Sumcheck-based zkSNARKs are Non-Malleable

Antonio Faonio[1] and Luigi Russo[2]

[1] EURECOM, Sophia Antipolis, France faonio@eurecom.fr
[2] TU Wien, Vienna, 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 [49] are powerful cryptographic primitives that enable scalable, privacy-preserving computation. At their core, zkSNARKs 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 [52], 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.* [20].

The risks of malleability are not merely hypothetical. For example, malleability attacks have historically involved over three hundred thousand Bitcoins [21]. Such vulnerabilities in state-of-the-art SNARKs are critical as they allow attackers to redirect funds or corrupt state transitions in decentralized ledgers.

In this work, we build on the line of research initiated by Faonio *et al.* [23,24] and Kohlweiss *et al.* [37], 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) [7] into non-interactive arguments using the Fiat–Shamir transform [27], together with polynomial commitment schemes [36].

The aforementioned works focus on compiling univariate PIOPs into zkSNARKs using KZG [36] 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+ [17], Gemini [10], Zeromorph [38], and Dory [41]. In this work, we focus on the PST scheme [50] for two main reasons: it provides the shortest proof size among multivariate polynomial commitment schemes, with the exception of the very recent Mercury [22] and Samaritan [32]; 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 [17]. This class employs the classical sumcheck protocol [47] 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 build-

ing block in several recent and influential zkSNARK constructions such as [10,17,53,56,57].

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 [24] 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 [57] by reducing the degrees of certain masking polynomials.
2. **Query minimization.** We then apply known optimizations [17], 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 [17], 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.* [17] proposed using the technique of [9] to construct a PST-HyperPlonk zkSNARK. 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 [17].

Similarly to previous generic frameworks [23,24,37], we establish our results in the Algebraic Group Model (AGM) [28]. Removing the AGM remains a significant challenge. First, a formal proof of knowledge soundness

for PST in the standard model was only newly established [5]. Second, recent results demonstrate that security proofs for optimized SNARKs face fundamental barriers. Specifically, Lipmaa *et al.* [46] showed that the *linearization trick* [24,29] fails to satisfy special soundness in the standard model. Furthermore, recent work by Lipmaa [44] 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.* [26] to prove simulation extractability.

1.2 Related Work

Several recent works [1,13,19,23,24,30,31,37,44] have established simulation extractability for a range of zkSNARKs including Basilisk [51], Bulletproofs [11], Jolt [2], Lasso [55], Lunar [12], Marlin [18], PLONK [29], Sonic [48], and Spartan [19]. These contributions can be grouped into two main categories. The first group includes works such as [13,19,30,31,44] that analyze the simulation extractability of (simple variants of) specific schemes. The second group includes general frameworks, such as those proposed in [23,24] and [37], 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 [42], 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 [31].

Additionally, Libert showed that PST evaluation proofs are not unique [43, 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 [33,34,39], as it accounts for schemes that can still effectively resist relevant and practical classes of attacks [4,8], 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.* [15,16].

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.* [23], 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 [23].

Also notice, we follow the approach of Faonio *et al.* [23] rather than that of Kohlweiss *et al.* [37], as the latter framework requires proof uniqueness, a property that PST does not satisfy, as shown in [43]. We state our first result below (see Theorem 2 in Section 4 for the formal version).

Informal Theorem 1. *The PST 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.*

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.* [24] 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 1 in Section 4 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 [24].*

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, one can prove a weaker form of zero-knowledge, called *distributional zero-knowledge*, with respect to that masking. For space reasons, we defer this to the full version [25].

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 con-

[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 [43], one can alter the evaluation proofs so that they become zero knowledge.

structions developed in recent years, and we refer to it as the PC+FS compiler.

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 [23,24,37], 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 [17] are PIOPs that rely on one or more invocations of the classic sumcheck protocol [47]. 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 distinguish low-degree and high-degree sumchecks: in the latter case, the univariate round polynomials sent during the sumcheck execution are themselves provided as oracles and must later be checked by the compiled zkSNARK.

We improve the zero-knowledge compiler of Xie *et al.* [57] by showing that for the class of high-degree sumcheck protocols in [17], 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.

A key observation in [17] 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 [17], treating them as a separate compiler that transforms one sumcheck-based PIOP into another. Along the way, we refine and extend

[5] While existing literature [13,19] has addressed simulation extractability for specific protocols such as Spartan and Lasso, these results are primarily *ad-hoc* works that, furthermore, rely on the framework of [26], which fundamentally requires *unique response* to establish security: however, as noted, this requirement is inherently incompatible with the linear-homomorphic properties of PST.

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 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. 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 [9]). 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 [6]. 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 [25].

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 [57], HyperPlonk [17], Spartan [53], and SuperSpartan [54]) and can be easily checked by protocol designers of future schemes.

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 $\{1, 2, \ldots, n\}$. 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. $L := (L_i)_{i \in [n]}$ is a shortcut for the list of n elements $(L_1, \ldots, L_n)$. 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 $\mathbb{F}[X_1, \ldots, X_\mu]$, 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.

Definition 1 ((n, d)-OMSDH, [24]). *Consider the experiment in Fig. 1. 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:*

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

Definition 2 (Algebraic algorithm, [28]). *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 $\mathsf{z} = \langle \boldsymbol{r}, \mathsf{elems} \rangle$.*

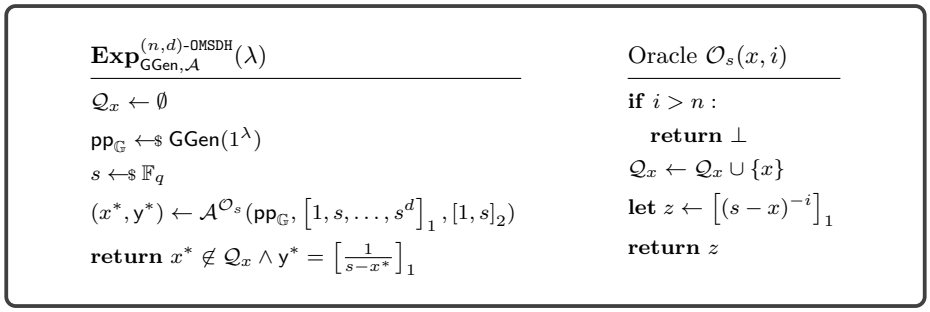

Fig. 1. The OMSDH experiment.

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 $(x_1, \ldots x_n) \rightarrow_{\mathsf{RO}} a$ 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 [47]).

2.1 Non-Interactive Zero-Knowledge Arguments

We define a PT relation $\mathcal{R}$ verifying triple $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ as in [35]. We say that $\mathbb{w}$ is a witness to the instance $\mathbb{x}$ being in the relation defined by the parameters pp 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 [26], 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 [26], 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 [23], 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.

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

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:

- $\mathsf{KGen}(\mathsf{pp}_{\mathbb{G}}, \mathsf{aux}) \to \mathsf{ck}$ takes as input $\mathsf{pp}_{\mathbb{G}} \leftarrow\!\!\$\, \mathsf{GGen}(1^\lambda)$, and any additional required parameters aux, and outputs a commitment key ck.
- $\mathsf{Com}(\mathsf{ck}, m) \to (\mathsf{c}, o)$ takes as input the commitment key ck, and a message $m \in \mathcal{M}$, and outputs a commitment c and an opening o.
- $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, m, o) \to b$ takes as input the commitment key ck, a commitment c, a message m and an opening o, and outputs a bit.

Besides correctness, a scheme CS can satisfy two more properties called *binding* and *hiding*. In this work we consider schemes that are computationally biding and succinct but that are not necessarily hiding.

CP-SNARKs. Commit-and-Prove SNARKs, or simply CP-SNARKs, are proof systems whose relations verify predicates over commitments [14]. 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 $\mathsf{KGen}(\mathsf{ck}) \to \mathsf{srs}$ 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 [50] (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 $[0, d_1] \times \cdots \times [0, d_\mu]$.

[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.

– $\mathsf{Com}(\mathsf{ck}, f(\boldsymbol{X}))$ 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 [36]. To emphasize the different settings, we define $\mathsf{CS}_{\mathtt{PST},\mu}$ to be the polynomial commitment scheme above where KGen fixes the last parameter to μ, thus $\mathsf{CS}_{\mathtt{KZG}} := \mathsf{CS}_{\mathtt{PST},1}$.

The PST commitment scheme allows for *evaluation proofs* which, in the framework of [14], 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 $e(\mathsf{c} - [y]_1, [1]_2) = \sum_{i\in[\mu]} e(\pi_i, [\beta_i - x_i]_2)$.

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

Prover $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathtt{PST},\mu}].\mathsf{Prove}(\mathsf{ek}, \mathbb{x} = (\mathsf{c}, \boldsymbol{x}, y), \mathbb{w} = f)$:
– Let $R_{\mu+1}(\boldsymbol{X}) \leftarrow f(\boldsymbol{X}) - y$
– For $i = \mu, \ldots, 1$:
  • 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 $[\pi_1(\boldsymbol{\beta})]_1, \ldots, [\pi_\mu(\boldsymbol{\beta})]_1$.

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, let $\mathbb{i} = (a_j)_{j\in[m]}, \mathbb{x} = ((\mathsf{c}_j)_{j\in[m]}, \boldsymbol{z}, \boldsymbol{x}, y), \mathbb{w} = (f_j)_{j\in[m]}$, then:

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

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 [25]. 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.* [24] 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*, 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 $\Pi_{\mathtt{geval}}$ for $\mathcal{R}_{\mathtt{geval}}$, that we simply call generalized PST, works as follows:

- The prover of $\Pi_{\mathtt{geval}}$ runs $\Pi_{\mathtt{evl}}$ 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 $\Pi_{\mathtt{geval}}$ runs the verifier of $\Pi_{\mathtt{evl}}$ recomputing the input $\mathbb{x}'$ from $\mathbb{i}$ and $\mathbb{x}$, as the prover does, and the PST proof.

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

On the zero-knowledge of (generalized) PST. As noted in Section 1, PST evaluation proofs are not zero-knowledge. We establish a weaker form of zero-knowledge, that we call *distributional zero-knowledge*, for PST. We observe that if the witness is sufficiently randomized via a carefully crafted masking algorithm msk, we can prove distributional zero-knowledge with respect to msk. For space reasons, we provide the details in [25].

3 Policy-Based Controlled-Malleability in the AGM

We recall the definitional framework of [23]. In their framework a policy is a pair $\Phi := (\Phi_0, \Phi_1)$ of PPT algorithms. The Φ-simulation extractability experiment defined by [23] 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 [15]. 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 [15] to allow instances from different relations.

Definition 6 (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. 2 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 [45]

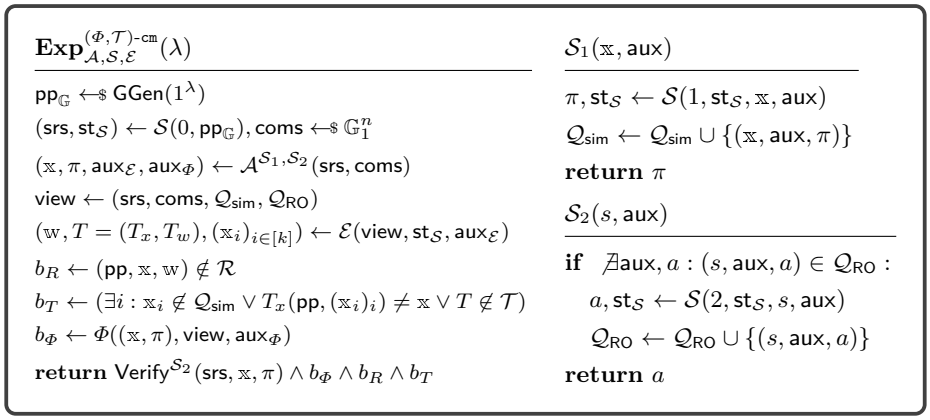

Fig. 2. The Φ-simulation $\mathcal{T}$-controlled-malleability security experiment.

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 $\mathbb{x} = T_x(\mathbb{x}_1, \ldots, \mathbb{x}_k)$. 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 $T(\mathbb{w}_1, \ldots, \mathbb{w}_k)$ would yield a valid witness for the forgery.

Definition 7 (Φ-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 $\mathbf{Exp}^{(\Phi,\mathcal{T})\text{-cm}}$ defined in Fig. 2 for some policy Φ. A NIZK Π for a relation $\mathcal{R}$ and simulator $\mathcal{S}$ for instances in $\mathcal{R}_S$ is Φ-simulation $\mathcal{T}$-controlled-malleable (briefly, $(\Phi,\mathcal{T})$-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:*

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

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

4 Controlled-Malleability of PST and KZG

In Definition 7, 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 $\Pi_{\mathtt{PST},\mu}$ (the generalized PST) in the presence of the zero-knowledge simulator of $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathtt{PST},\mu}]$ (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. For space reasons we defer the formal definition to [25], and here we only summarize it.

- A CP-SNARK $\Pi[\mathsf{CS}_{\mathtt{PST}}]$ for $\mathcal{R}$ is called PST-based if both its prover and its verifier internally invoke $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathtt{PST}}]$. One can view this as a *reduction of knowledge* [40] from $\mathcal{R}$ to $\mathcal{R}_{\mathtt{evl}}^k$ for some $k \in \mathbb{N}$.
- A CP-SNARK $\Pi[\mathsf{CS}_{\mathtt{PST}}]$ is called *algebraic* if, for any instance $\mathbb{x}$, the commitments in the derived instances for $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathtt{PST}}]$ are linear combinations of the group elements in the original instance $\mathbb{x}$.
- A CP-SNARK $\Pi[\mathsf{CS}_{\mathtt{PST}}]$ 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 $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathtt{PST}}]$.

For example, $\Pi_{\mathtt{PST}}$ 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.

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 1. *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 $\mathcal{E} = \mathcal{E}'$.

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.[8]

Although this class of adversaries is strictly more restrictive than those considered in [24], it is sufficient for our goal of proving that multivariate PIOP-based zkSNARKs are simulation extractable.

Faonio *et al.* [23] 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 8 (Algebraic Consistency, simplified). *Let $\mathcal{A}$ be a simplified adversary that has oracle access to a simulator $\mathcal{S}^{(\mu)}$ for $\Pi_{\mathtt{evl}}[\mathsf{CS}_{\mathtt{PST},\mu}]$ and receives as input a commitment key for $\mathsf{CS}_{\mathtt{PST},\mu}$ 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*

[8] In fact, in Section 4.2, 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.

interacting with $\mathcal{S}^{(\mu)}$ *is* algebraic consistent *if the linear system of multivariate polynomial equations, that we describe next, admits a solution.*

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* $Q_{k,j} \in \mathbb{F}_{\leq d}[\boldsymbol{X}]$. *For each simulation query* $(\mathsf{g}, \boldsymbol{x}, y)$ *we define a new equation:*

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

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 9 (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* $\mathcal{X} \subseteq \mathsf{coms} \cup \mathsf{proofs}$.

4.1 Controlled-Malleability of KZG

Faonio *et al.* [24] 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 $p(\mathsf{ck}, \boldsymbol{Y}) = [p(\boldsymbol{\beta}, \boldsymbol{Y})]_1$.

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

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

Nesting Level. To prove our result on KZG, we first recall the notion of *nesting level* introduced in [24]. 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 occur in 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 [24], 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 10 (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 $\Pi_{\mathtt{KZG}}$. 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 11. *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 12 (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 $\Phi^{\nu}_{\mathtt{gHC},\mathbb{i}}$ be a simplified*

policy that, upon input the forgery $(\mathtt{i}^*, \mathtt{x}^*, \pi^*)$ *, the view* view *and the auxiliary output* aux_Φ*, parses* $\mathtt{x}^* = ((\mathsf{c}_i^*)_i, \boldsymbol{z}^*, x^*, y^*)$*, 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* $\mathtt{i} = \mathtt{i}^*$*,*
- *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 $\boldsymbol{\Phi}^{\nu}_{\mathtt{gHC}} := \{\Phi^{\nu}_{\mathtt{gHC},\mathtt{i}} : \mathtt{i} \text{ are } \nu\text{-admissible}\}$*.*

Theorem 1 (Controlled-Malleability of KZG). *For any* $\nu \in \mathbb{N}$*, the scheme* $\Pi_{\mathtt{KZG}}$ *is* $\boldsymbol{\Phi}^{\nu}_{\mathtt{gHC}}$*-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 [24]. 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 $\boldsymbol{\Phi}^{\nu}_{\mathtt{gHC}}$ 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. We defer the formal proof to [25].

4.2 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.

Definition 13 (Generalized hash-check policy for PST). *For any index* $\mathtt{i} = (a_j)_{j\in[m]}$ *of* $\mathcal{R}_{\mathtt{geval}}$*, let* $\Phi^{\mathtt{PST}}_{\mathtt{gHC},\mathtt{i}}$ *be a simplified policy that, upon input the forgery* $(\mathtt{i}^*, \mathtt{x}^*, \pi^*)$ *and* $\mathtt{i} = \mathtt{i}^*$*, the view* view *and the auxiliary output* aux_Φ*, parses* $\mathtt{x}^* = ((\mathsf{c}_i^*)_i, \boldsymbol{z}^*, x^*, y^*)$*, and returns 1 if and only if:*

- *The set* $\mathcal{X}$ *of the commitments in the simulation queries (see Definition 9) is such that* $\mathcal{X} \subset \mathsf{coms}$ *and,*
- *if either (1)* $\forall i : a_i \in \mathbb{F}[\boldsymbol{Z}]$ *and* $(\mathsf{c}_i^*)_i \to_{\mathsf{RO}} \boldsymbol{z}^*$ *and* $(\sum_i a_i(\boldsymbol{z})\mathsf{c}_i^*) \to_{\mathsf{RO}} x^*$ *or (2)* $\forall i : a_i \in \mathbb{F}[X]$ *and* $(\mathsf{c}_i^*)_i \to_{\mathsf{RO}} x^*$*, we call this condition the generalized hash check.*

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

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

Theorem 2 (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 $\Pi_{\mathtt{PST},\mu-1}$ that internally makes use of an adversary against $\Pi_{\mathtt{PST},\mu}$. 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. We omit the analysis in this proof sketch, and refer the reader to [25] for the full proof.

Acknowledgments

The authors thank Benoît Libert for bringing to their attention his result on the non-uniqueness of PST, presented in the full version of his PKC paper [43], and Srinath Setty for helpful comments on the non-malleability of Nova and Spartan. The first author was supported by the French National Research Agency (ANR) through the CTRL-M project (ANR-25-CE39-5707).

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, *Advances in Cryptology – EUROCRYPT 2024, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 3–33, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.
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, *Advances in Cryptology – CRYPTO 2025, Part VII*, volume 16006 of *Lecture Notes in Computer Science*, pages 225–258, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.
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: 25th International Conference on Financial Cryptography and Data Security, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 457–475, Virtual Event, March 1–5, 2021. Springer Berlin Heidelberg, Germany.
5. 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, *Advances in Cryptology – CRYPTO 2025, Part VI*, volume 16005 of *Lecture Notes in Computer Science*, pages 584–616, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

6. 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: 45th International Colloquium on Automata, Languages and Programming*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:17, Prague, Czech Republic, July 9–13, 2018. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.
7. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, Beijing, China, October 31 – November 3, 2016. Springer Berlin Heidelberg, Germany.
8. 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: 22nd Theory of Cryptography Conference, Part I*, volume 15364 of *Lecture Notes in Computer Science*, pages 90–124, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.
9. 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, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 649–680, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.
10. Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. Gemini: Elastic SNARKs for diverse environments. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 427–457, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.
11. 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, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.
12. 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, *Advances in Cryptology – ASIACRYPT 2021, Part III*, volume 13092 of *Lecture Notes in Computer Science*, pages 3–33, Singapore, December 6–10, 2021. Springer, Cham, Switzerland.
13. Matteo Campanelli, Antonio Faonio, and Luigi Russo. SNARKs for virtual machines are non-malleable. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part IV*, volume 15604 of *Lecture Notes in Computer Science*, pages 153–183, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.
14. 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: 26th Conference on Computer and Communications Security*, pages 2075–2092, London, UK, November 11–15, 2019. ACM Press.
15. Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johans-

son, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 281–300, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany.
16. Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Succinct malleable NIZKs and an application to compact shuffles. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 100–119, Tokyo, Japan, March 3–6, 2013. Springer Berlin Heidelberg, Germany.
17. 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, *Advances in Cryptology – EUROCRYPT 2023, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 499–530, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.
18. 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, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 738–768, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.
19. Quang Dao and Paul Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 531–562, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.
20. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 566–598, Santa Barbara, CA, USA, August 19–23, 2001. Springer Berlin Heidelberg, Germany.
21. Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and MtGox. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014: 19th European Symposium on Research in Computer Security, Part II*, volume 8713 of *Lecture Notes in Computer Science*, pages 313–326, Wroclaw, Poland, September 7–11, 2014. Springer, Cham, Switzerland.
22. 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.
23. 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: 21st Theory of Cryptography Conference, Part III*, volume 14371 of *Lecture Notes in Computer Science*, pages 455–485, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland.
24. 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: 31st Conference on Computer and Communications Security*, pages 3138–3151, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press.
25. Antonio Faonio and Luigi Russo. Sumcheck-based zkSNARKs are non-malleable. Cryptology ePrint Archive, Paper 2026/335, 2026.
26. 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, *Progress in Cryptology - INDOCRYPT 2012: 13th International Conference in Cryptology in India*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79, Kolkata, India, December 9–12, 2012. Springer Berlin Heidelberg, Germany.

27. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer Berlin Heidelberg, Germany.
28. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.
29. 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.
30. 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: 13th International Conference on Security in Communication Networks*, volume 13409 of *Lecture Notes in Computer Science*, pages 735–760, Amalfi, Italy, September 12–14, 2022. Springer, Cham, Switzerland.
31. 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, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 397–426, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.
32. 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, *Advances in Cryptology – ASIACRYPT 2025, Part V*, volume 16249 of *Lecture Notes in Computer Science*, pages 456–489, Melbourne, VIC, Australia, December 8–12, 2025. Springer, Singapore, Singapore.
33. Paul Gerhart, Davide Li Calsi, Luigi Russo, and Dominique Schröder. Bounded-equivocable pseudorandom functions. Cryptology ePrint Archive, Report 2025/1894, 2025.
34. 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.
35. 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, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 698–728, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.
36. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer*

Science, pages 177–194, Singapore, December 5–9, 2010. Springer Berlin Heidelberg, Germany.

37. 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: 21st Theory of Cryptography Conference, Part III*, volume 14371 of *Lecture Notes in Computer Science*, pages 486–512, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland.
38. Tohru Kohrita and Patrick Towa. Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. *Journal of Cryptology*, 37(4):38, October 2024.
39. 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, San Jose, CA, USA, May 22–26, 2016. IEEE Computer Society Press.
40. Abhiram Kothapalli and Bryan Parno. Algebraic reductions of knowledge. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 669–701, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.
41. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 1–34, Raleigh, NC, USA, November 8–11, 2021. Springer, Cham, Switzerland.
42. Benoît Libert. Simulation-extractable KZG polynomial commitments and applications to HyperPlonk. In Qiang Tang and Vanessa Teague, editors, *PKC 2024: 27th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 14602 of *Lecture Notes in Computer Science*, pages 68–98, Sydney, NSW, Australia, April 15–17, 2024. Springer, Cham, Switzerland.
43. Benoit Libert. Simulation-extractable KZG polynomial commitments and applications to HyperPlonk. Cryptology ePrint Archive, Report 2024/854, 2024.
44. Helger Lipmaa. Plonk is simulation extractable in ROM under falsifiable assumptions. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025: 23rd Theory of Cryptography Conference, Part IV*, volume 16271 of *Lecture Notes in Computer Science*, pages 3–36, Aarhus, Denmark, December 1–5, 2025. Springer, Cham, Switzerland.
45. Helger Lipmaa, Roberto Parisella, and Janno Siim. Algebraic group model with oblivious sampling. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023: 21st Theory of Cryptography Conference, Part IV*, volume 14372 of *Lecture Notes in Computer Science*, pages 363–392, Taipei, Taiwan, November 29 – December 2, 2023. Springer, Cham, Switzerland.
46. 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, *Advances in Cryptology – CRYPTO 2025, Part VII*, volume 16006 of *Lecture Notes in Computer Science*, pages 362–395, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.
47. Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st Annual Symposium on Foundations*

of Computer Science, pages 2–10, St. Louis, MO, USA, October 22–24, 1990. IEEE Computer Society Press.
48. 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: 26th Conference on Computer and Communications Security*, pages 2111–2128, London, UK, November 11–15, 2019. ACM Press.
49. Silvio Micali. CS proofs (extended abstracts). In *35th Annual Symposium on Foundations of Computer Science*, pages 436–453, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press.
50. Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 222–242, Tokyo, Japan, March 3–6, 2013. Springer Berlin Heidelberg, Germany.
51. Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 774–804, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.
52. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553, New York, NY, USA, October 17–19, 1999. IEEE Computer Society Press.
53. Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 704–737, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.
54. Srinath Setty, Justin Thaler, and Riad Wahby. Customizable constraint systems for succinct arguments. Cryptology ePrint Archive, Report 2023/552, 2023.
55. Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Unlocking the lookup singularity with Lasso. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 180–209, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.
56. 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, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.
57. 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, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 733–764, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.