From Polynomial IOP and Commitments to Non-malleable zkSNARKs

Antonio Faonio, Dario Fiore, Markulf Kohlweiss, Luigi Russo, Michal Zajac

From Polynomial IOP and Commitments to Non-malleable zkSNARKs

Antonio Faonio[1], Dario Fiore[2], Markulf Kohlweiss[3], Luigi Russo[1], and Michal Zajac[4]

[1] EURECOM, Sophia Antipolis, France {faonio,russol}@eurecom.fr
[2] IMDEA Software Institute, Madrid, Spain dario.fiore@imdea.org
[3] University of Edinburgh and Input Output, markulf.kohlweiss@ed.ac.uk
[4] Nethermind, michal@nethermind.io

Abstract. We study sufficient conditions to compile simulation-extractable zkSNARKs from information-theoretic interactive oracle proofs (IOP) using a simulation-extractable commit-and-prove system for its oracles. Specifically, we define simulation extractability for opening and evaluation proofs of polynomial commitment schemes, which we then employ to prove the security of zkSNARKS obtained from polynomial IOP proof systems. To instantiate our methodology, we additionally prove that KZG commitments satisfy our simulation extractability requirement, despite being naturally malleable. To this end, we design a relaxed notion of simulation extractability that matches how KZG commitments are used and optimized in real-world proof systems. The proof that KZG satisfies this relaxed simulation extractability property relies on the algebraic group model and random oracle model.

1 Introduction

Non-interactive succinct zero-knowledge arguments of knowledge (zkSNARKs) [45] are the new Swiss army knife of blockchain scalability and privacy. They effectively deliver the twin dream of probabilistically checkable proofs (PCP) [3] and zero-knowledge proofs (ZKP) [34] while also being non-interactive, short, and efficiently verifiable. These features make zkSNARKs of high practical and theoretical relevance. They are an active area of research that has seen rapid progress in multiple aspects, such as efficiency [7,33,35,36], security and versatility of their setups [6,37], and proof composition [13,15].

Simulation-extractable zkSNARKs. *Knowledge-soundness* is the basic security notion of zkSNARKs: informally speaking, it guarantees that, in isolation, a prover producing a valid proof must know the corresponding witness. In contrast, there exist real-world deployments and cryptographic applications of zkSNARKs that require a stronger property called *simulation extractability* (SE, for brevity) [38,47]. Intuitively, this notion considers attackers that can see proofs for some statements and may use this information in order to produce a proof for some other statement without knowing the witness. Interestingly,

simulation extractability implies that proofs are *non-malleable* [23], a relevant property in practical applications. Most zkSNARKs in the literature are only proven to be knowledge-sound. In some cases, this is due to the fact that their proofs may indeed be malleable, e.g., as in [36] (see also [4]). In other cases, the lack of SE security proof is because it is challenging and may require more investigation.

From polynomial commitments to SNARKs. The design of modern zk-SNARKs follows the common cryptographic approach of starting with protocols that achieve information-theoretic security in idealized models and then compiling them into efficient protocols by employing a smaller computationally secure primitive. In the world of SNARKs, the corresponding concepts are (polynomial) interactive oracle proofs F-IOP [16,17,20,28,48] and (polynomial) functional commitments F-COM [12,40,41]. An F-IOP employs two (idealized) oracles that share their state: the prover calls the first oracle *to commit* to functions $f \in F$ and the verifier calls the second *to query* the committed functions. Concretely, the F-IOP to SNARK compiler uses F-COM to replace oracles with commitments, opening proofs, and query proofs. As this only removes reliance on idealized function oracles but not interaction, the compiler additionally employs the usual Fiat-Shamir transformation for public-coin protocols to obtain the final zkSNARK. The benefits of this compilation paradigm are modularity and separation of concerns: once the compiler is proven, a line of research can address the problem of improving F-IOPs while another research line can tackle the problem of realizing F-COM schemes (e.g., with better efficiency, from different assumptions, etc.): this approach has been successfully adopted to construct several recent zkSNARKs. All this recent work, though, only shows that schemes obtained via this paradigm are knowledge-sound.

1.1 Our work

We study the simulation extractability of a broad class of zkSNARKs built through this "natural" compilation approach. In particular, our primary goal includes showing that not only *existing* zkSNARKs but also any future zkSNARKs following this, by now standard, construction framework, provide simulation extractability. This goal has a twofold motivation. On the theoretical side, we are interested in understanding sufficient conditions on F-COM to compile an F-IOP into a simulation-extractable zkSNARK. On the practical side, by capturing existing compilers we can show that existing schemes that are under deployment, e.g., Plonk [28], have already this strong security property.[5] For this reason, in our work, we focus on the popular case of the compiler where the F-IOP is a polynomial IOP (i.e., the oracle functions F are low-degree polynomials), and F-COM is a polynomial commitment. Furthermore, in terms of instantiations we are interested in covering the celebrated KZG polynomial commitment

[5] In fact as Mahak Pancholi and Akira Takahashi recently informed us of a flaw in the trapdoor-less zero-knowledge simulation of [29] this is arguably the first proof of simulation extractability of Plonk with deterministic KZG commitments.

scheme [40] and on a polynomial IOP framework that is flexible enough to include recent constructions, e.g., [17,20,28,43,46]. The main contributions of our work are: (i) to introduce a relaxed notion of simulation extractability for polynomial commitments; (ii) to prove that the KZG scheme satisfies our relaxed SE notion in the algebraic group model (AGM) and random oracle model (ROM); and (iii) to prove that our notion is sufficient to compile a polynomial IOP into a simulation-extractable zkSNARK, using the usual compilation approach. By combining these results we obtain a simulation extractability proof for Plonk [28], Basilisk [46], and a slight variation of Marlin [20] and Lunar [17].

1.2 Our techniques

Background. For our work, we chose the class of Polynomial (Holographic) IOPs (PIOP) as defined by [17] as a generalization of [16].[6] The oracle of the prover commits to low-degree polynomials over a finite field while the queries of the verifier check polynomial equations over these polynomials. These polynomial equations can depend on additional field elements sent by the prover and/or the verifier during the execution of the protocol. Slightly more in detail, the verifier can query an oracle polynomial $p(X)$ (or multiple polynomials simultaneously) by specifying polynomials G and v to test equations of the form $G(X, p(v(X))) \equiv 0$. Therefore, to be compiled, PIOPs need a commit-and-prove SNARK (CP-SNARK) for proving the validity of such equations concerning the committed polynomials. Notably, one can easily build this CP-SNARK from a CP-SNARK for polynomial evaluations (e.g., KZG) by testing the equations on a random point chosen by the random oracle.

Simulation extractability challenges. Intuitively, the use of a simulation-extractable CP-SNARK in the above compiler should result in a simulation-extractable zkSNARK: the zero-knowledge simulator samples random commitments (relying either on hiding property of commitments, or the randomness in the committed functions p). It then simulates evaluations of p that satisfy the verification equation of the PIOP. The reduction to PIOP soundness extracts all committed polynomials from their opening proofs and the final polynomial evaluations from the evaluation proofs. However, this approach presents two major challenges:

- The PIOP could be arbitrary. For example, consider a PIOP obtained by the sequential composition of two PIOP protocols for two independent statements. Very likely, the set of queries to the polynomials made by the two sub-protocols are independent and (unless the PIOP specifies it explicitly) the evaluation queries of the first sub-protocol may be chosen based on the verifier's random challenges sent *before* the second sub-protocol even starts. The simulation extractability of the zkSNARK compiled from this protocol

[6] PIOPs can flexibly capture under the same hat all the most recent protocols based on the notions of [16], AHP [20], and ILDP [28].

might be affected because one could strip off the second set of evaluation proofs and replace them with those for another statement[7].
– One needs to prove that existing, efficient, and practically deployable instantiations of polynomial commitment schemes are simulation-extractable.

Our solutions. To solve the first challenge, motivated by our goal to show that existing zkSNARKs are simulation-extractable and that future zkSNARKs can seamlessly achieve simulation extractability, we define a (rather minimal) constraint on the PIOP. Namely, we require that at least one of the polynomial equations involves all the oracle polynomials and that the polynomial v chosen by the verifier (see above) is not constant.[8] Fortunately, this constraint is natural and easy to meet in practice: Plonk naturally meets our constraint meanwhile all the other proof systems based on Aurora's univariate sumcheck [8] can be easily (and at negligible cost) adapted by instantiating the proof of polynomial degree through an evaluation query on all the polynomials.

For the second challenge, unfortunately, the issue is that the most obvious candidate, the efficient and widely deployed KZG polynomial commitment scheme [40], is not simulation-extractable. Using bracket notation, KZG commitments are of the form $[p(s)]_1$ for a trapdoor secret s encoded in the parameters $([s^i]_1)_{i\in[0..d]}, [1, s]_2$, while evaluation proofs for an input x and output y are of the form $[\frac{p(s)-y}{s-x}]_1$. KZG is malleable: for example, given a commitment to p, anyone can compute $[p(s) + \Delta]_1$ and open it using the same proof to $(x, y + \Delta)$.

Our starting point is the observation that KZG retains a form of simulation extractability for evaluations at points that are randomly chosen *after* the commitment. Fortunately, this is the situation we encounter in the Fiat-Shamir part of the PIOP-to-SNARK compiler. The commitment forms part of the first commit-and-prove part of the statement which is hashed to determine the x of the second part of the statement. Thus, the evaluation point depends on the commitment and can be considered random in the RO model.

To formalize this important relaxation, we introduce the notion of policy-based simulation extractability (Φ-SE, w.r.t. a policy parameter Φ). In the standard simulation extractability experiment, the adversary can ask the simulator to generate proofs for statements of its choice and, eventually, must produce a *new* valid proof without knowing the witness. In Φ-SE, we consider a relaxation of the SE game in which all the simulation queries of the adversary must satisfy a predicate specified in Φ; similarly, Φ can constrain the winning condition of the adversary. For this reason, we refer to Φ as the *policy*. One can see that Φ-SE is a generalization of existing SE notions such as true-simulation extractability

[7] In particular, the adversary could have a simulated proof $\tilde{\pi} = (\pi, \pi')$ for $(\mathbb{x}, \mathbb{x}')$ and then could choose $\mathbb{x}''$ for which it knows a valid witness, and finally forge for $(\mathbb{x}, \mathbb{x}'')$ using (π, π''), where π'' is honestly generated. As the simulated proof π is reused, extraction fails. Notice that this attack works even when the Fiat-Shamir challenges for π'' are derived by hashing a transcript that contains π.

[8] This can be for example implemented via a common random point chosen at the end of the protocol, on which all oracles are evaluated.

(where the adversary can only see simulated proofs on true statements) [22], or weak simulation extractability (where the adversary only wins if it provides a proof for a new statement and, contrary to (strong) SE, loses if it provides a new proof for a statement previously asked to the simulation oracle). Once having defined this framework, we analyze which policies Φ are strong enough to achieve simulation extractability in the compiled zkSNARK, while at the same time being weak enough for instantiation by KZG under plausible assumptions (in the AGM [27] and RO). Specifically, we isolate the (simulation) extractability properties needed for the compiler and verify it for KZG in the AGM. This is the only part of our results where we need the AGM. Given the broad applications of KZG in the field of practical zkSNARKs and beyond, the characterization of its (non-)malleability is interesting in its own right. In fact, our policy highlights some malleability attacks that we discovered and that we needed to handle. Finally, for our Φ we prove that KZG is Φ-SE in the AGM and ROM. This proof turned out to be highly non-trivial and is one of our main technical contributions.

1.3 Related work

It is hard to be exhaustive, or even representative, in discussing related work on SNARKs. For the sake of our paper, we focus on related work on simulation extractability notions. Groth and Maller [38] give a simulation-extractable zkSNARK that consists of only 3 group elements. Their construction is neither universal nor updatable. The recent work of Ganesh, Orlandi, Pancholi, Takahashi and Tschudi [31] shows that Bulletproofs [14] are non-malleable in AGM. More recently, Dao and Grubbs show that Spartan and Bulletproofs are non-malleable even without AGM [21]. Both Ganesh et al. and Dao et al. work extend the framework introduced by Faust, Kohlweiss, Marson and Venturi in [25] to Fiat-Shamir applied to multi-round interactive arguments. On a similar path, the work of Ganesh, Khoshakhlagh, Kohlweiss, Nitulescu and Zajac [29] shows non-malleability for Plonk, Sonic and Marlin. Both [29,31] show that interactive *arguments* can be simulation-extractable after applying the Fiat-Shamir transform. In particular, their approach consists of defining new properties, like trapdoor-less zero-knowledge[9] and unique response[10] that *need to be proven on a case-by-case basis*. Namely, for each candidate SNARK (even if resulting from a generic compiler) one needs additional effort to show that it is simulation extractable. This is arguably more challenging and less generic than our approach. Thanks to our result, once having a Φ-SE polynomial commitment, one only needs to check a very simple property on the polynomial IOP.

The work of Abdolmaleki, Ramacher and Slamanig [2] shows a generic compiler to simulation-extractable SNARKs which requires key-homomorphic signatures. Their compiler produces universally-composable SNARKs (UC-SNARKs), which they prove through a black-box straight-line extractor. To obtain a black-box straight-line extractor, they append to the SNARK proof an encryption of

[9] That is zero-knowledge where the simulator does not rely on the SRS's trapdoor but on the programmability of the random oracle.

[10] That is, at some point of the protocol, the prover becomes a deterministic algorithm.

the witness, thus achieving a relaxed succinctness w.r.t. the size of the circuit describing the relation. The recent work of Ganesh, Kondi, Orlandi, Pancholi and Takahashi [30] shows how to regain full succinctness in UC-SNARKs in the ROM through Fischlin's transform [26].

1.4 Open problems

Our framework is general enough to handle compilation from polynomial commitment schemes different than KZG. Our contribution identifies a set of properties that a polynomial commitment scheme needs to have so that the resulting SNARK is simulation-extractable. We believe that thanks to the non-malleability of random oracles the FRI scheme [10] readily possesses the necessary properties, which would imply the simulation extractability of STARKs [6].

Another advantage of our formalization of PIOP over previous proposals such as [16] is that it naturally supports optimization tricks in the literature [17]. As an intermediate step of our compiler, we define a CP-SNARK for polynomial evaluations based on KZG. While we capture the important use case of batched evaluations on a common point, for the sake of simplicity, we leave further extensions and optimizations for future work. In particular, we do not capture the case of proving evaluations on *arbitrary* linear combinations of committed polynomials. We believe this extension could be handled at the PIOP level by extending the notion to *virtual* oracle polynomials obtained through linear combinations of other oracles (and thus using the homomorphic property of KZG). We leave open the problem to extend our result to other polynomial IOP models.

Recent works extend the polynomial evaluation proofs of KZG to multiple evaluation points [49,50]. Our simulation extractability strategy for KZG can be applied partially to these schemes; however, our technique uses a clever argument to separate the realm of commitments from the realm of proofs (in KZG proofs and commitments are both of the form $[p(s)]_1$ for some polynomial p) based on their degree as polynomials. Unfortunately, the same technique does not work when the degree of the polynomial in the proof depends on the number of evaluation points in the proved statement.

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: i.e., for every $c \in \mathbb{N}$ there is an integer λ_c such that $f(\lambda) \leq \lambda^{-c}$ for all $\lambda \geq \lambda_c$. For an integer $n \geq 1$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. 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 $(L_1, \ldots, L_n)$. To get a specific value from a list, we also use the "dot" notation; e.g., we use $L.b$ to access the second element of the list $L := (a, b, c)$. An asymmetric bilinear group $\mathbb{G}$ 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 $\mathsf{GroupGen}$ be some probabilistic polynomial-time (PPT) algorithm which on input 1^λ, where λ is the security parameter, returns a description of a bilinear group $\mathcal{G}$. Elements in $\mathbb{G}_i, i \in \{1, 2, T\}$ are denoted in implicit notation as $[a]_i := aP_i$, where $P_T := e(P_1, P_2)$. Every element in $\mathbb{G}_i$ can be written as $[a]_i$ for some $a \in \mathbb{Z}_q$, but note that, given $[a]_i$, it is in general hard to compute a (discrete logarithm problem). Given $a, b \in \mathbb{Z}_q$ we distinguish between $[ab]_i$, namely the group element whose discrete logarithm base $\mathcal{P}_i$ is ab, and $[a]_i \cdot b$, namely the execution of the multiplication of $[a]_i$ and b, and $[a]_1 \cdot [b]_2 = [a \cdot b]_T$, namely the execution of a pairing between $[a]_1$ and $[b]_2$. We do not use the implicit notation for variables, e.g. $\mathsf{c} = [a]_1$ indicates that c is a variable name for the group element whose logarithm is a.

Definition 1 (Algebraic algorithm, [27]). *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 $\mathsf{z} = \langle \boldsymbol{r}, \mathsf{elems} \rangle$.*

Definition 2 (Polynomial Commitment). *A polynomial commitment is a tuple of algorithms $\mathsf{PC} := (\mathsf{KGen}, \mathsf{Com}, \mathsf{VerCom})$ that works as follows:*

- $\mathsf{KGen}(\mathsf{pp}_\mathbb{G}, d) \to \mathsf{ck}$ *takes as input group parameters $\mathsf{pp}_\mathbb{G} \leftarrow\!\$\ \mathsf{GroupGen}(1^\lambda)$, and a degree bound d, and outputs a commitment key ck.*
- $\mathsf{Com}(\mathsf{ck}, f) \to (c, o)$ *takes as input the commitment key ck, and a low degree polynomial $f \in \mathbb{F}_{\leq d}[X]$, and outputs a commitment c and an opening o.*
- $\mathsf{VerCom}(\mathsf{ck}, c, f, o) \to b$ *takes as input ck, a commitment c, a polynomial f and an opening o, and accepts ($b = 1$) or rejects ($b = 0$).*

Definition 3 (Witness Sampleability, [39]). *A distribution $\mathcal{D}$ is witness samplable if there is a PPT algorithm $\tilde{\mathcal{D}}$ s.t. for any $\mathsf{pp}_\mathbb{G}$, the random variables $\boldsymbol{A} \leftarrow\!\$\ \mathcal{D}(\mathsf{pp}_\mathbb{G})$ and $\left[\tilde{\boldsymbol{A}}\right]_1$, where $\tilde{\boldsymbol{A}} \leftarrow\!\$\ \tilde{\mathcal{D}}(\mathsf{pp}_\mathbb{G})$, are equivalently distributed.*

Definition 4 ($\mathcal{D}_{\ell,k}$-Aff-MDH assumption). *Given a matrix distribution $\mathcal{D}_{\ell,k}$, the Affine Diffie-Hellman Problem is: given $\boldsymbol{A} \in \mathbb{G}_1^{\ell \times k}$, with $\boldsymbol{A} \leftarrow\!\$\ \mathcal{D}_{\ell,k}$, find a nonzero vector $\boldsymbol{x} \in \mathbb{Z}_q^\ell$ and a vector $\boldsymbol{y} \in \mathbb{Z}_q^k$ such that $\left[\boldsymbol{x}^\top \boldsymbol{A}\right]_1 = [\boldsymbol{y}]_1$.*

Definition 5 ((d, d')-Power Polynomial in the Exponent). *The (d, d')-PEA Assumption holds for a bilinear group generator $\mathsf{GroupGen}$ if for every PPT adversary $\mathcal{A}$ that receives as input $\left(\left[1, \ldots, s^d\right]_1, \left[1, \ldots, s^{d'}\right]_2\right)$ and outputs a polynomial $p(X)$ of degree at most $\max\{d, d'\}$, and a value y, the probability that $p(s) = y$ is negligible. When $d = d'$ we use the shortcut d-PEA.*

Definition 6 (d-Power Discrete Logarithm [42]). *Given a degree bound $d \in \mathbb{N}$, the d-Power Discrete Logarithm (d-DL) assumption holds for a bilinear group generator $\mathsf{GroupGen}$ if for every PPT adversary $\mathcal{A}$ that receives as input $\left(\left[1, \ldots, s^d\right]_1, \left[1, \ldots, s^d\right]_2\right)$, and outputs the value s', the probability that $s = s'$ is negligible. We also use DL as a shortcut for 1-DL.*

Lemma 1 (d-DL $\Rightarrow$ d-PEA). *We can make a reduction to the assumption that computes s. The reduction invokes the adversary, gets $p(X) - y$ of degree d, and computes s by factoring the polynomial $p(s) - y$. As $p(s) - y = 0$ we are guaranteed that s is a root.*

Lemma 2 (DL $\Rightarrow$ $\mathcal{U}_{\ell,k}$-Aff-MDH). *When considering the uniform random distribution $\mathcal{U}_{\ell,k}$, we can make a reduction to the assumption that computes s. The reduction samples at the exponent a uniformly random matrix $\boldsymbol{A} = (a_{i,j})_{i,j} \in \mathbb{Z}_q^{\ell \times k}$ and invokes the adversary on input $[(a_{i,j})_{i,j}]_1$. Finally, let $p_i(s)$ be the i-the row of $\boldsymbol{x}^\top \boldsymbol{A}$. The reduction computes s by factoring one of the k polynomials $p_i(s) - y_i$.*

3 Policy-based Simulation-Extractable NIZKs

We start by defining the basic syntax and properties for a Non-Interactive Zero-Knowledge Argument of Knowledge. Following Groth et al. [37], we define a PT relation $\mathcal{R}$ verifying triple $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$. We say that $\mathbb{w}$ is a witness to the instance $\mathbb{x}$ being in the relation defined by the parameters pp when $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ (equivalently, we sometimes write $\mathcal{R}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) = 1$). For example, pp could be the description of a bilinear group or additionally contain a commitment key or a common reference string. A NIZK for a relation $\mathcal{R}$ (and group generator $\mathsf{GroupGen}$) is a tuple of algorithms $\Pi = (\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify})$ where:

- $\mathsf{KGen}(\mathsf{pp}_\mathbb{G}) \to \mathsf{srs}$ is a probabilistic algorithm that takes as input the parameters $\mathsf{pp}_\mathbb{G} \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$ 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 an evaluation key ek, a statement $\mathbb{x}$, and a witness $\mathbb{w}$ such that $\mathcal{R}(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ holds, and returns a proof π.
- $\mathsf{Verify}(\mathsf{vk}, \mathbb{x}, \pi) \to b$ takes a verification key, a statement $\mathbb{x}$, and either accepts ($b = 1$) or rejects ($b = 0$) the proof π.

Definition 7 (Succinctness). *A NIZK Π is said succinct if the running time of* Verify *is* $\mathsf{poly}(\lambda + |\mathbb{x}| + \log |\mathbb{w}|)$ *and the proof size is* $\mathsf{poly}(\lambda + \log |\mathbb{w}|)$.

CP-SNARKs. Commit-and-Prove succinct arguments of knowledge, or simply CP-SNARKs, are knowledge-sound and succinct NIZKs whose relations verify predicates over commitments (see Campanelli, Fiore and Querol [18]). We consider the following syntax. Briefly speaking, we refer to a CP-SNARK for a relation $\mathcal{R}$ and a commitment scheme CS as a tuple of algorithms $\mathsf{CP} = (\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify})$ where $\mathsf{KGen}(\mathsf{ck}) \to \mathsf{srs}$ is an algorithm that takes as input a commitment key ck for CS and outputs $\mathsf{srs} := (\mathsf{ek}, \mathsf{vk}, \mathsf{pp})$; 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). Moreover, if we consider the key generation algorithm KGen' that upon group parameters $\mathsf{pp}_\mathbb{G}$ first runs $\mathsf{ck} \leftarrow\!\!\$\ \mathsf{CS.KGen}(\mathsf{pp}_\mathbb{G})$, runs $\mathsf{srs} \leftarrow\!\!\$\ \mathsf{CP.KGen}(\mathsf{ck})$ and outputs srs; then the tuple $(\mathsf{KGen}', \mathsf{Prove}, \mathsf{Verify})$ defines a SNARK.

Zero-Knowledge in the SRS (and RO) model. The zero-knowledge simulator $\mathcal{S}$ of a NIZK is a stateful PPT algorithm that can operate in three modes:

- $(\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$ takes care of generating 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 $\mathbb{x}$
- $(a, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s)$ takes care of answering random oracle queries

The state $\mathsf{st}_{\mathcal{S}}$ is updated after each operation. Similarly to [25,31], we define the following wrappers.

Definition 8 (Wrappers for NIZK Simulator). *The following algorithms are stateful and share their state* $\mathsf{st} = (\mathsf{st}_{\mathcal{S}}, \mathsf{coms}, \mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathsf{aux}})$ *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.*

- $\mathcal{S}_1(\mathbb{x}, \mathsf{aux})$ *is an oracle that returns the first output of* $\mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathbb{x}, \mathsf{aux})$.[11]
- $\mathcal{S}'_1(\mathbb{x}, \mathbb{w})$ *is an oracle that 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)* $\mathcal{S}_1(\mathbb{x})$.
- $\mathcal{S}_1^F(\mathbb{x}, \mathbb{w})$ *is an oracle parameterized by a function* F*; first, it checks if* $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$*, and then runs (and returns the output of)* $\mathcal{S}_1(\mathbb{x}, F(\mathbb{x}, \mathbb{w}))$. *As explained below, this is useful to model leaky-zero-knowledge.*
- $\mathcal{S}_2(s, \mathsf{aux})$ *is an oracle that 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* $\mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s)$. *Additionally, the oracle updates the set* $\mathcal{Q}_{\mathsf{RO}}$ *by adding the tuple* (s, aux, a) *to the set.*

Almost all the oracles in our definitions can take auxiliary information as additional input. We use this auxiliary information in a rather liberal form. For example, in the definition above, the auxiliary information for $\mathcal{S}_1$ refers to the (optional) leakage required by the simulator to work in some cases (see more in Definition 10), while the auxiliary information for $\mathcal{S}_2$ can contain, for example, the algebraic representations of the group elements in s (when we restrict to algebraic adversaries) or other information the security experiments might need.

Definition 9 (Zero-Knowledge). *A NIZK* NIZK *is (perfect) zero-knowledge if there exists a PPT simulator* $\mathcal{S}$ *such that for all adversaries* $\mathcal{A}$*:*

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

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

[11] More often, simulators need only the first three inputs, see Definition 9; abusing notation, we assume that such simulators simply ignore the auxiliary input aux.

We also introduce a weaker notion of zero-knowledge. A NIZK is F-leaky zero-knowledge if its proofs may leak some information, namely a proof leaks $F(\mathbb{x}, \mathbb{w})$, where $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. We formalize this by giving the zero-knowledge simulator the value $F(\mathbb{x}, \mathbb{w})$, which should be interpreted as a hint for the simulation of proofs. This notion could be seen as an extension of the bounded leaky zero-knowledge property defined in [17] and tailored for CP-SNARKs. Our notion is a special case of the leakage-resilient zero-knowledge framework of Garg, Jain and Sahai [32] where the leakage of the simulator is known ahead of time.

Definition 10 (Leaky Zero-Knowledge). *A NIZK* NIZK *is F-leaky zero-knowledge if there exists a PPT simulator $\mathcal{S}$ such that for all adversaries $\mathcal{A}$:*

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

3.1 Policy-Based Simulation Extractability

An extraction policy defines the constraints under which the extractor must extract the witness. For example, we could consider the policy that checks that the forged instance and proof were not queried/output by the zero-knowledge simulator (thus modeling the classical simulation extractability notion), or we could consider a policy that only checks that the forged instance was not queried to the simulator, thus obtaining a weaker flavor of classical simulation extractability. Clearly, the more permissive the policy the stronger the security provided.

In our work, we also consider policies that constrain the behavior of the zero-knowledge simulator. For example, we could consider the policy that checks that the queried instances belong to the relation, thus obtaining a notion similar to true-simulation extractability (see Dodis et al. [22]). Looking ahead, contrary to the true-simulation extractability notion in [22], our policy-based version of the true-simulation extractability rather than disallowing certain queries, punishes the adversary at extraction time. It is not hard to see that the two definitional flavors, namely disallowing illegal queries versus punishing an adversary that made an illegal query are equivalent in the context of simulation extractability, because the adversary's goal is computational[12].

Extraction policies. We define an extraction policy as a tuple $\Phi = (\Phi_0, \Phi_1)$ of PPT algorithms. This is used to define Φ-simulation extractability as follows. The security experiment starts by running the extraction policy algorithm Φ_0, which generates public information pp_Φ. The public information may contain, for example, random values that define the constraints later checked by Φ_1. Therefore, we feed pp_Φ to the adversary. In the case of commit-and-prove proof systems, the public information may contain commitments for which the adversary does not know openings (but on which it can still query simulated proofs).

[12] Observe that for decisional tasks disallowing and punishing flavors can result in different security notions, see Bellare, Hofheinz and Kiltz [5].

After receiving a forgery from the adversary, the security experiment runs the extraction policy Φ_1. The policy Φ_1 is a predicate that takes as input: (i) The public parameter pp_Φ; (ii) The forged instance and proof $(\mathbb{x}, \pi)$; (iii) The view of the experiment, denoted view. Such a view contains the public parameters, the set of simulated instances and proofs $\mathcal{Q}_{\mathsf{sim}}$, and the set $\mathcal{Q}_{\mathsf{RO}}$ of queries and answers to the random oracle[13]; (iv) Auxiliary information aux_Φ which might come along with the forged instance. We use aux_Φ to provide the adversary an interface with the policy.

Definition 11 (Simulation extractability). *A NIZK Π for a relation $\mathcal{R}$ and simulator $\mathcal{S}$ is Φ-simulation-extractable if for every PPT adversary $\mathcal{A}$ there is an efficient extractor $\mathcal{E}$ such that the following advantage is negligible in λ:*

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

Below, we give a definition that explicitly considers the sub-class of PPT algebraic adversaries. To fit algebraic adversaries into our definitional framework we let the algebraic adversaries return the representation vectors (1) for any query to the simulator $\mathcal{S}$ into the auxiliary information aux and (2) for the forgery into the auxiliary information $\mathsf{aux}_{\mathcal{E}}$.

Definition 12 (Simulation extractability in the AGM). *Let Π be a NIZK for a relation $\mathcal{R}$ with a simulator $\mathcal{S}$. Π is Φ-simulation-extractable (or simply Φ-SE) if there exists an efficient extractor $\mathcal{E}$ such that for every PPT algebraic adversary $\mathcal{A}$, the advantage $\mathbf{Adv}^{\Phi\text{-}se}_{\Pi,\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda)$ (cf. Definition 11) is negligible in λ.*

4 Simulation extractability of KZG in AGM

KZG [40] is a Polynomial Commitment scheme (see Definition 2) defined over bilinear groups $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, that consists of the following algorithms:

$\mathsf{KGen}(1^\lambda, d)$ on input the security parameter 1^λ, and a degree bound $d \in \mathbb{N}$, outputs $\mathsf{ck} := ((\left[s^j\right]_1)_{j\in[0,d]}, [1, s]_2)$, for secret $s \leftarrow\!\!\$\ \mathbb{F}_q$.
$\mathsf{Com}(\mathsf{ck}, f(X))$ on input ck, a polynomial $f(X)$, outputs $c := [f(s)]_1$.
$\mathsf{VerCom}(\mathsf{ck}, c, f(X))$ outputs 1 if $c = [f(s)]_1$.

The above scheme is (standard) binding under the d-DL assumption (see [35]), in fact, given two polynomials f and f' that evaluate to the same value on the secret point s, we can find s among the roots of the (non-zero) polynomial $f - f'$.

We consider a CP-SNARK $\mathsf{CP}_{\mathsf{evl}}$ for the relation $\mathcal{R}_{\mathsf{evl}}((x, y), f) := f(x) = y$, where f is committed as $[f(s)]_1$. The scheme constructed in this section requires one $\mathbb{G}_1$ element to commit to $f(X)$, one $\mathbb{G}_1$ element for the evaluation proof, and checking this proof of evaluation requires two pairings, and is knowledge extractable in the AGM [20]. This is taken from [17] but adapted to AGM only.

[13] Even if the given NIZK is not in the random oracle (namely neither the prover nor the verifier algorithms make random oracle queries) it still makes sense to assume the existence of the set $\mathcal{Q}_{\mathsf{RO}}$. This is useful to model security for NIZK protocols that eventually are used as sub-protocols in ROM-based protocols.

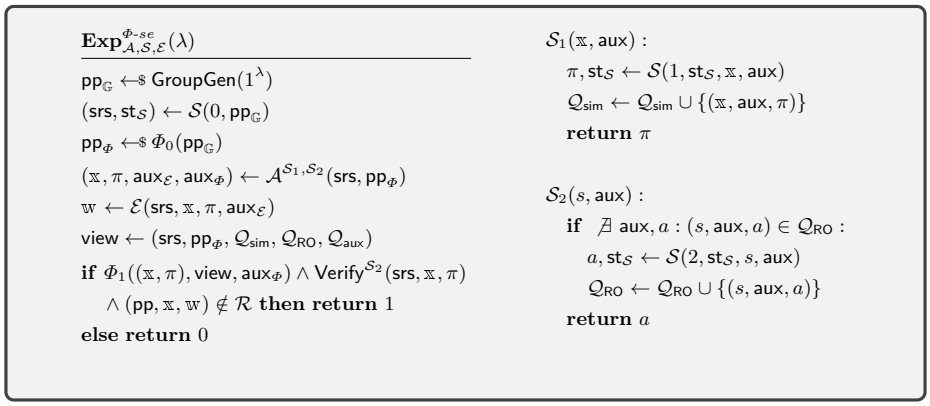

Fig. 1. The Φ-simulation extractability experiments in ROM. The extraction policy Φ takes as input the public view of the adversary view (namely, all the inputs received and all the queries and answers to its oracles). The set $\mathcal{Q}_{\mathsf{sim}}$ is the set of queries and answers to the simulation oracle. The set $\mathcal{Q}_{\mathsf{RO}}$ is the set of queries and answers to the random oracle. The set $\mathcal{Q}_{\mathsf{aux}}$ is the set of all the auxiliary information sent by the adversary (depending on the policy, this set might be empty or not). The wrappers $\mathcal{S}_1$ and $\mathcal{S}_2$ deal respectively with the simulation queries and the random oracle queries of $\mathcal{A}$ in the experiment.

$\mathsf{KGen}_{\mathtt{evl}}$**:** parse ck as $((\left[s^j\right]_1)_{j\in[0,d]}, [1,s]_2)$ and define $\mathsf{ek} := \mathsf{ck}$ and $\mathsf{vk} := [1,s]_2$, and return $\mathsf{srs} := (\mathsf{ek}, \mathsf{vk})$.

$\mathsf{Prove}_{\mathtt{evl}}(\mathsf{ek}, \mathbb{x} = (\mathsf{c}, x, y), \mathbb{w} = f)$**:** output $\pi := [\pi(s)]_1$, where $\pi(X)$ is the polynomial such that $\pi(X)(X - x) \equiv f(X) - y$.

$\mathsf{Verify}_{\mathtt{evl}}(\mathsf{vk}, \mathbb{x} = (\mathsf{c}, x, y), \pi)$**:** output 1 iff $e(\mathsf{c} - [y]_1, [1]_2) = e(\pi, [s - x]_2)$.

The extraction policy for $\mathsf{CP}_{\mathtt{evl}}$. We define $\boldsymbol{\Phi}^{\mathtt{s\text{-}adpt}}_{\mathtt{evl}} = \{\Phi_{\mathcal{D}}\}_{\mathcal{D}}$ as the family (indexed by a sampler $\mathcal{D}$) of *semi-adaptive* extraction policies for the KZG-based $\mathsf{CP}_{\mathtt{evl}}$ CP-SNARK. Indeed, as we show below, the evaluation points x_j for the instances for which the adversary can see simulated proofs are selectively chosen independently of the commitment key, while the evaluation values y can be adaptively chosen by the adversary. Each policy $\Phi_{\mathcal{D}}$ is a tuple of the form $(\Phi^{\mathcal{D}}_0, \Phi_1)$, as defined in Section 3.1, where $\Phi^{\mathcal{D}}_0$ outputs the parameters pp_Φ while Φ_1 outputs a bit. In particular, $\Phi^{\mathcal{D}}_0$ on input group parameters $\mathsf{pp}_{\mathbb{G}}$ outputs $\mathsf{pp}_\Phi := (\mathsf{coms}, \mathcal{Q}_x)$, where coms is a vector of commitments sampled from $\mathcal{D}$, and $\mathcal{Q}_x$ is a set of evaluation points.

For sake of clarity, we define the policy Φ_1 as the logical conjunction of a "simulator" policy Φ_{sim} and an "extractor" policy Φ_{ext}, i.e. $\Phi_1 = \Phi_{\mathsf{sim}} \wedge \Phi_{\mathsf{ext}}$. The first policy defines rules under which we can classify a simulation query *legal*, while the second one defines rules under which the extractor must be able to extract a meaningful witness.

Definition 13. *Let Φ_{sim} be the policy that returns 1 if and only if:*

1. **Points check:** *let $(\mathbb{x}_i, \mathsf{aux}_i, \pi_i)_i$ be all the entries of $\mathcal{Q}_{\mathsf{sim}}$. Recall that an instance $\mathbb{x}$ can be parsed as (c, x, y). Check that $\forall i : \mathbb{x}_i.x \in \mathcal{Q}_x$.*

2. **Commitment Check:** *For all* $i \in [Q_{\mathsf{sim}}]$*, parse* aux_i *as the representation vectors for* $\mathbb{x}_i.\mathsf{c}$ *and* π_i *such that* $\boldsymbol{r}_i = \boldsymbol{f}_i \| \boldsymbol{c}_i$ *is the algebraic representation of the commitment* $\mathbb{x}_i.\mathsf{c}$*. For any* i *check that* $\langle \boldsymbol{f}_i, \mathsf{ek} \rangle + \langle \boldsymbol{c}_i, \mathsf{coms} \rangle = \mathbb{x}_i.\mathsf{c}$.
3. **Algebraic Consistency:** *Let* $\mathcal{I}_j := \{i : \mathbb{x}_i.x = x_j\}$ *and let* $\boldsymbol{R}_j := (\boldsymbol{c}_i)_{i \in \mathcal{I}_j}$*. Check that* $\forall j$*: (i) the system of linear equations* $\boldsymbol{R}_j \cdot \boldsymbol{z} = \boldsymbol{y}_j$ *has at least a solution, where* $\boldsymbol{z}$ *are the variables and* $\boldsymbol{y}_j = (\mathbb{x}_i.y - \langle \boldsymbol{f}_i, (1, x_j, \ldots, x_j^d) \rangle)_{i \in \mathcal{I}_j}$.

In more intuitive terms, for every simulation query (c, x, y) made by the adversary: (1) ensures that x is in the set $\mathcal{Q}_x$ chosen at the beginning of the experiment (this is the semi-adaptive restriction); (2) ensures that c is computed as a linear combination of the simulated commitments and the $\mathbb{G}_1$ elements of the SRS, but *not of simulated proofs*; (3) ensures that overall the queried statements are plausibly true (e.g., the adversary does not ask to open the same (c, x) at two different $y \neq y'$). We notice that the "Algebraic Consistency" check is necessary since, if violated it would enable a class of generic attacks. We briefly mention one attack in the proof intuition and we refer the reader to [24] for the details.

Next, we define the policy Φ_{ext} as the logical disjunction of two policies, $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ and $\Phi^{\mathrm{der}}_{\mathsf{ext}}$. To this end, we first define some notation: let $g_c \colon \mathbb{G}_1 \times \{0,1\}^* \to \{0,1\}$ be a function that on inputs a group element c and a string s, that can be parsed as a list of group elements c_i followed by a second string $\tilde{s}$, outputs 1 iff $\exists i : \mathsf{c} = \mathsf{c}_i$.

Definition 14. *Let* $\Phi_{\mathsf{ext}}, \Phi^{rnd}_{\mathsf{ext}}$ *and* $\Phi^{der}_{\mathsf{ext}}$ *be predicates that, parsing the forgery instance* $\mathbb{x}^* = (\mathsf{c}^*, x^*, y^*)$*, are defined as follows:*

- $\Phi^{rnd}_{\mathsf{ext}}$ *returns 1 if and only if there exist a query* (s, aux, a) *to the random oracle and* aux *contains a non-constant polynomial* $h(X)$ *such that the following conditions are satisfied:*
  1. **Hashing check:** $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$*, note that* $\mathcal{Q}_{\mathsf{RO}}$ *is contained in* view*,*
  2. **Decoding check:** $g_c(c^*, s) = 1$.
  3. **Polynomial check:** $g_h(h, \mathsf{aux}) = 1$*, where* $g_h \colon \mathbb{F}[X] \times \{0,1\}^* \to \{0,1\}$ *is a function that on input a polynomial* $h(X)$ *and a string* aux *outputs 1 iff* $h(X)$ *is encoded in* aux*.*
  4. **Computation check:** $h(a) = x^*$.
- $\Phi^{der}_{\mathsf{ext}}$ *returns 1 iff* $\exists (\mathbb{x}, \cdot, \pi) \in \mathcal{Q}_{\mathsf{sim}}$ *s.t.* $\mathbb{x} := (\mathsf{c}^*, x^*, y')$ *and* $(y', \pi) \neq (y^*, \pi^*)$.
- Φ_{ext} *returns logical disjunction of* $\Phi^{rnd}_{\mathsf{ext}}$ *and* $\Phi^{der}_{\mathsf{ext}}$.

More intuitively, $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ checks that the point x^* is obtained from the random oracle *after* querying it on the commitment c^*, whereas $\Phi^{\mathrm{der}}_{\mathsf{ext}}$ checks if $\mathbb{x}^*$ is a strong forgery, namely it is a new evaluation proof for a statement (c^*, x^*) already queried to the simulation oracle.

Theorem 1. *For any witness samplable distribution* $\mathcal{D}$ *that is* $\mathcal{D}$*-Aff-MDH-secure (see Definition 4), any bilinear-group generator* $\mathsf{GroupGen}$ *that samples the generator of the group* $\mathbb{G}_1$ *uniformly at random,* $\forall \Phi_{\mathcal{D}} \in \boldsymbol{\Phi}^{\mathsf{s\text{-}adpt}}_{\mathsf{evl}}$*, KZG is* $\Phi_{\mathcal{D}}$*-simulation-extractable in the AGM. In particular, there exists* $\mathcal{E}$ *such that for any algebraic adversary* $\mathcal{A}$*:*

$$\mathbf{Adv}^{\Phi_{\mathcal{D}}\text{-}se}_{\mathsf{CP}_{\mathsf{evl}},\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) \leq O(\epsilon_{(Q_x+d+1)\text{-}\mathsf{DL}}(\lambda)) + O(\epsilon_{Aff\text{-}MDH}(\lambda)) + \mathsf{poly}(\lambda)\epsilon_h$$

where $Q_x := |\mathcal{Q}_x|$*,* d *is the maximum degree supported by* $\mathsf{CP}_{\mathtt{evl}}$*,* $\epsilon_{(Q_x+d+1)\text{-}\mathsf{DL}}(\lambda)$ *is the maximum advantage for any algebraic PT adversary against the* $(Q_x + d + 1)$*-strong Discrete-Log Assumption,* $\epsilon_{\mathit{Aff\text{-}MDH}}(\lambda)$ *is the maximum advantage for any algebraic PT adversary against the* $\mathcal{D}$*-Aff-MDH Assumption,* h *is the polynomial that satisfies the Polynomial check of* $\Phi_{\mathcal{D}}$*, and* $\epsilon_h = \frac{\deg(h)}{q}$*.*

We show in the full version [24] how to generalize the scheme to support hiding commitments, and we extend our result to the hiding setting. Also, we consider a scheme $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ for batch evaluations which follows from [28,44].

Proof intuition of Theorem 1. We consider an adversary whose forgery satisfies the predicate $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$. We first show an alternative way to simulate KZG proofs. This step allows one to move from a simulator whose trapdoor is a "secret exponent" s to a simulator whose trapdoor is a 'tower' of $\mathbb{G}_1$-elements $\left[s^i\right]_1$. The simulated SRS seen by the adversary includes only high-degree polynomials of the form $\left[p(s)s^i\right]_1$, while the simulator keeps the low-degree monomials $\left[s^i\right]_1$ for simulation. Here, p is a polynomial that vanishes in all the points to be asked in the simulation queries (this is reminiscent of the reduction technique for Boneh-Boyen signatures [11]). Since we program the SRS based on the queries our simulator is only *semi-adaptive*, namely it can simulate proofs for a (exponentially large) subset of all the statements. This first change essentially simplifies the objects involved in our analysis, from rational polynomials (with the formal variable being the trapdoor) to standard polynomials.

Next, we need to show that the adversary cannot mix the simulated commitments and the forgery material. In particular, we need to show that the forged proof is not derived as a linear combination involving simulated commitments. To show this, we use the fact that the degree of the proof must be smaller than the degrees of simulated commitments, otherwise we could break the d-DL assumption in the AGM. This intuitively comes from the fact that the verification equation *lifts* the degree of the polynomial in the forged proof (as it is multiplied by $(X - x^*)$). Similarly, we need to show that the forged instance cannot use a linear combination that involves the simulated commitments. For this, we use the Aff-MDH assumption to handle multiple evaluation proofs on different simulated commitments on the same evaluation point. In particular, we reduce the view of many simulated proofs over many commitments and many evaluation points to a view that only contains $\left[p(s)s^i\right]_1$ and (non-rational) polynomials $[p(s)/(s - x_j)]_1$. At this point, the attacker could still perform an attack if it could decide the evaluation point x^* arbitrarily. The attack works as follows: (i) the adversary asks a simulation proof π for $\mathbb{x} = (\mathsf{c}, x, y)$, and (ii) produces the forgery $\mathbb{x}^* = (\mathsf{c} + \alpha\pi, x - \alpha, y), \pi$, for any $\alpha \in \mathbb{Z}_q$. It is easy to check that the forgery satisfies the verification equation. However, for this attack to work the attacker needs to set the commitment in the forged instance as a function of $x^* = x - \alpha$. The last part of our analysis shows that, indeed, the algebraic representation of the commitment in the forgery cannot depend on x^* and that this attack cannot be mounted when x^* is chosen after the commitment with sufficient randomness. For the second case, we can reduce a $\Phi^{\mathrm{der}}_{\mathsf{ext}}$ forgery to a

$\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ forgery. In fact, such a forgery together with the simulated proofs set an algebraic inconsistency, a sub-case of the condition avoided by Item 3 of Definition 13, thus enabling an attack. In more detail, given a $\Phi^{\mathrm{der}}_{\mathsf{ext}}$-forgery $(\mathsf{c}, x, y), \pi$ and let $((\mathsf{c}, x, y'), \pi') \in \mathcal{Q}_{\mathsf{sim}}$ we can define a new $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$-forgery $(\mathsf{c}^*, x^*, y^*), \pi^*$ where $\mathsf{c}^* = (\pi' - \pi)$, $x^* = \mathsf{RO}(\mathsf{c}^*)$ and $\pi^* = \frac{\pi - \pi'}{x^* - x}$ and $y^* = \frac{y - y'}{x^* - x}$. We can prove that the verification equation holds noticing that $(\pi - \pi')(s - x) = [y - y']_1$ and by simple algebraic manipulations.

Proof (of Theorem 1). We stress that $\mathcal{A}$ is algebraic (cf. Definition 1), therefore for each group element output it additionally attaches a representation $\boldsymbol{r}$ of such a group element with respect to all the elements seen during the experiment (included elements in coms). In particular, we assume that for each query $(\mathbb{x}, \mathsf{aux})$ to the oracle $\mathcal{S}_1$ we can parse the value aux as $(\boldsymbol{r}, \mathsf{aux}')$ and $\boldsymbol{r}$ is a valid representation for $\mathbb{x}.\mathsf{c}$. Similarly, for the queries (s, aux) to $\mathcal{S}_2$, aux includes a valid representation for all the group elements g_i encoded in s, i.e. such that $g_c(\mathsf{g}_i, s) = 1$. Together with its forgery, the algebraic adversary encodes a polynomial $h(X)$ in aux_ϕ, and stores in $\mathsf{aux}_{\mathcal{E}}$ two representation vectors $\boldsymbol{r}_{c^*}$ and $\boldsymbol{r}_{\pi^*}$ for the two group elements c^* and π^*. We can parse the vectors $\boldsymbol{r}_\tau := \boldsymbol{f}_\tau \| \boldsymbol{c}_\tau \| \boldsymbol{o}_\tau$ for $\tau \in \{c^*, \pi^*\}$ where $\boldsymbol{f}_\tau$ is the vector of coefficients associated to group elements ek, $\boldsymbol{c}_\tau$ is the vector of coefficients associated to group elements $\mathsf{coms} = ([c_i]_1)_{i \in [Q_c]}$, and $\boldsymbol{o}_\tau$ is the vector of coefficients associated to the group elements of the simulated proofs proofs. Namely, for $\tau \in \{c^*, \pi^*\}$ we have:

$$\tau = \langle \boldsymbol{f}_\tau, \mathsf{ek} \rangle + \langle \boldsymbol{c}_\tau, \mathsf{coms} \rangle + \langle \boldsymbol{o}_\tau, \mathsf{proofs} \rangle.$$

We can assume w.l.g. that all the simulation queries and the forgery of the adversary $\mathcal{A}$ agree with the policy $\Phi_{\mathcal{D}}$, as otherwise the adversary would automatically lose the experiment. We assume that $\boldsymbol{f}_{i,j} = \boldsymbol{0}, \forall i, j$, i.e., the adversary asks simulation queries on commitments that are a linear combination of coms only: this is also w.l.g. as we briefly show below. Given a commitement $\mathsf{c}_{i,j} = \mathbb{x}_{i,j}.\mathsf{c}$, whose representation is $\boldsymbol{r}_{i,j} = \boldsymbol{f}_{i,j} \| \boldsymbol{c}_{i,j}$, the adversary could compute a proof $\pi_{i,j}$ for the point x_j and the evaluation value y as follows:

1. let $y' = f_{i,j}(x_j)$, $\mathcal{A}$ computes the commitment $\mathsf{c}' \leftarrow \mathsf{Com}(\mathsf{ck}, f_{i,j}(X))$, and the "honest" proof π' for (c', x_j, y')
2. asks the simulation oracle to provide a proof $\tilde{\pi}$ for the instance $(\mathsf{c} - \mathsf{c}', x_j, y - y')$ with representation $\boldsymbol{0} \| \boldsymbol{c}_{i,j}$
3. recombines the proof $\pi_{i,j} = \pi' + \tilde{\pi}$

We define our extractor to be the *canonical* extractor that returns the polynomial $f(X) \leftarrow \langle \boldsymbol{f}_{c^*}, (1, X, \ldots, X^d) \rangle$. We start by proving that for any algebraic adversary $\mathcal{A}$ whose forgery satisfies the predicate $\Phi^{\mathrm{der}}_{\mathsf{ext}}$, there exists an algebraic adversary $\mathcal{B}$ whose forgery satisfies the predicate $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$. Let $\{\Phi'_{\mathcal{D}}\}_{\mathcal{D}}$ be the family of policies defined exactly as $\boldsymbol{\Phi}^{\mathtt{s\text{-}adpt}}_{\mathtt{evl}}$ with the difference that the extracion policy Φ_{ext} is equal to $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ (i.e., there is no logical disjunction with $\Phi^{\mathrm{der}}_{\mathsf{ext}}$).

Lemma 3. *For any algebraic adversary $\mathcal{A}$ there is an algebraic adversary $\mathcal{B}$:*

$$\mathbf{Adv}^{\Phi_{\mathcal{D}}\text{-}\mathtt{se}}_{\mathsf{CP}_{\mathtt{evl}},\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda) = \mathbf{Adv}^{\Phi'_{\mathcal{D}}\text{-}\mathtt{se}}_{\mathsf{CP}_{\mathtt{evl}},\mathcal{B},\mathcal{S},\mathcal{E}}(\lambda)$$

Proof. First, we notice that once we fix a commitment c, a point x, and a value y, there is a unique proof π that can satisfy the KZG verification equation. Thus, the predicate $\Phi^{\mathrm{der}}_{\mathsf{ext}}$ can be simplified as requiring that an adversary outputs a valid proof π^* and a value y^* such that $\exists((\mathsf{c}^*, x^*, y'), \cdot, \pi) \in \mathcal{Q}_{\mathsf{sim}}$ and $y^* \neq y'$.

The reduction $\mathcal{B}$ internally runs $\mathcal{A}$ forwarding all the simulation queries, up to the forgery $(\mathbb{x}^*, \pi^*)$, where $\mathbb{x}^* = (\mathsf{c}^*, x^*, y^*)$. If the simulation queries and/or the forgery of the adversary $\mathcal{A}$ do not agree with the policy $\Phi_{\mathcal{D}}$, i.e. $\mathcal{A}$ automatically loses its game, $\mathcal{B}$ aborts. Otherwise, it must be true that the forgery of $\mathcal{A}$ either (i) satisfies the extraction predicate $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ or (ii) satisfies the extraction predicate $\Phi^{\mathrm{der}}_{\mathsf{ext}}$. Both cases can be efficiently checked by $\mathcal{B}$. In case (i) $\mathcal{B}$ would simply forward the forgery of $\mathcal{A}$ retaining the same advantage of $\mathcal{A}$. Otherwise, before submitting the forgery, $\mathcal{B}$ retrieves from $\mathcal{Q}_{\mathsf{sim}}$ the statement $\mathbb{x} \coloneqq (\mathsf{c}^*, x^*, y')$, where $y' \neq y^*$, and the corresponding proof π output by $\mathcal{S}_1$. Then $\mathcal{B}$ produces the forgery:

$$\hat{c} \leftarrow \pi^* - \pi, \qquad \hat{x} \leftarrow h(a), \qquad \hat{\pi} \leftarrow \frac{\pi - \pi^*}{\hat{x} - x^*}, \qquad \hat{y} \leftarrow \frac{y' - y^*}{\hat{x} - x^*}$$

which satisfies the verification equation (cf [24]), and the extraction predicate $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ when $(\hat{c}, h, a) \in \mathcal{Q}_{\mathsf{RO}}$. □

Thanks to Lemma 3 we can assume that the forgery of $\mathcal{A}$ satisfies the extraction predicate $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$. We let $\mathbf{H}_0$ be the $\mathbf{Exp}^{\Phi_{\mathcal{D}}\text{-}\mathsf{se}}_{\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda)$ experiment, and we denote by ϵ_i the advantage of $\mathcal{A}$ to win $\mathbf{H}_i$, i.e. $\epsilon_i \coloneqq \Pr[\mathbf{H}_i = 1]$.

Hybrid $\mathbf{H}_1$. Recall that $\mathcal{D}$ is witness samplable, thus according to Definition 3 there exists a PPT algorithm $\tilde{\mathcal{D}}$ associated with the sampler $\mathcal{D}$. The hybrid experiment $\mathbf{H}_1$ is identical to the previous one, but the group elements in coms are "sampled at exponent", i.e. we use $\tilde{\mathcal{D}}$ to generate the field elements $\boldsymbol{\gamma}$, and we let $\mathsf{coms} \leftarrow [\boldsymbol{\gamma}]_1$; we also add $\boldsymbol{\gamma}$ to $\mathsf{st}_{\mathcal{S}}$. By the witness sampleability of $\mathcal{D}$, $\mathbf{H}_0$ and $\mathbf{H}_1$ are perfectly indistinguishable, thus $\epsilon_1 = \epsilon_0$.

Hybrid $\mathbf{H}_2$. In this hybrid, we change the way we generate the SRS srs and the way in which $\mathcal{S}_1$ simulates the proofs. Let $\left((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), [1]_1, [1]_2\right) \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$, sample $s \leftarrow\!\!\$\ \mathbb{F}$ and compute $\left[s, \ldots, s^{D+d}\right]_1, [1, s]_2$, where $D \leftarrow Q_x + 1$. Let $x_r \leftarrow\!\!\$\ \mathbb{F}$, and let $p(X)$ be the vanishing polynomial in $\mathcal{Q}_x \cup \{x_r\}$, namely$p(X) \coloneqq (X - x_r) \prod_{x \in \mathcal{Q}_x} (X - x)$. Let also $p_j(X) \coloneqq p(X)(X - x_j)^{-1}$, for $j \in [Q_x]$. In $\mathbf{H}_2$ we have that:

- $\mathsf{pp}_{\mathbb{G}} \coloneqq ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), [p(s)]_1, [1]_2)$,
- $\mathsf{srs} \coloneqq (\mathsf{ek}, \mathsf{vk})$, where $\mathsf{ek} \leftarrow \left[p(s), p(s)s, \ldots, p(s)s^d\right]_1$ and $\mathsf{vk} \leftarrow [1, s]_2$,
- $\mathsf{st}_{\mathcal{S}} \coloneqq \left[1, s, \ldots, s^{D+d}\right]_1, [1, s]_2, \boldsymbol{\gamma}$.

Upon a query of the form $(\mathbb{x} = (\mathsf{c}, x_j, y_k), \mathsf{aux} = (\boldsymbol{r}_c, \mathsf{aux}'))$ to $\mathcal{S}_1$, the latter outputs the proof $\pi \leftarrow [(\langle \boldsymbol{r}_c, \boldsymbol{\gamma} \rangle - y_k) \cdot p_j(s)]_1$, and updates $\mathcal{Q}_{\mathsf{sim}}$ accordingly.

We now show that $\mathbf{H}_1 \equiv \mathbf{H}_2$, i.e., the view offered to the adversary $\mathcal{A}$ is identically distributed in the two experiments.

Lemma 4. $\epsilon_2 = \epsilon_1$.

Proof. Notice that in $\mathbf{H}_2$ we sample from $\mathsf{GroupGen}$ the description of the group, and then we set the generator of $\mathbb{G}_1$ to $[p(s)]_1$ which, thanks to the random root x_r, is distributed uniformly at random even given the value s. It is not hard to verify that the simulated proofs generated by the hybrid $\mathbf{H}_2$ pass the verification equations, in fact, we are assuming that queried commitment c are of the form $\langle \boldsymbol{r}_c, \mathsf{coms} \rangle$. Additionally, since the proofs are uniquely determined given the SRS and the statements, the simulated proofs created in $\mathbf{H}_2$ are distributed as the simulated proofs generated by the simulator $\mathcal{S}_1$ in $\mathbf{H}_1$. Thus the advantage of $\mathcal{A}$ is the same in the two experiments. □

Given an algebraic adversary $\mathcal{A}$ we can define a new adversary, $\mathcal{A}_{\mathsf{c}}$, that we call the *core* adversary. Whenever the adversary $\mathcal{A}$ outputs a group element g it provides a representation vector $\boldsymbol{r}_{\mathsf{g}} := \boldsymbol{f}_{\mathsf{g}} \| \boldsymbol{c}_{\mathsf{g}} \| \boldsymbol{o}_{\mathsf{g}}$ for g such that:

$$\mathsf{g} = \langle \boldsymbol{f}_{\mathsf{g}}, \mathsf{ek} \rangle + \langle \boldsymbol{c}_{\mathsf{g}}, \mathsf{coms} \rangle + \langle \boldsymbol{o}_{\mathsf{g}}, \mathsf{proofs} \rangle.$$

$\mathcal{A}_{\mathsf{c}}$ runs internally $\mathcal{A}$ and forwards all the queries and answers from $\mathcal{A}$ to its simulation oracle. However, the way of simulating RO queries must ensure to not alter the result of the extractor policy, i.e. the "hash-check" for x^*. This is why we cannot simply forward the queries of $\mathcal{A}$ to the random oracle. Therefore, we keep track of the queries made by $\mathcal{A}$ in the list $\mathcal{Q}_{\mathsf{RO},\mathcal{A}}$ and the list of queries made by the core adversary in $\mathcal{Q}_{\mathsf{RO}}$. More in detail, when $\mathcal{A}$ queries the RO with (s, aux), the adversary $\mathcal{A}_{\mathsf{c}}$ makes a "core" RO query $(s_{\mathsf{c}}, \mathsf{aux}_{\mathsf{c}})$ such that:

1. Let s be parsed as $(\mathsf{g}_i)_{i \in [k]}$ (the group elements in s whose representations $\boldsymbol{r}_{\mathsf{g}_i} := \boldsymbol{f}_{\mathsf{g}_i} \| \boldsymbol{c}_{\mathsf{g}_i} \| \boldsymbol{o}_{\mathsf{g}_i}$ are in aux) and a string $\tilde{s}$. Notice, since the adversary is algebraic we can un-ambiguously parse s as such.
2. For each i, $\mathcal{A}_{\mathsf{c}}$ computes the group elements $\mathsf{g}'_i = \mathsf{g}_i - \langle \boldsymbol{f}_{\mathsf{g}_i}, \mathsf{ek} \rangle$. $\mathcal{A}_{\mathsf{c}}$ encodes into the string s' the group elements $(\mathsf{g}_i, \mathsf{g}'_i)_{i \in [k]}$.
3. $\mathcal{A}_{\mathsf{c}}$ queries the RO with $(s_{\mathsf{c}}, \mathsf{aux}_{\mathsf{c}})$, where $s_{\mathsf{c}} := s' \| \tilde{s}$, and $\mathsf{aux}_{\mathsf{c}}$ contains the representations of all the group elements in s' and the same function h encoded in aux. Finally, it forwards the output to $\mathcal{A}$, i.e. it adds (s, aux, a) to $\mathcal{Q}_{\mathsf{RO},\mathcal{A}}$, and adds (s, s_{c}) to (the initially empty) $\mathcal{Q}_s$.

Eventually, $\mathcal{A}$ outputs as forgery a string s and the tuple $(\mathsf{c}', x', y', \pi')$, together with representation vectors $\boldsymbol{r}_{c'}$ and $\boldsymbol{r}_{\pi'}$. Let $f(X) := \langle \boldsymbol{f}_{c'}, (1, X, \ldots, X^d) \rangle$, $y := f(x')$, and $q(X)$ be such that $q(X)(X - x') = f(X) - y$. Let $\boldsymbol{f}_q$ be the vector of the coefficients of $q(X)$, namely $q(X) := \langle \boldsymbol{f}_q, (1, X, \ldots, X^d) \rangle$. The core adversary $\mathcal{A}_{\mathsf{c}}$ returns for its forgery the string s_{c} such that $(s, s_{\mathsf{c}}) \in \mathcal{Q}_s$, and the tuple $(\mathsf{c}^*, x', y^*, \pi^*)$, where $y^* \leftarrow y' - f(x')$ and:

$$\mathsf{c}^* \leftarrow \mathsf{c}' - \underbrace{[f(s)p(s)]_1}_{\mathsf{Com}(\mathsf{ck}, f(X))}, \qquad \pi^* \leftarrow \pi' - \underbrace{[q(s)p(s)]_1}_{\mathsf{Com}(\mathsf{ck}, q(X))}$$

inserting into aux_{Φ} the (correct) algebraic representations $(\mathbf{0} \| \boldsymbol{c}_{c'} \| \boldsymbol{o}_{c'})$ for c^* and $((\boldsymbol{f}_{\pi'} - \boldsymbol{f}_q) \| \boldsymbol{c}_{\pi'} \| \boldsymbol{o}_{\pi'})$ for π^*.

Hybrid $\mathbf{H}_3$. This hybrid is exactly the same of $\mathbf{H}_2$ but instead of running the experiment with the adversary $\mathcal{A}$ we run it with the core adversary $\mathcal{A}_{\mathsf{c}}$.

Lemma 5. $\epsilon_3 = \epsilon_2$.

Proof. First, by construction, it is easy to verify that $\mathcal{A}_{\mathsf{c}}$ is algebraic. Thus we need to show that the forgery of $\mathcal{A}$ is valid if and only if the forgery of $\mathcal{A}_{\mathsf{c}}$ is valid. By the verification equation of the forgery of $\mathcal{A}_{\mathsf{c}}$, we have:

$$
\begin{aligned}
&e(\mathsf{c}^* - [y^*]_1, [1]_2) - e(\pi^*, [s - x^*]_2) = \\
&e(\mathsf{c}' - [f(s)p(s)]_1 - [y' - f(x')]_1, [1]_2) - e(\pi' - [q(s)p(s)]_1, [s - x^*]_2) = \\
&e(\mathsf{c}' - [y']_1, [1]_2) - e(\pi', [s - x']_2) - [f(s)p(s) - f(x') - q(s)p(s)(s - x^*)]_T = \\
&e(\mathsf{c}' - [y']_1, [1]_2) - e(\pi', [s - x']_2),
\end{aligned}
$$

where the last equation holds since $q(X)(X - x') = (f(X) - f(x'))$ and $x^* = x'$. Finally, notice that a forgery is valid for $\mathcal{A}$ if it provides a string s that satisfies the "hash check" of Φ_{ext}. We have that there exist s, aux, a, and $h(X)$ such that: (i) $g_c(\mathsf{c}^*, s) = 1$, (ii) $g_h(h, \mathsf{aux}) = 1$, (iii) $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO},\mathcal{A}}$, and (iv) $x^* = h(a)$ for the forgery of $\mathcal{A}$. The way $\mathcal{A}_{\mathsf{c}}$ simulates the RO queries ensures that for the query s of $\mathcal{A}$ to the RO, the core adversary sent the "core" RO query s_{c} that encodes both c' and c^*, thus we have that (i) $g_c(\mathsf{c}^*, s_{\mathsf{c}}) = 1$, (ii) $g_h(h, \mathsf{aux}_{\mathsf{c}}) = 1$, (iii) $(s_{\mathsf{c}}, \mathsf{aux}_{\mathsf{c}}, a) \in \mathcal{Q}_{\mathsf{RO}}$, and (iv) $x^* = h(a)$ for the forgery of $\mathcal{A}_{\mathsf{c}}$. □

Notice that if we run the canonical extractor on the outputs of the core adversary $\mathcal{A}_{\mathsf{c}}$, the extractor sets the extracted witness to be the zero polynomial.

Hybrid $\mathbf{H}_4$. The hybrid $\mathbf{H}_4$ additionally checks that $\boldsymbol{f}_{\pi^*} \neq \mathbf{0} \vee \boldsymbol{c}_{\pi^*} \neq \mathbf{0}$, and if the condition holds the adversary $\mathcal{A}_{\mathsf{c}}$ loses the game.

Lemma 6. $\epsilon_3 \leq \epsilon_4 + \epsilon_{(Q_x+d+1)\text{-}\mathsf{DL}}$

Proof. Recall that from the definition of the experiment, upon a query $(\mathbb{x}, \mathsf{aux})$ from $\mathcal{A}_{\mathsf{c}}$ to the simulation oracle of the form $\mathbb{x} = (\mathsf{c}, x_j, y_k)$ and $\mathsf{aux} = \boldsymbol{r}$ where $\mathsf{c} = \langle \boldsymbol{r}, \mathsf{coms} \rangle$, the adversary receives the proof $[\pi_{\boldsymbol{r},j,k}(s)]_1$ where:

$$\pi_{\boldsymbol{r},j,k}(X) \coloneqq (\langle \boldsymbol{r}, (\gamma_i)_i \rangle - y_k) p_j(X).$$

Consider the following polynomials:

$$
\begin{aligned}
c^*(X) &\coloneqq \sum_{i \in [Q_c]} c_{c^*,i} \cdot \gamma_i p(X) + \sum_{\boldsymbol{r},j,k} o_{c^*,\boldsymbol{r},j,k} \cdot \pi_{\boldsymbol{r},j,k}(X) \\
\pi^*(X) &\coloneqq \sum_{i \in [Q_c]} c_{\pi^*,i} \cdot \gamma_i p(X) + \sum_{\boldsymbol{r},j,k} o_{\pi^*,\boldsymbol{r},j,k} \cdot \pi_{\boldsymbol{r},j,k}(X) + \sum_{i \in [d+1]} f_{\pi^*,i} X^{i-1} p(X) \\
v(X) &\coloneqq c^*(X) - y^* p(X) - (X - x^*)\pi^*(X)
\end{aligned}
$$

By the guarantees of the AGM, we have $\mathsf{c}^* = [c^*(s)]_1$ and $\pi^* = [\pi^*(s)]_1$, moreover, if the verification equation is satisfied by the forgery of $\mathcal{A}_{\mathsf{c}}$, then $v(s) = 0$.

Next, we show that when the forgery of the adversary is valid the probability of $\boldsymbol{f}_{\pi^*} \neq \mathbf{0}$ or $\boldsymbol{c}_{\pi^*} \neq \mathbf{0}$ is bounded by $\epsilon_{(Q_x+d+1)\text{-}\mathsf{DL}}$. First, notice that if the verification equation for $\mathcal{A}_{\mathsf{c}}$ holds then the polynomial $v(X)$ must be equivalent

to the zero polynomial with overwhelming probability. In fact, $v(s) = 0$ when the verification equation holds; if $v(X)$ is not the zero polynomial then, by Lemma 1, we can reduce $\mathcal{A}_{\mathsf{c}}$ to an adversary to the $(Q_x + d + 1)$-DL assumption. Thus:

$$c^*(X) - y^* p(X) - (X - x^*)\pi^*(X) = v(X) = 0. \tag{1}$$

By the guarantees of the AGM, the polynomial $\pi^*(X)$ is a linear combination of elements that depend on $X^{i-1}p(X)$ for $i \in [d+1]$ and $p_j(X)$ for $j \in [Q_x]$. However, when the verification equation holds, the degree of $\pi^*(X)$ must be strictly less than the degree of $p(X)$, because, by Eq. (1), $v(X)$ would contain a non-zero coefficient of degree Q_x+d+1 which in particular implies that $v(X) \not\equiv 0$. Then it must be the case that the forged proof $\pi^*(s)$ is a linear combination of the simulated proofs only, thus both $\boldsymbol{f}_{\pi^*}$ and $\boldsymbol{c}_{\pi^*}$ are null. □

The representation of c^* and π^* computed by the adversary (possibly) depends on the elements $\pi_{\boldsymbol{r},j,k}$ (i.e. the proof for the linear combination $\boldsymbol{r}$ of the elements of coms with evaluation point x_j and evaluation value y_k) of proofs. However, it is much more convenient to give a representation that depends on the polynomials $p_j(X)$. This motivates the definition of our next hybrid.

Hybrid $\mathbf{H}_5$. The hybrid $\mathbf{H}_5$ finds coefficients $\boldsymbol{o}''_\tau$, for $\tau \in \{c^*, \pi^*\}$ such that:

$$\langle \boldsymbol{o}_\tau, \mathsf{proofs} \rangle = \langle \boldsymbol{o}''_\tau, ([p_j(s)]_1)_j \rangle. \tag{2}$$

Moreover, if $\boldsymbol{o}_{c^*} \neq \boldsymbol{0}$ but $\boldsymbol{o}''_{c^*} = \boldsymbol{0}$ the adversary loses the game.

Lemma 7. $\epsilon_4 \leq \epsilon_5 + \epsilon_{\mathit{Aff\text{-}MDH}}$

Proof. We begin by showing that the hybrid can compute such alternative representations efficiently. We proceed in steps. Let us parse the simulated proofs $\mathsf{proofs} := (\pi_{j,\ell})_{j,\ell}$ such that $\pi_{j,\ell}$ is the ℓ-th simulated proof obtained by $\mathcal{S}_1$ on a query involving the j-th point x_j, i.e., $((x_j, \hat{\mathsf{c}}_{j,\ell}, y_{j,\ell}), \mathsf{aux}_{j,\ell})$. Also, let $\boldsymbol{c}_{j,\ell}$ be the algebraic representation for the group element $\hat{\mathsf{c}}_{j,\ell}$ in $\mathsf{aux}_{j,\ell}$. For every $j \in [Q_x]$, we define $\boldsymbol{R}_j$ as the $Q_c \times Q_c$ matrix whose ℓ-th column is $\boldsymbol{c}_{j,\ell}$. By construction of $\mathcal{S}_1$ in this hybrid we have that for every $j \in [Q_x]$ it holds

$$\pi_{j,\ell} := \left[(\boldsymbol{c}_{j,\ell}^\top \cdot \boldsymbol{\gamma} - y_{j,\ell})\, p_j(s)\right]_1$$

and thus $\boldsymbol{\pi}_j := \left[(\boldsymbol{R}_j^\top \boldsymbol{\gamma} - \boldsymbol{y}_j) p_j(s)\right]_1$ with $\boldsymbol{y}_j := (y_{j,\ell})_\ell$. Without loss of generality, we assume that for each x_j the adversary makes the maximum number of simulation queries (i.e., $\ell \in [Q_c]$); therefore $\boldsymbol{R}_j$ is a full rank matrix (this follows from the fact that the simulation queries of the adversary satisfy the policy Φ_{sim}, and in particular the algebraic consistency of the policy, see Item 3). Given any vector $\boldsymbol{o}_\tau$ with $\tau \in \{\mathsf{c}^*, \pi^*\}$, its m-th entry $o_{\tau,m}$ corresponds to the m-th simulated proof in proofs. Therefore, similarly to above, we denote by $o_{\tau,j,\ell}$ the entry corresponding to proof $\pi_{j,\ell}$ and we define $\boldsymbol{o}_{\tau,j} := (o_{\tau,j,\ell})_\ell$. Then, for every $j \in [Q_x]$ we define $\boldsymbol{o}'_{\tau,j} \leftarrow \boldsymbol{R}_j \cdot \boldsymbol{o}_{\tau,j}$ and $\boldsymbol{\pi}'_j \leftarrow (\boldsymbol{R}_j^\top)^{-1} \cdot \boldsymbol{\pi}_j$, from which we derive:

$$\forall \tau \quad \sum_j \langle \boldsymbol{o}'_{\tau,j}, \boldsymbol{\pi}'_j \rangle = \sum_j \langle \boldsymbol{R}_j \cdot \boldsymbol{o}_{\tau,j}, (\boldsymbol{R}_j^\top)^{-1} \cdot \boldsymbol{\pi}_j \rangle = \sum_j \langle \boldsymbol{o}_{\tau,j}, \boldsymbol{\pi}_j \rangle$$

which is equal to $\langle \boldsymbol{o}_\tau, \mathsf{proofs}\rangle$, up to a permutation of the indices j.

For all $j \in [Q_x]$ let $\boldsymbol{z}_j \coloneqq (\boldsymbol{R}_j^\top)^{-1} \cdot \boldsymbol{y}_j$, and note that $\boldsymbol{\pi}'_j = [(\boldsymbol{\gamma} - \boldsymbol{z}_j)p_j(s)]_1$ namely $\pi'_{j,i}$ is a valid proof for the instance $(\mathsf{c}_i, x_j, z_{j,i})$ w.r.t. the simulated SRS.

$\mathbf{H}_5$ computes $o''_{\tau,j} \leftarrow \langle \boldsymbol{o}'_{\tau,j}, (\boldsymbol{\gamma} - \boldsymbol{z}_j)\rangle$, and $\boldsymbol{o}''_\tau \leftarrow (o''_{\tau,j})_{j\in[Q_x]}$. By construction:

$$\sum_{j\in[Q_x]} \langle \boldsymbol{o}'_{\tau,j}, \boldsymbol{\pi}'_j\rangle = \sum_{j\in[Q_x]} o''_{\tau,j} \cdot [p_j(s)]_1 .$$

which proves the first part of the lemma, i.e., computing $\boldsymbol{o}''_{\tau,j}$ satisfying Eq. (2).

In what follows, we prove that if the event that $\mathbf{H}_5$ outputs 0 but $\mathbf{H}_4$ would output 1, namely that all the conditions of $\mathbf{H}_4$ hold but $\boldsymbol{o}_{\mathsf{c}^*} \neq \mathbf{0} \wedge \boldsymbol{o}''_{\mathsf{c}^*} = \mathbf{0}$, then we can break the Aff-MDH assumption. First, notice that for any j $\boldsymbol{o}_{\mathsf{c}^*,j} \neq \mathbf{0}$ implies that $\boldsymbol{o}'_{\mathsf{c}^*,j} \neq \mathbf{0}$, because the linear transformation applied to compute $\boldsymbol{o}'_{\mathsf{c}^*,j}$ is full rank. Second, take an index j^* such that $\boldsymbol{o}_{\mathsf{c}^*,j^*} \neq \mathbf{0}$ and set $\boldsymbol{A} \leftarrow \boldsymbol{o}'_{\mathsf{c}^*,j^*}$ and $\zeta \leftarrow \langle \boldsymbol{z}_{j^*}, \boldsymbol{o}'_{\mathsf{c}^*,j^*}\rangle$. By the above definition of the values o''_{c^*,j^*} and our assumption that the "bad event" of this hybrid is $\boldsymbol{o}''_{\mathsf{c}^*} = \mathbf{0}$, we have that:

$$\langle \boldsymbol{A}, [\boldsymbol{\gamma}]_1\rangle = [\underbrace{\langle \boldsymbol{o}'_{\mathsf{c}^*,j^*}, (\boldsymbol{\gamma} - \boldsymbol{z}_{j^*})\rangle}_{o''_{\mathsf{c}^*,j^*}=0}]_1 + [\underbrace{\langle \boldsymbol{o}'_{\mathsf{c}^*,j^*}, \boldsymbol{z}_{j^*}\rangle}_{\zeta}]_1 = [\zeta]_1 .$$

The reduction $\mathcal{B}$ to the $\mathcal{D}$-Aff-MDH Assumption takes as input a distribution $[\boldsymbol{\gamma}]_1$ and runs the experiment as in $\mathbf{H}_4$ (it perfectly emulates $\mathbf{H}_4$, and in particular the simulation oracle, because it knows the trapdoor s "at the exponent"). Then $\mathcal{B}$ computes the coefficients $(A_i)_{i\in[Q_c]}$ and the value ζ as described above, which is a valid $\mathcal{D}$-Aff-MDH solution. □

Hybrid $\mathbf{H}_6$. The hybrid $\mathbf{H}_6$ additionally checks that $\boldsymbol{r}_{c^*} \neq \mathbf{0}$, and if the condition holds the adversary $\mathcal{A}_{\mathsf{c}}$ loses the game.

Lemma 8. $\epsilon_5 \leq \epsilon_6 + \epsilon_{\mathit{Aff\text{-}MDH}} + 2\epsilon_{(Q_x+1+d)\text{-}\mathsf{DL}} + \mathsf{poly}(\lambda)\frac{\deg(h)}{q}$

Proof. We bound the probability that the adversary loses in $\mathbf{H}_6$ but not in $\mathbf{H}_5$, namely, the probability that $\boldsymbol{r}^*_c \neq \mathbf{0}$ but the conditions of $\mathbf{H}_5$ hold. We show a reduction $\mathcal{B}$ to the Aff-MDH when this event happens. First of all, we can assume that the core adversary outputs coefficients $\boldsymbol{f}_{c^*} = \boldsymbol{f}_{\pi^*} = \boldsymbol{c}_{\pi^*} = \mathbf{0}$, i.e. the adversary only makes use of previous commitments $\mathsf{c}_i \in \mathsf{coms}$ and simulated proofs $\pi_{\boldsymbol{r},j,k} \in \mathsf{proofs}$ to represent c^*, and only uses the simulated proofs to represent the proof π^*. The reduction $\mathcal{B}$ takes as input a distribution $[\boldsymbol{\gamma}]_1$ and runs the experiment as in $\mathbf{H}_5$. $\mathcal{B}$ aborts if the forgery $(\mathsf{c}^*, x^*, y^*, \pi^*)$ returned by the adversary is not valid (i.e. either the extraction predicate or the verification equation is not satisfied) or $\boldsymbol{r}_{c^*} = \mathbf{0}$. Otherwise, we have that:

$$e(\mathsf{c}^* - [p(s)y^*]_1, [1]_2) = e(\pi^*, [s - x^*]_2) \text{ and } \boldsymbol{r}_{c^*} \neq \mathbf{0}$$

where $\boldsymbol{r}_{c^*} \neq \mathbf{0}$ if $\boldsymbol{o}_{c^*} \neq \mathbf{0} \vee \boldsymbol{c}_{c^*} \neq \mathbf{0}$. We can then rewrite the commitment and the proof of forgery of the core adversary as a function of the coefficients $\boldsymbol{o}''_{c^*}$

and $\boldsymbol{o}''_{\pi^*}$ (as computed in the $\mathbf{H}_5$):

$$\mathsf{c}^* \coloneqq \sum_{i\in[Q_c]} c_{c^*,i}\,[\gamma_i p(s)]_1 + \sum_{j\in[Q_x]} o''_{c^*,j}\,[p_j(s)]_1\,, \qquad \pi^* \coloneqq \sum_{j\in[Q_x]} o''_{w^*,j}\,[p_j(s)]_1$$

Since the verification equation is satisfied, and plugging in the AGM representations we have:

$$\sum_{i\in[Q_c]} c_{c^*,i}\gamma_i p(s) + \sum_{j\in[Q_x]} o''_{c^*,j} p_j(s) - p(s)y^* = \sum_{j\in[Q_x]} o''_{\pi^*,j} p_j(s)(s-x^*) \tag{3}$$

For all $j \in [Q_x]$, we define $\delta_j \coloneqq x_j - x^*$. We can rewrite the r.h.s. of Eq. (3) as:

$$\sum_{j\in[Q_x]} o''_{\pi^*,j} p_j(s)(s-x^*) = \sum_{j\in[Q_x]} o''_{\pi^*,j}(p(s) + p_j(s)\delta_j).$$

In Eq. (3), we group all the terms that depend on $p(s)$ on the left side, and we move all the terms that depend on $p_j(s)$ to the right side, thus obtaining:

$$\underbrace{\Big(\sum_{i\in[Q_c]} c_{c^*,i}\gamma_i - \sum_{j\in[Q_x]} o''_{w^*,j} - y^*\Big)}_{A} p(s) = \sum_{j\in[Q_x]} \underbrace{\Big(o''_{w^*,j}\delta_j - o''_{c^*,j}\Big)}_{B_j} p_j(s) \tag{4}$$

Let $f(X) \coloneqq Ap(X) - \sum_{j\in[Q_x]} B_j p_j(X)$. Notice that because of Eq. (4) we have $f(s) = 0$, thus we can assume $f(X) \equiv 0$, as otherwise we can reduce, by Lemma 1, to the $(Q_x + d + 1)$-DL assumption. It must be the case that both $\sum_{j\in[Q_x]} B_j p_j(s) = 0$ and $A = 0$ because the degree of $p(X)$ and of $p_j(X)$ for any j are different. Moreover, the polynomials $p_j(X)$ are linearly independent, namely the only linear combination $\sum_j a_j p_j(X) = 0$ is the trivial one where the coefficients $a_j = 0$[14], thus $B_j = 0$ for all j. We have that $o''_{w^*,j}\delta_j - o''_{c^*,j} = 0, \forall j$. Thus we can rewrite the coefficients $o''_{\pi^*,j} = \frac{o''_{c^*,j}}{\delta_j}$, $\forall j$. Since A must be 0:

$$\sum_{i\in[Q_c]} c_{c^*,i}\gamma_i - \sum_{j\in[Q_x]} \frac{o''_{c^*,j}}{\delta_j} - y^* = 0. \tag{5}$$

$\mathcal{B}$ can plug the definition of the coefficients $o''_{c^*,j}$ in Eq. (5) and derive:

$$\begin{aligned} 0 &= \sum_{i\in[Q_c]} c_{c^*,i}\gamma_i - \sum_{i,j} \frac{o'_{c^*,i,j}(\gamma_i - z_{j_i})}{\delta_j} - y^* \\ &= \sum_{i\in[Q_c]} (c_{c^*,i} - \sum_{j} \tfrac{o'_{c^*,i,j}}{\delta_j})\gamma_i + \sum_{i,j} \tfrac{o'_{c^*,i,j} z_{j_i}}{\delta_j} - y^*. \end{aligned}$$

[14] To see this, $\forall x_j \in \mathcal{Q}_x$ we have that $\sum_{j'} a_{j'} p_{j'}(x_j) = a_j p_j(x_j)$ since $p_j(x_j) \neq 0$ and $p_{j'}(x_j) = 0$ for $j \neq j'$, and $a_j p_j(x_j) = 0$ iff $a_j = 0$

Above, in the last step we have grouped the terms depending on γ_i. In particular, the last equation shows that $\mathcal{B}$ can make a forgery in the Aff-MDH game since it knows $z \coloneqq y^* - \sum_{i,j} \frac{o'_{\mathsf{c}^*,i,j} z_{j_i}}{\delta_j}$ and coefficients $A_i \coloneqq c_{\mathsf{c}^*,i} - \sum_j \frac{o'_{\mathsf{c}^*,i,j}}{\delta_j}$ such that: $\sum_{i \in [Q_c]} A_i \left[\gamma_i\right]_1 = [z]_1$. For this to be a valid solution in the Aff-MDH game, we need the existence of at least an index i such that $A_i \neq 0$. We show that this occurs with all but negligible probability, i.e., $\Pr[\exists i \in [Q_c] : A_i \neq 0] \geq 1 - \mathsf{negl}(\lambda)$.

To this end, consider an arbitrary $\mu \in [Q_c]$, then we have $\Pr[\forall i \in [Q_c] : A_i = 0] \leq \Pr[A_\mu = 0]$. Thus, for any μ, we have:

$$\Pr[\exists i \in [Q_c] : A_i \neq 0] = 1 - \Pr[\forall i \in [Q_c] : A_i = 0] \geq 1 - \Pr[A_\mu = 0].$$

Below, we argue that $\Pr[A_\mu = 0]$ is negligible based on the randomness of x^* which is chosen by the random oracle after defining A_μ, and we make use of the assumption that $\boldsymbol{r}_{\mathsf{c}^*} \neq 0$. We claim that the value $A_\mu = c_{\mathsf{c}^*,\mu} - \sum_j \frac{o'_{\mathsf{c}^*,j,\mu}}{(x^* - x_j)}$ can be fixed before the random oracle query $\mathbb{x}^*$ is made. To this end, we start by showing that $\boldsymbol{o}'_{\mathsf{c}^*,j}$ does not depend on x^*. Let $B(j) \subseteq [Q_c]$ be the subset of indices of the simulation queries that involve x_j and that occurred before the random oracle query that returned x^*. We observe that for every $\eta \in B(j)$ it must be $o_{\mathsf{c}^*,j,\eta} = 0$ since the simulated proof $\pi_{j,\eta}$ is not in the view of the adversary. Therefore:

$$o'_{\mathsf{c}^*,j,i} = \sum_{\eta \in [Q_c]} \boldsymbol{R}_{j,\eta,i} \cdot o_{\mathsf{c}^*,j,\eta} = \sum_{\eta \in B(j)} \boldsymbol{R}_{j,\eta,i} \cdot o_{\mathsf{c}^*,j,\eta}$$

and observe that all the rows of $\boldsymbol{R}_j$ belonging to $B(j)$ can all be defined before x^* is sampled. Hence, we have that A_μ depends on the values $\boldsymbol{c}_{\mathsf{c}^*}, x^*, \{x_j\}_j$, and $\boldsymbol{o}_{\mathsf{c}^*,j}$ which can all be defined before the random oracle query $\mathbb{x}^*$ is made.

Now, we bound $\Pr[A_\mu = 0]$. Recall that, since the extractor policy Φ_{ext} holds true, we have that $x^* = h(a)$ and $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$ where $g_c(\mathsf{c}^*, s) = 1$ and the function h is the polynomial encoded in aux_ϕ: the adversary may want to encode up to $n \in \mathsf{poly}(\lambda)$ different polynomials h_i into aux_ϕ to maximize its advantage, and the extractor policy does not impose any restriction on this. Moreover, by the AGM, since $\mathcal{A}_{\mathsf{c}}$ sends a query s (where c^* is encoded in s) to the random oracle it also defines coefficients for c^* before the value a, and therefore $x^* = h(a)$, is defined. Also, it is not hard to see that the representation vector of c^* defined by $\mathcal{A}_{\mathsf{c}}$ when querying the random oracle must be the same representation vector used for the forgery. As otherwise we would break the $(Q_x{+}d{+}1)$-DL assumption. Thus the coefficients $\boldsymbol{c}_{\mathsf{c}^*}$ and $\boldsymbol{o}'_{\mathsf{c}^*,j}$ are defined by the adversary before seeing the random value x^*. Notice that, once the coefficients $\boldsymbol{c}_{\mathsf{c}^*}$ and $\boldsymbol{o}'_{\mathsf{c}^*,j}$ are fixed, the coefficient A_μ can be seen as function of $x^* \in \mathbb{Z}_q$, i.e. $A_\mu = A_\mu(x^*)$, where:

$$A_\mu(X) = c_{\mathsf{c}^*,\mu} + \sum_j \frac{o'_{\mathsf{c}^*,j,\mu}}{X - x_j} = \frac{c_{\mathsf{c}^*,\mu} \prod_j (X - x_j) + \sum_j (o'_{\mathsf{c}^*,j,\mu} \prod_{j' \neq j} (x_{j'} - X))}{\prod_j X - x_j}.$$

Notice that $A_\mu(X)(\prod_j (X - x_j))$ vanishes in at most Q_x points in $\mathbb{F} \setminus \mathcal{Q}_x$ and vanishes in the set of points $\mathcal{Q}_x$. Let $\mathcal{R}$ be the set of the roots of such a polynomial,

since $\forall i \in [n]$, h_i is defined before x^* is computed, and by union bound:

$$\Pr[\exists i : h_i(\mathsf{RO}(s)) \in \mathcal{R}] \leq \sum_{r \in \mathcal{R}} \Pr[\exists i : h_i(\mathsf{RO}(s)) = r] \leq nQ_x \frac{\max_i \mathsf{deg}(h_i)}{q}$$

for each string s that encodes c^*, To conclude, we notice that $\mathcal{A}$ can submit at most Q_{RO} queries to the RO with strings encoding c^*, say $s_1, \ldots s_{Q_{\mathsf{RO}}}$. Thus the probability that there exist $i \in [n], j \in [Q_{\mathsf{RO}}]$ such that $h_i(\mathsf{RO}(s_j)) \in \mathcal{R}$ is bounded by $nQ_{\mathsf{RO}}Q_x \frac{\max_i \mathsf{deg}(h_i)}{q}$. □

Hybrid $\mathbf{H}_7$. The hybrid $\mathbf{H}_7$ additionally checks that $y^* \neq 0$, and if the condition holds the adversary $\mathcal{A}_{\mathsf{c}}$ loses the game. For space reasons, we give in [24] the proof of the following lemma.

Lemma 9. $\epsilon_6 \leq \epsilon_7 + \epsilon_{(Q_x+1+d)\text{-DL}} + \mathsf{poly}(\lambda)\frac{\mathsf{deg}(h)}{q}$

Finally, we have that the probability that the adversary wins in $\mathbf{H}_7$ is null, namely $\epsilon_7 = 0$. Indeed, the canonical extractor $\mathcal{E}$ outputs the 0 polynomial, moreover because of the condition introduced in $\mathbf{H}_6$, we have $\mathsf{c}^* = [0]_1$, and because of the condition introduced in $\mathbf{H}_7$ we have $y^* = 0$, thus the witness extracted is valid for the instance $\mathbb{x}^* = (\mathsf{c}^* = [0]_1, x^*, y^* = 0)$. □

5 Simulation-Extractable Universal zkSNARKs

We provide a technical overview of our compiler for universal SNARKs based on polynomial IOPs. Rather than delving into extensive formal definitions and analysis, we aim to present this section in a more informal (and also more compact) manner and refer the reader to [24] for all the details.

We define an indexed relation $\mathcal{R}$ verifying tuple $(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \mathbb{w})$. We say that $\mathbb{w}$ is a witness to the instance $\mathbb{x}$ being in the relation defined by the pp and index $\mathbb{i}$ when $(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$. Briefly, we say that a NIZK NIZK is *universal* if there exists a deterministic algorithm Derive that takes as input a (universal) srs and an index $\mathbb{i}$, and outputs a specialized verification key for such an index. We say NIZK is a SNARK if the verification keys and the proofs are succinct. We say that a SNARK is *universal* if there exists a deterministic algorithm Derive that takes as input a (universal) srs and an index $\mathbb{i}$, and outputs a specialized and succinct verification key for such an index.

Polynomial Interactive Oracle Proofs. A Polynomial (Holographic) IOP [17] consists of an r-rounds interaction between a prover P, sending oracle polynomials p_i (and additional messages π_i), and a verifier V, who sends uniformly random messages ρ_i; finally, V outputs a set of polynomial identities to be checked on the prover's polynomials of the form $(G^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})$, that is satisfied if and only if $F^{(k)}(X) \equiv 0$ where:

$$F^{(k)}(X) := G^{(k)}(X, \{p_i(v_i^{(k)}(X))\}_i, \{\pi_i\}_i). \tag{6}$$

In our work, we use PIOPs with two slight refinements.[15] The first one, called *(non-adaptive) algebraic verifiers*, says that the above polynomials $v_j^{(k)}$ do not depend on the instance and can be expressed as polynomial functions of $\mathcal{V}$'s random coins, i.e., $v_j^{(k)}(X) = \tilde{v}_j^{(k)}(X, \boldsymbol{\rho})$ for some instance-independent $\tilde{v}_j^{(k)}$. The second one is a more restrictive[16] concept of soundness called *state-restoration straight-line knowledge soundness*. This notion combines the notion of state-restoration soundness from [9] with the concept of straight-line extractability from [17]. For further clarification, the malicious prover engages in a game with the honest verifier and has the additional ability to *roll back* the interaction with the verifier to a previous state. At some point, the interaction may reach a final state. The prover is considered successful if it produces an accepting transcript, while the extractor, given such a transcript that includes all the oracle polynomials, fails to produce a valid witness. Similarly to previous work, we use the notion of bounded zero-knowledge of [20,17].

Compilation-safe PIOP. We must incorporate an additional element into the classical recipe. As stated in the introduction, mix-and-match attacks on compiled protocols, involving two or more independent sub-protocols, are unavoidable. Therefore, we identify a structural restriction on the PIOP that prevents such problematic scenarios. The restriction is easy to state and easy to meet:

Definition 15 (Compiler-safe PIOP). *A PIOP* PIOP *is* compiler-safe *if for any* $\mathtt{i}, \mathtt{x}$ *and* $\boldsymbol{\rho} \coloneqq \rho_1, \ldots, \rho_{r-1}$ *and any tuple* $(G^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})_{j \in [n_e]} \leftarrow \mathcal{V}(\mathbb{F}, \mathtt{i}, \mathtt{x}, \boldsymbol{\rho})$ *there exists an index* k *such that for all* j *the polynomials* $v_j^{(k)}$ *are of degree at least one.*

The Compilation-Ready CP-SNARK. Instead of compiling directly a PIOP through a polynomial commitment in its simplest form (i.e., an evaluation proof for each polynomial queried in the PIOP), we take an alternative road similar to [17]. Namely, we assume the existence of a CP-SNARK that, w.r.t. a tuple of commitments $(\mathsf{c}_j)_{j \in [n]}$, is capable of proving either knowledge of polynomials $(p_j)_{j \in [n]}$ opening these commitments or that the committed polynomials satisfy a statement like the one in Eq. (6) (i.e., that the oracles committed in $(\mathsf{c}_j)_{j \in [n]}$ would make the PIOP verifier accept)[17]. We call this building block a *compilation-ready* CP-SNARK (CP, shortly), and informally we refer to the former type of statements as "proof of knowledge" and to the latter as "PIOP verifier". While our compilation strategy follows previous work, our novel contribution is to properly define the properties that this CP-SNARK must satisfy in order to argue that the result of the compiler is simulation-extractable, and not only knowledge-sound. These properties are mainly three. The first one is

[15] All the PIOPs that we are aware of satisfy both these properties.

[16] The (classical) notion of knowledge extractability implies state-restoration soundness through complexity leveraging [9].

[17] The reason to assume a single CP-SNARK for both kinds of statements has to do with the security guarantees when we compose protocols in the AGM [1].

that the CP prover can "append" arbitrary messages to the proven instances. Looking ahead to our compiler, this feature is used so that the prover and the verifier can append the (hash of the) protocol's transcript to the proven instance, in such a way that a CP proof acts as a *signature of knowledge for the transcript* [19]. Note that this hashing of the transcript already happens in the standard PIOP compiler due to the application of the Fiat-Shamir transform; here, we highlight it explicitly as it plays an important role in the proof of simulation extractability. The second property, referred to as the *commitment simulator for PIOP*, intuitively requires the existence of a strategy to simulate commitments such that: adding them to the view preserves zero-knowledge, and the simulation respects the "commitment check" constraint in Item 2 of Definition 13. This is a very mild property that is trivially satisfied when employing hiding commitments, and is met by existing simulation strategies based on deterministic commitments to randomized polynomials [28,17]. The third property of CP is that it must be *simulation-extractable* w.r.t. a policy $\hat{\mathbf{\Phi}}$ such that:

- The adversary can ask simulated proofs for "PIOP verifier" statements where all the $v_j^{(k)}$ of Eq. (6) are fixed at the beginning of the experiment.
- If the forgery of the adversary is a "proof of knowledge" for commitments $\mathbf{c}^*$, then the adversary must return as auxiliary output yet another forgery for a "PIOP verifier" statement such that: (1) All the commitments $\mathbf{c}^*$ appear in the second forgery, (2) the second forgery is valid according to the extractor policy described next.
- If the forgery of the adversary is for the "PIOP verifier" statement, then the statement-proof pair returned by the adversary must not be in the list of simulated statements-proofs, and (similarly to Definition 15) there exists a k such that for all j the polynomial $v_j^{(k)}$ has degree at least 1.

Theorem 2 (Informal). *Let* PIOP *be a PIOP for an indexed relation* $\mathcal{R}$ *that is state-restoration straight-line extractable, bounded zero-knowledge, and compiler-safe (cf. Definition 15). Let* CP *be a compilation-ready CP-SNARK for* PIOP*. There exists a compiler that produces a simulation-extractable Universal zkSNARK for* $\mathcal{R}$*.*

We follow the classical compilation strategy where: for each of the r rounds, the zkSNARK prover sends commitments of the PIOP oracle polynomials (along with a proof of knowledge) and then computes the PIOP verifier's challenges using Fiat-Shamir; in the last round, the prover sends a CP proof that the PIOP verifier accepts, i.e., Eq. (6) holds w.r.t. all the commitments sent earlier. Notably, this CP proof is produced using the statement and the hash of the transcript as "message" for the signature of knowledge.

We briefly discuss how the properties of PIOP and CP play a role in the security of the compiled zkSNARK Π. We recall that in the simulation-extractability experiment, we have an adversary $\mathcal{A}$ who makes simulation queries for statements of its choice and eventually comes up with a forgery, which is a statement-proof that is new and valid. The goal is to show that for such an adversary there is an

extractor that outputs a valid witness with overwhelming probability. Roughly speaking, we build this extractor by first extracting the committed oracle polynomials from the CP "proof of knowledge" in the random oracle query of $\mathcal{A}$ in each round,[18] and then by running the PIOP extractor to obtain the witness.

For this extraction strategy to work, we need two conditions: (A) The "proof of knowledge" extraction must be valid. (B) The zkSNARK extractor feeds the PIOP extractor with polynomials that pass the PIOP verification equations. A technicality about relying on CP extraction for (A) and (B) is that we actually have to make a reduction to its *policy-based simulation-extractability*. In particular, this means that we have to turn $\mathcal{A}$ into CP adversaries that comply with the policy $\hat{\mathbf{\Phi}}$. To obtain (A), we use the second property of $\hat{\mathbf{\Phi}}$ mentioned above, which ensures a valid extraction if the adversary later provides a valid proof of polynomial evaluation. This is however the case for us since a successful adversary must provide such proof. For (B), we rely on the following observations. If $\mathcal{A}$ produces a forgery for a new statement of Π then the CP proof (aka signature of knowledge) must use a new message, and thus we can build a CP adversary returning a new statement-proof pair. If $\mathcal{A}$ produces a forgery for a statement queried to the simulation oracle, then by strong simulation extractability the proof must be new, which means that: either the commitments in the transcript are different, or the commitments are all the same but the "PIOP verifier" proof is different. In the former case, we get a different transcript, which yields a CP forgery with a new message, as in the previous case. In the latter case, the transcript is the same and we get a CP forgery with the same message but fresh proof. Notably, in all the cases, the CP forgeries respect the degree-1 condition thanks to the compiler-safe property of the PIOP. Finally, the reduction CP adversaries that we build satisfy the first property of $\hat{\mathbf{\Phi}}$ thanks to the algebraic verifier property of PIOP, which allows us to precompute the instance-independent polynomials $\tilde{v}_j^{(k)}$, and to the programmability of the random oracle that allows us to presample the verifier's challenges $\boldsymbol{\rho}$, define $v_j^{(k)}(X) = \tilde{v}_j^{(k)}(X, \boldsymbol{\rho})$, and later program the random oracle to use these coins $\boldsymbol{\rho}$.

Compilation-ready CP-SNARK from KZG. To connect together Section 4 and the results of this section, we show a simple compilation-ready CP-SNARK in the ROM based on batched KZG evaluation proofs. For a "PIOP verifier" statement, the prover RO-hashes the instance and obtains a random point ξ, evaluates the polynomials $v_j^{(k)}(\xi)$ for any j and outputs the evaluations $p_j(v_j^{(k)}(\xi))$ together with a batch evaluation proof for all of them. For a "proof of knowledge" statement, the prover does not output an explicit proof element (we call this a *vacuous proof*), and we rely on the AGM to argue its extractability. The idea is that, for an algebraic adversary that produces an alleged commitment c and its algebraic representation, we can find a way to *open* c, under some circumstances. For example, consider the adversary that, during the simulation-extractability experiment, hashes (i.e., makes a random oracle query) the commitment c, and

[18] Note, this avoids rewinding, since extraction is performed in the same moment when the adversary sends the proof of knowledge through a RO call.

later includes c in a "PIOP verifier" instance. Then the algebraic representation of c returned at hashing time must coincide with the same polynomial extracted at forgery time, otherwise one can break the standard binding of the commitment. Crucially, this scenario fits exactly the second part of the policy $\hat{\mathbf{\Phi}}$. As for the third part of the policy, we notice that an attack similar to the mix-and-match malleability attack mentioned in the introduction applies to our compilation-ready CP-SNARK. For example, the adversary could ask a simulation for an instance that tests two (fake) commitments on constant values defined by the $v_j^{(k)}$, and then it can produce a forgery that includes one of the commitments by copying part of the simulated proof. Intuitively, this is why we require that the $v_j^{(k)}$ have a degree at least 1: when evaluated on a fresh random point ξ, a valid proof for $p_j(v_j^{(k)}(\xi))$ ensures that the prover knows p_j.

Acknowledgements This work received funding from MESRI-BMBF French-German joint project named PROPOLIS (ANR-20-CYAL-0004-01), the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under project PICOCRYPT (grant agreement No. 101001283), and from the Spanish Government under projects PRODIGY (TED2021-132464B-I00) and ESPADA (PID2022-142290OB-I00). The last two projects are co-funded by European Union EIE, and NextGenerationEU/PRTR funds.

References

1. M. Abdalla, M. Barbosa, J. Katz, J. Loss, and J. Xu. Algebraic adversaries in the universal composability framework. In M. Tibouchi and H. Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 311–341. Springer, Heidelberg, Dec. 2021.
2. B. Abdolmaleki, S. Ramacher, and D. Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 2020*, pages 1987–2005. ACM Press, Nov. 2020.
3. S. Arora and S. Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd FOCS*, pages 2–13. IEEE Computer Society Press, Oct. 1992.
4. K. Baghery, M. Kohlweiss, J. Siim, and M. Volkhov. Another look at extraction and randomization of groth's zk-snark. In N. Borisov and C. Díaz, editors, *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 457–475. Springer, 2021.
5. M. Bellare, D. Hofheinz, and E. Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? *Journal of Cryptology*, 28(1):29–48, Jan. 2015.
6. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable zero knowledge with no trusted setup. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732. Springer, Heidelberg, Aug. 2019.

7. E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, Aug. 2013.
8. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.
9. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, Oct. / Nov. 2016.
10. E. Ben-Sasson, L. Goldberg, S. Kopparty, and S. Saraf. DEEP-FRI: Sampling outside the box improves soundness. In T. Vidick, editor, *ITCS 2020*, volume 151, pages 5:1–5:32. LIPIcs, Jan. 2020.
11. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Heidelberg, May 2004.
12. D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081, 2020. https://eprint.iacr.org/2020/081.
13. D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Halo infinite: Proof-carrying data from additive polynomial commitments. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 649–680, Virtual Event, Aug. 2021. Springer, Heidelberg.
14. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.
15. B. Bünz, A. Chiesa, P. Mishra, and N. Spooner. Recursive proof composition from accumulation schemes. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 1–18. Springer, Heidelberg, Nov. 2020.
16. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020.
17. M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In M. Tibouchi and H. Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Heidelberg, Dec. 2021.
18. M. Campanelli, D. Fiore, and A. Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, Nov. 2019.
19. M. Chase and A. Lysyanskaya. On signatures of knowledge. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, Aug. 2006.
20. A. Chiesa, Y. Hu, M. Maller, P. Mishra, P. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.
21. Q. Dao and P. Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In C. Hazay and M. Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 531–562. Springer, Heidelberg, Apr. 2023.

22. Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631. Springer, Heidelberg, Dec. 2010.
23. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991.
24. A. Faonio, D. Fiore, M. Kohlweiss, L. Russo, and M. Zajac. From polynomial iop and commitments to non-malleable zksnarks. Cryptology ePrint Archive, Paper 2023/569, 2023. https://eprint.iacr.org/2023/569.
25. S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In S. D. Galbraith and M. Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Heidelberg, Dec. 2012.
26. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, Aug. 2005.
27. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, Aug. 2018.
28. A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.
29. C. Ganesh, H. Khoshakhlagh, M. Kohlweiss, A. Nitulescu, and M. Zajac. What makes fiat-shamir zksnarks (updatable SRS) simulation extractable? In C. Galdi and S. Jarecki, editors, *Security and Cryptography for Networks, SCN 2022*, volume 13409 of *Lecture Notes in Computer Science*, pages 735–760. Springer, 2022.
30. C. Ganesh, Y. Kondi, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Witness-succinct universally-composable snarks. Cryptology ePrint Archive, Paper 2022/1618, 2022. https://eprint.iacr.org/2022/1618.
31. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In O. Dunkelman and S. Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 397–426. Springer, Heidelberg, May / June 2022.
32. S. Garg, A. Jain, and A. Sahai. Leakage-resilient zero knowledge. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 297–315. Springer, Heidelberg, Aug. 2011.
33. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.
34. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.
35. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, Dec. 2010.
36. J. Groth. On the size of pairing-based non-interactive arguments. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.
37. J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In H. Shacham

and A. Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, Aug. 2018.
38. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, Aug. 2017.
39. C. S. Jutla and A. Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, Dec. 2013.
40. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, Dec. 2010.
41. J. Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In K. Nissim and B. Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 1–34. Springer, Heidelberg, Nov. 2021.
42. H. Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, Mar. 2012.
43. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, Nov. 2019.
44. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updateable structured reference strings. Cryptology ePrint Archive, Report 2019/099, 2019. https://eprint.iacr.org/2019/099.
45. S. Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, Nov. 1994.
46. C. Ràfols and A. Zapico. An algebraic framework for universal and updatable SNARKs. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, Aug. 2021. Springer, Heidelberg.
47. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, Oct. 1999.
48. A. Szepieniec. Polynomial IOPs for linear algebra relations. Cryptology ePrint Archive, Report 2020/1022, 2020. https://eprint.iacr.org/2020/1022.
49. A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In C. Galdi and V. Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 45–64. Springer, Heidelberg, Sept. 2020.
50. A. Zapico, V. Buterin, D. Khovratovich, M. Maller, A. Nitulescu, and M. Simkin. Caulk: Lookup arguments in sublinear time. In H. Yin, A. Stavrou, C. Cremers, and E. Shi, editors, *ACM CCS 2022*, pages 3121–3134. ACM Press, Nov. 2022.