X}_{:\mu-1}, \beta_\mu, \boldsymbol{Y}) = \tilde{f}(\boldsymbol{X}_{:\mu-1}, \beta_\mu) + \sum \hat{c}_i Y_i$.

By the controlled-malleability of $\Pi_{\mathtt{PST},\mu-1}$ we additionally have:

$$\hat{c}(\boldsymbol{x}_{:\mu-1}, \boldsymbol{y}) = \tilde{y} + (\beta_\mu - \tilde{x}_\mu) \cdot (p(\tilde{\boldsymbol{x}}_{:\mu-1}, \beta_\mu) + \textstyle\sum_j c_i y_i)$$

Moreover, we have $\hat{c}(\boldsymbol{x}_{:\mu-1}, \boldsymbol{y}) = \tilde{f}(\tilde{\boldsymbol{x}}_{:\mu-1}, \beta_\mu)$. We need to show that the conjunction of these two equations holds even in the realm of polynomials. Namely we show:

$$\tilde{f}(\tilde{\boldsymbol{x}}_{:\mu-1}, X) = \tilde{y} + (X - \tilde{x}_\mu) \cdot (p(\tilde{\boldsymbol{x}}_{:\mu-1}, X) + \textstyle\sum_j c_i y_i) \tag{5}$$

This last equation implies that $\tilde{f}(\tilde{\boldsymbol{x}}) = \tilde{y}$.

We show Eq. (5) using the $(1, d)$-OMSDH Assumption. In particular, consider the bad event that Eq. (5) does not hold but $\left[\tilde{f}(\tilde{\boldsymbol{x}}_{:\mu-1}, \beta_\mu)\right]_1 = [\tilde{y} + (\beta_\mu - \tilde{x}_\mu) \cdot p(\tilde{\boldsymbol{x}}_{:\mu-1}, \beta_\mu)]_1$. This bad event implies that β_μ is a root of the non-zero polynomial $\tilde{f}(\tilde{\boldsymbol{x}}_{:\mu-1}, X) - (\tilde{y} + (X - \tilde{x}_\mu) \cdot p(\tilde{\boldsymbol{x}}_{:\mu-1}, X))$. Thus we can consider the reduction $\mathcal{B}'$ that runs similarly to $\mathcal{B}$ but with the following differences:

- It gets as input $\mathsf{ck} = \left(([\beta_\mu^i]_1)_{i\in[q]}, [1, \beta_\mu]_2\right)$ and samples $\beta_1, \ldots, \beta_{\mu-1} \leftarrow\!\!\$\ \mathbb{F}^{\mu-1}$ and $\alpha_i \leftarrow\!\!\$\ \mathbb{F}^n$ and sets $\mathsf{coms} = ([\alpha_i]_1)_{i\in[n]}$. It generates $\mathsf{ck}^{(\mu)}$ similarly to $\mathcal{B}$ with the knowledge of $\beta_1, \ldots, \beta_{\mu-1}$.
- It responds to the i-th simulation oracle query, with instance $(\mathsf{c}_k, \boldsymbol{x}, y)$ for some k, by querying the one-more SDH oracle on $(x_\mu, 1)$ receiving $\left[(\beta_\mu - x_\mu)^{-1}\right]_1$ and using such a value, and the knowledge of α_k to compute $\pi_{i,\mu}$, while the remaining proof material can be computed using the associated $R_i(\boldsymbol{X})$.
- Eventually the adversary outputs its forgery, the reduction computes $\tilde{f}(\tilde{\boldsymbol{x}}_{:\mu-1}, X)$ and $p(\tilde{\boldsymbol{x}}_{:\mu-1}, X)$ from the representations of $\tilde{c}$ and $\tilde{\pi}_\mu$ and find β_μ among the roots of the polynomial described in Eq. (5).

To summarize, we have shown that the polynomial $\tilde{c}(\boldsymbol{X}, \boldsymbol{Y})$ is such that:

1. $\tilde{c}(\tilde{\boldsymbol{x}}, \boldsymbol{y}) = \tilde{y}$ where $(c_i, \tilde{\boldsymbol{x}}, y_i)$, by construction of $\mathcal{B}$, are queried by $\mathcal{A}$ to its own simulation oracle.
2. $[\tilde{c}(\boldsymbol{\beta}, \mathsf{coms})]_1 = \tilde{\mathsf{c}}$
3. For any i, if $\mathsf{deg}_{Y_i}(\tilde{c}) > 0$ then $(\mathsf{c}_i, \tilde{\boldsymbol{x}}, y_i)$ was queried by $\mathcal{A}$ to its own simulation oracle.

Let $\boldsymbol{Y}^+$ be the variables $(Y_j)_{\hat{c}_j \neq 0}$ and, similarly, $\boldsymbol{Y}^-$ be the variables $(Y_j)_{\hat{c}_j = 0}$. We can re-write the algebraic representation of the commitments $\tilde{\mathsf{c}}_i$ as follows:

$$\tilde{c}_i(\boldsymbol{X}, \boldsymbol{Y}) = f_i(\boldsymbol{X}, \boldsymbol{Y}^+) + g_i(\boldsymbol{X}, \boldsymbol{Y}),$$

where the monomials in g_i have non-zero degree in at least one of the variables in $\boldsymbol{Y}^-$. We need to show that, for any i, the polynomial $g_i(\boldsymbol{X}, \boldsymbol{Y}) = 0$.

Let's assume the contrary, then we can show $\sum_i a_i(\boldsymbol{Z}, X_1) \cdot g_i(\boldsymbol{X}) \neq 0$. Since the $(a_j)_{j\in[m]}$ are linearly independent, for any assignment $\boldsymbol{x}, \boldsymbol{y}$ the polynomial $\sum_i a_i(\boldsymbol{Z}, X) g_i(\boldsymbol{x}, \boldsymbol{y})$ is non-zero. Thus, $\sum_i a_i(\boldsymbol{Z}, X_1) g_i(\boldsymbol{X}, \boldsymbol{Y})$ is non zero.

Notice that, by the policy $\Phi^{\nu}_{\mathtt{gHC,i}}$, we have either $(\mathsf{c}^*_j)_{j\in[m]} \rightarrow_{\mathsf{RO}} \tilde{\boldsymbol{z}}$ or $(\mathsf{c}^*_j)_{j\in[m]} \rightarrow_{\mathsf{RO}} \tilde{\boldsymbol{x}}$, in both cases we can apply by the Schwartz-Zippel lemma, because the random valus $\boldsymbol{z}$ and $\boldsymbol{x}$ are sampled independently of $\sum_i a_i(\boldsymbol{Z}, X_1) \cdot g_i(\boldsymbol{X}, \boldsymbol{Y})$. Thus unless with probability $\frac{d^*}{|\mathbb{F}|}$, where we recall $d^* = \sum d_i$, we have $\sum a_i(\tilde{\boldsymbol{z}}, \tilde{x}_1) \cdot g_i(\boldsymbol{X}, \boldsymbol{Y}) \neq 0$.

On the other hand, by Item 3 of the properties of $\tilde{\mathsf{c}}$, it must be that $\sum_i a_i(\tilde{\boldsymbol{z}}, \tilde{x}_1) g_i(\boldsymbol{\beta}, \mathsf{coms}) = [0]_1$. Notice that the commitments coms are sampled uniformly at random, thus we can either break a representation problem which in turns implies breaking the DLOG assumption or break the one-more SDH assumption. For the case of DLOG, the algebraic reduction upon input a challenge element $[\alpha]_1$ generates the commitments $\mathsf{c}_i = r_i \cdot [\alpha]_1$, runs the controlled-malleability experiment with $\mathcal{A}$ by sampling the trapdoor $\boldsymbol{\beta}$ and instantiatiating the simulator. Eventually, the reduction can find the polynomial $(g_j)_{j\in[i]}$ from the algebraic representations, since $f'([\alpha]_1) = \sum_i a_i(\boldsymbol{z}^*, x^*_1) g_i(\boldsymbol{\beta}, \boldsymbol{r} \cdot [\alpha]_1) = [0]_1$, by factorizing f' the reduction can find α and break the DLOG assumption. The case of the one-more SDH is very similar. We can partition this case in μ different subcases. For the case of extracting β_μ we can reduce to the one-more SDH assumption running the controlled-malleability experiment similarly to the reduction $\mathcal{B}'$ described above, and eventually finding the root β_μ in the polynomial $\sum_i a_i(\boldsymbol{z}^*, x_1^*) g_i(\boldsymbol{X}, \boldsymbol{Y})$. For the other case, we can actually reduce to the plain SDH assumption because we can run the controlled-malleability experiment using the same strategy of the reduction $\mathcal{B}$.

We finalize the lemma by first noticing that both the DLOG and the SDH assumptions can be reduced to the one-more SDH assumption. Namely, an adversary for the DLOG (resp. SDH) assumption is also an adverary against the one-more SDH assumption. Moreover, we set $\mathcal{B}^*$ in the statement of the lemma to be the loosest of the reduction we showed which has a $1/\mu$ multiplicative loose factor. □

6 Polynomial Interactive Oracle Proofs

We recall the formalism of oracle relations introduced in [18]. In a nutshell, an oracle relation can be seen as the *oracle-world* counterpart of a commit-and-prove relation. In particular, oracle relations are a useful abstraction that allows to define predicates over the oracles sent by the prover in the execution of a PIOP.

Definition 19 (Oracle Relations, [18,26]). *An oracle (indexed) relation $\mathcal{R}$ is an (indexed) relation when the instances $\mathbb{x}$ of $\mathcal{R}$ contain pointers to oracle polynomials over some field $\mathbb{F}$. The actual polynomials corresponding to the oracles are contained in the witness.*

We denote the pointer to the oracle polynomial f by $[\![f]\!]$, let $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ we denote with $\mathsf{oracles}(\mathbb{x}) = \{[\![f_1]\!], [\![f_2]\!], \ldots, [\![f_k]\!]\}$ *for some k the pointers to the polynomial oracles in $\mathbb{x}$ and* $\mathbb{w} = (f_1, f_2, \ldots, f_k)$.

Similarly to [18], we consider the notion of virtual oracles. A virtual oracle to $[\![f]\!] := g([\![f_1]\!], \ldots, [\![f_k]\!])$ for some function g is the list of oracles $\{[\![f_1]\!], \ldots, [\![f_k]\!]\}$ together with the description of g. To evaluate $g([\![f_1]\!], \ldots, [\![f_k]\!])$ at some point $\boldsymbol{x}$, we compute $y_i = f_i(\boldsymbol{x})$, $\forall i \in [k]$ and output $g(y_1, \ldots, y_k)$. Equivalently, given commitments to polynomials, we can construct a *virtual commitment* to a function of these polynomials in the same manner. If g is an additive function, namely $g(X_1, \ldots X_k) = \sum_{i \in [k]} a_i X_i$ and the polynomial commitment is additively homomorphic, such as PST, then we can use the homomorphism to do the evaluation. In this case we call the virtual oracle polynomial $[\![f]\!]$ a *linear* virtual polynomial.

Definition 20 ((Holographic) Multivariate Polynomial IOP). *Let $\mathcal{F}$ be a family of finite fields, let $\mathcal{R}$ be an oracle indexed relation. A (public-coin non-adaptive) holographic multivariate polynomial Interactive Oracle Proof (PIOP) over $\mathcal{F}$ for $\mathcal{R}$ is a tuple $\mathsf{IP} := (r, n, m, \nu, d, \mathsf{I}, \mathsf{P}, \mathsf{V})$ where $r, n, m, \nu, d \colon \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions, and $\mathsf{I}, \mathsf{P}, \mathsf{V}$ are three algorithms for the indexer, prover and verifier respectively, that work as follows.*

Offline phase: *The indexer $\mathsf{I}(\mathbb{F}, \mathbb{i})$ is executed on input a field $\mathbb{F} \in \mathcal{F}$ and a relation description $\mathbb{i}$, and it returns $n(0)$ oracle polynomials $\{[\![p_{0,j}]\!] : p_{0,j} \in \mathbb{F}_{\nu(0)}[X]\}_{j \in [n(0)]}$ encoding the relation $\mathbb{i}$.*

Online phase: *The prover $\mathsf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$ and the verifier $\mathsf{V}^{\mathsf{I}(\mathbb{F}, \mathbb{i})}(\mathbb{F}, \mathbb{x})$ are executed for $r(|\mathbb{i}|)$ rounds; the prover has a tuple $(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and the verifier has an instance $\mathbb{x}$ and oracle access to the index polynomials.*

In the i-th round, P sends $m(i)$ messages $\{\pi_{i,j} \in \mathbb{F}\}_{j \in [m(i)]}$, and $n(i)$ oracle polynomials $\{[\![p_{i,j}]\!] : p_{i,j} \in \mathcal{F}_{d(i),\nu(i)}[X]\}_{j \in [n(i)]}$, while V replies (except for the last round) with a uniformly random message $\rho_i \in \mathbb{F}$.

Decision phase: *The verifier queries the oracle polynomials at arbitrary points and outputs a decision bit. More in details, the verifier can make queries of the form* $(\llbracket p \rrbracket, \boldsymbol{x}, y; p) \in \mathcal{R}_{\mathtt{evl}}$ *where* $\llbracket p \rrbracket$ *are virtual oracles derived from* $\{\llbracket p_{i,j} \rrbracket : i \in r(|\mathbb{i}|), j \in n(i)\}$.

Important complexity measures for efficient zkSNARK compilation include the *number of verifier queries* and the *degree of virtual polynomials.* Notably, we can focus on the number of linear virtual polynomials queried by the verifier. In fact:

- Any PIOP can be transformed into one that queries only linear virtual polynomials by decomposing higher-degree virtual polynomials into multiple linear queries, at the cost of increasing the number of queries.
- The homomorphic property of PST allows each linear virtual polynomial query to correspond to a *single proof* in the polynomial commitment scheme.

Definition 21. *We call the* (virtual) query complexity *of* IP *the number of oracle queries made by the verifier. Moreover, we say that* IP *is* strictly linear virtual queries *if its verifier makes only linear virtual polynomials queries.*

Security properties. An important security property we require from a PIOP is *state-restoration straight-line knowledge soundness.* The notion is defined by a security game in which the malicious prover $\tilde{\mathsf{P}}$ engages with an honest verifier and has the ability to *roll back* the interaction with the verifier to a previous state. Eventually, the interaction may reach a final state and the prover is considered successful if it outputs an accepting transcript, while the extractor, given the oracles in the accepting transcript, fails to produce a valid witness. State-restoration knowledge soundness is implied by standard knowledge soundness [8], even though the reduction is lossy, and it is considered the correct notion of soundness for multi-round public-coin PIOPs compiled through the Fiat–Shamir transform since it prevents the so-called *grinding attacks* [62].

1. The challenger initializes the list $\mathsf{SeenStates}$ to be empty.
2. Repeat the following until the challenger halts:
   (a) $\tilde{\mathsf{P}}$ either (1) chooses a complete verifier state cvs in $\mathsf{SeenStates}$ or (2) sends a fresh tuple $(\mathbb{i}, \mathbb{x}, \{\pi_{1,j}\}_j, \{p_{1,j}\}_j)$ to the challenger.
   (b) If (1) the challenger sets the verifier to cvs:
      i. if $\mathsf{cvs} = (\mathbb{i}, \mathbb{x}, \{\pi_{1,j}\}_j, \{p_{1,j}\}_j \| \rho_1 \| \dots \| \{\pi_{i,j}\}_j, \{p_{i,j}\}_j)$ and $i < r(\mathbb{x})$: $\tilde{\mathsf{P}}$ outputs $\{\pi_{i-1,j}\}_j$, $\{p_{i-1,j}\}_j$; V samples ρ_i and sends it to $\tilde{\mathsf{P}}$; the game appends $\mathsf{cvs}' := (\mathsf{cvs} \| \{\pi_{i-1,j}\}_j \| \{p_{i-1,j}\}_j \| \rho_i)$ to the list $\mathsf{SeenStates}$;
      ii. if $\mathsf{cvs} = (\mathbb{i}, \mathbb{x}, \{\pi_1, j\}_j, \{p_1, j\}_j \| \rho_1 \| \dots \| \rho_{r-1})$: $\tilde{\mathsf{P}}$ outputs $\{\pi_{r,j}\}_j$ and $\{p_{r,j}\}_j$; the challenger runs $\mathsf{I}(\mathbb{F}, \mathbb{i})$ and V performs the decision phase of the PIOP. The challenger sets cvs to be the final cvs, sets the decision bit d as the output of the verifier V and halts.
   (c) If (2) the verifier samples ρ_1 and sends it to $\tilde{\mathsf{P}}$; the game appends the state $\mathsf{cvs}' := (\mathbb{i}, \mathbb{x}, \{\pi_{1,j}\}_j, \{p_{1,j}\}_j \| \rho_1)$ to the list $\mathsf{SeenStates}$.
3. The game computes the extraction bit $b \overset{\text{def}}{=} (\mathbb{i}, \mathbb{x}, \mathcal{E}(\mathsf{cvs})) \in \mathcal{R}$ where the index $\mathbb{i}$ and the instance $\mathbb{x}$ are the ones included in the final cvs. The game returns $(d \wedge \neg b)$, i.e., the malicious prover convinces the verifier but the extractor fails.

Fig. 5. The state-restoration soundness experiment $\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\mathsf{IP},\mathcal{E}}(\mathbb{F})$.

Definition 22 (State-restoration (straight-line) knowledge-soundness). *Let* $\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\mathsf{IP},\mathcal{E}}(\mathbb{F})$ *be the experiment in Fig. 5. A PIOP* IP *is* state-restoration (straight-line) knowledge-sound *if there exists an extractor* $\mathcal{E}$ *such that for any* $\tilde{\mathsf{P}}$ *and any* $\mathbb{F}$*:*

$$\Pr\left[\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\mathsf{IP},\mathcal{E}}(\mathbb{F}) = 1\right] \leq \mathsf{negl}(|\mathbb{F}|)$$

6.1 Sumcheck-based PIOP

We recall that a sumcheck PIOP is a multivariate PIOP for the oracle relation $\mathcal{R}_{\mathrm{SUM}}$ that consists of all the tuples $(\mathbb{x}, \mathbb{w})$ where $\mathbb{x} = (v, [\![f]\!])$ and $\mathbb{w} = f$ is a μ-variate polynomial of degree d such that $\sum_{\boldsymbol{b}\in\{0,1\}^\mu} f(\boldsymbol{b}) = v$.

Sumcheck Protocol with high-degree round polynomials. Given a pair $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathrm{SUM}}$:

- Set $v_0 \leftarrow v$
- For $i = 1, \ldots, \mu$:
  - The prover computes $q_i(X) \coloneqq \sum_{\boldsymbol{b}\in\{0,1\}^{\mu-i-1}} f(\boldsymbol{b}, X, r_{i+1}, \ldots, r_\mu)$ and sends the univariate oracle polynomial $[\![p_i]\!]$, of degree at most $d-2$, to the verifier, along with the value $q_i(0)$, where
  $$p_i(X) \coloneqq \tfrac{q_i(X)-(1-X)q_i(0)-Xq_i(1)}{X(1-X)}$$
  - The verifier computes $q_i(1) \leftarrow v_{i-1} - q_i(0)$, and sets $v_i \leftarrow p_i(r_i)(1-r_i)r_i + (1-r_i)q_i(0) + r_i q_i(1)$. Then, samples $r_i \leftarrow\!\!\$\ \mathbb{F}$, and sends r_i to the prover.
- The verifier accepts if $f(r_1, \ldots, r_\mu) = v_\mu$

This protocol was proposed by [18] and reduces the communication and verifier complexity with respect to the standard sumcheck protocol of [51].

In this work, we restrict our attention to a particular class of the multivariate PIOPs, namely those in which the prover and the verifier reduce to one or more invocations of the *sumcheck PIOP* defined above. This kind of PIOP is not only very flexible, but it has inerently some "structure" that is retained in the compilation to a zkSNARK and plays a key role in the strong non-malleability properties of the compiled object: as we elaborate next, the compilation of this kind of PIOP results into a simulation-extractable zkSNARK as long as it has some properties that seem natural and easy to check.

Definition 23 (Sumcheck-based multivariate PIOP). *A* (k_1, k_2, k_3, t, ν) *sumcheck-based PIOP is a PIOP where:*

1. *The indexer* I *returns a single* ν*-variate multilinear oracle polynomial* $[\![p_0]\!]$*.*
2. *For all* $i \in [k_1]$*,* P *sends a* ν*-variate multilinear oracle polynomial* $[\![p_i]\!]$*.* V *sends challenge* ρ_i*.*
3. *At the end of the* k_1*-th round,* P *and* V *sequentially run* k_2 *sumcheck PIOPs.*
   For all $i \in [k_2]$*, the* i*-th sumcheck is over a polynomial* $f_i \coloneqq h_i(g_{i,1}, \ldots, g_{i,c_i})$ *where* $\mathsf{deg}(f_i) = \nu_i$*, the polynomial* h_i *is a public polynomial and each* g_j *is a multilinear polynomial obtained by partially evaluating one of the* k_1 *polynomials sent by the prover in the first phase. Namely, there exists indexes* $k_{i,j}$*, sets* $S_{i,j}$ *and boolean vectors* $\mathbf{b}_{i,j}$ *(possibly function of* $(\rho_j)_{j\in[k_1]}$*) such that* $g_{i,j} \coloneqq (p_{k_{i,j}})_{|X_{S_{i,j}}=\boldsymbol{b}_{i,j}}$*.*
   The i*-th sumcheck protocol defines the following verifier's queries:*

- *If the i-th sumcheck is an* high degree *sumcheck, namely $\nu_i > t$, we associate with the i-th sumcheck protocol the oracles $(\llbracket p_{i,j} \rrbracket)_j$ where $\llbracket p_{i,j} \rrbracket$ is the round oracle sent at the j-th round and we define the verifier answers with randomness $r_{i,j}$. The i-th sumcheck protocol execution defines verifier's queries:*

$$Q'_i := \{(\llbracket p_{i,j} \rrbracket, r_{i,j}, v_{i,j}) : j \in [\nu_i]\}$$

- *A query to the (virtual) oracle polynomial $\llbracket f_i \rrbracket$ on the point $\boldsymbol{r}_i$. Namely, the prover sends evaluation v_i and the verifier queries $(\llbracket f_i \rrbracket, \boldsymbol{r}_i, v_i)$.*
  For any j, let $\bar{\boldsymbol{r}}_{i,j}$ be such $\bar{\boldsymbol{r}}_{|S_j} = \boldsymbol{b}_j$ and $\bar{\boldsymbol{r}}_{|\bar{S}_j} = \boldsymbol{r}_i$, the virtual oracle query defines the following queries to the oracle polynomials:

$$Q_i := \{(\llbracket p_{k_{i,j}} \rrbracket, \bar{\boldsymbol{r}}_{i,j}, \bar{g}_{i,j}) : j \in [c_i]\}$$

  and, additionally, the verifier checks $h_i(\bar{g}_{i,1}, \ldots, \bar{g}_{i,c_i}) = v_i$.

Let $\bar{k}_2$ be the number of high-degree sumechecks.

4. *Moreover, for all $j \in [k_3]$, V queries one of the oracles $\{\llbracket p_1 \rrbracket, \ldots, \llbracket p_{k_1} \rrbracket\}$ on arbitrary point $\bar{\boldsymbol{r}}_{k_2+1,j} \in \mathbb{F}^\nu$. These queries define a set Q_{k_2+1} of verifier's queries of the form*

$$\left(\left\llbracket p_{k_{k_2+1,j}} \right\rrbracket, \bar{\boldsymbol{r}}_{k_2+1,j}, \bar{g}_{k_2+1,j}\right)_{j \in [k_3]}$$

The points $(\bar{g}_{k_2+1,j})_j$ are deterministically derived from $(\bar{\boldsymbol{r}}_{k_2+1,j})_j$ and the public oracle p_0.

It is easy to check that, without further optimizations, the query complexity of a (k_1, k_2, k_3, t, ν)-sumcheck-based PIOP is at least $\Omega(k_2 + k_3)$ and it can be $O(k_2 \cdot \nu + k_3)$ in the worst case when $\nu_i > t$ for all i.

7 Sumcheck-based zkSNARks are simulation-extractable

In this section we show the conditions under which a multivariate sumcheck-based PIOP can be compiled to a simulation-extractable zkSNARK.

7.1 The efficiency compiler for sumcheck-based PIOP

The work of [18] introduces two optimizations that reduce the number of verifier queries to a single linear virtual query to a univariate polynomial and a single linear virtual query to a ν-variate multi-linear polynomial.

The compiler in Algorithm 2 formalizes and slightly improves these optimizations. We analyze the compiler and formally prove the state-restoration soundness of our modifications. Additionally, we show that the compilation process enhances security by leveraging a simpler structural property of the original PIOP, which we explain next.

We consider a sumcheck-based PIOP where the k_2 sumcheck protocols jointly involve all oracle polynomials sent in the first k_1 messages. Specifically, we define the set of indexes

$$\mathcal{I}(\mathsf{IP}) := \{k_{i,j} \mid i \in [k_2], j \in [c_i]\},$$

where the indexes $k_{i,j}$ are defined in Item 3 of Definition 23. We require $\mathcal{I}(\mathsf{IP}) = [k_1]$, and we call this a *compiler-safe* sumcheck-based PIOP. In practice, any reasonable sumcheck-based PIOP

Algorithm 2 Efficiency Compiler $\Sigma_{\texttt{Batch}}[\mathsf{IP}]$

```
1: P and V run IP for $(\mathbb{x},\mathbb{w}) \in \mathcal{R}$, V does not make any oracle query.
2: Let $\forall i \in [k_2] : (p_{i,j})_{j\in[\nu]}, \boldsymbol{r}_i, (v_{i,j})_{j\in[\nu]}, (p_{k_{i,j}})_{j\in[c_1]}, (\bar{\boldsymbol{r}}_{i,j})_{j\in[c_i]}, (\bar{g}_{i,j})_{j\in[c_i]}$ be as in Definition 23.
   /* Halo Infinite's batch polynomial evaluations for $r_{i,j}$ pair-wise different: */
3: Let $T := \{r_{i,j} : i \in [k_2], \nu_i > t, j \in [\nu_i]\}$, and let $T_{i,j} := T \setminus \{r_{i,j}\}$, define the polynomial
       $$f(X) := \sum_{i,j} Z_{T_{i,j}}(X)(p_{i,j}(X) - v_{i,j})$$
4: P sends the oracle polynomial $q \leftarrow f/Z_T$.
5: V samples a challenge $z \leftarrow\$ \mathbb{F}$ and queries the linear virtual (univariate) polynomial $[\![l]\!]$ on $z$ evaluating to 0:
       $$l(X) := \sum_{i,j} Z_{T_{i,j}}(z)(p_{i,j}(X) - v_{i,j}) - q(X)Z_T(z) \quad (6)$$
   /* HyperPlonk's batch multivariate polynomial evaluations */
6: V sends to P a random vector $\boldsymbol{t} \leftarrow\$ \mathbb{F}^\ell$ and $\ell = \lceil \log(k_2 + 1 + \bar{k}) \rceil$ where $\bar{k} = \max(k_2, k_3)$.
7: Define sum $s := \sum_{i\in[k_2+1],j\in[c_i]} eq(\boldsymbol{t}, \langle i\|j\rangle) \cdot \bar{g}_{i,j}$ where $\langle i\|j\rangle \in \{0,1\}^\ell$.
8: Let $h$ be the MLE for $(h_{i,j,\boldsymbol{b}})_{i\in[k_2+1],j\in[\bar{k}],\boldsymbol{b}\in\{0,1\}^\mu}$, where:
       $$h_{i,j,\boldsymbol{b}} := \begin{cases} eq(\boldsymbol{t}, \langle i\|j\rangle) \cdot p_{k_{i,j}}(\boldsymbol{b}) & j \le c_i \\ 0 & j > c_i \end{cases}$$
9: Let $\tilde{eq}$ be the MLE for $(eq(\boldsymbol{b}, \bar{\boldsymbol{r}}_{i,j}))_{i\in[k_2+1],j\in[\bar{k}],\boldsymbol{b}\in\{0,1\}^\mu}$ such that
       $$\tilde{eq}(\langle i\|j\rangle, \boldsymbol{b}) = \begin{cases} eq(\bar{\boldsymbol{r}}_{i,j}, \boldsymbol{b}) & j \le c_i \\ 0 & j > c_i \end{cases}$$
10: P and V run a sumcheck PIOP for $(s, [\![w]\!]; w) \in \mathcal{R}_{\mathrm{SUM}}$, where $w := h \cdot \tilde{eq}$. Let $(\boldsymbol{a}_1, \boldsymbol{a}_2) \in \mathbb{F}^{\ell+\mu}$ be the sumcheck challenge vector and $\bar{w}$ be the claimed value for $w(\boldsymbol{a}_1, \boldsymbol{a}_2)$ at the end of the sumcheck protocol.
11: V verifies that $[\![w]\!]$ at $(\boldsymbol{a}_1, \boldsymbol{a}_2)$ evaluates to the claimed value $\bar{w}$, namely:
    – V locally computes $\bar{h} \leftarrow \bar{w}/\tilde{eq}(\boldsymbol{a}_1, \boldsymbol{a}_2)$.
    – V queries that the linear virtual polynomial $[\![l']\!]$ evaluates to $\bar{h}$ at $\boldsymbol{a}_2$ where:
       $$l'(\boldsymbol{X}) := \sum_{i,j} \tilde{eq}(\langle i\|j\rangle, \boldsymbol{a}_1) \cdot eq(\boldsymbol{t}, \langle i\|j\rangle) \cdot p_{k_{i,j}}(\boldsymbol{X}). \quad (7)$$
```

satisfies this property. To illustrate, suppose a PIOP does not satisfy this condition. In such a case, there exists a polynomial p_{i^*} that is not queried by any virtual polynomial query of the sumcheck protocols. We can then redefine the PIOP to exclude the transmission of p_{i^*} while keeping the rest of the protocol unchanged. This results in a more efficient PIOP than the original one.

It is important to note that this reasoning assumes the prover is aware of which oracle polynomials will be involved in the sumchecks, which is a natural assumption in most practical scenarios.

Theorem 4. *Let* IP *be a PIOP for* $\mathcal{R}$ *and consider the PIOP* $\Sigma_{\texttt{Batch}}[\mathsf{IP}]$ *as in Algorithm 2,* $\Sigma_{\texttt{Batch}}[\mathsf{IP}]$ *is a PIOP for* $\mathcal{R}$*, moreover:*

– *Let* $q_{\tilde{\mathsf{P}}}$ *be total number of the* $\tilde{\mathsf{P}}$ *during the state-restoration experiment,* d^* *be an upper bound to the degree of the polynomials the prover can commit to,* ϵ_{SUM} *(resp.* ϵ_{IP}*) be the state-restoration soundness error of the sumcheck protocol (resp. the* IP *protocol) then:*

$$\Pr\left[\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\Sigma_{\texttt{Batch}}[\mathsf{IP}],\mathcal{E}}(\mathbb{F})\right] \le q_{\tilde{\mathsf{P}}}\left(\frac{q_{\tilde{\mathsf{P}}} + d^* \cdot |T| + \ell}{|\mathbb{F}|} + \epsilon_{SUM}\right) + \epsilon_{\mathsf{IP}}.$$

– $\Sigma_{\texttt{Batch}}[\mathsf{IP}]$*'s query complexity is 2,*

Proof. Assume that IP is state-restoration knowledge-sound and there exist $\mathcal{E}$ and a function ϵ_{IP} such that for any $\tilde{\mathsf{P}}$ we have $\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\Sigma_{\texttt{Batch}}[\mathsf{IP}],\mathcal{E}}(\mathbb{F}) \leq \epsilon_{\mathsf{IP}}(\mathbb{F})$.

Let $\mathbf{H}_1$ be the same as $\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\Sigma_{\texttt{Batch}}[\mathsf{IP}],\mathcal{E}}(\mathbb{F})$ but where the hybrid returns $(d \wedge \neg b \wedge d_1)$ and d_1 is set to 1 if the values $r_{i,j}$ in the final cvs are pair-wise different for $i \in [k_2]$ and $j \in [\nu]$.

We have $\Pr\left[\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\Sigma_{\texttt{Batch}}[\mathsf{IP}],\mathcal{E}}(\mathbb{F})\right] \leq \Pr[\mathbf{H}_1] + \Pr[\neg d_1]$. Moreover, using a standard analysis of the birthday attack, we have that $\Pr[\neg d_1] \leq \frac{q^2_{\tilde{\mathsf{P}}}}{2|\mathbb{F}|}$, where $q_{\tilde{\mathsf{P}}}$ is the total number of queries of $\tilde{\mathsf{P}}$ during the state-restoration experiment.

Let $\mathbf{H}_2$ be the same as $\mathbf{H}_1$ but where the hybrid returns $(d \wedge \neg b \wedge d_1 \wedge d_2)$ and d_2 is set to 1 if $\forall i,j : p_{i,j}(r_{i,j}) = v_{i,j}$. Similarly to the previous hybrid, we have $\Pr[\mathbf{H}_1] \leq \Pr[\mathbf{H}_2] + \Pr[\neg d_2 \wedge d \wedge d_1]$. Thus we need to bound $\Pr[\neg d_2 \wedge d \wedge d_1]$. Notice such an event implies that $\exists i^*, j^* : p_{i^*,j^*}(r_{i^*,j^*}) \neq v_{i^*,j^*}$ and $l(z) = 0$ where we recall that :

$$l(z) = \sum_{i,j} Z_{T_{i,j}}(z)(p_{i,j}(z) - v_{i,j}) - q(z)Z_T(x)$$

Notice that the polynomial $Z_{T_{i^*,j^*}}(X)(p_{i^*,j^*}(X) - v_{i^*,j^*})$ is not divisible by $Z_T(X)$; morever, condtioned on $d_1 = 1$, the polynomials $Z_{T_{i,j}}(X)(p_{i,j}(X) - v_{i,j})$ are all independent. Therefore, for any polynomial $q'(X)$ the polynomial $\sum_{i,j} Z_{T_{i,j}}(X)(p_{i,j}(X) - v_{i,j}) - q'(X)Z_T(X)$ is non-zero. All these polynomials are defined by $\tilde{\mathsf{P}}$ before seeing the challenge $z \in \mathbb{F}$ contained in the final cvs; additionally, notice that, because of the state-restoration game, $\tilde{\mathsf{P}}$ has at most $q_{\tilde{\mathsf{P}}}$ chance to resample z. By union-bound and by the Schwartz-Zippel lemma we can conclude that

$$\Pr[\neg d_2 \wedge d \wedge d_1] \leq \frac{q_{\tilde{\mathsf{P}}} \cdot d^* \cdot |T|}{|\mathbb{F}|}$$

where d^* is un upper bound to the degree of a polynomial the prover can commit to.

Let $\mathbf{H}_3$ be the same as $\mathbf{H}_2$ but where the hybrid sets d_3 to 1 if $(s, [\![w]\!]\,; w) \in \mathcal{R}_{\text{SUM}}$ and returns $(d \wedge \neg b \wedge \wedge_{i\in[3]} d_i)$. Let ϵ_{SUM} be the state-restoration knowledge soundness of the sumcheck protocol for polynomial w of degree 2, it is clear that $\Pr[\mathbf{H}_2] \leq \Pr[\mathbf{H}_3] + q_{\tilde{\mathsf{P}}} \cdot \epsilon_{\text{SUM}}$.

Let $\mathbf{H}_4$ be the same as $\mathbf{H}_3$ but where the hybrid sets d_4 to 1 if forall $k_{i,j}$ we have $p_{k_{i,j}}(\bar{\boldsymbol{r}}_{i,j}) = \bar{g}_{i,j}$. We bound $\Pr[d_3 \wedge \neg d_4]$. In particular, by definition of h, $\tilde{eq}$ and s, and by $(e, [\![w]\!]\,; w) \in \mathcal{R}_{\text{SUM}}$, we have:

$$\sum_{i,j} eq(\boldsymbol{t}, \langle i\|j\rangle) \cdot p_{k_{i,j}}(\bar{\boldsymbol{r}}_{i,j}) = \sum_{i,j,\boldsymbol{b}} w(\langle i\|j\rangle, \boldsymbol{b}) = s = \sum_{i,j} eq(\boldsymbol{t}, \langle i\|j\rangle) \cdot \bar{g}_{i,j}.$$

Consider the polynomial $g(\boldsymbol{T}) = \sum_{i,j} eq(\boldsymbol{T}, \langle i\|j\rangle) \cdot (p_{k_{i,j}}(\bar{\boldsymbol{r}}_{i,j}) - \bar{g}_{i,j})$. Because of $\neg d_4$ we have that $g(\boldsymbol{T})$ is non-zero because at least one of the coefficient $(p_{k_{i,j}}(\bar{\boldsymbol{r}}_{i,j}) - \bar{g}_{i,j}) \neq 0$. Moreover, by the equation above we have $g(\boldsymbol{t}) = 0$. Thus, applying the Schwartz-Zippel lemma over a multi-linear polynomial with ℓ variables, and the union bound we have $\Pr[d_3 \wedge \neg d_4] \leq \frac{q_{\tilde{\mathsf{P}}} \cdot \ell}{|\mathbb{F}|}$.

Finally, let cvs be the finalized cvs in the hybrid game $\mathbf{H}_4$, and let $\mathsf{cvs}_{\mathsf{IP}}$ the prefix of cvs that contains only the protocol executions of IP, we observe that $d \wedge d_2 \wedge d_4$ implies that the verification procedure of IP would accept on the $\mathsf{cvs}_{\mathsf{IP}}$. We define $\tilde{\mathsf{P}}_{\mathsf{IP}}$ be the state-restoration knowledge-soundness adversary for IP that simply runs $\tilde{\mathsf{P}}$ and simulates the verifiers messages for the last two challenges (Item 5 and Item 6 of Algorithm 2). The adversary $\tilde{\mathsf{P}}_{\mathsf{IP}}$ additionally asserts the events defined by the bits d_1 and d_3, and if such assertion fails, it returns an error message.

We have that $\Pr[\mathbf{H}_4] = \Pr\left[\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}}_{\mathsf{IP}},\mathsf{IP},\mathcal{E}_{\mathsf{IP}}}(\mathbb{F})\right]$. Chaining the disequations together, we have:

$$\Pr\left[\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\Sigma_{\mathtt{Batch}}[\mathsf{IP}],\mathcal{E}}(\mathbb{F})\right] \leq q_{\tilde{\mathsf{P}}}\left(\frac{q_{\tilde{\mathsf{P}}} + d^* \cdot |T| + \ell}{|\mathbb{F}|} + \epsilon_{\mathrm{SUM}}\right) + \epsilon_{\mathsf{IP}}.$$

7.2 The zero-knowledge compiler for sumcheck-based PIOP

We describe and improve the compiler from [18] that transforms a sumcheck-based PIOP into one that is zero-knowledge. Their general framework consists of two parts. The first part is to mask the oracle polynomials so that their oracle query answers do not reveal the information of the original polynomial in a way that the evaluations over the boolean hypercube remain equivalent. The second part is making the underlying sumcheck PIOP zero-knowledge. Our improvement is in the second part, where we improve the ZK sumcheck of Xie *et al.* [64] for the high-degree case when the round-polynomials are sent as oracles.

Polynomial masking. We refer the reader to Definition 10 for the definition of a masking algorithm, Algorithm 1 for the specific masking algorithm used in this work, and Lemma 1 for the corresponding lemma.

We describe the zero-knowledge compiler in Algorithm 3. As in [18], for polynomials $f, g \in \mathcal{F}_{d,\mu}$ we denote with $\mathsf{merge}(f, g)$ the polynomial in $\mathcal{F}_{d,\mu+1}$ as:

$$\mathsf{merge}(f, g) := (1 - X_0) \cdot f(X_1, \ldots, X_\mu) + X_0 \cdot g(X_1, \ldots, X_\mu)$$

Abusing of notation, we define $\mathsf{merge}((f_i)_{i\in[n]})$ as the polynomial h such that $h(\langle i\rangle, X_1, \ldots, X_\mu) = f_i(X_1, \ldots, X_\mu)$ where $\langle i\rangle$ is the $\log n$-bits binary decomposition of $i \in \mathbb{N}$.

Definition 24. *For any j, i and ℓ, we say that the variable ℓ of the oracle polynomial p_j is* free *if, for any k, let us consider the k-th sumcheck protocol, then either the polynomial p_j does not appear in the k-th sumcheck or, if it appears, let j' be the index such that $g_{i,j'} = p_j$ we have $\ell \notin S_{i,j'}$ where the set $S_{i,j'}$ is the of indexes of the variables assigned in $g_{i,j'}$ (see Item 3 of Definition 23).*

We now provide a definition of zero-knowledge for sumcheck-based PIOPs that our compiler can achieve. This definition is tailored to PIOP-to-zkSNARK compilations where the underlying polynomial commitment scheme is not hiding and, in particular, leaks an evaluation point. It requires that the zero-knowledge simulator (1) reproduces the information leaked by the non-hiding commitments, and (2) fully simulates the round polynomials in the sumcheck protocol.

Definition 25 (Zero-Knowledge for sumcheck-based PIOP). *A sumcheck-based PIOP IP is ϵ-zero-knowledge if there exists a simulator $\mathcal{S}$ such that $\mathcal{S}$ on input the description of the field $\mathbb{F}$, the index $\mathbb{i}$, the input $\mathbb{x}$ and a vector $\boldsymbol{\beta}$, outputs a simulated view $\tilde{\mathsf{view}}$ and simulated round polynomials $(\tilde{p}_{i,j}(X))_{i\in[k_1],j\geq 0}$, and for every index $\mathbb{i}$, and $(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, the following random variables are within ϵ statistical distance:*

$$\mathbf{REAL}_{\mathsf{IP}}(\mathbb{F}, \mathbb{i}, \mathbb{x}) := \left(\boldsymbol{\beta}, \mathsf{view}\left(\mathsf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w}), \mathsf{V}^{\mathsf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})\right), (p_j(\boldsymbol{\beta}))_j, (p_{i,j}(\beta_1))_{i,j,\nu_i>t}\right)$$

$$\mathbf{IDEAL}_{\mathcal{S}}(\mathbb{F}, \mathbb{i}, \mathbb{x}) := \left(\boldsymbol{\beta}, \tilde{\mathsf{view}}, (\tilde{p}_j)_j, (\tilde{p}_{i,j}(\beta_1))_{i,j,\nu_i>t} \;:\; \begin{array}{l}\tilde{\mathsf{view}}, (\tilde{p}_{i,j})_{i,j} \leftarrow \mathcal{S}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \boldsymbol{\beta}),\\ \forall k : \tilde{f}_k \leftarrow\!\!\$\; \mathbb{F}\end{array}\right)$$

Algorithm 3 ZK Compiler for Sumcheck-Based PIOPs $\Sigma_{\mathsf{ZK}}[\mathsf{IP}]$

```
1: The prover P set $\tilde{h}_0 := \sum_{j\in[\mu]} h_{0,j}(X_0, X_j)$ and $\tilde{h}_i := \sum_{j\in[\mu_i]} \tilde{h}_{i,j}(X_j)$, where, for any $i \in [k_2]$ and for any $j$, the polynomial $\tilde{h}_{0,j}(X_0, X_j) \leftarrow\$ \mathbb{F}_{\le(k_2,3)}[X_0, X_j]$ and the polynomial $\tilde{h}_{i,j}(X_j) \leftarrow\$ \mathbb{F}_{\le t}[X_j]$ when $\nu_i < t$ or $\tilde{h}_{i,j} \leftarrow 0$.
2: P sends oracle polynomial $\tilde{h}$ where $\tilde{h} \leftarrow \mathsf{merge}((\tilde{h}_j)_{j\in[0,k_2]})$ and messages $(\tilde{H}_i)_{i\in[0,k_2]}$ where $\tilde{H}_i = \sum_{\boldsymbol{b}\in B_{\mu_i}} \tilde{h}_j(\boldsymbol{b})$.
3: P and V run IP for $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ with the following changes.
4: for $i = 1$ to $k_1$ when P outputs oracle $p_i$, send oracle $p_i^* \leftarrow \mathsf{msk}(p_i, t_i + 1, \ell_i)$.
5: for $i = 1$ to $k_2$ do
6:     Receive challenge $\rho_i$, and let $\bar{f}_i := h_i(g_1^*, \dots, g_{c_i}^*)$.
7:     if $\nu_i < t$ then
8:         Let $f_i^* := \bar{f}_i + \rho_i \cdot \tilde{h}_i$ run low-degree sumcheck on $(f_i^*, H_i + \rho\tilde{H}_i)$
9:     else
10:        Let $f_i^* := \bar{f}_i(X_1, \dots, X_{\mu_i}) + \rho_i \cdot \tilde{h}_0(X_0, X_1, \dots, X_\mu)$ run high-degree sumcheck on $(f_i^*, H_i + \rho\tilde{H}_0)$
11:    end if
12: end for
13: for $i = 1$ to $k_3$ do
14:     Verifier queries $p_{j_i}^*(c_i)$ and checks it equals $y_i$
15: end for
```

We assume μ_i be the number of variables in the virtual oracle polynomial $\bar{f}_i$ for the i-th sumecheck in IP and $\mu = \max_i \mu_i$. For any $i \in [k_1]$, we let ℓ_i be a free variable of p_i and t_i be the number of appearances of p_i, namely $t_i := |\{i' : \exists i', j', k_{i',j'} = i\}|$.

where $(p_j)_j$ are the multivariate polynomials sent in the first k_1 rounds and $(p_{i,j})$ the univariate round polynomials returned by the prover P *and $\boldsymbol{\beta}$, in both the random variables, is a vector with coefficient in $\mathbb{F}$ sampled uniformly at random.*[12] *When ϵ is a negligible function of $|\mathbb{i}|+|\mathbb{x}|$ we simply say that* IP *is zero-knowledge.*

Theorem 5. *Let* IP *be a sumcheck-based PIOP and assume that, for any $i \in [k_1]$ there exists a value ℓ_i such that the variable ℓ_i of the oracle polynomial p_i is free. Let $\mathsf{IP}' := \Sigma_{\mathsf{ZK}}[\mathsf{IP}]$ be the sumcheck-based PIOP in Algorithm 3, then* IP′ *is zero-knowledge. Moreover, if* IP *is state-restoration sound then so is* IP′. *In particular, let $q_{\tilde{\mathsf{P}}}$ be total number of the $\tilde{\mathsf{P}}$ during the state-restoration experiment, and ϵ_{IP} be the state-restoration soundness error of the* IP *protocol then:*

$$\Pr\left[\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\mathsf{IP}',\mathcal{E}}(\mathbb{F})\right] \le \frac{q_{\tilde{\mathsf{P}}}}{|\mathbb{F}|} + \epsilon_{\mathsf{IP}}$$

Proof. To prove zero-knowledge we can assume the existence of a zero-knowledge simulator for the low-degree sumcheck derived from [64,18]. In particular, let Prove_{lSC} be the proving algorithm of the low-degree sumcheck, there exists a simulator $\mathcal{S}_{lSC}$ such that for any $\bar{f}_i, \tilde{h}_i$ and any $\rho_i \neq 0$ such that $\tilde{h}_i$ is sampled as described in Algorithm 3, the distributions below are indistinguishable:

$$\mathsf{Prove}_{lSC}(f_i^*, H_i + \rho_i\tilde{H}_i) \approx \mathcal{S}_{lSC}(H_i + \rho_i\tilde{H}_i, (\bar{g}_{i,j})_{j\in[c_i]}) \tag{8}$$

where, we recall that the values $\bar{g}_{i,j} \in \mathbb{F}$ are the evaluations of the oracle polynomials defining the virtual polynomial f_i, namely, $p_{k_{i,j}}(\bar{\boldsymbol{r}}_{i,j}) = \bar{g}_{i,j}$ for any j (see Definition 23) and $\boldsymbol{r}_i$ is the vector

[12] The choice of the variable β_1 for evaluating the polynomials $p_{i,j}$ (resp. $\tilde{p}_{i,j}$) is arbitrary. For convenience, we assume that univariate polynomials are associated with the variable X_1.

of challenges in the simulated view while the vector $\bar{\boldsymbol{r}}_{i,j}$ are the vectors defined assigning partially some of the position in the vector $\boldsymbol{r}_i$. We define a zero-knowledge simulator for the high-degree sumchecks. The simulator $\mathcal{S}_{hSC}$, on input $(S, (\bar{g}_{i,j})_j)$, samples a random polynomial $\tilde{f}$ with degree $\mathsf{deg}_{X_j}(\tilde{f}) = 6$ for any j conditioned on $\sum_{\boldsymbol{b}\in B_{\mu+1}} \tilde{f}(\boldsymbol{b}) = S$ and $\tilde{f}(\boldsymbol{r}_i) = h_i(\bar{g}_{i,1}, \dots, \bar{g}_{i,c_i})$ and then run the high-degree sumcheck protocol described in Section 6.1. We show that for any $(\bar{f}_i)_{i\in[k_2],\nu_i>t}$ and for $\tilde{h}_0$ sampled as described in Algorithm 3 the distributions below are indistinguishable:

$$\Big(\beta_1, \mathsf{Prove}_{hSC}(f_i^*, H_i + \rho_i \tilde{H}_i), (\tilde{p}_{i,j}(\beta_1))_{j\in[0,\mu_i]}\Big)_{i\in[k_2],\nu_i>t} \tag{9}$$
$$\approx$$
$$\Big(\beta_1, \mathcal{S}_{lSC}(H_i + \rho_i \tilde{H}_i, (\bar{g}_{i,j})_{j\in[c_i]}), (\tilde{p}_{i,j}(\beta_1))_{j\in[0,\mu_i]}\Big)_{i\in[k_2],\nu_i>t}$$

where $\beta_1 \leftarrow\!\!\$\, \mathbb{F}$ is uniformly random. Notice that the transcript of the i-th (high-degree) sumcheck proof contains elements $(e_{i,k,0}, e_{i,k,1}, e'_{i,k})_{k\in[0,\mu_i]}$ where:

- $e_{i,k,b'} = \sum_{\boldsymbol{b}\in B_{\mu-k}} f_i^*(r_1, \dots, r_k, \boldsymbol{b}) + \sum_{\boldsymbol{b}\in B_{\mu-k}} \tilde{h}_0(r_0, r_1, \dots, r_{k-1}, b', \boldsymbol{b})$ for $b' \in \{0,1\}$
- $e'_{i,k} = \sum_{\boldsymbol{b}\in B_{\mu-k}} f_i^*(r_1, \dots, r_k, \boldsymbol{b}) + \sum_{\boldsymbol{b}\in B_{\mu-k}} \tilde{h}_0(r_0, r_1, \dots, r_k, \boldsymbol{b})$.

Moreover, the k-th round polynomial is masked with:

$$\begin{aligned} m_{i,k}(X) &:= \sum_{\boldsymbol{b}\in B_{\mu-k}} \tilde{h}_0(r_{i,0}, r_{i,1}, \dots, r_{i,k-1}, X, \boldsymbol{b}) \\ &= \sum_{j<k} \tilde{h}_{0,j}(r_0, r_j) + \tilde{h}_{0,k}(r_{i,0}, X) + \sum_{\boldsymbol{b}\in B_{\mu-k}} \sum_j \tilde{h}_0(r_{i,0}, b_j) \end{aligned}$$

where $\boldsymbol{r}_k = (r_{k,j})_{j\in[0,\mu_i]}$. Notice that when the values $(r_{j,0})_j$ are all distinct then the linear map from the coefficients of $\tilde{h}_{0,k}(X_0, X)$ to $(\tilde{h}_{0,k}(r_{0,j}, X))$ is invertible, as in fact, it is a Vandermonde's matrix. Thus there is a bijection between evaluations of the mask $(m_{i,k}(X))_{i,k}$ and the evaluations of the polynomial $\tilde{h}_0$, moreover, the total degree of $\tilde{h}_0$ is $4 \times \bar{k}_2$ where we recall that $\bar{k}_2 = |\{i : \nu_i > t\}|$ and thus the four evaluations $(m_{i,k}(\beta_1), m_{i,k}(0), m_{i,k}(1), m_{i,k}(r_{i,j}))$ of the k-th round polynomial are distributed uniformly at random (unless $\beta_1 = r_{i,j}$ which happens only with negligible probability). Therefore, the evaluations of the round polynomials $\tilde{p}_{i,k}$ are uniformly distributed.

Finally, we can define the zero-knowledge simulator for the sumcheck-based PIOP.

The simulator $\mathcal{S}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \boldsymbol{\beta})$:

- Honestly generate the public polynomial p_0.
- Pick arbitrary polynomials $(\tilde{p}_i^*)_{i\in[k_1]}$ conditioned on the sumcheck relations over $f_1, \dots, f_{k_2}$ to hold.
- Sample the mask polynomials $\tilde{h}_i$ for $i \in [0, k_2]$ as described in Algorithm 3.
- Run the simulators $\mathcal{S}_{lSC}$ and $\mathcal{S}_{hSC}$ where, in the i-th sumcheck, if it is low-degree, we run the simulator on $\mathcal{S}_{lSC}(H_i + \rho_i \tilde{H}_i, (\bar{g}_{i,j})_j)$ else, if it is high-degree, we run the simulator $\mathcal{S}_{hSC}(H_i + \rho_i \tilde{H}_0, (\bar{g}_{i,j})_j)$. Where, in both cases, $\bar{g}_{i,j} \leftarrow \mathbb{F}$ for any $j \in [c_i]$. The high-degree sumcheck simulator additionally outputs the round polynomials $\tilde{p}_{i,k}(X)$ for $i \in [k_2]$ and $\nu_i > t$ and $k \in [0, \mu_i]$.
- Return the view consisting of the k_2 simulated sumcheck protocol executions and return the polynomials $(\tilde{p}_{i,j})_{i,j:\nu_i>t,j\in[0,\mu_i]}$.

We set $\mathbf{H}_0 := \mathbf{IDEAL}_{\mathcal{S}}(\mathbb{F}, \mathbb{i}, \mathbb{x})$, and consider the following list of hybrids:

- $\mathbf{H}_1$ the same as $\mathbf{H}_0$ but where $\tilde{p}_i^*$ are sampled by first choosing arbitrary polynomials $\tilde{p}_i$ conditioned on the sumcheck relations and then $\tilde{p}_i^* \leftarrow \mathsf{msk}(\tilde{p}_i, t_i + 1, \ell_i)$. Moreover, the values $\bar{g}_{i,j}$ are set as $\tilde{p}^*_{k_{i,j}}(\bar{\boldsymbol{r}}_{i,j})$ and the random values $\tilde{f}_i$ are subsituted with $\tilde{p}_i^*(\boldsymbol{\beta})$. We prove that $\mathbf{H}_0 \approx \mathbf{H}_1$. Notice that the number of queries to the polynomials $\tilde{p}_i^*$ is, by definition, bounded by t_i, on top of this, we can add the extra query on point $\boldsymbol{\beta}$. By definition, the index ℓ_i is free (see Definition 24), thus, with overwhelming probability, all the queries to i-th polynomial have distinct values at position ℓ_i. Therefore, we can apply Lemma 1 to prove the indistinguishability.
- $\mathbf{H}_2$ same as $\mathbf{H}_1$ but the polynomials $(\tilde{p}_i)_{i\in[k_1]}$ are replaced with the polynomials defined by running the prover of the protocol. The two views are equivalent because the view of the verifier does not change at all.
- $\mathbf{H}_3$ is the same as $\mathbf{H}_2$ but we run the prover for the low-degree sumcheck protocols instead of running the simulator. The two hybrids are indistinguishable because of the zero-knowledge property showed by [64,18] and described above.
- $\mathbf{H}_4$ is the same as $\mathbf{H}_3$ but we run the prover for the high-degree sumecheck protocols instead of running the simulator. Similarly to before, the two view are indistinguishable.

Finally notice that $\mathbf{H}_4$ is equivalent to the $\mathbf{REAL}$ random variable.

Let $\bar{s}_i = \sum_{\boldsymbol{x}\in B_{\mu_i}} \bar{f}_i(\boldsymbol{x})$, $\tilde{s}_i = \sum_{\boldsymbol{x}\in B_\mu} \tilde{h}_i(\boldsymbol{x})$ for any i. Let $\delta_i(X) := H_i - \bar{s}_i + X(\tilde{H}_i - \tilde{s}_i)$ for $\nu_i < t$ and $\delta_i(X) := H_i - \bar{s}_i + X(\tilde{H}_i - \tilde{s}_0)$ for $\nu_i \geq t$ and for any i. For the state-restoration soundness, we notice that unless the bad event $\exists i : \delta_i(\rho_i) = 0$ but $\delta_i(X) \neq 0$, the state-restoration soundness of IP and IP' are the same. By union bound on the number of attempts, which is bounded by $q_{\tilde{\mathsf{P}}}$, and applying Swartz-Zippel noticing that δ_i are linear functions, we have the bound in the statement of the theorem.

7.3 Putting it all together

A sumcheck-based PIOP can be compiled to a zkSNARK by chaining the zero-knowledge compiler in Algorithm 3 (unless the sumcheck-based PIOP is already zero-knowledge) and the efficiency compiler in Algorithm 2, using PST and applying the Fiat–Shamir transform.

We describe the compiler $\Sigma_{\mathtt{ZK+Batch+PC+FS}}[\mathsf{IP}] := \Sigma_{\mathtt{FS+PC}} \circ \Sigma_{\mathtt{Batch}} \circ \Sigma_{\mathtt{ZK}}$ from sumcheck-based PIOP to zkSNARKs, where $\Sigma_{\mathtt{FS+PC}}[\mathsf{IP}]$ is the compiler described in Algorithm 4. In particular:

1. We define an instance $\mathbb{x}_{\mathtt{KZG}}$ for Eq. (6) in Algorithm 2 of Algorithm 2, belonging to the language associated with $\mathcal{R}_{\mathtt{geval}}$ for KZG, using the index polynomials $\mathbb{i}_{\mathtt{KZG}} = (Z_{T_{i,j}}(X))_{i,j}, Z_T$. We notice that the index polynomials are linearly independent. The instance $\mathbb{x}_{\mathtt{KZG}}$ specifies the evaluation of the commitments $(\mathsf{c}_{i,j})_{i,j}$ and the commitment $[q(\beta_1)]_1$ at an evaluation point z, which is sampled uniformly at random from $\mathbb{F}$ after the commitments have been included in the transcript.
2. We define an instance $\mathbb{x}_{\mathtt{PST}}$ for Eq. (7) in Algorithm 2 of Algorithm 2, belonging to the language associated with $\mathcal{R}_{\mathtt{geval}}$ for PST, using the index polynomials $\mathbb{i}_{\mathtt{PST}} = (\tilde{eq}(\langle i\|j\rangle, \boldsymbol{Z}_1) \cdot eq(\boldsymbol{Z}_2, \langle i\|j\rangle))_{i,j}$. We notice that the index polynomials are linearly independent. The instance $\mathbb{x}_{\mathtt{PST}}$ specifies the evaluation of the commitments $(\mathsf{c}_{k_{i,j}})_{i,j}$ where the index polynomials are evaluated at $\boldsymbol{z} = \boldsymbol{a}_1\|\boldsymbol{t}$ and the commitments at $\boldsymbol{a}_2$. Also in this case, both $\boldsymbol{a}_1, \boldsymbol{t}$ and $\boldsymbol{a}_2$ are defined after the commitments are included in the transcript.

Algorithm 4 Sumcheck-PIOP to zkSNARK compiler $\Sigma_{\texttt{FS+PC}}[\mathsf{IP}]$.
We assume that IP is compiled with $\Sigma_{\texttt{Batch}}$ (Algorithm 2).

```
1: procedure KGen(srs)
2:     Run $\mathsf{IP.I}(\mathbb{F}, \mathbb{i})$ returning $p_0$.
3:     return Output $\mathsf{srs}, \mathsf{vk} = [p_0(\boldsymbol{\beta})]_1$.
4: end procedure
5: procedure Prove(srs, $\mathbb{i}, \mathbb{x}, \mathbb{w}$)
6:     Set transcript $\mathtt{trns} \leftarrow (\mathsf{vk}, \mathbb{x})$.
7:     Run PIOP prover $\mathsf{IP.P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$.
8:     At each round $\mathsf{P}$ sends an oracle $[\![p]\!]$ and (possibly) a message $\boldsymbol{v}$,
       compute $\mathsf{c}_p \leftarrow [p(\boldsymbol{\beta})]_1$ and $\mathtt{trns} \leftarrow \mathtt{trns} \| (\mathsf{c}, \boldsymbol{v})$.
9:     Let $([\![l]\!], z, 0)$ be the linear virtual oracle query in Eq. (6) of Algorithm 2.
       Call the prover of $\Pi_{\texttt{KZG}}$ with indexes set to $\mathbb{i} = (Z_{i,j})_{i,j}, Z_T$
       and polynomials $(p_{i,j} - v_{i,j})_{i,j}, q$.
       Let $\pi_{\texttt{KZG}}$ be the resulting proof.
10:    Let $([\![l']\!], \boldsymbol{a}_2, \bar{h})$ be the linear virtual oracle query in Eq. (7) of Algorithm 2.
       Call the prover of $\Pi_{\texttt{PST}}$ with indexes $\mathbb{i} = (eq(\langle i\|j\rangle, \boldsymbol{Z}_1) \cdot eq(\boldsymbol{Z}_2, \langle i\|j\rangle))_{i,j}$,
       polynomials $(p_{k_{i,j}})_{i,j}$, evaluation point $\boldsymbol{z} \leftarrow (\boldsymbol{a}_1, \boldsymbol{t})$ and $\boldsymbol{x} \leftarrow \boldsymbol{a}_2$.
11:    return $\mathtt{trns} \| (\pi_{\texttt{KZG}}, \pi_{\texttt{PST}})$
12: end procedure
13: procedure Verify(vk, $\mathbb{x}, \pi$)
14:    Parse $\pi = \mathtt{trns} \| \pi_{\texttt{KZG}}, \pi_{\texttt{PST}}$,
15:    From trns and RO recompute interaction of PIOP prover and verifier.
16:    Call $\Pi_{\texttt{KZG}}$ verifier on proof $\pi_{\texttt{KZG}}$ and $\mathbb{x}_{\texttt{KZG}} = \big(((\mathsf{c}_{p_{i,j}} - [v_{i,j}]_1)_{i,j}, \mathsf{c}_q), z, 0\big)$,
       and indexes $\mathbb{i}_{\texttt{KZG}} = (Z_{i,j})_{i,j}, Z_T$.
17:    Call $\Pi_{\texttt{PST}}$ verifier on proof $\pi_{\texttt{PST}}$ and $\mathbb{x}_{\texttt{PST}} = \Big(((\mathsf{c}_{p_{k_{i,j}}})_{i,j}, \mathsf{c}_q), (\boldsymbol{a}_1, \boldsymbol{t}), \boldsymbol{a}_2, \bar{h}\Big)$,
       and indexes $\mathbb{i}_{\texttt{PST}} = (\tilde{eq}(\langle i\|j\rangle, \boldsymbol{Z}_1) \cdot eq(\boldsymbol{Z}_2, \langle i\|j\rangle))_{i,j}$.
18: end procedure
```

We show that the resulting zkSNARK is weak simulation-extractable, namely, the adversary cannot forge a proof for a new instance without knowing its witness even in the presence of simulated proof for different (but arbitrary correlated) instances[13].

Definition 26. *Let Φ^{wSE} the policy that, upon input the forgery $\mathbb{x}^*, \pi^*$ and the view of the adverary, return 1 iff $\nexists \pi : (\mathbb{x}^*, \pi) \in \mathcal{Q}_{\mathsf{sim}}$. We say that a Π is weak simulation extractable in the AGM if it is Φ^{wSE}-simulation extactable.*

Theorem 6 (Simulation-extractable zkSNARK from sumcheck-based PIOPs). *Let IP be a sumcheck-based PIOP for a relation $\mathcal{R}$ that is state-restoration straightline knowledge-sound (see Definition 22) and such that $\mathcal{I}(\mathsf{IP}) = [k_1]$. The zkSNARK derived from the compiler $\Sigma_{\texttt{ZK+Batch+PC+FS}}[\mathsf{IP}]$ instantiated with PST is weak simulation-extractable in the AGM under the OMSDH assumption.*

Proof. Let $\mathsf{IP}^* := \Sigma_{\texttt{ZK+Batch}}[\mathsf{IP}]$. Let Q be the number of simulation queries issued by the adversary $\mathcal{A}_\Pi$. We start by defining the zero-knowledge simulator for the zkSNARK.

The simulator $\mathcal{S}$:

[13] One can achieve strong simulation extractability using the folklore technique of sampling a public key for a one-time signature, including it in the transcript, and signing the final proof.

- At setup stage, sample $\mathsf{srs}, \boldsymbol{\beta} \leftarrow \mathcal{S}_{\mathtt{PST}}$ honestly generate the public index polynomial p_0 and commit to it.
- For $j \in [Q]$, pick a fresh list of PST simulated commitments $\mathsf{coms}^{(j)} \leftarrow\$ \mathbb{G}_1^{k_1}$.
- On input $\mathbb{x}$:
  1. Run the zero-knowledge simulator of IP^*, and parse the output as k_2 simulated sumcheck protocol executions, including the list of round polynomials $(p_{i,k})_{i,k}$, together with the claimed evaluations of the oracle polynomials sent by the prover IP^*.
  2. For $i = 1$ to k_1, add to the transcript the i-th commitment $\mathsf{coms}_i^{(j)}$.
  3. For $i = 1$ to k_2, add to the transcript the honestly-generated commitments $[p_{i,j}(\beta_1)]_1$ of the round polynomials $p_{i,k}$ if $\nu_i > t$ (or the field elements defining such polynomials if the sumcheck is low-degree) and program the output of RO output of the transcript to match the verifier challenges included in the simulated proof.
  4. Finally, define the two final instances, say $\mathbb{x}_{\mathtt{KZG}}$ and $\mathbb{x}_{\mathtt{PST}}$, where $\mathbb{x}_{\mathtt{KZG}}$ is associated with the check in Eq. (6) and $\mathbb{x}_{\mathtt{PST}}$ is associated with the check in Eq. (7). For the former, run the honest prover of $\Pi_{\mathtt{KZG}}$, using as witness the round polynomials $(p_{i,k})_{i,k}$ generating a proof $\pi_{\mathtt{KZG}}$; for the latter, invoke the simulator of $\Pi_{\mathtt{PST}}$ generating a proof $\pi_{\mathtt{PST}}$.

     Add these two proofs to the transcript and output the transcript (without the verifier challenges) to the adversary.
- For RO queries issued by the adversary, output uniformly random elements, unless the RO output has been previously programmed during the proof simulation.

We notice that the zero-knowledge guarantees of the simulator follow from the zero-knowledge of IP^* and the distributional zero-knowledge of the scheme $\Pi_{\mathtt{PST}}$.

Lemma 9. *The zkSNARK derived from the compiler is zero-knowledge.*

Proof. Since we already defined a simulator we need only to show that the ideal and real views are indistinghuishable. We fix an adversary $\mathcal{A}_{zk}$ against zero-knowledge, we define a series of hybrids and let $\epsilon_i := \Pr[\mathbf{H}_i]$.

- $\mathbf{H}_0$ is the real-world experiment in which $\mathcal{A}_{zk}$ has oracle access to the real prover.
- $\mathbf{H}_1$ is the same as $\mathbf{H}_0$ but where the low-degree sumcheck protocols are simulated using the simulator $\mathcal{S}_{lSC}$ described in Eq. (8). We recall that $\mathcal{S}_{lSC}$, at its i-th call, takes as input the values $\bar{g}_{i,j} \in \mathbb{F}$ which are the evaluations of the oracle polynomials defining the virtual polynomial f_i, namely, $p_{k_{i,j}}(\bar{\boldsymbol{r}}_{i,j}) = \bar{g}_{i,j}$ for any j (see Definition 23) and $\boldsymbol{r}_i$ is the vector of challenges in the simulated view. Additionally, the hybrid needs to re-program the random oracle to appropriately generate the challenges for the sumcheck protocol executions. Since the transcripts includes commitments to the random polinomial $\tilde{h}$, the bad event that the hybrid re-program an already queried location happens with probability at most $\frac{q_{\mathsf{RO}}}{q}$ where q_{RO} is the number of query to the random oracle and $q = |\mathbb{G}_1|$, so we can safely ignore it. By the indistinghuishability property of $\mathcal{S}_{lSC}$ we obtain $\epsilon_0 \approx \epsilon_1$.
- $\mathbf{H}_2$ is the same as $\mathbf{H}_1$ but, similarly to the previous step, the high-degree sumcheck protocols are simulated using the simulator $\mathcal{S}_{hSC}$ described in Eq. (9). Notice the simulator $\mathcal{S}_{hSC}$ generates univariate polynomials that are honestly committed using KZG commitment scheme. As in the previous step we have $\epsilon_1 \approx \epsilon_2$.

- $\mathbf{H}_3$ proceeds as the previous hybrid except that PST proofs are generated by the distributional zero-knowledge simulator $\mathcal{S}^{(\mu)}$ from Section 4. We notice we can reduce to the distributional zero-knowledge of PST w.r.t. to msk. The reduction runs both $\mathcal{A}_{zk}$ and runs the prover of the zkSNARK with the modifications from $\mathbf{H}_1$ and $\mathbf{H}_2$. However, whenever the adverary $\mathcal{A}_{zk}$ asks a query, the reduction, instead of sampling the multivariate polynomials $p_1^*, \ldots, p_{k_2}^*$ directly, following the specification of the ZK-compiler in Algorithm 3 in Algorithm 3, queries the interface provided by the security game in Fig. 4, namely, sending queries of the form $(\mathtt{sample}, i, p_i, t_i + 1, \ell_i)$. As result of such queries the reduction receives the commitments $\mathsf{c}_i = [p_i^*(\boldsymbol{\beta})]_1$ for any i, which it can includes in the transcript of the proof. Notice that the masking algorithm for the compiled zkSNARK and in the distributional zero-knowledge security game are set to be the same, thus the multivariate polynomials are sampled from the same distribution as in the hybrids. Moreover notice that thanks to the changes introduced in the previous two hybrids all the computation on the multivariate polynomials are evaluations, which the reduction can query to the challenger of the security game. Notice that the checker will always accept the queries, by definition of the ZK compiler. Thus the reduction perfectly simulates one of the hybrids, and in particular, either it perfectly simulates $\mathbf{H}_2$ when the challenger provides real proofs, or $\mathbf{H}_3$ when the challenger provides simulated proofs, thus $\epsilon_2 \approx \epsilon_3$.
- $\mathbf{H}_4$ reverts the changes introduced in $\mathbf{H}_1$ and $\mathbf{H}_2$, thus $\epsilon_3 \approx \epsilon_4$.
- $\mathbf{H}_5$ is the ideal-world in which $\mathcal{A}_{zk}$ has oracle access to the the simulator. We notice that the difference between the two hybrids is that in the latter we are using the zero-knowledge simulator of the sumcheck-based PIOP IP^* while in the former we run (and compile) the PIOP prover. Thus $\epsilon_4 \approx \epsilon_5$ because of the zero-knwowledge property of the sumcheck-based PIOP.

Chaining together we have $\epsilon_0 \approx \epsilon_5$ which concludes the proof of zero-knowledge.

As for simulation-extractability, since we focus on a relation $\mathcal{R}$ whose instances do not contain any commitment, we can easily show that the view of the adversary is algebraically consistent, i.e., the simulator only computes algebraically consistent statements and that $\mathcal{X}$, namely, the set of the commitments in the query to $\mathcal{S}^{(\mu)}$, is such that $\mathcal{X} \subseteq (\mathsf{coms}^{(j)})_j$.

We now need to show that for the adversary $\mathcal{A}_\Pi$ there is an extractor $\mathcal{E}_\Pi$ that outputs a valid witness with overwhelming probability whenever $\mathcal{A}_\Pi$ submits a valid forgery.

This proof follows a similar argument to the one in [26], and we briefly summarize the main steps hereafter.

Roughly speaking, $\mathcal{E}_\Pi$ first tries to (straight-line) extract the committed oracle polynomials from their algebraic representation, then runs the PIOP extractor $\mathcal{E}_{\mathsf{IP}}$ (on input these polynomials) to obtain the actual witness: we can show that this is possible with overwhelming probability by reducing to the controlled-malleability of PST and the state-restoration straight-line extractability of IP, as follows.

To analyze the success probability of the extractor $\mathcal{E}_\Pi$, we define an adversary $\mathcal{A}_{\mathsf{IP}}$ for the state-restoration game of the PIOP: roughly, this reduction answers all the simulation queries of $\mathcal{A}_\Pi$ as the zero-knowledge simulator $\mathcal{S}$ would do. When $\mathcal{A}_\Pi$ issues a RO query, it outputs a random element as $\mathcal{S}$ would do, unless the query does not contain any simulated material; in such a case, it parses the query as a (partial) transcript and tries to extract the (oracle) multilinear polynomials from the corresponding commitments; then, it sets the cvs of the state-restoration game to match the extracted polynomials and the transcript in the query, receives the verifier challenge, and forwards the answer to the adversary $\mathcal{A}_\Pi$. Eventually, $\mathcal{A}_\Pi$ outputs its forgery and $\mathcal{A}_{\mathsf{IP}}$ tries to apply the same extraction procedure to obtain all the multilinear polynomials, sets the final cvs, which triggers the decision bit of the state-restoration game. By the state-restoration straight-line extractability of IP, we notice that no adversary, and in particular $\mathcal{A}_{\mathsf{IP}}$, can force the decision bit d to be 1 but triggering a failure of the extraction, namely forcing the extraction bit b to be 0.

By the controlled-malleability of PST, we can bound the probability that the extraction of the oracle polynomials fails. Given as input the instance-proof pair $(\mathbb{x}_{\mathtt{PST}}, \pi_{\mathtt{PST}})$, the probability that the adversary $\mathcal{A}_{\mathsf{IP}}$ fails to extract any of the polynomials is at most the probability that the PST extractor fails. As observed above, the zero-knowledge simulator would only issue queries that are algebraic-consistent, and so would do $\mathcal{A}_{\mathsf{IP}}$ too. Moreover, it is not hard to see that the forgery satisfies a policy $\Phi \in \boldsymbol{\Phi}^{\mathtt{PST}}_{\mathtt{gHC}}$ since the points $\boldsymbol{z}^*$ and $\boldsymbol{x}^*$ included in $\mathbb{x}_{\mathtt{PST}}$ are derived as the RO output of the transcript, which includes the commitments of the oracle polynomials. The extractor would either output a valid witness, or an admissible transformation $T \in \mathcal{T}_{\mathtt{LH}}$; however, the probability that the latter happens is at most negligible when the statement $\mathbb{x}^*$ for which the adversary $\mathcal{A}_{\Pi}$ submitted the forgery is fresh since this would require a collision in the choice of the verifier challenges. Finally, we notice that all the commitments sent in the first k_1 rounds are part of the instance $\mathbb{x}_{\mathtt{PST}}$ because IP is a compiler-safe sumcheck-based PIOP and hence $\mathcal{I}(\mathsf{IP}) = [k_1]$; thus, all the multilinear polynomials, associated with the transcript derived from $\mathbb{x}^*$, are extracted correctly.

To guarantee that the extracted polynomials are indeed valid, we need to additionally check that the round polynomials are valid too. This follows in a similar way to the strategy depicted above, but this time when reducing to the controlled-malleability of PST commitment, we use as forgery $(\mathbb{x}_{\mathtt{KZG}}, \pi_{\mathtt{KZG}})$: notice that we still need to simulate PST proofs and that a KZG forgery can be naturally parsed as a forgery for PST.

Finally, we can conclude that $\mathcal{E}_{\Pi}$ fails only with negligible probability more than the probability that the extractor for the state-restoration game of the PIOP fails as all the polynomials are extracted correctly from the proof of $\mathcal{A}_{\Pi}$ with overwhelming probability. □

7.4 Concrete Instantiations of sumcheck-based SE-zkSNARKs

As a direct corollary of Theorem 6, we obtain that a wide range of concrete sumcheck-based zkSNARKs are simulation-extractable. In this section, we briefly mention some illustrative examples.

The HyperPlonk PIOP [18] is explicitly defined as a sumcheck-based PIOP. SuperSpartan can also be interpreted as a sumcheck-based PIOP, as noted in [60, page 22 and Section 4], where the authors informally refer to certain efficiency optimizations that we formalize in this work. The same holds for Spartan [59].

Another example is Libra [64], which uses the sumcheck and GKR protocols [37], and can be naturally viewed as a sumcheck-based PIOP.

In all these cases, we consider the versions of the schemes that have been instantiated with zero knowledge, incorporate the optimizations formalized in this work, and use the PST commitment scheme.

Corollary 1. *For $X \in \{\mathsf{HyperPlonk}, \mathsf{Spartan}, \mathsf{SuperSpartan}, \mathsf{Libra}\}$, the zkSNARK $\Sigma_{\mathtt{ZK+Batch+FS+PC}}[X]$ with PST is weak simulation-extractable in the AGM.*

Proof. By Theorem 6, it is sufficient to show that X is a compiler-safe sumcheck-based PIOP. Finally, we notice that for each oracle polynomial $[\![p_i]\!]$ sent by the prover, there exists a sumcheck over a function that has a non-zero dependence on the oracle $[\![p_i]\!]$, which implies that the sumcheck PIOP is compiler-safe.

References

1. Behzad Abdolmaleki, Matteo Campanelli, Quang Dao, and Hamidreza Khoshakhlagh. On the simulation-extractability of proof-carrying data. Cryptology ePrint Archive, Report 2025/2037, 2025.
2. Arasu Arun, Srinath T. V. Setty, and Justin Thaler. Jolt: SNARKs for virtual machines via lookups. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 3–33. Springer, Cham, May 2024.
3. Christian Badertscher, Matteo Campanelli, Michele Ciampi, Luigi Russo, and Luisa Siniscalchi. Universally composable SNARKs with transparent setup without programmable random oracle. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VII*, volume 16006 of *LNCS*, pages 225–258. Springer, Cham, August 2025.
4. Karim Baghery, Markulf Kohlweiss, Janno Siim, and Mikhail Volkhov. Another look at extraction and randomization of groth's zk-SNARK. In Nikita Borisov and Claudia Díaz, editors, *FC 2021, Part I*, volume 12674 of *LNCS*, pages 457–475. Springer, Berlin, Heidelberg, March 2021.
5. Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 121–151. Springer, Cham, August 2020.
6. Juraj Belohorec, Pavel Dvorák, Charlotte Hoffmann, Pavel Hubácek, Kristýna Masková, and Martin Pastyrík. On extractability of the KZG family of polynomial commitment schemes. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 584–616. Springer, Cham, August 2025.
7. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018.
8. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016.
9. Jan Bobolz, Pooya Farshim, Markulf Kohlweiss, and Akira Takahashi. The brave new world of global generic groups and UC-secure zero-overhead SNARKs. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part I*, volume 15364 of *LNCS*, pages 90–124. Springer, Cham, December 2024.
10. Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Halo infinite: Proof-carrying data from additive polynomial commitments. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 649–680, Virtual Event, August 2021. Springer, Cham.
11. Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. Gemini: Elastic SNARKs for diverse environments. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 427–457. Springer, Cham, May / June 2022.
12. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.
13. Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Cham, December 2021.
14. Matteo Campanelli, Antonio Faonio, and Luigi Russo. SNARKs for virtual machines are non-malleable. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part IV*, volume 15604 of *LNCS*, pages 153–183. Springer, Cham, May 2025.
15. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019.
16. Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 281–300. Springer, Berlin, Heidelberg, April 2012.

17. Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Succinct malleable NIZKs and an application to compact shuffles. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 100–119. Springer, Berlin, Heidelberg, March 2013.
18. Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 499–530. Springer, Cham, April 2023.
19. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Cham, May 2020.
20. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Berlin, Heidelberg, April / May 2002.
21. Quang Dao and Paul Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 531–562. Springer, Cham, April 2023.
22. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Berlin, Heidelberg, August 2001.
23. Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and MtGox. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014, Part II*, volume 8713 of *LNCS*, pages 313–326. Springer, Cham, September 2014.
24. Liam Eagen and Ariel Gabizon. MERCURY: A multilinear polynomial commitment scheme with constant proof size and no prover FFTs. Cryptology ePrint Archive, Report 2025/385, 2025.
25. Antonio Faonio, Dario Fiore, Markulf Kohlweiss, Luigi Russo, and Michal Zajac. From polynomial IOP and commitments to non-malleable zkSNARKs. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part III*, volume 14371 of *LNCS*, pages 455–485. Springer, Cham, November / December 2023.
26. Antonio Faonio, Dario Fiore, and Luigi Russo. Real-world universal zkSNARKs are non-malleable. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3138–3151. ACM Press, October 2024.
27. Antonio Faonio, Dario Fiore, and Luigi Russo. Real-world universal zkSNARKs are non-malleable. Cryptology ePrint Archive, Report 2024/721, 2024.
28. Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Berlin, Heidelberg, December 2012.
29. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.
30. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018.
31. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019.
32. Chaya Ganesh, Hamidreza Khoshakhlagh, Markulf Kohlweiss, Anca Nitulescu, and Michal Zajac. What makes Fiat-Shamir zkSNARKs (updatable SRS) simulation extractable? In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22*, volume 13409 of *LNCS*, pages 735–760. Springer, Cham, September 2022.
33. Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat-Shamir bulletproofs are non-malleable (in the algebraic group model). In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 397–426. Springer, Cham, May / June 2022.
34. Chaya Ganesh, Sikhar Patranabis, and Nitin Singh. Samaritan: Linear-time prover SNARK from new multilinear polynomial commitments. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part V*, volume 16249 of *LNCS*, pages 456–489. Springer, Singapore, December 2025.
35. Paul Gerhart, Davide Li Calsi, Luigi Russo, and Dominique Schröder. Bounded-equivocable pseudorandom functions. Cryptology ePrint Archive, Report 2025/1894, 2025.
36. Paul Gerhart, Davide Li Calsi, Luigi Russo, and Dominique Schröder. Fully-adaptive two-round threshold Schnorr signatures from DDH. Cryptology ePrint Archive, Report 2025/1478, 2025.

37. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.
38. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Cham, August 2018.
39. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010.
40. Markulf Kohlweiss, Mahak Pancholi, and Akira Takahashi. How to compile polynomial IOP into simulation-extractable SNARKs: A modular approach. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part III*, volume 14371 of *LNCS*, pages 486–512. Springer, Cham, November / December 2023.
41. Tohru Kohrita and Patrick Towa. Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. *Journal of Cryptology*, 37(4):38, October 2024.
42. Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858. IEEE Computer Society Press, May 2016.
43. Abhiram Kothapalli and Bryan Parno. Algebraic reductions of knowledge. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 669–701. Springer, Cham, August 2023.
44. Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 359–388. Springer, Cham, August 2022.
45. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 1–34. Springer, Cham, November 2021.
46. Benoît Libert. Simulation-extractable KZG polynomial commitments and applications to HyperPlonk. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14602 of *LNCS*, pages 68–98. Springer, Cham, April 2024.
47. Benoit Libert. Simulation-extractable KZG polynomial commitments and applications to HyperPlonk. Cryptology ePrint Archive, Report 2024/854, 2024.
48. Helger Lipmaa. Plonk is simulation extractable in ROM under falsifiable assumptions. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025, Part IV*, volume 16271 of *LNCS*, pages 3–36. Springer, Cham, December 2025.
49. Helger Lipmaa, Roberto Parisella, and Janno Siim. Algebraic group model with oblivious sampling. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 363–392. Springer, Cham, November / December 2023.
50. Helger Lipmaa, Roberto Parisella, and Janno Siim. On knowledge-soundness of plonk in ROM from falsifiable assumptions. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VII*, volume 16006 of *LNCS*, pages 362–395. Springer, Cham, August 2025.
51. Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st FOCS*, pages 2–10. IEEE Computer Society Press, October 1990.
52. Lurk Lab. Lurk: A programming language for recursive succinct arguments. `https://github.com/lurk-lang/lurk`, 2023.
53. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.
54. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.
55. Wilson Nguyen, Dan Boneh, and Srinath Setty. Revisiting the nova proof system on a cycle of curves. Cryptology ePrint Archive, Report 2023/969, 2023.
56. Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 222–242. Springer, Berlin, Heidelberg, March 2013.
57. Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, August 2021. Springer, Cham.

58. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.
59. Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737. Springer, Cham, August 2020.
60. Srinath Setty, Justin Thaler, and Riad Wahby. Customizable constraint systems for succinct arguments. Cryptology ePrint Archive, Report 2023/552, 2023.
61. Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Unlocking the lookup singularity with Lasso. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 180–209. Springer, Cham, May 2024.
62. StarkWare. ethSTARK documentation. Cryptology ePrint Archive, Report 2021/582, 2021.
63. Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018.
64. Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764. Springer, Cham, August 2019.