Real-world Universal zkSNARKs are non-malleable

Antonio Faonio, Dario Fiore, Luigi Russo

Real-world Universal zkSNARKs are non-malleable

Antonio Faonio[1], Dario Fiore[2], and Luigi Russo[1]

[1] EURECOM, Sophia Antipolis, France {faonio,russol}@eurecom.fr
[2] IMDEA Software Institute, Madrid, Spain dario.fiore@imdea.org

Abstract. Simulation extractability is a strong security notion of zkSNARKs that guarantees that an attacker who produces a valid proof must know the corresponding witness, even if the attacker had prior access to proofs generated by other users. Notably, simulation extractability implies that proofs are non-malleable and is of fundamental importance for applications of zkSNARKs in distributed systems. In this work, we study sufficient and necessary conditions for constructing simulation-extractable universal zkSNARKs via the popular design approach based on compiling polynomial interactive oracle proofs (PIOP). Our main result is the first security proof that popular universal zkSNARKs, such as PLONK and Marlin, *as deployed in the real world*, are simulation-extractable. Our result fills a gap left from previous work (Faonio et al. TCC'23, and Kohlweiss et al. TCC'23) which could only prove the simulation extractability of the "textbook" versions of these schemes and does not capture their optimized variants, with all the popular optimization tricks in place, that are eventually implemented and deployed in software libraries.

1 Introduction

Zero-knowledge proofs [GMR85] allow a prover to convince a verifier that a statement is true without revealing any information beyond that. A notable class of zero-knowledge proofs are *zkSNARKs* (zero-knowledge succinct non-interactive arguments of knowledge) [Mic94,BCCT12] in which, after an initial setup phase, the prover can generate proofs that are short and easy to verify, without the need of any interaction. The simultaneous achievement of these properties makes zkSNARKs an unbeatable tool in several applications, as they enable not only privacy-preserving computation but also scalability.

The fundamental security concept of zkSNARKs is knowledge-soundness, which essentially states that a prover generating a valid proof must possess the corresponding witness. However, this notion only considers provers in isolation. In simpler terms, knowledge-soundness fails to address attackers who have access to proofs for certain statements and may exploit this advantage to create a proof for another statement without possessing its corresponding witness. This gap in the security definition of zero-knowledge proofs was identified early on by Sahai [Sah99], who introduced the concept of simulation soundness, subsequently expanded upon as simulation extractability (SE, for brevity) [DDO+01].

For zkSNARKs, the notion of SE was first studied by Groth and Maller [GM17] who proposed a pairing-based construction and an application to succinct signatures of knowledge for NP (aka Snarky signatures). Their scheme, however, is of the "first generation" and requires a trusted setup for a single circuit. In this work, we focus on the SE of *universal zkSNARKs*, the emerging generation of zkSNARKs where the initial setup is reusable for any circuit within a given size bound, a property that makes their practical application more versatile.

The state of SE-zkSNARKs. While the last decade has seen tremendous progress in zkSNARKs, producing a multitude of schemes, only a handful of them are known to be simulation extractable. Many schemes are in an unknown situation as they lack a security proof of SE. It is indeed the typical workflow to first focus on proving knowledge soundness of new schemes (which may already entail its own challenges) and to leave SE for future work. The reason is that proving SE is often a challenging task that does not follow via a straightforward extension of the knowledge-soundness proof.

Several recent results [GOP+22,DG23,GKK+22,FFK+23,KPT23], motivated by this state of affairs, prove the SE of existing zkSNARKs, such as Bulletproofs [BBB+18], Spartan [DG23], Sonic [MBKM19],

PLONK [GWC19], Marlin [CHM+20], Lunar [CFF+21] and Basilisk [RZ21]. We can divide these results into two main categories: those such as [GOP+22,DG23,GKK+22] that prove the SE of specific zkSNARKs; those like [FFK+23,KPT23] that prove the SE of a broad class of zkSNARKs such as those built via the popular approach combining polynomial interactive oracle proofs (PIOPs) and polynomial commitments. The works in the latter category are of particular interest as they give an SE recipe that is generic and thus it can benefit both existing and future schemes.

Given this state of the art, one may therefore ask if there is more to know about the SE of universal zkSNARKs based on PIOPs. However, a closer look at the recent results reveals *two important gaps that do not allow concluding that the "real world" versions of schemes like PLONK and Marlin are simulation extractable.*

Theory vs. Implementation The first gap lies in that the versions of these schemes that offer the best performance and are eventually implemented in software libraries[3] slightly depart from the ones obtained through the PIOP-to-zkSNARK vanilla compilation. The difference is in the last step. In order to maximize efficiency, they apply an optimization (that we call *linearization trick*, also known as Maller's optimization) [GWC19,OL] that leverages the homomorphic properties of the KZG polynomial commitment to reduce the number of field elements in the proof. This optimization though changes the zkSNARK verification algorithm in a way that escapes the SE security analysis in previous work [FFK+23,KPT23].

Delegation phase The second gap is that the aforementioned frameworks of Faonio *et al.* [FFK+23] and Kohlweiss *et al.* [KPT23] capture PIOPs in which some polynomials are evaluated on a random challenge chosen in the last round. This is however not the case for Marlin and Lunar in which polynomials involving the witness are evaluated on a challenge chosen before the last round, which is witness-independent and needed only for verifier's efficiency (what we call a *delegation phase*). For this reason, the work of [FFK+23,KPT23] can only argue the SE of small variants of Marlin and Lunar.

Our Results. In this work, we resolve the two limitations above and we give the first proof of SE of the "real world" optimized versions of zkSNARKs which include PLONK, Marlin, Lunar, and Basilisk. To achieve these results, we improve the techniques of [FFK+23] in several ways: (1) we formalize the compilation recipe based on the linearization trick and we prove that, under a set of minimal constraints, PIOPs can be compiled to SE-zkSNARKs using the linearization trick optimization; (2) we refine the set of conditions to compile a PIOP to a zkSNARK, notably removing the artificial one from [FFK+23] that prevented capturing Marlin and Lunar, and thus we broaden the class of PIOPs that can be compiled in an SE manner; (3) we simplify and generalize the conditions under which KZG can be proved simulation-extractable.

As a byproduct of (1) and (3), we obtain the first security analysis of the linearization trick optimization. We show potentially insecure instantiations as well as a characterization of the conditions that make it secure even in terms of plain knowledge-soundness, in the AGM with oblivious sampling (AGMOS) [LPS23].

Some of our definitions and techniques to prove SE may appear rather convoluted. We would like to note that this is due to the wish of capturing SE for *existing* protocols, without introducing any change, which is a challenging goal. As an example, [FFK+23] gave a simple condition to safely compile a PIOP: that a witness-dependent polynomial is evaluated on a random challenge chosen in the last round. However, this condition is not met by some protocols, which in this work we eventually prove to be SE. This required us to elaborate more complex conditions to explain why this is possible.

Given the technicalities of the aforementioned gaps and the compilation strategy of [FFK+23], we provide a more comprehensive explanation of our results in the following section.

2 A Technical Overview of Our Results

2.1 Revisiting PIOP-based zkSNARKs

A common approach to design zkSNARKs is to first construct an information-theoretic protocol that achieves the desired functionality in an idealized model and then remove the idealized component by compiling it into

[3] E.g., https://github.com/dusk-network/plonk, https://github.com/arkworks-rs/marlin

a zkSNARK via the use of a computationally-secure primitive [Ish19,Ish20]. The most popular instantiation of this approach uses PIOP [GWC19,BFS20,CHM+20,Sze20,CFF+21] for the information-theoretic part, and polynomial commitments [KZG10] for the computational one.

In a PIOP, the prover uses one oracle *to commit* to polynomials while the verifier calls a second oracle *to query* the committed polynomials. In the compiled PIOP, instead, the prover commits to polynomials with a polynomial commitment, and then computes the results of verifier's queries and uses the *evaluation opening* to vouch for their correctness. Finally, to remove interaction the compiler employs the Fiat-Shamir transformation to obtain the zkSNARK. The details of the (kind of) verifier's queries often diverge in different implementations of this paradigm. Arguably, the simplest form of queries is the evaluation of polynomials, namely, queries checking that a committed polynomial p at evaluation point x evaluates to $y = p(x)$; this is the model used in [CHM+20,BFS20]. Other PIOP variants [GWC19,CFF+21] consider more general queries that state the validity of polynomial equations over (a subset) of the committed polynomials.

Our generalization: $\mathcal{R}$-PIOP. To keep all these different notions of PIOP under the same umbrella, in our work, we define the notion of $\mathcal{R}$-PIOP where the verifier's queries are instances belonging to the *oracle* relation $\mathcal{R}$. Roughly speaking, an oracle relation is an NP-relation where the instances can refer to polynomial oracles [CBBZ23]. Under this definition, Marlin uses an $\mathcal{R}_{\mathtt{evl}}$-PIOP, where $\mathcal{R}_{\mathtt{evl}}$ is the relation that checks that a polynomial oracle evaluates to y at point x, while PLONK [GWC19] and Lunar [CFF+21] are $\mathcal{R}_{\mathtt{poly}}$-PIOPs, where $\mathcal{R}_{\mathtt{poly}}$ is the relation that checks polynomial equations over polynomial oracles.

How to compile $\mathcal{R}$-PIOPs? Unfortunately, zkSNARKs obtained (mechanically) applying compilations from $\mathcal{R}_{\mathtt{evl}}$-PIOPs are often sub-optimal proof systems, due to the fact that one should include in the proof a field element for each evaluation of a polynomial oracle. In particular, such compilations cannot leverage on the homomorphic property that many polynomial commitments, such as KZG [KZG10], have. Thus, subsequent optimizations usually accompany, and slightly change, the formally analyzed zkSNARKs. Instead, zkSNARKs compiled from $\mathcal{R}_{\mathtt{poly}}$-PIOPs can defer all the optimizations to the richer and more expressive (sub)-proof systems for $\mathcal{R}_{\mathtt{poly}}$.[4] Yet, in practice, the latter proof systems are often reduced to the former via a random point evaluation.

One of the most common optimizations, based on homomorphic commitments, is the so-called *linearization trick*, sometimes referred as the Mary Maller's optmization [GWC19,OL]. This optimization allows reducing the number of field elements in the final proofs. For example, to prove that $A(x)B(x) + C(x) = y$ holds for committed polynomials A,B,C, and values x and y, one can prove that $B(x) = y_b$ for some y_b, the verifier uses the homomorphism of the polynomial commitment to obtain the commitment to the *linearization* polynomial $L(X) \coloneqq A(X)y_b + C(X)$, and then the prover proves that $L(x) = y$, saving from naively evaluating all the polynomials on x.

Building on this idea, PLONK [GWC19] describes a general recipe to compile $\mathcal{R}_{\mathtt{poly}}$-PIOP to zkSNARK. The procedure first finds the minimal sub-set of polynomials that one should evaluate in order to generate the linearization polynomial, and then it (batch) evaluates all the polynomials in this subset and the linearization polynomial on a fresh random point.[5]

On the (in)security of the linearization trick. It turns out that this general recipe is not always sound. In fact, the work of [LPS23] shows a counter-example to the extractability of the linearization trick when using the KZG polynomial commitment. In particular, assume the adversary does not know the representation of a group element c (using the lingo of [LPS23], c is an obliviously sampled element), and sets the three polynomial commitments of the example above as $(\mathsf{c}_A, \mathsf{c}_B, \mathsf{c}_C) = (\mathsf{c}, [b]_1, -b \cdot \mathsf{c})$. According to [LPS23], only the commitment c_B is extractable in the algebraic group model, namely the adversary can give an algebraic representation (under the basis of the elements of the SRS) only for c_B. The linearization commitment would be equal to $\mathsf{c}_L = \mathsf{c}b - b\mathsf{c} = [0]_1$, which is independent of the evaluation point x. The adversary can clearly provide an evaluation proof at x for c_L, in spite of not knowing the polynomials implicitly committed

[4] On the downside, PIOPs based on polynomial equations, while at an informal level are easier to describe, tend to have harder-to-parse full specifications.

[5] Such a random point is needed to reduce polynomial identity tests into equations over field elements, through the Schwartz-Zippel Lemma.

in c_A and c_C. In particular, this counter-example shows that we can only extract the second of the three polynomials, under the (more realistic) algebraic group model where the adversary gets to see group elements, besides the SRS, for which it does not know their algebraic representations.

Here we generalize the attack of [LPS23] by considering the general case where, for committed polynomials $(A_i, B_i)_{i\in[n]}$, we want to prove that $\sum_i A_i(x)B_i(x) = y$. In particular, we let $\mathcal{R}_{\mathsf{lin}}$ be the relation where the instances are tuples of the form $((\mathsf{a}_i, \mathsf{b}_i)_{i\in[n]}, x, y)$ such that for field elements x and y, and any i, a_i (resp. b_i) is a commitment to A_i (resp. B_i) for which the equation above holds. What we call (the zkSNARK for) the linearization trick for KZG is the proof system that proves $y_i = A_i(x)$ for any i and then, using the homomorphic property of KZG, generates the linearization commitment $\mathsf{c}_L = \sum_i y_i \cdot \mathsf{b}_i$, and proves $L(x) = \sum_i y_i B_i(x) = y$.

Our generic attack works whenever the polynomials A_i are linearly dependent. The attacker can set, for example, the commitments for the polynomials B_i to $\mathsf{b}_i = \alpha_i \cdot \mathsf{c}$, for an obliviously sampled group element c and for carefully chosen values $(\alpha_i)_i$ such that $\sum_i \alpha_i A_i(X) = 0$. It is easy to see that this adversary can generate a proof for $\sum_i A_i(x^*)B_i(x^*) = 0$, for any x^*, without knowing all the polynomials B_i.

We do not formalize this attack further in our paper as we use it mainly as a motivation for our constructive results. Indeed, we use the intuition behind this counter-example in order to show that the independence of the polynomials A_i is the missing piece of the puzzle to prove extractability of (the zkSNARK defined from) the linearization trick. In particular, the correct recipe for the general compiler proposed by [GWC19] should look not only for the sub-set that minimizes the number of polynomials to open, but should also make sure that the polynomials in such a sub-set are linearly independent. Luckily, the linear independence holds for the subset of polynomials chosen for this optimization in PLONK.[6] We obtain a formalization of this security claim as a corollary of our results on the SE of KZG (see next section).

To summarize, our first set of results deals with proving that the linearization trick for KZG is, under certain conditions, simulation-extractable. We do this in two main steps. First, we consider the SE of KZG evaluation proofs in which the commitment is obtained by a linear combination of other commitments (cf. Sections 2.2 and 5.1). Second, we analyze the sufficient conditions on the A_i polynomials that make the linearization trick simulation extractable (cf. Sections 2.3 and 5.2).

2.2 Simulation Extractability of KZG for linearized commitments

The work of [FFK+23] introduces the notion of policy-based SE, that, roughly speaking, ensures that a zkSNARK is simulation-extractable whenever the adversary complies with a pre-defined policy. This generalized notion of SE is convenient (and necessary) to formalize the simulation-extractable properties of malleable schemes such as KZG.

We summarize the security game of SE for KZG in the algebraic group model. The adversary obtains a list of obliviously-sampled commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n$ where $\mathsf{c}_i \in \mathbb{G}_1$ and it has oracle access to a simulation oracle that, upon input tuples (c, x, y), outputs simulated proofs $\pi = (\mathsf{c} - [y]_1)(s - x)^{-1}$. Additionally, the adversary has oracle access to a random oracle.[7] Eventually, the adversary outputs its forgery π^* for an instance (c^*, x^*, y^*). Standard simulation extractability would just require that the instance was never queried to the simulation oracle. [FFK+23] additionally requires that:

1. The queries of the adversary do not create an algebraic inconsistency in terms of the proved statements. For example, the adversary cannot require simulated proofs for (c, x, y) and (c, x, y') with $y \neq y'$. This constraint is strictly necessary to prove SE for KZG.
2. The evaluation points x for the simulation queries belong to an arbitrary, but fixed ahead of time, set $\mathcal{Q}_x$. This property is called semi-adaptive queries.

[6] Here we are simplifying: the verification in PLONK uses the linearization trick on a mix of polynomials that comes both from the prover and the indexer (i.e., polynomials committed in the specialized reference string). Indexer's polynomials are trivially extractable as they are part of the statement, thus we can refine the property of linear independence by focusing on the polynomials that are not *coupled* with indexer's ones.

[7] This is not strictly necessary, and it could be modeled differently. However, it is a convenient model since the PIOP-to-zkSNARK compiler uses the Fiat-Shamir transformation.

3. The group elements c asked in the simulation queries could not be (algebraically) derived using previously obtained simulated proofs.
4. The forgery's evaluation point x^* must be random and independent of c^*. To enforce this, we check that x^* is derived by applying the random oracle to a string that contains an encoding of c^*.

In this paper, we substantially simplify the policy above by removing the second and third constraints. Besides providing a cleaner and simpler notion of security, removing these constraints has two extra benefits: Removing the second constraint allows proving the PIOP-to-zkSNARK compiler secure in the *non-programmable* random oracle model; removing the third constraint allows extending the PIOP-to-zkSNARK compiler to work with *commit-and-prove* relations (namely, the relation proved by the zkSNARK can have commitments as part of the instance). One limitation of our technique to remove the second constraint is that we need to make a stronger cryptographic assumption than the q-SDH assumption that we call *one-more* q-SDH assumption. This assumption additionally provides an oracle that can be adaptively queried on field element x and (small) natural number i and returns $\left[(s-x)^{-i}\right]_1$. We show that the *one-more* q-SDH assumption holds in the generic group model and that we can reduce a non-adaptive version of the *one-more* SDH to the plain q-SDH assumption.

Moreover, we generalize the fourth constraint from [FFK+23]. Specifically, we change the constraint by allowing c^* to be a commitment to a linearization polynomial. To do so we check that x^* is derived from the random oracle with inputs commitments $(\mathsf{b}_i)_i$ and polynomials[8] $(A_i)_i$ and that $\mathsf{c}^* = \sum A_i(x^*)\mathsf{b}_i$. Proving SE using the latter generalization turns out to be the necessary heavy lifting to perform in order to, then, show that the linearization trick is (policy-based) simulation-extractable, as summarized in the following theorem.

Informal Statement of Theorem 1. *The evaluation proof of KZG polynomial commitment is (policy-based) simulation-extractable in the algebraic group model under our simplified and generalized policy where the forgery can contain a commitment to a linearization polynomial.*

The obliviously-sampled commitments $\mathsf{c}_1, \ldots, \mathsf{c}_n$ not only allow to give an interesting notion of SE in the algebraic group model, they also naturally extend our model to include the algebraic group model with obliviously-sampled elements, as considered in [LPS23]. For this reason, by proving SE of the linearization trick for KZG w.r.t. this new set of constraints, we can derive the following Corollary by considering the subclass of adversaries that do not query the simulation oracle.

Corollary 1. *PLONK and Marlin are knowledge-sound in the AGMOS.*

2.3 Simulation extractability of the linearization trick

Unfortunately, when the adversary can make simulation oracles, the condition of linear independence, sufficient and necessary to restore the (plain) knowledge extractability of the linearization trick, is not sufficient. In fact, consider an adversary, holding an obliviously sampled group element c, that asks for a simulated proof on $(\mathsf{c}, 0, 0)$, namely a proof that the polynomial committed in the *commitment* c evaluates 0 at point 0. Let $\pi = \mathsf{c}/s$ be the simulated proof. The adversary can generate an instance for $\mathcal{R}_{\mathsf{lin}}$ that is not extractable even if the polynomials A_i are linearly independent. It could set the polynomials $A_1(X) = 1$ and $A_2(X) = -X$, thus $(\mathsf{c}_{A_1}, \mathsf{c}_{A_2}) = ([1]_1, [-s]_1)$, and then sets $(\mathsf{c}_{B_1}, \mathsf{c}_{B_2}) = (\mathsf{c}, \pi)$ as the (un-extractable) commitments to the polynomials B_1 and B_2 respectively. Now, for any arbitrary x^* it can generate a forgery, by setting the forged proof π^* to c/s. Its validity follows from the fact that $1 \cdot \mathsf{c} - x^*\mathsf{c}/s = (s - x^*)\mathsf{c}/s$. The reason why this attack works is that KZG proofs and KZG commitments belong to the same domain; thus a proof can be reinterpreted as a commitment. Through this lens, the simulation oracle allows the adversary to push down *the degree of* the un-extractable polynomials. Looking at the counter example from the perspective of formal polynomials, we have that:

$$\sum_{i=1,2} A_i(X)B_i(X) = A_1(X) \cdot \mathsf{c} + A_2(X) \cdot \tfrac{\mathsf{c}}{X} = A_1(X) \cdot \mathsf{c} + \tfrac{A_2(X)}{X} \cdot \mathsf{c} = 0.$$

[8] Technically, we threat these latter polynomials as auxiliary information that the adversary must "declare" before seeing x^*.

The problem is that, while A_1 and A_2 are linearly independent, the polynomials $A_1(X)$ and $A_2(X)/X$ are not so.

As our technical contribution, we show that *higher-degree* linear independence between the polynomials A_i is necessary and sufficient to obtain SE of the linearization trick for the KZG commitment scheme. In particular, instead of defining independence of the polynomials A_i as the condition $\sum \alpha_i A_i \neq 0$ for any choice of $\alpha_i \in \mathbb{F}$, we define their independence w.r.t. any of $\alpha_i \in \mathbb{F}_{\leq \nu}[X]$, for a parameter $\nu \in \mathbb{N}$.

It remains to understand how to set such a parameter ν. To do so, we define a notion of level for *proofs of proofs* that, roughly speaking, indicates how many times the adversary sequentially queried the simulation oracle on an obliviously sampled group element or elements algebraically derived from it. For example, the level of an obliviously sampled element is zero, the level of a proof on it is one, and the level of a proof of a proof could possibly increase to two if we queried twice on the same evaluation point, or remain the same otherwise[9], and so on.

Informal Statement of Theorem 2. *The linearization trick for KZG polynomial commitment is (policy-based) simulation-extractable in the AGM under our simplified and generalized policy and assuming that the extracted polynomials $A_1, \ldots, A_n$ are independent for a parameter ν and the maximum level reached by simulated proofs queried by the adversary is smaller or equal to ν.*

The above theorem completes the set of results that we need to obtain SE when instantiating the PIOP-to-zkSNARK compiler with KZG. Next, we address the SE requirements at the PIOP level.

2.4 Capturing PIOPs with delegation phase

The work of [FFK+23] showed that only a subset of all the PIOPs can be compiled to SE-zkSNARKs. For example, if we take a PIOP for the product relation $\mathcal{R} \times \mathcal{R}$ which, internally, sequentially runs two instances of a PIOP for $\mathcal{R}$, we can incur copy-paste attacks that re-use a simulated proof for the first instance in $\mathcal{R}$ and honestly prove the knowledge for the second instance. To avoid these pathological cases, [FFK+23] introduced the notion of *compilation-safeness* that gives a sufficient condition for a PIOP to be compiled to SE-zkSNARK. In a nutshell, a PIOP is compilation-safe if it has a *witness-dependent* last round. Here, by "witness-dependent round", we mean that the polynomials sent at such a round store information that depends on the witness and are necessary to extract the full witness at the PIOP level.

However, Marlin [CHM+20] and other proof systems [CFF+21,RZ21] based on Checkable Subspace Sampling Arguments [RZ21] are not compilation-safe. They have a two-phase algorithm for the prover where the first phase is witness-dependent, while the second phase, which we call *delegation phase*, is witness-independent, and in particular is performed to enable succinct verification. For PIOPs with delegation, we need a more careful compilation strategy. To avoid copy-paste attacks that would copy the witness-dependent transcript from a simulated proof and compute a fresh witness-independent suffix for the forgery, we need to make sure that (1) the polynomial oracles sent during the delegation phase are committed using a deterministic commitment and that (2) the delegation phase is unique at the PIOP level, namely, there is only one possible answer that any malicious prover for the PIOP can send in the delegation phase, once fixed the messages of all the previous rounds[10]. With this characterization of PIOPs we can prove the following theorem:

Informal Statement of Theorem 4. *PIOPs with delegation phase can be compiled to simulation-extractable commit-and-prove zkSNARKs with the linearization trick optimization. Also, security holds in the algebraic group model with oblivious sampling and in the non-programmable random oracle assuming the one-more q-SDH assumption.*

In the informal theorem above we swept under the rug many details. In particular, the reader may wonder about the connection of the independence parameter ν for the security of the linearization trick

[9] Briefly, the reason why the level does not increase in this case is because the rational function $1/((X-x_1)(X-x_2))$ is in the linear span of $(X-x_1)^{-1}$ and $(X-x_2)^{-1}$.

[10] For example, Marlin compiled using hiding KZG, or with FRI-based polynomial commitments, is provably not *strong* simulation extractable, while it could still be proved *weak* simulation extractable.

and the requirements for the compilation above. What we show in the proof of the compiler is that the parameter ν can be kept very low. In fact, the reduction from SE-zkSNARK to the SE of linearization trick (obliviously) samples fresh commitments for each simulation query made by the adversary and, assuming that the PIOP is zero-knowledge, the reduction needs to query (linearization trick) simulated proofs only for this fresh batch of commitments, thus bounding the level of proof of proofs to, at maximum, the number of evaluations needed by a single execution of the PIOP.

In Section 6.4 we show that PLONK and Marlin have PIOPs fulfilling our requirements. Combining this with the theorem above, we obtain our main results on the SE of these schemes.

3 Preliminaries

A function f is negligible in λ (we write $f \in \mathsf{negl}(\lambda)$) if it approaches zero faster than the reciprocal of any polynomial. For an integer $n \geq 1$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. Vectors and matrices are denoted in boldface. Calligraphic letters denote sets, while set sizes are written as $|\mathcal{X}|$. Lists are represented as ordered tuples, e.g. $L := (L_i)_{i\in[n]}$ is a shortcut for the list of n elements $(L_1, \ldots, L_n)$. 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 $L = (a, b, c)$. The difference between lists and vectors is that elements of vectors are of the same type.

Definition 1. *Let $\mathcal{A} = \{A_i\}_{i\in[n]}$ be a set of polynomials in $\mathbb{F}[X]$ and $\nu \in \mathbb{N}$. We say that $\mathcal{A}$ are ν-independent polynomials if $\forall(\alpha_j)_j \in \mathbb{F}_{\leq\nu}[X]$: $\sum_j \alpha_j A_j(X) \neq 0$.*

Lemma 1. *Let $J \in \mathbb{N}$, and let $\{x_j\}_{j\in[J]} \subset \mathbb{F}_q$, $(\nu_j)_{j\in[J]} \in \mathbb{N}^J$, $\nu^* = \sum_j \nu_j$ and let $S := \mathsf{Span}(\{1\} \cup \{(X - x_j)^{-k}\}_{j\in[J],k\in[\nu_j]})$. Consider the function ϕ with domain S that maps rational functions Ω to polynomials $\Omega \cdot \prod_j (X - x_j)^{\nu_j}$. The function ψ maps is a morphism with image the set $\mathbb{F}_{\leq\nu^*}[X]$.*

Proof. First, notice that any element $v \in S$ can be written as a linear combination of the form $\alpha + \sum_{j,k} \beta_{j,k}(X - x_j)^{-k}$. It holds that ψ carries over the basis since $\psi(v) = \psi(\alpha + \sum_{j,k} \beta_{j,k}(X - x_j)^{-k}) = \alpha\psi(1) + \sum_{j,k} \beta_{j,k}\psi((X - x_j)^{-k})$. The only thing left to prove is that $\psi(1) \cup \psi((X - x_j)^{-k})$ is indeed a basis for $\mathbb{F}_{\leq\nu^*}[X]$. We notice that of $\psi((X - x_j)^{-k})$ is of the form $n_{j,k}(X) := (X - x_j)^{\nu_j - k} \prod_{j'\neq j} (X - x'_j)^{\nu_{j'}}$, and $\psi(1)$ is simply $n(X) := \prod_j (X - x_j)^{\nu_j}$. To conclude the proof, we show that these polynomials are linearly independent. Given the fact that they are $\nu^* + 1$ polynomials of degree at most ν^*, this is equivalent to prove that they span $\mathbb{F}_{\leq\nu^*}[X]$.

Let $f(X) := \alpha n(X) + \sum_{j,k} \alpha_{j,k} n_{j,k}(X)$. We need to prove that $f(X) \equiv 0$ if and only if $\alpha = \alpha_{j,k} = 0$. For all j, it must be that $f(x_j) = 0$. We have that for any x_j: $f(x_j) = \alpha n(x_j) + \sum_{j,k} \alpha_{j,k} n_{j,k}(x_j) = \alpha_{j,\nu_j} \prod_{j'\neq j} (x_j - x_{j'})$, which is equal to 0 if and only if $\alpha_{j,\nu_j} = 0$. We can rewrite $f(X)$ as:

$$f(X) = \alpha n(X) + \sum_{j,k\leq\nu_j} \alpha_{j,k\leq\nu_j} n_{j,k}(X) = f'(X) \prod_j (X - x_j)$$

where $f'(X)$ is equal to:

$$\alpha \prod_j (X - x_j)^{\nu_j - 1} + \sum_{j,k} \alpha_{j,k} (X - x_j)^{\nu_j - k - 1} \prod_{j'} (X - x_{j'})^{\nu_{j'} - 1}$$

Note that if $f(X) \equiv 0$, also $f'(X) \equiv 0$. We can recursively apply the same argument, proving that all the coefficients $\alpha_{j,k} = 0$, for all $k > 0$. In the final step, we can write $f(X)$ as $\alpha \prod_j (X - x_j)$ which is equal to 0 iff $\alpha = 0$. □

Asymmetric Bilinear groups. An asymmetric bilinear group is a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q, the elements P_1, P_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently-computable non-degenerate bilinear map, and there is no efficiently

computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Let $\mathsf{GroupGen}$ be some probabilistic polynomial-time (PPT) algorithm which on input 1^λ, where λ is the security parameter, returns a description $\mathsf{pp}_\mathbb{G}$ of a bilinear group. 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 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 $e([a]_1, [b]_2)$. 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.

$\mathbf{Exp}^{(n,d)\text{-OMSDH}}_{\mathsf{GroupGen},\mathcal{A}}(\lambda)$

$\mathcal{Q}_x \leftarrow \emptyset$

$\mathsf{pp}_\mathbb{G} \leftarrow\!\$\ \mathsf{GroupGen}(1^\lambda)$

$s \leftarrow\!\$\ \mathbb{F}_q$

$(x^*, \mathsf{y}^*) \leftarrow \mathcal{A}^{\mathcal{O}_s}(\mathsf{pp}_\mathbb{G}, \left[1, s, \ldots, s^d\right]_1, [1, s]_2)$

return $x^* \notin \mathcal{Q}_x \wedge \mathsf{y}^* = \left[\frac{1}{s - x^*}\right]_1$

Oracle $\mathcal{O}_s(x, i)$

if $i > n$:

  return $\bot$

$\mathcal{Q}_x \leftarrow \mathcal{Q}_x \cup \{x\}$

let $z \leftarrow \left[(s - x)^{-i}\right]_1$

return z

Fig. 1. The OMSDH experiment.

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

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

With the lingo of [BFL20], OMSDH is a special case of an adaptive Uber Assumption for Rational Fractions. When the set of points $\mathcal{Q}_x$ is fixed before the experiment starts, the assumption falls back to an Uber Assumption for Rational Fractions and Flexible Target, as defined in [BFL20], that is reducible to discrete log in the AGM. We defer the proof to Appendix A.

Definition 3 (Algebraic algorithm, [FKL18]). *An algorithm $\mathcal{A}$ is* algebraic *if for all group elements* z *that $\mathcal{A}$ outputs (either as returned by $\mathcal{A}$ or by invoking an oracle), it additionally provides the representation of* z *relative to all previously received group elements. That is, if* elems *is the list of group elements that $\mathcal{A}$ has received so far, then $\mathcal{A}$ must also provide a vector $\boldsymbol{r}$ such that* $\mathsf{z} = \langle \boldsymbol{r}, \mathsf{elems} \rangle$.

Since in our work we mostly focus on algebraic adversaries receiving as input a structured refererence string of the form $(\left[s^{i-1}\right]_1)_{i \in [d]}$, we parse the first d coefficients of $\boldsymbol{r}$ as an encoding of the polynomial $f(X) := \langle (r_i)_{i \in [d]}, (X^{i-1})_{i \in [d]} \rangle$.

Finally, we state these assumptions on distributions.

Definition 4 (Witness Samplability, [JR13]). *A distribution $\mathcal{D}$ is witness samplable if there exists 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 5 ($\mathcal{D}_{\ell,k}$**-Aff-MDH assumption, [FFK+23]).** *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 $\left[\boldsymbol{x}^\top \boldsymbol{A}\right]_1 = [\boldsymbol{y}]_1$.*

4 Simulation-Extractable CP-SNARKS in AGM

We define a PT relation $\mathcal{R}$ verifying triple $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ as in [GKM+18]. 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 (non-interactive) proof system 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 group 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 $\mathbb{w}$ s.t. $\mathcal{R}(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ holds, and returns a proof.

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

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

Zero-Knowledge in the SRS (and RO) model. 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 $\mathbb{x}$
- $(a, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s)$ answers random oracle queries

The state $\mathsf{st}_{\mathcal{S}}$ is updated after each operation. Similarly to [FKMV12,GOP+22,FFK+23], we define the following wrappers.

Definition 6 (Wrappers for NIZK Simulator). *The following oracles are stateful and share their state* $\mathsf{st} = (\mathsf{st}_{\mathcal{S}}, \mathsf{coms}, \mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathsf{srs}}, \mathcal{Q}_{\mathsf{aux}})$ *where* $\mathsf{st}_{\mathcal{S}}$ *is initially set to be the empty string, and* $\mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}$ *and* $\mathcal{Q}_{\mathsf{aux}}$ *are initially set to be the empty sets.*

- $\mathcal{S}_1(\mathbb{x}, \mathsf{aux})$ *returns the first output of* $\mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathbb{x}, \mathsf{aux})$.[11]
- $\mathcal{S}'_1(\mathbb{x}, \mathbb{w})$ *first checks* $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ *where* pp *is part of* srs *and then runs (and returns the output of)* $\mathcal{S}_1(\mathbb{x})$.
- $\mathcal{S}_2(s, \mathsf{aux})$ *first checks if the query* s *is already present in* $\mathcal{Q}_{\mathsf{RO}}$ *and in case answers accordingly, otherwise it returns the first output* a *of* $\mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s)$. *Additionally, the oracle updates the set* $\mathcal{Q}_{\mathsf{RO}}$ *by adding the tuple* (s, aux, a) *to the set. In the case of* non-programmable *random oracle model,* $\mathcal{S}$ *is notified of the RO query but cannot control the answer* a.

Almost all the oracles in our definitions can take auxiliary information as additional input. 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 8), 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.

[11] More often, simulators need only the first three inputs, see Definition 7; abusing notation, we assume that such simulators simply ignore the auxiliary input aux.

Definition 7 (Zero-Knowledge). *A NIZK* NIZK *is (perfect) zero-knowledge if there exists a PPT simulator* $\mathcal{S}$ *such that for all adversaries* $\mathcal{A}$*:*

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

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.

As in [CFF+21,FFK+23], we introduce a weaker notion of zero-knowledge. A NIZK is L-leaky zero-knowledge if its proofs may leak some information, namely a proof leaks $L(\mathbb{x},\mathbb{w})$, where $(\mathbb{x},\mathbb{w}) \in \mathcal{R}$. This is formalized by giving the zero-knowledge simulator the value $L(\mathbb{x},\mathbb{w})$ that should be interpreted as a hint for the simulation of proofs.

Definition 8 (Leaky Zero-Knowledge, [FFK+23]). *A NIZK* NIZK *is* L*-leaky zero-knowledge if there exists a PPT simulator* $\mathcal{S}$ *such that for all adversaries* $\mathcal{A}$*:*

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

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

1. KGen takes as input group parameters $\mathsf{pp}_{\mathbb{G}} \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$ and outputs a commitment key ck.
2. Com takes the commitment key ck, and a message $m \in \mathcal{M}$, and outputs a commitment c and an opening o.
3. VerCom takes as input the commitment key ck, a commitment c, a message $m \in \mathcal{M}$ and an opening o, and it accepts $(b = 1)$ or rejects $(b = 0)$.

We consider polynomial commitment schemes where the message space is $\mathbb{F}_{\leq d}[X]$ for a degree parameter d given as additional input to KGen. Besides correctness, a scheme CS satisfies two more properties.

Definition 9 (Binding). *A commitment scheme* CS *is (computationally) binding if no PPT adversary can find, unless with negligible probability, a commitment* c*, two messages* $m \neq m'$ *and two openings* o, o'*:*

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

Definition 10 (Hiding:). *A commitment scheme* CS *is (statistically) hiding if* $\forall m, m'$, $\forall \mathsf{ck}$*:*

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

CP-SNARKs. Commit-and-Prove SNARKs, or simply CP-SNARKs, are proof systems whose relations verify predicates over commitments [CFQ19]. We refer to a CP-SNARK for a relation $\mathcal{R}$ and a commitment scheme CS as a tuple of algorithms $\Pi := (\mathsf{KGen},\mathsf{Prove},\mathsf{Verify})$ where $\mathsf{KGen}(\mathsf{ck}) \to \mathsf{srs}$ is an algorithm that takes as input a commitment key ck for CS and outputs $\mathsf{srs} := (\mathsf{ek},\mathsf{vk})$[12]; 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 $\mathsf{pp}_{\mathbb{G}}$, runs $\mathsf{ck} \leftarrow\!\!\$\ \mathsf{CS.KGen}(\mathsf{pp}_{\mathbb{G}})$ and $\mathsf{srs} \leftarrow\!\!\$\ \Pi.\mathsf{KGen}(\mathsf{ck})$, and outputs srs; then the tuple $(\mathsf{KGen}',\mathsf{Prove},\mathsf{Verify})$ defines a SNARK.

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

RO transcript. In our work, we often need to enforce that a point x is random and independent w.r.t. a bunch of elements. To capture this scenario, we check that x is derived by applying the random oracle (RO) to a string that either (i) contains an encoding of the elements or (ii) the output of another RO query that satisfies the first condition, and so on. We use the shortcut $(x_1, \dots x_n; y_1, \dots, y_m) \rightarrow_{\mathsf{RO}} a$ to indicate that there is a list of tuples $(s_1, \mathsf{aux}_1), \dots, (s_k, \mathsf{aux}_k)$ and a list $(a_i)_{i \in [k-1]}$ such that

1. $\forall i \in [k-1] : \mathsf{RO}(s_i, \mathsf{aux}_i) = a_i$, and $\mathsf{RO}(s_k, \mathsf{aux}_k) = a$
2. $\forall i \in [k-1] : a_i$ is a substring of s_{i+1}
3. $\forall j \in [n], \exists i \in [k] : x_j$ is a substring of s_i
4. $\forall j \in [m], \exists i \in [k] : y_j$ is contained in aux_i

4.1 Policy-Based Simulation Extractability

We recall the definitional framework of [FFK+23]. A policy is a tuple $\Phi := (\Phi_0, \Phi_1)$ of PPT algorithms. The Φ-simulation extractability experiment starts by running the policy algorithm Φ_0, which generates public information pp_Φ. The public information may contain parameters that define the constraints later checked by Φ_1. In the case of commit-and-prove proof systems, the public information may contain a list of commitments $\mathsf{coms} := (\mathsf{c}_i)_i$ for which the adversary does not know openings (i.e., obliviously sampled), but on which it can query simulated proofs. Therefore, we feed pp_Φ to the adversary. After receiving a forgery from the adversary, the security experiment runs the extraction policy Φ_1. The policy Φ_1 is a predicate that decides whether the attack is legitimate, e.g., it is not a trivial one such as returning a proof received by the simulation oracle. To decide this, Φ_i takes as input: (i) The public parameters pp_Φ; (ii) The forged instance and proof $(\mathbb{x}, \pi)$; (iii) The view view of the experiment that 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.[14]

We extend the definitional framework of [FFK+23] to the $\mathcal{F}$-extractability setting, introduced by [BCKL08], where the extractor extracts a function of the witness. Notice that the simulation policy may depend on the function $\mathcal{F}$. Clearly, when $\mathcal{F}$ is the identity function, we obtain the policy-based notion of simulation extractability defined by [FFK+23].

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

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

Moreover, given a family of policies $\boldsymbol{\Phi}$ and a family of functions $\boldsymbol{\mathcal{F}}$, we say that a NIZK Π is $(\boldsymbol{\Phi}, \boldsymbol{\mathcal{F}})$-simulation-extractable if Π is $(\Phi, \mathcal{F})$-simulation-extractable for any $\Phi \in \boldsymbol{\Phi}$ and $\mathcal{F} \in \boldsymbol{\mathcal{F}}$. We say that Π is $\boldsymbol{\Phi}$-simulation-extractable if Π is $(\boldsymbol{\Phi}, id)$-simulation-extractable and id is the identity function.

4.2 Simulation Extractability for KZG-based CP-SNARKs

We specialize the notion introduced in the previous section for CP-SNARKs based on the KZG commitment scheme. First, we specialize the definition of policy-based SE to the algebraic group model.

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

[13] As noted in [FFK+23], even if the given NIZK is not in the random oracle it still makes sense to assume the existence of the set $\mathcal{Q}_{\mathsf{RO}}$ (e.g., to model security for NIZK protocols that eventually are used as sub-protocols in ROM-based protocols)

[14] We recall that the presence of aux_Φ is exploited to provide the adversary an interface with the policy, namely, to provide evidences that the forgery belongs to set of instances for which the SE is guaranteed.

$\mathbf{Exp}^{(\Phi,\mathcal{F})\text{-}se}_{\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda)$

$\mathsf{pp}_{\mathbb{G}} \leftarrow\!\$\ \mathsf{GroupGen}(1^\lambda)$
$(\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$
$\mathsf{pp}_{\Phi} \leftarrow\!\$\ \Phi_0(\mathsf{pp}_{\mathbb{G}})$
$(\mathbb{x}, \pi, \mathsf{aux}_{\mathcal{E}}, \mathsf{aux}_{\Phi}) \leftarrow \mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}(\mathsf{srs}, \mathsf{pp}_{\Phi})$
$\mathbb{w}_{\mathcal{F}} \leftarrow \mathcal{E}(\mathsf{srs}, \mathbb{x}, \pi, \mathsf{aux}_{\mathcal{E}})$
$\mathsf{view} \leftarrow (\mathsf{srs}, \mathsf{pp}_{\Phi}, \mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathsf{aux}})$
if $\Phi_1((\mathbb{x}, \pi), \mathsf{view}, \mathsf{aux}_{\Phi}) \wedge \mathsf{Verify}^{\mathcal{S}_2}(\mathsf{srs}, \mathbb{x}, \pi)$
$\quad \wedge\, \forall\, \mathbb{w}$ s.t. $\mathcal{F}(\mathbb{w}) = \mathbb{w}_{\mathcal{F}} : (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}$
$\quad$ **then return** 1
else return 0

$\mathcal{S}_1(\mathbb{x}, \mathsf{aux})$

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

$\mathcal{S}_2(s, \mathsf{aux})$

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

Fig. 2. The $(\Phi, \mathcal{F})$-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 $\mathcal{A}$.

KZG commitment scheme. We recall the (non-hiding version of the) commitment scheme of Kate, Zaverucha and Goldberg [KZG10] that is a fundamental building block of all our CP-SNARKs. KZG is a polynomial commitment scheme defined over a bilinear group $\mathbb{G}$ that consists of the following algorithms:

- $\mathsf{KGen}(\mathsf{pp}_{\mathbb{G}}, d)$ on input the group parameters and a degree bound $d \in \mathbb{N}$,outputs $((\left[s^j\right]_1)_{j\in[0,d]}, [1, s]_2)$ where $s \leftarrow\!\$\ \mathbb{F}_q$.
- $\mathsf{Com}(\mathsf{ck}, f(X))$ outputs a commitment $\mathsf{c} := [f(s)]_1$.
- $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, f(X))$ outputs 1 iff $\mathsf{c} = [f(s)]_1$.

KZG commitment scheme allows for simple and efficient *evaluation proofs* which, in the framework of [CFQ19], is a CP-SNARK Π_{evl} for the relation $\mathcal{R}_{\mathsf{evl}}((\mathsf{c}, x, y), f) = 1$ iff $f(x) = y \wedge \mathsf{c} = [f(s)]_1$. We describe such a CP-SNARK below:

- $\mathsf{Prove}_{\mathsf{evl}}(\mathsf{ek}, \mathbb{x} = (\mathsf{c}, x, y), \mathbb{w} = f)$ outputs $\pi := [\pi(s)]_1$, where $\pi(X)$ is the polynomial such that $\pi(X)(X - x) \equiv f(X) - y$.
- $\mathsf{Verify}_{\mathsf{evl}}(\mathsf{vk}, \mathbb{x} = (\mathsf{c}, x, y), \pi)$ outputs 1 iff $e(\mathsf{c} - [y]_1, [1]_2) = e(\pi, [s - x]_2)$.

The above CP-SNARK is knowledge extractable in the AGM [CHM+20] and in the AGMOS [LPS23]. The ZK simulator is $\mathcal{S} := (\mathcal{S}_0, \mathcal{S}_1)$, where $\mathcal{S}_0$ outputs the trapdoor s together with the srs, and $\mathcal{S}_1$ simulates proofs for $\mathbb{x} := (\mathsf{c}, x, y)$ outputting $\pi := (\mathsf{c} - [y]_1)(s - x)^{-1}$.

KZG-based CP-SNARKs. Informally, we say that a CP-SNARK is KZG-based if it internally calls, implicitly or explicitly, the CP-SNARK Π_{evl} defined in the previous paragraph. This definition is rather informal, thus, we give below a formal notion that includes all the KZG-based CP-SNARKs.

Definition 13 (KZG-based CP-SNARK). *We say that Π is KZG-based if Π is a CP-SNARK (for some relation $\mathcal{R}$ and) for the KZG commitment scheme, where: the proofs can be parsed as vectors of elements in $\mathbb{G}_1$ and $\mathbb{F}$, and the verification on $(\mathbb{x}, \pi)$ consists of equations of the form:*

$$\textstyle\sum_i e(\mathsf{x}_i, [p_i(s)]_2) + \sum_i e(\mathsf{q}_i, [p'_i(s)]_2) = [p''(s)]_T$$

where $(\mathsf{x}_i)_i$ are the $\mathbb{G}_1$-elements of the instance $\mathbb{x}$, $(\mathsf{q}_i)_i$ are the $\mathbb{G}_1$-elements of the proof π, and the (linear) polynomials p_i, p'_i and p'' are functions of the instance $\mathbb{x}$, the proof π, and possibly of the random oracle.

Algebraic Consistency. Faonio *et al.* [FFK+23] defines a necessary property to achieve extractability in the presence of a simulation oracle for any KZG-based SNARKs. The property is motivated by the generalization of the simple attack where, for a commitment c, an adversary is given two simulated KZG evaluation proofs π_1, π_2 on the same evaluation point x and for two different evaluation values y_1 and y_2. By the homomorphic property of KZG, the adversary can forge an evaluation proof on the statement $((\alpha+\beta)\mathsf{c}, x, \alpha y_1 + \beta y_2)$ by setting the proof $\alpha\pi_1 + \beta\pi_2$. This attack can be generalized whenever the adversary can leverage *algebraic inconsistencies* provided by simulated proofs, as we explain hereafter.

Let $\boldsymbol{A} \in \mathbb{F}[X]^{m\times n}$, and let $\boldsymbol{b} \in \mathbb{F}[X]^m$. We have that $(\boldsymbol{A}\|\boldsymbol{b})$ describes a linear system of polynomial equations that admits a solution if there exists a vector $\boldsymbol{z} \in (\mathbb{F}[X])^n$ such that $\boldsymbol{A}\cdot\boldsymbol{z} = \boldsymbol{b}$.

Definition 14 (Algebraic Consistency). *Let Π be a KZG-based CP-SNARK. Let* view *be the view of $\mathcal{A}$ at the end of the SE game* $\mathbf{Exp}^{(\Phi,\mathcal{F})\text{-}se}_{\Pi,\mathcal{A},\mathcal{S},\mathcal{E}}$ *for an adversary $\mathcal{A}$. We say that the view* view *is* algebraic consistent *if the linear system S of polynomial equations, that we describe next, admits a solution.*

Let coms *be the list of simulated commitments in* pp_Φ*, where* $\mathsf{coms} := (\mathsf{c}_k)_k$ *and* $\forall k : \mathsf{c}_k \in \mathbb{G}_1$*, and* proofs *be the list of simulated proofs* $\mathsf{proofs} := (\pi_k)_k$ *(where* $\pi_k := (\mathsf{q}_{k,j})_j, \boldsymbol{y}_k$ *and* $\forall k,j : \mathsf{q}_{k,j} \in \mathbb{G}_1, y_{k,j} \in \mathbb{F}$*) included in the view* view*. We assign to each simulated commitment* c_k *in* view *a formal variable (defining a polynomial) Z_k, similarly we assign to each $\mathbb{G}_1$-group element of the simulated proofs* $\mathsf{q}_{k,j}$ *formal variables (defining polynomials) $Q_{k,j} \in \mathbb{F}_{\leq d}[X]$. For each simulation query we define new equations derived by the verification equations of Π and from the algebraic representations of the instances queried to the simulation oracle. In particular, for the k-th simulation query with instance* $\mathbb{x}_k$ *and whose $\mathbb{G}_1$-elements are* $(\mathsf{x}_{k,j})_j$ *and simulated proof π_k, we can associate the polynomials of the verification equation $p_{k,i}, p'_{k,i}$ and p''_k and we add the following equation to the linear system of polynomial equations S:*

$$\textstyle\sum_i \left(f_{k,i}(X) + \langle \boldsymbol{c}_{k,i}, \boldsymbol{Z}\rangle + \langle \boldsymbol{o}_{k,i}, \boldsymbol{Q}\rangle\right) p_{k,i}(X) + \sum_i Q_{k,i}\cdot p'_{k,i}(X) = p''_k(X)$$

where $\boldsymbol{Z}$ is the vector of all variables Z_j for any j and $\boldsymbol{Q}$ is the vector of all the variables $Q_{i,j}$ for any i,j, and the algebraic representation of $\mathsf{x}_{k,i}$ *is $(\boldsymbol{f}_{k,i}, \boldsymbol{c}_{k,i}, \boldsymbol{o}_{k,i})$ and $f_{k,i}(X) = \sum_j (\boldsymbol{f}_{k,i})_j X^j$.*

As a concrete example, for the KZG-based CP-SNARK $\Pi_{\mathtt{evl}}$, from the k-th simulation query with instance (c, x, y) we can derive and add to the linear system of polynomial equations the equation:

$$(f(X) + \langle \boldsymbol{c}, \boldsymbol{Z}\rangle + \langle \boldsymbol{o}, \boldsymbol{Q}\rangle) - y - Q_k(X - x) = 0,$$

where $\mathsf{c} = [f(s)]_1 + \langle \boldsymbol{c}, \mathsf{coms}\rangle + \langle \boldsymbol{o}, \mathsf{proofs}\rangle$.

Notice that once we have computed a solution for S, the linear system of polynomial equations, we can represent it in a reduced form.

Definition 15 (Reduced solution). *Given a solution $(\boldsymbol{z}, \boldsymbol{q})$ for a linear system S defining the algebraic consistency of SE experiment (see Definition 14), we say that $\boldsymbol{z}$ is its* reduced solution.

Given a *reduced solution* for S, it is possible to determine the (non-rational) polynomials $q_i(X)$. In fact, once we assign the values for the variables $\boldsymbol{Z}$ to $\boldsymbol{z}$, the linear system has $|\mathsf{proofs}|$ variables and $|\mathsf{proofs}|$ (independent) equations, thus admits one solution.

5 Simulation Extractability of KZG

In [FFK+23], $\Pi_{\mathtt{evl}}$ was proved simulation-extractable under a semi-adaptive policy. The main limits of that policy are that the adversary can query simulated proofs on instances (c_j, x_j, y_j) where only the evaluation values y_j can be adaptively chosen. Instead, the evaluation points x_j must be selectively chosen independently of the commitment key, and the commitments c_j cannot depend on the simulated proofs. In the next section we generalize the scheme $\Pi_{\mathtt{evl}}$ and prove simulation extractability for two classes of policies. As corollary, we derive that KZG achieves simulation extractability in AGM and RO:

1. under q-SDH assumption, for a semi-adaptive policy more flexible than the one in [FFK+23]
2. under the OMSDH assumption, for a fully adaptive policy.

5.1 Simulation Extractability of batched KZG

We consider a batched version of $\Pi_{\mathtt{evl}}$ described in Section 4.2 that we name $\Pi_{\mathtt{m\text{-}evl}}$. This batched version, given in the ROM, follows from [GWC19,MBKM19] and relies on the linearity of the polynomials and the homomorphic properties of KZG.

- $\mathsf{Prove}_{\mathtt{m\text{-}evl}}(\mathsf{ek}, \mathbb{x} = (x, (\mathsf{c}_{i\in[n]}, y_i)_{i\in[n]}), \mathbb{w} = (f_i)_i)$ computes for $i \in [n] : \pi_i \leftarrow \mathsf{Prove}_{\mathtt{evl}}(\mathsf{ek}, (\mathsf{c}_i, x, y_i), f_i)$, $\rho \leftarrow \mathsf{RO}(\mathsf{vk}\|\mathbb{x})$ and returns $\sum_i \rho^{i-1}\pi_i$.
- $\mathsf{Verify}_{\mathtt{m\text{-}evl}}(\mathsf{vk}, \mathbb{x} = (x, (\mathsf{c}_i, y_i)_i), \pi)$ computes $\mathsf{c} \leftarrow \sum_i \rho^{i-1}\mathsf{c}_i$, $y \leftarrow \sum_i \rho^{i-1} y_i$, $\rho \leftarrow \mathsf{RO}(\mathsf{vk}\|\mathbb{x})$, and returns $\mathsf{Verify}_{\mathtt{evl}}(\mathsf{vk}, (x, \mathsf{c}, y))$.

We describe our extraction policy. First, we notice that to prove simulation extractability for the KZG-based CP-SNARK $\Pi_{\mathtt{m\text{-}evl}}$ (and in general for any KZG-based CP-SNARK), we can consider the (stronger) SE experiment where the simulation oracle returns simulated proofs for $\Pi_{\mathtt{evl}}$. In fact, we can consider the reduction that, at any simulation oracle call for $\Pi_{\mathtt{m\text{-}evl}}$ from the adversary, would first call the simulation oracle for $\Pi_{\mathtt{evl}}$ and then assemble a valid simulated proof for $\Pi_{\mathtt{m\text{-}evl}}$.

To enable the adversary to ask simulation proofs for commitments c whose representation depends on previously obtained simulated proofs (what we call a *proof of a proof*), we need to introduce the following definition.

Definition 16 (Nesting level of a proof). *Let* view *be the view of an adversary at the end of the SE game for a KZG-based CP-SNARK, and let $x_1, \ldots, x_n$ be the list of all the evaluation points in the simulation queries. For each (single-eval) simulation statement $((\mathsf{c}, x, y), \pi) \in \mathcal{Q}_{\mathsf{sim}}$ let $\boldsymbol{c}$ (resp. $\boldsymbol{o}$) be the coefficients associated with the commitments $\mathsf{coms} := (\mathsf{c}_j)_j$ (resp. simulated proofs $\mathsf{proofs} := (\pi_j)_j$) in the algebraic representation of c. Let b_k be equal to 1 if $x = x_k$ and 0 otherwise. Let $b_{j,k}$ be equal to 1 if $x = x_k$ and $\mathsf{c}_j \neq 0$, and 0 otherwise.*

For all $j \in [|\mathsf{coms}|]$, $k \in [n]$, the nesting level $\nu_\pi(j,k)$ of the simulated proof π on the simulated commitment c_j and the point x_k is equal to:

$$\nu_\pi(j,k) := \max_{i:o_i\neq 0 \wedge \nu_{\pi_i}(j,k)\neq 0} \{\nu_{\pi_i}(j,k) + b_k\} \cup \{b_{j,k}\}$$

We define the maximum nesting level $\bar{\nu} := \max_j \sum_k \max_{\pi_i} \nu_{\pi_i}(j,k)$.

Informally, the idea behind the maximum nesting level $\bar{\nu}$ is that each *proof of a proof* involving at some point one of the simulated commitments can (possibly but not always) increase the degree of the denominator of the rational function associated with such a simulated proof. The value $\bar{\nu}$ is the minimal upper bound on the degree of (the denominators of) the rational functions associated with the simulated proofs (see Lemma 1). We consider the following constraints, parametrized by a set $\mathcal{I} \subseteq [n]$.

Point check: given a set of points $\mathcal{Q}_x \in \mathsf{pp}_\Phi$, return 1 if $\forall \mathbb{x}$ queried to $\mathcal{S}_1$, we have that $\mathbb{x}.x \in \mathcal{Q}_x$

Hash check with Linearized Commitment (and parameter $\mathcal{I}$): Parsing the forgery instance $\mathbb{x}^* := (x^*, (\mathsf{c}_i^*, y_i^*)_{i\in[n]})$, return 1 if and only if there exist group elements $(\mathsf{b}_{i,r})_r$, polynomials $A_{i,r}(X)$, a non-constant polynomial h such that:

- $\forall i \in [n] : \mathsf{c}_i^* = \sum_r A_{i,r}(x^*)\mathsf{b}_{i,r}$
- $\forall i \in \mathcal{I} : ((\mathsf{b}_{i,r})_r; (A_{i,r})_r, h) \rightarrow_{\mathsf{RO}} a$.
- $h(a) = x^*$
- $\forall i \in \mathcal{I} : \{A_{i,r}\}_r$ are $\bar{\nu}$-independent polynomials, where $\bar{\nu}$ is the maximum nesting level (cf. Definition 16)

Looking ahead, the *point check* does require some form of programmability of the RO at SNARK level, while the *hash check* (with linearized commitment) essentially consists of checking the hash of the "virtual" representation of a group element and is weak enough to capture schemes tailored for optimizations, like the linearization trick [GWC19,OL].

	[FFK+23]	$\boldsymbol{\Phi}^{\texttt{s-adpt}}_{\texttt{m-evl}}$	$\boldsymbol{\Phi}^{\texttt{adpt}}_{\texttt{m-evl}}$
Hash check w/ L.C.		✓	✓
Hash check	✓		
Point check	✓	✓	
Commitment check	✓		
Assumption (AGM)	$(Q+d+1)$-DL	$(Q+d+1)$-DL	(Q,d)-OMSDH

Table 1. Comparison of extraction policies in terms of constraints and security assumptions with related work.

Definition 17. *Let $\boldsymbol{\Phi}^{\texttt{adpt}}_{\texttt{m-evl},\mathcal{I}}$ (resp. $\boldsymbol{\Phi}^{\texttt{s-adpt}}_{\texttt{m-evl},\mathcal{I}}$) be the set of policies $\Phi_{\mathcal{D}} = (\Phi_0^{\mathcal{D}}, \Phi_1)$ for a distribution $\mathcal{D}$ where: 1. $\Phi_0^{\mathcal{D}}$ on input group parameters $\mathsf{pp}_{\mathbb{G}}$ outputs $\mathsf{pp}_{\Phi} := \mathsf{coms}$, where coms is a vector of commitments sampled from $\mathcal{D}$ (resp. additionally it outputs a set $\mathcal{Q}_x \subseteq \mathbb{F}$). 2. Φ_1 is the hash check with parameter $\mathcal{I}$ defined above. (Resp. Φ_1 is the logical conjunction of the hash check, with parameter $\mathcal{I}$, and the point check.) 3. $\mathcal{D}$ is witness sampleable and the $\mathcal{D}$-Aff-MDH assumption holds.*

In Table 1 we compare our new extraction policies with the extraction policy of [FFK+23]. We stress that our hash check with linearized commitment is more permissive than their hash check constraint, therefore, our theorem is stronger. In the table, Q is the number of simulation queries and d is the maximum degree supported by the scheme.

For any set $\mathcal{I} \subseteq [n]$, let us denote with $\sigma_{\mathcal{I}}$ the $\mathcal{I}$-projection function, namely the function that takes as input a list $(a_1, \ldots, a_n)$ and returns the list $(a_i)_{i\in\mathcal{I}}$.

Theorem 1. *$\forall \mathcal{I} \subseteq [n]$, $\Pi_{\texttt{m-evl}}$ is $(\boldsymbol{\Phi}^{\texttt{adpt}}_{\texttt{m-evl},\mathcal{I}}, \sigma_{\mathcal{I}})$-SE under the OMSDH assumption and is $(\boldsymbol{\Phi}^{\texttt{s-adpt}}_{\texttt{m-evl},\mathcal{I}}, \sigma_{\mathcal{I}})$-SE under the $(Q_{\mathsf{sim}} + d)$-dlog assumption in the AGM.*

Proof intuition. We consider an algebraic adversary $\mathcal{A}$ whose forgery satisfies the extraction policy. In particular, the view is algebraic consistent, thus there exists a solution for the polynomial system of linear equations defined by the view. As the first important step of the proof, we simplify this system of equations and find alternative representations where each simulated proof depends either from one single simulated commitment or from one single simulated proof. This simplification allows rewriting the forged linearized commitment in the more manageable form $\mathsf{c}^* = [m_0(s)]_1 + \sum [\log(\mathsf{c}_i) \cdot m_i(s)]_1$ where c_i are the simulated commitments. Here, we can prove that $m_i(X) \equiv 0$ for $i > 0$. In fact, assume otherwise and assume the commitments are uniformly random[15], then we can break the representation problem finding $\log(\sum m_i(x^*)\mathsf{c}_i) = y^* - m_0(x^*)$ where the forgery of the adversary is (c^*, x^*, y^*).

We are still not done because $m_0(X)$ is a rational function of the form $f(X) - \sum A_i(X)(\sum_j o_j q_{i,j}(X))$ where f is the polynomial we would like to extract, the $q_{i,j}$ are rational functions whose degree is bounded by the maximum nesting level $\bar{\nu}$ and the o_j are the coefficients in the algebraic representation of c^* that depend on the simulated proofs material. If we assume that the forgery is valid then we would obtain $m_0(x^*) = y^*$, otherwise we could break the OMSDH assumption, moreover, we can show this case happens when $\sum A_i(x^*)(\sum_j o_j q_i(x^*)) = 0$ but, the extractor would still fail if there exists at least $o_j \neq 0$. Here we crucially use our hypothesis on $\bar{\nu}$-independence of the A_i to show that this cannot happen and thus all $o_j = 0$.

One might wonder if this last step is an artifact of our proof technique, and whether the independence is necessary. We show the latter is the case with an attack similar to the one presented in Section 2.3. The attack asks for a simulated proof on $(\mathsf{c}, 0, 1)$ for a simulated commitment c and sets the forged linearized commitment to $\mathsf{c}^* = \mathsf{c} - x^*\pi$ for an arbitrary evaluation point x^* and $y^* = 1$, the attack works because $\mathsf{c}^* - [1]_1 = \mathsf{c} - x^*\mathsf{c}/s - x^*/s + 1 = \pi(s - x^*)$. The formal polynomial associated to c^* would be of the form $0 - (1 - X \cdot \frac{1}{X}) + Z(1 - X \cdot \frac{1}{X})$ where Z is the formal variable associated to the simulated commitment, $o_1 = 1$ and $A_1(X) = 1$ and $A_2(X) = -X$ and where the latter polynomials are 1-linearly dependent.

[15] In our proof we consider the more general case where the simulated commitments are sampled from an Aff-MDH-secure distribution.

Proof (of Theorem 1). We recall that the set $\mathcal{I}$ contains the indexes i such that $\mathcal{E}$ needs to extract the witness polynomials f_i committed in c_i^*.

By the definition of algebraic adversary (cf. Definition 3) for each group element output, $\mathcal{A}$ additionally attaches a representation $(f, \boldsymbol{r})$ of such a group element with respect to all the elements seen during the experiment (included elements in coms and the simulated proofs). In particular, we assume that for each query $(\mathbb{x}, \mathsf{aux})$ to the oracle $\mathcal{S}_1$ we can parse the value aux as $((f_i, \boldsymbol{r}_i)_i, \mathsf{aux}')$, where $(f_i, \boldsymbol{r}_i)$ is a valid representation for $\mathbb{x}.\mathsf{c}_i$.

The adversary also encodes a polynomial $h(X)$ in aux_ϕ. The commitments c_i^* of the forgery come along with representation $(A_{i,r}(X), \mathsf{b}_{i,r})_r$, stored in $\mathsf{aux}_\mathcal{E}$; the adversary also stores $(f_{\mathsf{b}_{i,r}}, \boldsymbol{r}_{\mathsf{b}_{i,r}})$ to represent the group element $\mathsf{b}_{i,r}$. Namely, given the commitments coms and all the proofs $\mathsf{proofs}_\mathcal{A}$ output by $\mathcal{S}_1$, it holds that $\mathsf{c}_i^* = \sum_r A_{i,r}(x^*)(f_{\mathsf{b}_{i,r}}(s) + \langle \boldsymbol{r}_{\mathsf{b}_{i,r}}, \mathsf{coms} \| \mathsf{proofs}_\mathcal{A} \rangle)$.

Without loss of generality, we restrict the class of the algebraic adversaries that we consider. Given an algebraic adversary $\mathcal{A}$ we can define a new adversary $\mathcal{A}'$ such that:

- $\mathcal{A}'$ makes (single-eval) simulation queries, i.e., each statement $\mathbb{x}$ given as input to $\mathcal{S}_1$ can be parsed as (c, x, y)
- each commitment in $\mathbb{x}$ is a linear combination of simulated commitments and proofs, but not of elements of the SRS

The adversary $\mathcal{A}'$ runs internally $\mathcal{A}$ and forwards all its queries and answers to the simulation oracle in the following way:

- Upon query $\mathbb{x} := (x, (\mathsf{c}_i, y_i))$ to $\mathcal{S}_1$, with representation $(f_i, \boldsymbol{r}_i)$ such that $\mathsf{c}_i = [f_i(s)]_1 + \langle \boldsymbol{r}_i, \mathsf{coms} \| \mathsf{proofs}_\mathcal{A} \rangle$, $\mathcal{A}'$ queries n times the simulation oracle with $(\mathsf{c}_i - [f_i(s)]_1, x, y_i - f_i(x))$, receiving the proof π_i. Finally, returns the proof $\pi' := \sum_i \rho^{i-1}(\pi_i + \mathsf{Prove}_{\mathsf{evl}}(\mathsf{ek}, ([f_i(s)]_1, x, f_i(x)), f))$, where $\rho \leftarrow \mathsf{RO}(\mathsf{vk} \| \mathbb{x})$

By the homomorphic properties of $\Pi_{\mathtt{m\text{-}evl}}$, the correctness of the proofs readily holds.

We define our extractor $\mathcal{E}$ to be the extractor that returns, for all $i \in \mathcal{I}$ the polynomial $f_i(X) := \sum_r A_{i,r}(x^*) f_{\mathsf{b}_{i,r}}(X)$; this turns out to be equivalent to the canonical extractor in the AGM, because, as we show, the remaining entries in the representation sum up to zero.

We let $\mathbf{H}_0$ be the $\mathbf{Exp}^{(\Phi,\mathcal{F})\text{-}se}_{\mathcal{A},\mathcal{S},\mathcal{E}}$ experiment, and we denote by $\epsilon_i := \Pr[\mathbf{H}_i = 1]$.

Hybrid $\mathbf{H}_1$. We set $\mathbf{H}_1$ to be the same experiment but with the *alternative* adversary $\mathcal{A}'$ defined below:

1. The alternative adversary runs $\mathcal{A}$ forwarding all its queries until $\mathcal{A}$ sends its forgery. Let $\bar{\mathbb{x}} = (\bar{x}, (\bar{\mathsf{c}}_i, \bar{y}_i)_i), \bar{\pi}$ be its forgery. Let $\mathcal{Q}_x$ be the set of points x_j for which the adversary queried $\mathcal{S}_1$.
2. If $\bar{x} \in \mathcal{Q}_x$, namely when the adversary made a simulation query with evaluation point set to $\bar{x}$, then it finds values y_i such that $\mathbb{x} := (\bar{x}, (\bar{\mathsf{c}}_i, y_i)_i)$ is algebraic consistent with the view of the adversary, and queries the simulation oracle $\mathcal{S}_1$ with $\mathbb{x}'$ receiving back π. (Else it outputs $\bar{\mathbb{x}}, \bar{\pi}$.)
3. It computes the forgery $\mathbb{x}^* = (\mathsf{c}^*, x^*, y^*), \pi^*$, where:

$$\mathsf{c}^* \leftarrow (\bar{\pi} - \pi) \qquad \pi^* \leftarrow \frac{\bar{\pi} - \pi}{\bar{x} - x^*} \qquad y^* \leftarrow \frac{\sum_i \rho^{i-1}(\bar{y} - y)}{\bar{x} - x^*}$$

   the forgery point $x^* \leftarrow \mathsf{RO}(s)$, and s is a string never queried to the RO by $\mathcal{A}$ and containing c^* as substring, which yields $\mathsf{c}^* \rightarrow_{\mathsf{RO}} x^*$.
4. It aborts if $x^* \in \mathcal{Q}_x$, otherwise it outputs the forgery.

We show that, unless it occurs the bad event that $x^* \in \mathcal{Q}_x$, the forgery of the adversary $\mathcal{A}'$ is valid whenever the forgery of $\mathcal{A}$ is valid. First we notice, by the verification equation of KZG, that $(\bar{\pi} - \pi)(s - \bar{x}) = \sum_i \rho^{i-1} [y_i - \bar{y}_i]_1$. Thus:

$$\pi^*(s - x^*) = \tfrac{\bar{\pi}-\pi}{\bar{x}-x^*}(s - x^* + \bar{x} - \bar{x}) = \tfrac{\bar{\pi}-\pi}{\bar{x}-x^*}(-x^* + \bar{x}) + \tfrac{\bar{\pi}-\pi}{x^*-x}(s - \bar{x}) = \mathsf{c}^* - [y^*]_1 \tag{1}$$

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

Hybrid $\mathbf{H}_2$. Recall that $\mathcal{D}$ is witness sampleable, thus according to Definition 4 there exists a PPT algorithm $\tilde{\mathcal{D}}$ associated with the sampler $\mathcal{D}$. The hybrid $\mathbf{H}_2$ is identical to the previous one, but the group elements in coms are sampled "at the exponent", i.e., we use $\tilde{\mathcal{D}}$ to generate the field elements $\boldsymbol{\gamma}$, and we let $\mathsf{coms} \leftarrow [\boldsymbol{\gamma}]_1$. By the witness sampleability of $\mathcal{D}$, $\mathbf{H}_1 \equiv \mathbf{H}_2$, thus $\epsilon_2 = \epsilon_1$.

Hybrid $\mathbf{H}_3$. In this hybrid we add some more entries to the list of simulated proofs $\mathcal{Q}_{\mathsf{sim}}$.

The experiment runs $\mathcal{A}$ until completion. Since the view of the adversary is algebraic consistent, we can define a set of polynomial equations that admits solutions derived from the simulation queries of $\mathcal{A}$. Let $(z_i(X))_{i\in[Q_{\mathsf{sim}}]}$ be a reduced solution (Definition 15) for this set of polynomial equations. The experiment submits additional queries to $\mathcal{S}_1$ as follows. First, for all j, k, define the value $\bar{\nu}_{j,k} \coloneqq \max_\pi \nu_\pi(j,k)$.

Then $\forall j,k$ such that $\bar{\nu}_{j,k} \neq 0$, does the following. It queries $\mathcal{S}_1$ with $(\mathsf{coms}_j, x_k, z_j(x_k))$, and let $\bar{\pi}_{j,k,1}$ be the output of $\mathcal{S}_1$ on each of these queries; then, for $l \in [\bar{\nu}_{j,k} - 1]$, obtains the proof $\bar{\pi}_{j,k,l+1}$ on the statement $(\bar{\pi}_{j,k,l}, x_k, q_{j,k,l}(x_k))$, where: $q_{j,k,1}(X) \coloneqq z_j(X)$ and $\forall l > 0$ the polynomial $q_{j,k,l+1}(X) \coloneqq (q_{j,k,l}(X) - q_{j,k,l}(x_k))(X - x_k)^{-1}$. We notice all these additional proofs are of the form:

$$\bar{\pi}_{j,k,l} = \left[\frac{\gamma_j - \sum_{l'\in[l]}(s-x_k)^{l'-1} q_{j,k,l'}(x_k)}{(s-x_k)^l}\right]_1 \tag{2}$$

We call them "core" proofs, as they will play an important role in the next hybrid, and we denote with proofs the vector of all the core proofs, to distinguish them from the adversary's proofs $\mathsf{proofs}_{\mathcal{A}}$, namely the set of simulated proofs requested by the adversary. These additional simulation queries are algebraic consistent w.r.t. the view of $\mathcal{A}$: in particular, $\forall j,k$ each proof $\bar{\pi}_{j,k,1}$ is an evaluation proof for the commitment coms_j on the point x_k and the evaluation value $z_j(x_k)$, which by definition of the polynomials $z_j(X)$ is algebraic consistent. For all $l > 1$ the proof $\bar{\pi}_{j,k,l}$ is a proof on the point x_k for a commitment that is a quotient derived from $z_j(X)$, and the evaluation value is chosen to be consistent with it. The change introduced in this hybrid does not alter the winning probability of $\mathcal{A}$, hence $\epsilon_3 = \epsilon_2$.

Hybrid $\mathbf{H}_4$. In this hybrid we change the representation of the forgery of $\mathcal{A}$. In particular, once $\mathcal{A}$ has submitted the (successful) forgery $(\mathbb{x}^*, \pi^*)$, attaching its representation, we replace it with new coefficients that only depend on coms, and proofs, but not on the adversary's proofs $\mathsf{proofs}_{\mathcal{A}}$.

The change introduced in this hybrid is only syntactical and does not alter the winning probability of $\mathcal{A}$, as we show hereafter.

Lemma 2. $\epsilon_4 = \epsilon_3$

Proof. We give a recursive procedure that rewrites the algebraic representations of all the Q_{sim} adversary's proofs $\mathsf{proofs}_{\mathcal{A}}$ in the base defined by proofs. We prove by induction on the number of simulation queries made by the adverary the correctness of the procedure.

Base. Let π be the first proof computed by $\mathcal{S}_1$, for an instance (c, x_{k^*}, y), where the commitment $\mathsf{c} = \sum_j c_j [\gamma_j]_1$. We have that, by the correctness of the proof, $\pi(s - x_{k^*}) = \sum_j c_j [\gamma_j]_1 - y$. By algebraic consistency, we have that there exists an equation of the form $\sum_j (c_j Y_j(x_{k^*})) = y$ with variables (the coefficients of) the polynomials Y_j, and the list of polynomials $(z_j(X))_j$ is a reduced solution by the change introduced in $\mathbf{H}_3$. We derive that π is equal to:

$$\left[\frac{\sum_j c_j(\gamma_j - z_j(x_{k^*}))}{s - x_{k^*}}\right]_1 = \left[\frac{\sum_j c_j(\gamma_j - q_{j,k^*,1}(x_{k^*}))}{s - x_{k^*}}\right]_1 = \sum_j c_j \bar{\pi}_{j,k*,1}$$

Inductive Step. Let now assume that all the first t proofs can be expressed as linear combination of elements of proofs. We show that also the $(t+1)$-th proof can be written in the same way.

Let (c, x_{k^*}, y) be the $(t+1)$-th (valid) query submitted to $\mathcal{S}_1$, where $\mathsf{c} = \sum_j c_j [\gamma_j]_1 + \sum_{j\le t} o_j \pi_j$ and $\mathsf{proofs}_{\mathcal{A}} = (\pi_j)_{j\in[Q_{\mathsf{sim}}]}$. By induction, there exist coefficients $o'_{j,k,l}$ such that: $\mathsf{c} = \sum_j c_j [\gamma_j]_1 + \sum_{j,k,l} o'_{j,k,l} \bar{\pi}_{j,k,l}$. The proof π computed by $\mathcal{S}_1$ (we set $\pi_{t+1} \coloneqq \pi$) is such that $\pi(s - x_{k^*}) = \mathsf{c} - [y]_1$. Let $\pi = [p]_1$, then we

have that:

$$p = \frac{1}{s-x_{k^*}} \left(\sum_j c_j \gamma_j + \sum_{j,k,l} o'_{j,k,l} \frac{\gamma_j - \sum_{l' \in [l]} (s-x_k)^{l'-1} q_{j,k,l'}(x_k)}{(s-x_k)^l} - y \right)$$

Also, $y = \sum_j c_j z_j(x_{k^*}) + \sum_{j,k,l} o'_{j,k,l} q_{j,k,l}(x_{k^*})$ by algebraic consistency, and it can be expanded as:

$$\sum_j c_j z_j(x_{k^*}) + \overbrace{\sum_{j,k \neq k^*,l} o'_{j,k,l} \frac{z_j(x_{k^*}) - \sum_{l' \in [l]} (x_{k^*}-x_k)^{l'-1} q_{j,k,l'}(x_k)}{(x_{k^*}-x_k)^l}}^{\bar{y}} + \sum_{j,l} o'_{j,k^*,l} q_{j,k^*,l}(x_{k^*})$$

We first notice that, by plugging the equation above, we can rewrite p as:

$$\sum_j c_j \frac{\gamma_j - z_j(x_{k^*})}{s-x_{k^*}} + r + \sum_{j,l} o'_{j,k^*,l} \frac{\gamma_j - \sum_{l' \in [l+1]} (s-x_{k^*})^{l'-1} q_{j,k^*,l'}(x^*_k)}{(s-x_{k^*})^{l+1}}$$

where $r = \frac{1}{s-x_{k^*}} \left(\sum_{j,k \neq k^*,l} o'_{j,k,l} \frac{\gamma_j - q_{j,k,l}(x_k)}{s-x_k} - \bar{y} \right)$. Note that the first and the third addends of $[p]_1$ are linear combination of elements in proofs. The only thing left to prove is that $[r]_1$ can be written as linear combination of elements of proofs.

Let $N_{j,k,l} \coloneqq \gamma_j - \sum_{l' \in [l]} (s-x_k)^{l'-1} q_{j,k,l'}(x_k)$. For all j, k, l, we have that:

$$\begin{aligned} \frac{\bar{\pi}_{j,k,l}}{s-x_{k^*}} &= \frac{[N_{j,k,l}]_1}{(s-x_k)^l (s-x_{k^*})} \\ &= [N_{j,k,l}]_1 \left(\sum_{\ell=0}^{l-1} \underbrace{(-1)^\ell \frac{(x_k - x_{k^*})^{-(\ell+1)}}{(s-x_k)^{l-\ell}}}_{\alpha_{j,k,l,\ell}} + \underbrace{(-1)^l \frac{(x_k - x_{k^*})^{-l}}{(s-x_{k^*})}}_{\beta_{j,k,l}} \right) \end{aligned}$$

Also, for all $\ell \in [0, l-1]$, we have that $[N_{j,k,l}]_1 \, \alpha_\ell$ is equal to:

$$(-1)^\ell \bar{\pi}_{j,k,l-\ell} + (-1)^{\ell+1} (x - x_{k^*})^{\ell-1} \sum_{l' \in [\ell]} [q_{j,k,l'}(x_k)]_1 \, (s-x_k)^{l'-1}$$

Also, we have that $[N_{j,k,l}]_1 \, \beta$ is equal to:

$$\begin{aligned} (-1)^l \bar{\pi}_{j,k^*,l} + (-1)^{l+1} \left[\frac{z_j(x_{k^*}) - \sum_{l' \in [l]} (s-x_k)^{l'-1} q_{j,k,l'}(x_k)}{s-x_{k^*}} \right]_1 = \\ (-1)^l \bar{\pi}_{j,k^*,l} + \left[\frac{z_j(x_{k^*}) - \sum_{l' \in [l]} (x_k - x_{k^*})^{l'-1} q_{j,k,l'}(x_k)}{(s-x_{k^*})(x^*_k - x_k)^l} \right]_1 = \\ + \sum_{\ell=0}^{l-1} (-1)^\ell (x - x_{k^*})^{\ell-1} \sum_{l' \in [\ell]} [q_{j,k,l'}(x_k)]_1 \, (s-x_k)^{l'-1} \end{aligned}$$

Using the above equations, we can write $[r]_1$ as:

$$\sum_{j,k\neq k^*,l} o'_{j,k,l} \frac{[N_{j,k,l}]_1}{(s-x_k)^k(s-x_{k^*})} - \frac{[\bar{y}]_1}{s-x_{k^*}} =$$

$$\sum_{j,k\neq k^*,l} o'_{j,k,l} [N_{j,k,l}]_1 \left(\sum_{\ell=0}^{l-1} \alpha_{j,k,l,\ell} + \beta_{j,k,l}\right) - \frac{[\bar{y}]_1}{s-x_{k^*}} =$$

$$\sum_{j,k\neq k^*,l} o'_{j,k,l} \left(\sum_{\ell=0}^{l-1} (-1)^\ell \bar{\pi}_{j,k,l-\ell} + (-1)^l \bar{\pi}_{j,k^*,l}\right) + \frac{[\bar{y}]_1}{s-x_k^*} - \frac{[\bar{y}]_1}{s-x_{k^*}} =$$

$$\sum_{j,k\neq k^*,l} o'_{j,k,l} \left(\sum_{\ell=0}^{l-1} (-1)^\ell \bar{\pi}_{j,k,l-\ell} + (-1)^l \bar{\pi}_{j,k^*,l}\right)$$

□

Before moving to the next hybrid, we set some notation. First, $\forall i, r$, let parse $\boldsymbol{r}_{\mathsf{b}_{i,r}} = \boldsymbol{c}_{i,r} \| \boldsymbol{o}_{i,r}$. From the definition of $\mathbf{H}_4$, we have that $\mathsf{c}_i^* = [f_i(s)]_1 + \sum_r A_{i,r}(x^*)(\langle \boldsymbol{c}_{i,r}, \mathsf{coms}\rangle + \langle \boldsymbol{o}_{i,r}, \mathsf{proofs}\rangle)$

From the change introduced in $\mathbf{H}_3$, $\mathcal{S}_1$ outputs "core" proofs $\bar{\pi}_{j,k,l}(s, \mathsf{coms}) \in \mathsf{proofs}$, where $\bar{\pi}_{j,k,l}(X, \boldsymbol{Y}) \coloneqq (Y_j - \sum_{l' \in [l]} (X - x_k)^{l'-1} q_{j,k,l'}(x_k))(X - x_k)^{-l}$.

By the guarantees of the AGM, for all $i \in [n]$ we can write $\mathsf{c}_i^* = [c_i^*(s, \mathsf{coms})]_1$, where $c_i^*(X, \boldsymbol{Y})$ is equal to:

$$\sum_r A_{i,r}(x^*) f_{\mathsf{b}_{i,r}}(X) + \sum_r A_{i,r}(x^*) \underbrace{\sum_j (c_{i,r,j} Y_j + \sum_{k,l} o_{i,r,j,k,l} \pi_{j,k,l}(X, \boldsymbol{Y}))}_{B_{i,r}(X,Y)} \tag{3}$$

and, if the verification equation is satisfied, we have that:

$$v(s) = \pi^*(s - x^*)$$

where $v(X) \coloneqq \sum_i \rho^{i-1}(c_i^*(X, \mathsf{coms}) - y_i^*)$.

Hybrid $\mathbf{H}_5$. This hybrid is equal to $\mathbf{H}_4$ but it returns 0 if there exists $i \in [n]$ such that $c_i^*(x^*, \mathsf{coms}) \neq y_i^*$.

Lemma 3. $\epsilon_5 \leq \epsilon_4 + \epsilon_{\mathsf{OMSDH}} + n/q$

Proof. If there exists $i \in [n]$ such that $c_i^*(x^*, \mathsf{coms}) \neq y_i^*$, with overwhelming probability $1 - n/q$, we have that $v(x^*) \neq 0$ because, by the hash ckeck, ρ is chosen uniformly at random after the polynomials $c_i^*(X, \boldsymbol{Y})$ are determined. Then, we can make a forgery to the OMSDH[16] assumption as follows.

The reduction gives the adversary the same SRS generated by the OMSDH challenger. The oracle $\mathcal{O}_s$ allows the reduction to compute the proofs for any statement $\mathbb{x} \coloneqq (\mathsf{c}, x, y)$, where c is a linear combination of elements of the SRS, coms and previously seen simulated proofs. As shown in the previous hybrid, a proof for $\mathbb{x}$ can be computed using group elements of the form $[s - x_k]_1^{-l}$ (that can be retrieved using $\mathcal{O}_s$), the coefficients γ_j and the algebraic representation of c, which is all known to the reduction. Let $q(X), r$ be such that $v(X) = q(x) + r(X - x^*)$. The reduction submits the forgery (x^*, y^*), where $\mathsf{y}^* \coloneqq r^{-1}(\pi^* - [q(s)]_1)$. This is a valid forgery because y^* is equal to $\left[(s - x^*)^{-1}\right]_1$ and x^* was never queried to $\mathcal{O}$ by the change introduced in $\mathbf{H}_1$. □

Hybrid $\mathbf{H}_6$. Let $\mathbf{H}_6$ return 0 if there is an index $i \in \mathcal{I}$ such that f_i extracted by $\mathcal{E}$ is not a valid witness.

Lemma 4. $\epsilon_6 \leq \epsilon_5 + |\mathcal{I}| \epsilon_{\mathsf{Aff\text{-}MDH}} + \mathsf{deg}(h)(\sum_{i \in \mathcal{I}} \max_r \mathsf{deg}(A_{i,r}) + \bar{\nu})/q$

[16] When the policy is semi-adaptive, we can reduce to dlog because of Theorem 5.

Proof. We prove it through a series of n hybrids. Let $\mathbf{H}_{6,0} \equiv \mathbf{H}_5$, and let $\mathbf{H}_{6,i}$ be the same as $\mathbf{H}_{6,i-1}$ and if $i \in \mathcal{I}$ it additionally returns 1 if f_i is not a valid witness. Clearly, for $i \notin \mathcal{I}$, it holds that $\epsilon_{6,i} = \epsilon_{6,i-1}$.

For $i \in \mathcal{I}$, let E_i be the event that $\sum_r A_{i,r}(x^*)B_{i,r}(X) \equiv 0$.

Case 1. We show that $\Pr[\mathbf{H}_{6,i} = 1 \wedge E_i] = 0$. We recall that the extractor $\mathcal{E}$ returns the polynomial $f_i(X) \coloneqq \sum_r A_{i,r}(x^*) f_{\mathsf{b}_{i,r}}(X)$. Conditioning on E_i, we have that $c_i^*(X, \boldsymbol{Y}) = f_i(X)$, and $\mathsf{c}_i^* = [f_i(s)]_1$. $\mathcal{E}$ returns a valid witness if $f_i(x^*) = y_i^*$, which is enforced by the check introduced in $\mathbf{H}_5$.

Case 2. We show that $\Pr[\mathbf{H}_{6,i} = 1 \wedge \neg E_i] \leq \epsilon_{\mathsf{Aff\text{-}MDH}} + \mathsf{deg}(h)(\bar{\nu} + \max_r \mathsf{deg}(A_{i,r}))/q$. First, it must be that there exist indexes r^*, j^*, k^*, l^* such that either $c_{i,r^*,j^*} \neq 0$ or $o_{i,r^*,j^*,k^*,l^*} \neq 0$, as otherwise $B_{i,r} \equiv 0, \forall r$.

Let $\hat{c}_i(X, \boldsymbol{Y}) \coloneqq m_0(X) + \sum m_j(X) Y_j$ be a multi-linear polynomial with coefficient in $\mathbb{F}_{\leq q}(X)$ where:

$$m_0(X) = f_i(X) - \sum_r A_{i,r}(X) \sum_{j,k,l} o_{i,r,j,k,l} \frac{\sum_{l' \in [l]} (X - x_k)^{l'-1} q_{j,k,l'}(x_k)}{(X - x_k)^l}$$

$$m_j(X) = \sum_r A_{i,r}(X) \underbrace{(c_{i,r,j} + \sum_{k,l} \tfrac{o_{i,r,j,k,l}}{(X-x_k)^l})}_{m_{j,r}(X)}, \ \forall j > 0$$

Notice that by definition we have that: $\hat{c}_i(x^*, \boldsymbol{Y}) = c_i^*(x^*, \boldsymbol{Y})$.

$\forall j$ let $p_j(X) \coloneqq \prod_k (X - x_k)^{\bar{\nu}_{j,k}}$, and notice that the set $\{p_j(x)\} \cup \{\frac{p_j(x)}{(X-x_k)^l}\}_{k,l}$ is an independent set of polynomials w.r.t $\mathbb{F}$ and $m_{j,k}(X) \cdot p_j(X)$ is in the span of such a set of polynomials, thus, because of the condition of Case 2, $m_{j^*,r^*}(X) \neq 0$.

Let $\nu_j^* \coloneqq \sum_k \bar{\nu}_{j,k}$. By definition, $\nu_j^* \leq \bar{\nu}$ that we recall is equal to $\max_j \sum_k \bar{\nu}_{j,k}$. Because of the *Hash check*, we have that $\{A_{i,r}\}_r$ are $\bar{\nu}$-independent polynomials; moreover, by Lemma 1 there is a morphism between the span of the set $\{1\} \cup \{(X - x_k)^{-l}\}_{k,l \in \bar{\nu}_{j,k}}$ and $\mathbb{F}_{\leq \nu_j^*}[X]$. Thus we conclude that $m_{j^*}(X) \neq 0$.

Since $c_i^*(x^*, \mathsf{coms}) = [y^*]_1$ by the check introduced in $\mathbf{H}_5$, and $c_i^*(x^*, \mathsf{coms}) = \hat{c}_i(x^*, \mathsf{coms})$ by definition, we can reduce to Aff-MDH as follows. The reduction generates the SRS and simulates using the trapdoor s, while the commitments coms are received by the Aff-MDH challenger[17]. The reduction outputs $((\mu_j)_j, \hat{y})$, where the coefficients $\mu_j \leftarrow m_j(x^*)$ and $\hat{y} = y^* - m_0(x^*)$.

Given that the *Hash check* is satisfied, $((\mathsf{b}_{i,r})_r; (A_{i,r})_r; h) \to_{\mathsf{RO}} a$, and $h(a) = x^*$, which implies that c_i^* is a function of the coefficients $c_{i,r,j}, o_{i,r,j,k,l}$ and the polynomials $A_{i,r}$ that are fixed before a (and hence x^*) is computed. By Schwartz-Zippel, we derive that the coefficient $\mu_{j^*} = m_{j^*}(x^*)$ is null only with negligible probability $\mathsf{deg}(h)(\bar{\nu} + \max_r \mathsf{deg}(A_{i,r}))/q$. □

Finally, we notice that in $\mathbf{H}_6$, $\mathcal{E}$ successfully extracts all the witness polynomials f_i, for $i \in \mathcal{I}$. Then we conclude that $\epsilon_6 = 0$. □

5.2 Simulation Extractability of the Linearization Trick

In this section we formalize the linearization trick for KZG commitments [GWC19,OL] as a CP-SNARK for the relation $\mathcal{R}_{\mathsf{lin}}$ that upon instance:

$$\mathbb{x} \coloneqq ((\mathsf{c}_j)_{j \in [m]}, (\mathsf{b}_i)_{i \in [n]}, (G_i)_{i \in [n]}, x, y),$$

whose witness $\mathbb{w} = (C_j)_{j \in [m]}, (B_i)_{i \in [n]}$ are polynomials committed in the instance, and that outputs 1 if and only if

$$\sum_{i=1}^{n} A_i(x) B_i(x) = y,$$

and where $A_i(X) \coloneqq G_i((C_j(X))_j, X)$ with $G_i \in \mathbb{F}[X_1, \ldots, X_m, X]$.

[17] In particular, this means that the commitments are sampled from $\mathcal{D}$, which is identically distributed to $\tilde{\mathcal{D}}$, as argued in $\mathbf{H}_2$.

We call the polynomials C_j (resp. commitments c_j) the *core polynomials* (resp. commitments); moreover, we call the polynomials A_i and B_i (resp. the commitments b_i) the *left* and *right polynomials* (resp. commitments).

We define Π_{lin} that uses $\Pi_{\mathtt{m\text{-}evl}}$ as inner scheme:

$\mathsf{Prove}_{\mathsf{lin}}(\mathsf{ek}, \mathbb{x}, \mathbb{w})$**:** compute $\pi_{\mathtt{m\text{-}evl}} \leftarrow \mathsf{Prove}_{\mathtt{m\text{-}evl}}(\mathbb{x}_{\mathtt{m\text{-}evl}}, ((C_j)_j, R))$, where $R(X) := \sum_i A_i(x)B_i(X)$, $\mathsf{r} := \sum_i A_i(x)\mathsf{b}_i$, and $\mathbb{x}_{\mathtt{m\text{-}evl}} := (x, (\mathsf{c}_j, C_j(x))_j, (\mathsf{r}, y))$. Output $\pi := (\pi_{\mathtt{m\text{-}evl}}, (C_j(x))_j)$

$\mathsf{Verify}_{\mathsf{lin}}(\mathsf{vk}, \mathbb{x}, \pi)$**:** parse π as $(\pi_{\mathtt{m\text{-}evl}}, (y_j)_j)$, compute r as $\sum_i G_i((y_j)_j, x)\mathsf{b}_i$. Output $\mathsf{Verify}_{\mathtt{m\text{-}evl}}(\mathsf{vk}, \mathbb{x}_{\mathtt{m\text{-}evl}}, \pi_{\mathtt{m\text{-}evl}})$, where $\mathbb{x}_{\mathtt{m\text{-}evl}} := (x, ((\mathsf{c}_j, y_j)_j, (\mathsf{r}, y)))$

This scheme is not zero-knowledge as the proofs leak some information on the witness, that are the values $y_j = C_j(x)$. Formally, it achieves L_{lin}-leaky zero-knowledge where $L_{\mathsf{lin}}(\mathbb{x}, \mathbb{w}) := (\mathbb{w}.C_j(\mathbb{x}.x))_j$. We define the simulator $\mathcal{S} := (\mathcal{S}_0, \mathcal{S}_1)$, where $\mathcal{S}_0$ outputs the trapdoor information s together with the srs, and $\mathcal{S}_1$ simulates proofs for $\mathbb{x} := ((\mathsf{c}_j)_{j\in[m]}, (\mathsf{b}_i)_{i\in[n]}, (G_i)_{i\in[n]}, x, y)$ and leakage $(y_j)_{j\in[m]}$ computing $\pi_{\mathtt{m\text{-}evl}} := (s - x)^{-1}(\sum_j \rho^{i-1}(\mathsf{c}_j - [y_j]_1) + \rho^m(\mathsf{r} - y))$, where $\rho := \mathsf{RO}(\mathsf{vk}\|x, (\mathsf{c}_j, C_j(x))_j, (\mathsf{r}, y))$, $\mathsf{r} := \sum_i A_i(x)\mathsf{b}_i$ and outputs the proof $\pi := (\pi_{\mathtt{m\text{-}evl}}, (y_j)_j)$.

The extraction policy. Let $\Phi_{\mathsf{lin}}^{\mathcal{J},\nu}$ be the policy parametrized by $\nu \in \mathbb{N}$ and $\mathcal{J} \subseteq [n]$, for $n \in \mathbb{N}$, described below:

Hash Check (for the linearization trick): parse the forged instance $\mathbb{x}^* := ((\mathsf{c}_j^*)_j, (\mathsf{b}_i^*)_i, (G_i^*)_i, x^*, y^*)$, return 1 if and only if there exists a polynomial h such that:
- $((\mathsf{c}_j^*)_j, (\mathsf{b}_i^*)_i; (G_i^*)_i, h) \to_{\mathsf{RO}} a$ and $h(a) = x^*$;
- $\forall j : \nu > \sum_k \max_{\pi\in\mathsf{proofs}} \nu_\pi(j,k)$ where proofs is the list of simulated proofs.

Partial-Extraction Check: parse $\mathsf{aux}_{\mathcal{E}}$, find polynomials $(B_i^*)_{i\in\mathcal{J}}$ and return 1 iff b_i^* commits to B_i^*, $\forall i \in \mathcal{J}$.

Definition 18. *Let $\boldsymbol{\Phi}_{\mathsf{lin}}^{\mathcal{J},\nu}$ be the set of policies $\Phi_{\mathcal{D}} = (\Phi_0^{\mathcal{D}}, \Phi_{\mathsf{lin}}^{\mathcal{J},\nu})$ for a distribution $\mathcal{D}$ where:*

- *$\Phi_0^{\mathcal{D}}$ on input group parameters $\mathsf{pp}_{\mathbb{G}}$ outputs $\mathsf{pp}_\Phi := \mathsf{coms}$, where coms is a vector of commitments sampled from $\mathcal{D}$.*
- *$\mathcal{D}$ is witness sampleable and the $\mathcal{D}$-Aff-MDH assumption holds.*

The *Partial-Extraction Check* allows to define the concept of *partial extractability* (see [BCF+21,CFH+22]) within the framework of Φ-simulation extractability. The definition of partial extractability allows the adversary to provide to the extractability experiment one part of the witness, while the extractor must find the remaining part. Looking ahead, this check allows to define more flexible notions of extractability, for example, PLONK's verifier needs to check two linearization trick instances on a non-disjunct set of polynomials, thus we can partition the polynomials to extract between the two instances and, in doing so, we can loosen the independence requirements from the two instances. We give more details in Section 6.4.

To formalize the extractability of the linearization trick we crucially rely on the framework of $\mathcal{F}$-extractability. In particular, we consider the function $\mathcal{F}_{\mathcal{J},\nu}(\mathbb{w})$, for parameters $\mathcal{J} \subseteq [n]$ and $\nu \in \mathbb{N}$, that parses $\mathbb{w}$ as $(C_j)_j, (B_i)_i$, computes for all i the polynomial $A_i(X) := G_i((C_j(X))_j, X)$, and outputs $\mathbb{w}$ if $(A_i)_{i\notin\mathcal{J}}$ are ν-independent, otherwise outputs only $(C_j^*)_j$.

The $\mathcal{F}_{\mathcal{J},\nu}$-extractability and the *Hash Check* go hand in hand, the former specifies the condition under which extraction of the right polynomials can happen while the latter sets the rules, for the adversary, so that such condition holds.

Theorem 2. *For any $n, \nu \in \mathbb{N}, \mathcal{J} \subseteq [n]$, Π_{lin} is $(\boldsymbol{\Phi}_{\mathsf{lin}}^{\mathcal{J},\nu}, \mathcal{F}_{\mathcal{J},\nu})$-simulation-extractable in the AGM under the OMSDH assumption.*

Proof Intuition. Thanks to the heavy lifting of Theorem 1 the proof of Theorem 2 is not much different than a proof of (standard) extractability in the AGMOS [LPS23] would be. In fact, the proof can be summarized as two direct reductions to the SE of $\Pi_{\mathtt{m\text{-}evl}}$. In the first reduction, which is almost straight-forward, we show

how to extract the core polynomials. On the other hand, the second reduction needs a careful analysis as, in fact, the extractor of $\Pi_{\mathtt{m\text{-}evl}}$ extracts $R(X) = \sum A_i(x^*)B_i(X)$ while we need to show how to extracts the polynomials $(B_i(X))_i$. For simplicity, assume that the adversary obtains an obliviously sampled element c, thus we can write $\mathsf{b}_i = [B_i(s)]_1 + \bar{B}_i(s) \cdot \mathsf{c}$. We need to show that $\bar{B}_i \equiv 0$, and we can assume, thanks to the SE of $\Pi_{\mathtt{m\text{-}evl}}$, that $\sum A_i(x^*)\bar{B}_i(X) \equiv 0$. In proving knowledge extractability, we can just rely on the linear independence of the polynomials A_i and the Schwartz-Zippel lemma, for simulation extractability we additionally use the ν-independence and the second item of the Hash Check property.

Proof. We define our extractor $\mathcal{E}_{\mathcal{J},\nu}$ to be the extractor that, by looking at the algebraic representations, returns the polynomials $C_j(X) \coloneqq f_{\mathsf{c}_j}(X)$ for all $j \in [m]$, computes the polynomials $A_i(X)$, and if $\{A_i\}_{i\notin\mathcal{J}}$ are ν-independent, it additionally returns $B_i(X) \coloneqq f_{\mathsf{b}_i}(X)$ for all $i \in [n]$.

We let $\mathbf{H}_0$ be the $\mathbf{Exp}^{\mathcal{J},\nu\text{-}se}_{\mathcal{A},\mathcal{S},\mathcal{E}}$ experiment, and we denote by $\epsilon_i \coloneqq \Pr[\mathbf{H}_i = 1]$.

Hybrid $\mathbf{H}_1$. This hybrid is the same as $\mathbf{H}_0$ and it returns 0 if there exists $j \in [m]$ such that $\mathsf{c}_j \neq [C_j(s)]_1$ or $C_j(x^*) \neq y^*_j$.

Lemma 5. $\epsilon_1 \leq \epsilon_0 + \epsilon_{\mathtt{m\text{-}evl}}$

Proof. Recall that $\sigma_{\mathcal{I}}$ is the $\mathcal{I}$-projection function. We reduce to the $(\boldsymbol{\Phi}^{\mathtt{adpt}}_{\mathtt{m\text{-}evl}}, \sigma_{[m]})$-simulation extractability of $\Pi_{\mathtt{m\text{-}evl}}$. We recall that the extractor of the experiment does only guarantee (if the policy is satisfied) to extract the first m polynomials.

The reduction $\mathcal{B}$ takes as input the SRS and the commitments coms from the challenger and forwards them to $\mathcal{A}$. It trivially answers the queries of $\mathcal{A}$:

- **RO-query:** Proxy the query to $\mathcal{S}_2$.
- **SIM-query:** On input an instance $\mathbb{x}_{\mathsf{lin}}$, and leakage $(y_j)_{j\in[m]}$ define the multi-eval instance $\mathbb{x}_{\mathtt{m\text{-}evl}}$ as the honest prover would do, and query $\mathcal{S}_1$ on it.

Upon forgery $(\mathbb{x}^*, \pi^*)$ from $\mathcal{A}$, where $\mathbb{x}^* = (x^*, (\mathsf{c}_j)_j, (\mathsf{b}_i)_i, y^*)$, and $\pi^* = (\pi^*_{\mathtt{m\text{-}evl}}, (y^*_j)_j)$, $\mathcal{B}$ returns a multi-eval forgery $(\mathbb{x}^*_{\mathtt{m\text{-}evl}}, \pi^*_{\mathtt{m\text{-}evl}})$ where the statement is $\mathbb{x}^*_{\mathtt{m\text{-}evl}} = (x^*, ((\mathsf{c}_j)_j, \mathsf{r}), ((y^*_j)_j, y^*))$, with $\mathsf{r} = \sum_i G_i((y_j)_j, x)\mathsf{b}_i$, and $\pi^*_{\mathtt{m\text{-}evl}}$ is the proof in π^*.

Notice that this forgery passes the *Hash check* predicate for $\Pi_{\mathtt{m\text{-}evl}}$ when the forgery of $\mathcal{A}$ passes the *Hash check* for Π_{lin}: in particular, for some polynomial h we have that, for all $j \in [m]$, $(\mathsf{c}_j; h) \to_{\mathsf{RO}} a$, and $h(a) = x^*$. We have that the (canonical) extractor for $\Pi_{\mathtt{m\text{-}evl}}$ would succesfully extract, unless with probability $\epsilon_{\mathtt{m\text{-}evl}}$, all the witness polynomials $C_j(X)$ associated with the commitments c_j and such that $C_j(x^*) = y^*_j$. □

Hybrid $\mathbf{H}_2$. This hybrid is the same as $\mathbf{H}_1$, except it returns 0 if $\{A_i\}_{i\notin\mathcal{J}}$ are ν-independent polynomials and:

- $\sum_{i\in[n]} A_i(x^*)B_i(x^*) \neq y^*$
- or there exists $i \in [n]$ such that $\mathsf{b}_i \neq [B_i(s)]_1$

Lemma 6. $\epsilon_2 \leq \epsilon_1 + \epsilon_{\mathsf{Aff\text{-}MDH}} + \epsilon_{\mathtt{m\text{-}evl}} + \frac{\deg(h)(\max_i \deg(A_i)+\bar{\nu})}{q}$

Proof. Notice that for all $i \in \mathcal{J}$, $\mathsf{b}_i = [B_i(s)]_1$, where the polynomials $(B_i)_{i\in\mathcal{J}}$ are output by the adversary itself, by definition of the *Partial-Extraction Check* in the extraction policy. If the distinghuishing event occurs, namely $\{A_i\}_{i\notin\mathcal{J}}$ are ν-independent and $\sum_{i\in[n]} A_i(x^*)B_i(x^*) \neq y^*$ or $\mathsf{b}_i \neq [B_i(s)]_1$ for $i \notin \mathcal{J}$, we can reduce to the simulation extractability of $\Pi_{\mathtt{m\text{-}evl}}$ as follows.

The reduction $\mathcal{B}$ simulates the experiment for $\mathcal{A}$ as in the reduction described in Lemma 5. Then, upon forgery $(\mathbb{x}^*, \pi^*)$ from $\mathcal{A}$, where $\mathbb{x}^* = ((\mathsf{c}_j)_j, (\mathsf{b}_i)_i, (G^*_i)_i, x^*, y^*)$, and $\pi^* = (\pi^*_{\mathtt{m\text{-}evl}}, (y^*_j)_j)$, $\mathcal{B}$ computes the values:

$$q_j \coloneqq \left[(C_j(s) - C_j(x^*))(s - x^*)^{-1}\right]_1, \forall j \in [m]$$
$$q_{m+1} \coloneqq \sum_{i\in\mathcal{J}} \left[(A_i(s)B_i(s) - A_i(x^*)B_i(x^*))(s - x^*)^{-1}\right]_1.$$

The adversary $\mathcal{B}$ sets as forgery the statement $\mathbb{x}'_{\texttt{m-evl}} \coloneqq (x^*, \mathsf{r}', y')$ and proof $\pi'_{\texttt{m-evl}}$, where:

$$\mathsf{r}' \leftarrow \sum_{i\in[n]} A_i(x^*)\mathsf{b}_i - \sum_{i\in J} [A_i(s)B_i(s)]_1$$
$$y' \leftarrow y^* - \sum_{i\in\mathcal{J}} A_i(x^*)B_i(x^*)$$
$$\pi'_{\texttt{m-evl}} \leftarrow \rho^{-m}(\pi^*_{\texttt{m-evl}} - \sum_{j\in[m+1]} \rho^{j-1} q_j)$$

and $\rho \leftarrow \mathsf{RO}(\mathsf{vk}\|\mathbb{x}^*_{\texttt{m-evl}})$. Notice that the above forgery is for a multi-eval of size 1, namely it is a single-eval forgery, and thus the batch coefficient $\rho' \leftarrow \mathsf{RO}(\mathsf{vk}\|\mathbb{x}'_{\texttt{m-evl}})$ is actually never used by the verifier to check the proof: this is why it passes the verification equation when the forgery of $\mathcal{A}$ satisfies the verification equation.

Differently from the reduction in Lemma 5, the extractor of $\Pi_{\texttt{m-evl}}$ can extract only one witness, i.e., the polynomial committed in r'.

If the *Hash check* for Π_{lin} is satisfied, so does the *Hash check* for $\Pi_{\texttt{m-evl}}$: in particular, by definition of the distinguishing event, the polynomials $(A_i(X))_{i\notin\mathcal{J}}$ are ν-independent. We have that, unless with probability $\epsilon_{\texttt{m-evl}}$, the canonical extractor of $\Pi_{\texttt{m-evl}}$ would extract from $\mathsf{r}' \coloneqq \sum_{i\notin\mathcal{J}} A_i(x^*)\mathsf{b}_i$ the polynomial

$$R'(X) \coloneqq \sum_{i\notin\mathcal{J}} A_i(x^*)B_i(X)$$

such that $R'(x^*) = y'$.

Similarly to the proof of Theorem 1 (just before Hybrid $\mathbf{H}_5$), for all $i \notin \mathcal{J}$ we can associate the commitment b_i with a polynomial equal to $B_i(X) + \tilde{B}_{i,0}(X) + \sum_j Y_j \tilde{B}_{i,j}(X)$, where $\tilde{B}_{i,0}(X)$ depends only on the simulated proofs, while $\tilde{B}_{i,j}(X)$ depends on the simulated proofs and simulated commitment c_j, and we can associate the commitment r' with a polynomial $R'(X, \boldsymbol{Y})$ such that $\mathsf{r}' = R'(s, \mathsf{coms})$ and $R'(X, \boldsymbol{Y})$ is equal to $M_0(X) + \sum_j M_j(X)Y_j$, where:

$$M_0(X) = \sum_{i\notin\mathcal{J}} A_i(x^*)B_i(X) + \sum_{i\notin\mathcal{J}} \tilde{B}_{i,0}(X),$$
$$M_j(X) = \sum_{i\notin\mathcal{J}} A_i(x^*)\tilde{B}_{i,j}(X), \quad \forall j > 0,$$

and it holds that:

1. $\tilde{B}_{i,0}(X) \equiv 0$ if for all $j > 0 : \tilde{B}_{i,j} \equiv 0$.
2. For all i, j, $\tilde{B}_{i,j}$ is an element of a space isomorphic to $\mathbb{F}_{\leq\nu}[X]$.
3. $R'(x^*, \mathsf{coms}) = [y']_1$.

To see Item 1, notice that a simulated proof for (c, x, y) is equal to $\mathsf{c}/(s-x) + [-y/(s-x)]_1$. The rational function $\tilde{B}_{i,0}(X)$ accounts the second addends $-y/(X-x)$ from all the simulated proofs while the $\tilde{B}_{i,j}$ for $j > 0$ take care of the remaining addends. If the latter rational functions are 0 then it means that the adversary did not query the simulation oracle and therefore also the $\tilde{B}_{i,0}$ are 0 polynomials.

When the *Hash Check* holds the polynomials $\tilde{B}_{i,j}$ are fixed before x^* is computed by the RO, similarly to the proof of Theorem 1, these polynomials are fixed by the algebraic representation of the commitment r. If $\mathsf{r} \to_{\mathsf{RO}} x^*$, then the claim holds.

First, we notice that if for all $i \notin \mathcal{J}$ and for all j, the polynomial $\tilde{B}_{i,j} \equiv 0$, then $R'(X, \boldsymbol{Y}) = R'(X)$. We derive that

$$\sum_{i\in[n]} A_i(x^*)B_i(x^*) = \sum_{i\in\mathcal{J}} A_i(x^*)B_i(x^*) + R'(x^*) = y^*$$

We now bound the probability that there exist indexes i, j such that $\tilde{B}_{i,j} \not\equiv 0$. Assume, to reach a contradiction, that there exist i^*, j^* such that $\tilde{B}_{i^*,j^*} \not\equiv 0$. First, we notice that, by Item 1, we can assume

$j^* > 0$. Second, we notice that $M_{j^*} \not\equiv 0$ because $\{A_i\}_{i \notin \mathcal{J}}$ are ν-independent by definition, and, by Item 2, for all i, j $\tilde{B}_{i,j}$ is an element of a space isomorphic to $\mathbb{F}_{\leq \nu}[X]$. More in detail, let $\hat{M}_{j^*}(X) = \sum_i A_i(X)B_i(X)$, because of the ν-independence and the bound on the degree of the $\tilde{B}_{i,j}$ we have that $\hat{M}_{j^*}(X) \not\equiv 0$. Now assume $M_{j^*}(X) \equiv 0$, then $M_{j^*}(x^*) = 0$ and thus $\hat{M}_{j^*}(x^*) = 0$. However notice that $\hat{M}_{j^*}$ is defined by the forged instance and therefore before x^* is sampled, which means that applying the Schwartz-Zippel lemma $M_{j^*}(X) \equiv 0$ only with probability:

$$\mathsf{deg}(M_{j^*} \circ h)/q = \mathsf{deg}(h)(\max_i \mathsf{deg}(A_i) + \bar{\nu})/q.$$

When $M_{j^*} \not\equiv 0$, we can reduce to Aff-MDH. The reduction generates the SRS and simulates using the trapdoor s, while the commitments coms are received by the Aff-MDH challenger. The reduction to Aff-MDH outputs $((\mu_j)_j, \hat{y})$, where the coefficients $\mu_j \leftarrow m_j(x^*)$ and $\hat{y} = y' - m_0(x^*)$. Item 3 implies the correctness of the forgery of the reduction to Aff-MDH.

Putting together, the distinghuishing event implies either a forgery for $\Pi_{\mathtt{m\text{-}evl}}$ or a forgery for the Aff-MDH assumptions, therefore the statement of the lemma follows. □

Finally, we notice that $\epsilon_2 = 0$ because the extractor returns the witness polynomials C_j, for all $j \in [m]$, by the change introduced in $\mathbf{H}_1$. Also, if $\{A_i\}_{i \notin \mathcal{J}}$ are ν-independent, then it also extracts valid witnesses B_i, for all $i \in [n]$, because of the change introduced in $\mathbf{H}_2$. □

Removing the Hash Check. $\Phi_{\mathsf{lin}}^{\mathcal{J},\nu}$ is a sufficient ingredient of our compiler to prove the simulation extractability of zkSNARKs such as PLONK or Marlin: as we explain in Section 6, in these two protocols the verifier checks that the polynomials sent by the prover satisfy some predicate on some *random* points, which allows us to reuse the proofs of an adversary to the SNARKs in the reduction to the simulation extractability of Π_{lin} since they match the *Hash check*. Looking ahead, we call these checks *focal* as they play an important role in our compilation strategy.

However, there may be other protocols that involve also checking equations over some fixed, or not sufficiently random, points. In this paragraph, we show that we can prove Π_{lin} simulation extractable even when the Hash Check is not satisfied, as long as the adversary is able to produce a proof algebraic inconsistent w.r.t. view. This allows us to enlarge the class of protocols that our compiler to zkSNARKs captures. A similar result was proved for the scheme $\Pi_{\mathtt{m\text{-}evl}}$ in [FFK+23]. Let $\Phi_{\mathsf{lin}+}$ be the policy that performs the following check:

- **Algebraic Check:** let $\mathbb{x} := ((\mathsf{c}_j)_j, (\mathsf{b}_i)_i, (G_i)_i, x, y)$ and $\pi := (\hat{\pi}, (y_j)_j)$, returns 1 if and only if there exists a tuple $(\mathbb{x}' = ((\mathsf{c}_j)_j, (\mathsf{b}_i)_i, (G_i)_i, x, y'), \pi' = \hat{\pi}', (y'_j)_j)$ in $\mathcal{Q}_{\mathsf{sim}}$ such that

$$y' \neq y \vee \exists j : y_j \neq y'_j \tag{4}$$

Theorem 3. *Π_{lin} is $(\Phi_{\mathsf{lin}+}, id)$-simulation-extractable in the AGM under the OMSDH assumption.*

Proof. We show a reduction $\mathcal{B}$ to the $(\boldsymbol{\Phi}_{\mathsf{lin}}^{\emptyset,0}, id)$-simulation extractability of Π_{lin}. For the simulation, the reduction simply proxies all the queries back and forth between the adversary and the challenger. At forgery time, $\mathcal{B}$ finds the commitments $(\mathsf{c}_j)_{j\in[m]}$, $(\mathsf{b}_i)_{i\in[n]}$ in the forgery instance $\mathbb{x}$, and derives the linearization commitment r as the verifier would do when verifying the forgery proof π, namely using G_i, x and the field elements y_j contained in it and the commitments b_i. Also, let ρ be the batch coefficient used by the verifier in this step, and let y'_j be the field elements contained in the simulated proof π' that is not algebraic consistent with π: if the adversary complies with the extraction policy, this proof exists; this means that, for all j, $\mathcal{B}$ can find in π' the values y'_j such that the pairs (c_j, y'_j) are algebraic consistent with the current view view; $\mathcal{B}$ can also find y'_r such that (r, y'_r) is algebraic consistent with view. Then $\mathcal{B}$ queries $\mathcal{S}_1$ to get (single-eval) proofs $\bar{\pi}_j$ and $\bar{\pi}_r$ associated with the pairs defined above, and defines

$$\bar{\pi} := \sum_j \rho^{j-1}\bar{\pi}_j + \rho^m \bar{\pi}_r$$

If $\bar{\pi} = \pi$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ submits the instance forgery $\mathbb{x}^* := ((\mathsf{c}^*, [0]_1), x^*, 0)$ and the proof $\pi^* := (\hat{\pi}^*, y^*)$ where:

$$\begin{aligned}
&\mathsf{c}^* \leftarrow (\bar{\pi} - \pi) \\
&\hat{\pi}^* \leftarrow (\bar{\pi} - \pi)(x - x^*)^{-1} \\
&y^* \leftarrow (\sum_j \rho^{j-1}(y'_j - y_j) + \rho^m(y'_r - y))(x - x^*)^{-1}
\end{aligned}$$

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

First, we show that the probability that $\mathcal{B}$ aborts when the adversary complies with the policy and submits a successful forgery is only negligible. Let $\pi_j := (\mathsf{c}_j - y_j)(s - x^*)^{-1}$, and let $\pi_r := (\mathsf{r} - y)(s - x^*)^{-1}$. We have that the proof $\pi = \sum_j \rho^{j-1}\pi_j + \rho^m \pi_r$. Also, by definition of Eq. (4), and because the KZG (single-eval) proofs $\bar{\pi}_j$ and $\bar{\pi}_r$ are unique, either $\pi_r \neq \bar{\pi}_r$ or there exist j such that $\bar{\pi}_j \neq \pi_j$. Since the coefficient ρ is chosen uniformly at random after the proofs of the adversary are fixed, and is also independent of the proofs $\bar{\pi}_j, \bar{\pi}_r$ of the reduction, we conclude by Schwarz-Zippel that $\pi = \bar{\pi}$ with probability at most equal to $\frac{m}{q}$.

Second, we observe that $\mathcal{B}$ complies with the policy in the extractability experiment: the Hash Check is satisfied because we define the point x^* as the output of a RO query including the commitment c^*, and moreover the Partial-Extraction Check is trivially satisfied because the reduction does not have to add any polynomial in $\mathsf{aux}_{\mathcal{E}}$.

Finally, the forgery $(\mathbb{x}^*, \pi^*)$ satisfies the verification equation. In the verification procedure, the linearization commitment is set to be $y^* \cdot [0]_1 = [0]_1$. Because of the homomorphic properties of KZG, then, the verifier has only to check that $\hat{\pi}^*$ is a valid proof for the (single-eval) statement (c^*, x^*, y^*), whose correctness follows from Eq. (1) □

6 Generalizing Polynomial Interactive Oracle Proofs

We generalize PIOPs by allowing the verifier's queries to be (arbitrary) predicates over the prover's oracles. To this end, we use the formalism of oracle relations introduced in [CBBZ23]. Roughly speaking, an oracle relation could be seen as the *oracle-world* counterpart of commit-and-prove relation. In particular, as we use them in the next definition, oracle relations are a useful abstraction which allows to define predicates over the oracles sent by the prover in the execution of a PIOP.

Definition 19 (Oracle Relations, [CBBZ23]). *An oracle (indexed) relation $\mathcal{R}$ is an (indexed) relation when the instances $\mathbb{x}$ of $\mathcal{R}$ contain pointers to oracle polynomials over some field $\mathbb{F}$. The actual polynomials corresponding to the oracles are contained in the witness. We denote the pointer to the oracle polynomial f by $[\![f]\!]$, let $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ we denote with $\mathsf{oracles}(\mathbb{x}) = \{[\![f_1]\!], [\![f_2]\!], \ldots, [\![f_k]\!]\}$ for some k the pointers to the polynomial oracles in $\mathbb{x}$ and $\mathbb{w} = (f_1, f_2, \ldots, f_k)$.*

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

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

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

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

Decision phase: *Let* $r := r(|\mathbb{i}|)$, $n := \sum_{k=0}^{r} n(k)$, $m := \sum_{k=1}^{r} m(k)$. *After the* r*-th round, let the verifier* $\mathsf{V}^{\mathsf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x}, \boldsymbol{\pi}, \boldsymbol{\rho})$, *on input the description of the field* $\mathbb{F}$, *the verifier messages* $\boldsymbol{\rho} := (\rho_1, \ldots, \rho_{r-1})$, *the messages of the prover* $\boldsymbol{\pi} := (\pi_1, \ldots, \pi_m)$ *outputs the instance* $\hat{\mathbb{x}}$ *whith* $\mathsf{oracles}(\hat{\mathbb{x}}) \subseteq \{[\![p_1]\!], \ldots, [\![p_n]\!]\}$. *The verifier accepts if* $\hat{\mathbb{x}} \in \mathcal{L}_{\hat{\mathcal{R}}}$.

We simply say that IP is an r-rounds PIOP if the number of rounds is constant and independent of the size of the index. We give some additional notation. In the following, we will use two different ways to index (the pointers to) the oracle polynomials in a PIOP protocol's execution. We will refer to the oracle polynomials sent by the prover and by the indexer either as $[\![p_{i,j}]\!]$, with double indexes, or as the $[\![p_k]\!]$, with a single index, where $k = \sum_{i'=1,\ldots,i-1} n(i') + j$. We define the oracle index $\mathsf{index}([\![f]\!])$ as the index in the transcript associated with (the pointer to) the polynomial oracle $[\![f]\!]$. Similarly to the set $\mathsf{oracles}(\hat{\mathbb{x}})$, we define the set $\mathsf{indexes}(\hat{\mathbb{x}}) := \{\mathsf{index}([\![f]\!]) : [\![f]\!] \in \mathsf{oracles}(\hat{\mathbb{x}})\}$ the indexes in the transcript associated with (the pointers to) the polynomial oracles involved in $\hat{\mathbb{x}}$.

Security Properties for PIOPs. A list $\mathcal{L} = \{(i_1, y_1), \ldots\}$ is $(\boldsymbol{b}, C)$-bounded where $\boldsymbol{b} \in \mathbb{N}^n$ and C is a PT algorithm if $\forall i \in [n] : |\{(i, y) : (i, y) \in \mathcal{L}\}| \leq \boldsymbol{b}_i$ and $\forall (i, y) \in \mathcal{L} : C(i, y) = 1$.

Definition 21 ($\boldsymbol{b}$-bounded Zero-Knowledge). *A PIOP* IP *is* $\boldsymbol{b}$*-Zero-Knowledge if there exists a checker* C *such that* $\Pr[C(i, x) = 0] \leq \mathsf{negl}(|\mathbb{F}|)$ *over random* x *such that for every index* $\mathbb{i}$, *and* $(\mathsf{pp}, \mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, *and every* $(\boldsymbol{b}, C)$*-bounded list* $\mathcal{L}$, *the following random variables are within* ϵ *statistical distance:*

$$\Big(\mathsf{view}\big(\mathsf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w}), \mathsf{V}^{\mathsf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})\big), (p_i(y))_{(i,y)\in\mathcal{L}}\Big) \approx_\epsilon \mathcal{S}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathcal{L})$$

where $p_1, \ldots, p_n$ *are the polynomials returned by the prover* P *and* $\mathcal{S}$ *is a PPT simulator. Moreover,* IP *is* independent leakage *if* $\mathcal{S}$ *can be divided in two algorithms* $(\mathcal{S}_0, \mathcal{S}_1)$ *where* $\mathcal{S}_0$ *outputs the simulated transcript, and* $\mathcal{S}_1$ *the leakage, i.e., for all randomness* r' *we have* $\mathcal{S}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathcal{L}; r') = \mathcal{S}_0(\mathbb{F}, \mathbb{i}, \mathbb{x}; r'), \mathcal{S}_1(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathcal{L}; r')$.

Hereafter, we introduce the notion of a simulation-friendly polynomial oracles to abstract how our compiler generates instance-independent commitments for the oracles sent during a PIOP protocol's execution.

Definition 22 (PIOP with simulation-friendly polynomial oracles). *A PIOP* IP *has simulation-friendly polynomial oracles if for every* $\mathbb{F}$ *and* $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ *the distribution* $(\mathsf{Com}(\mathsf{ck}, p_i))_i$ *is computationally indistinguishable from the uniform distribution over* $\mathcal{C}^n$ *where* $\mathcal{C}$ *is the commitment space and where* $(p_i)_i$ *are the oracles sent by the prover* $\mathsf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$ *in the interaction with* $\mathsf{V}(\mathbb{F}, \mathbb{x})$.

If the commitment scheme is hiding then this property is trivially true. For the case of non-hiding commitments, one may rely on Decisional Uber Assumption [Boy08] that reduces to discrete log for algebraic adversaries [RS20].

State-restoration. In a state-restoration setting [BCS16,FFK+23] 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, and the prover is considered successful if is able to produce an accepting transcript (consisting of oracle polynomials), while the extractor fails to produce a valid witness given the transcript.

Definition 23 (State-restoration (straight-line) proof of knowledge). *Let* $\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\mathsf{IP},\mathcal{E}}(\mathbb{F})$ *be the experiment in Fig. 3. A PIOP* IP *is* state-restoration (straight-line) proof of knowledge *if there exists an extractor* $\mathcal{E}$ *such that for any* $\tilde{\mathsf{P}}$ *and any* $\mathbb{F}$*:*

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

Verifier Checks. It is often the case that the relation $\hat{\mathcal{R}}$, for an $\hat{\mathcal{R}}$-PIOP, is the logical conjunction of a (sub)relation. In this case, we consider $\hat{\mathbb{x}} := (\hat{\mathbb{x}}_k)_k$ and the verifier returns 1 when all the checks $\hat{\mathbb{x}}_i$ are satisfied. When looking at concrete examples of PIOPs, in the rest of this section, we will assume that this

$\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\mathsf{IP},\mathcal{E}}(\mathbb{F})$

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

Fig. 3. The $\mathbf{Exp}^{sr}_{\tilde{\mathsf{P}},\mathsf{IP},\mathcal{E}}(\mathbb{F})$ experiment.

natural approach is used by the verifier: for sake of simplicity, we extend Definition 20 of an $\hat{\mathcal{R}}$-PIOP and allow the verifier to output n_e checks $(\hat{\mathtt{x}}_k)_k$, and the verifier accepts iff $\hat{\mathtt{x}}_k \in \mathcal{L}_{\hat{\mathcal{R}}}$ for all $k \in [n_e]$.

PIOPs with Delegation. There are cases in which the PIOP can be thought of as a two-phase protocol, sharing the same indexer I where: (i) in the first phase of the protocol, the prover P_1 takes as input the field $\mathbb{F}$, the index $\mathtt{i}$, the instance $\mathtt{x}$ and the witness $\mathtt{w}$, and interacts for a certain number of rounds with the verifier, while (ii) in the second phase, the prover P_2 that, crucially, does not take as input the witness $\mathtt{w}$, interacts with the verifier for only two rounds.[18] Since we require the output of P_2 to be uniquely determined by its input (which is also computable by an "inefficient" verifier), we call this last (witness-independent) phase a *delegation phase*.

The reason to add this new definition is to enlarge the class of PIOPs for which the technical condition in [FFK+23] (sufficient to prove Simulation Extractability of the compiled SNARK) holds.

Definition 24 (Delegation Phase for a PIOP). *Let IP be an $r+1$-rounds $\hat{\mathcal{R}}$-PIOP over $\mathcal{F}$ for $\mathcal{R}$. We say that IP is $\hat{\mathcal{R}}$-PIOP with delegation phase if we can parse P (resp. V) as P_1 and P_2 (resp. as V_1 and V_2) such that there exists a verifier $\tilde{\mathsf{V}}$ taking as additional input the index $\mathtt{i}$ where (1) $\mathsf{IP}_1 = (\mathsf{P}_1, \tilde{\mathsf{V}})$ is a $(r-1)$-rounds $\hat{\mathcal{R}}$-PIOP over $\mathcal{F}$ for $\mathcal{R}$ and the queries of $\tilde{\mathsf{V}}$ and V_1 are identical for any inputs, (2) $\mathsf{IP}_2 = (\mathsf{P}_2, \mathsf{V}_2)$ is a 2-rounds $\hat{\mathcal{R}}$-PIOP over $\mathcal{F}$ for the (P-)language of strings $(\mathbb{F}, \mathtt{i}, (\mathtt{x}, (\boldsymbol{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]}))$ where $\tilde{\mathsf{V}}(\mathbb{F}, \mathtt{i}, \mathtt{x}, (\boldsymbol{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]})) = 1$ assuming that the $\tilde{\mathsf{V}}$'s queries to $\hat{\mathcal{R}}$ are answered positively.*

Uniqueness of delegation phase. *Moreover, we have that for all $\mathbb{F}, \mathtt{i}, \mathtt{x}, (\boldsymbol{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]}$ the probability, taken over the V_2's message $\rho_r \xleftarrow{\$} \mathbb{F}$, that V_2 on input $(\mathbb{F}, \mathtt{x}, (\boldsymbol{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]})$ accepts on two transcripts, that are different in the first tuple of messages and polynomials, is negligible in* $\log|\mathbb{F}|$.

In the following, we simply refer to an *r-rounds $\hat{\mathcal{R}}$-PIOP with a Delegation Phase*, denoting it as $\mathsf{IP}_1 \| \mathsf{IP}_2$, as the $(r+1)$-rounds $\hat{\mathcal{R}}$-PIOP in which the prover P first runs P_1 and interacts with the verifier V_1 for r rounds, then runs P_2 in the last phase, while the verifier outputs the checks of V_1 and V_2, and accepts if

[18] We could consider a more general setting with multiple delegation rounds; however, all the optimized constructions we are aware of only require two.

and only if all the checks are satisfied.[19] Note, we say that a PIOP with delegation has simulation-friendly polynomial oracles if so does IP_1.

6.1 Polynomial $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP

Similarly to Section 5.2, let $\hat{\mathcal{R}}_{\mathsf{lin}}$ be the oracle indexed relation that upon an instance

$$\hat{\mathbb{x}} := ((\llbracket c_j \rrbracket)_{j\in[m]}, (\llbracket b_i \rrbracket)_{i\in[n]}, (G_i)_{i\in[n]}, x, y),$$

outputs 1 if and only if $\sum_i a_i(x)b_i(x) = y$, where for all i, we have that $a_i(X) := G_i(c_1(X), \ldots, c_m(X), X)$. We refer to the polynomial oracles $(\llbracket c_j \rrbracket)_j$ as the *core polynomial oracles*, while the polynomial oracles $(\llbracket a_i \rrbracket)_i$ and $(\llbracket b_i \rrbracket)_i$ as the *left* and *right polynomial oracles* respectively. We use the shorthand $\hat{\mathbb{x}}.a_i$ to refer to the a_i defined above.

Below we formalize a class of $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOPs in which each evaluation point x chosen by the verifier for a $\hat{\mathcal{R}}_{\mathsf{lin}}$ query is a function $x = \tilde{v}(\boldsymbol{\rho})$ of its random coins, where $\tilde{v}$ is a polynomial that can be defined by the verifier depending only on the index $\mathbb{i}$ and the instance $\mathbb{x}$. The $\hat{\mathcal{R}}_{\mathsf{lin}}$ checks relying on a $\tilde{v}$ which is non-constant in the $r-1$-th random coin are called "focal", as they have a focal role to ensure extractability.

Definition 25 (Structured $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP and focal checks). *An r-rounds $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP IP is* structured *if there exists a deterministic PT algorithm $\tilde{\mathsf{V}}$ such that for all $\mathbb{i}$, $\mathbb{x}$, $\boldsymbol{\pi}$, $\boldsymbol{\rho}$, $k \in [n_e]$ we have that:*

$$\tilde{v}_k(\boldsymbol{\rho}) = \hat{\mathbb{x}}_k.x$$

where $(\tilde{v}_k)_k \leftarrow \tilde{\mathsf{V}}(\mathbb{F}, \mathbb{i}, \mathbb{x})$ and $(\hat{\mathbb{x}}_k)_k \leftarrow \mathsf{V}^{\mathsf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x}, \boldsymbol{\pi}, \boldsymbol{\rho})$.

If $\mathsf{deg}_{r-1}(\tilde{v}_k) \geq 1$ we say that the check $\hat{\mathbb{x}}_k$ is focal*. We denote by $\mathcal{K}_f$ the set of all indexes k such that $\hat{\mathbb{x}}_k$ is focal.*

Finally, we introduce the notion of compilation-safeness for $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOPs. The idea of the definition below is that focal checks can be ordered in such a way that we can incrementally extract all the polynomials, starting from the trivially extractable polynomials, namely the index polynomials, and using the partial extractability property derived from Definition 18.

Definition 26 (Compiler-safe $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP). *An r-rounds $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP IP is* compiler-safe *if for any $\mathbb{i}$ and $\mathbb{x}$, for any $\boldsymbol{\pi}$ and any $\boldsymbol{\rho}$ there are not polynomial oracles in the last message of the prover and there is an ordering of the focal checks $(\hat{\mathbb{x}}_k)_{k\in\mathcal{K}_f}$ such that (1) for any $k \in \mathcal{K}_f$ we have $\{\hat{\mathbb{x}}_k.a_i : \hat{\mathbb{x}}_k.b_i \notin \mathcal{J}_{k-1}\}$ are ν-independent and (2) we have $\mathcal{J}_{n_e}$ is the set of all the polynomials including index polynomials sent by the prover, where:*

- *$\mathcal{J}_0$ is the set of $n(0)$ index polynomials*
- *for all $k \notin \mathcal{K}_f$, $\mathcal{J}_k := \mathcal{J}_{k-1}$*
- *for all $k \in \mathcal{K}_f$, $\mathcal{J}_k := \mathcal{J}_{k-1} \cup \{\hat{\mathbb{x}}_k.c_j : j \in [m]\} \cup \{\hat{\mathbb{x}}_k.b_i : i \in [n]\}$*
- *ν is the maximum number of distinct points for which the verifier evaluates a non-index polynomial, i.e.,*

$$\nu := \max_{i>n(0)} |\{\hat{\mathbb{x}}_k : \llbracket p_i \rrbracket \in \mathsf{oracles}(\mathbb{x}_k), k \in [n_e]\}|$$

Moreover, an $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP with a delegation phase $\mathsf{IP}_1 \| \mathsf{IP}_2$ is structured *(resp.* compiler-safe*) if IP_1 and IP_2 are both structured (resp.* compiler-safe*).*

[19] The prover can sends all the messages of the first round of IP_2 on the r-th round of IP_1, thus yelding an $r+1$ (rather than $r+2$) rounds protocol.

6.2 Polynomial $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP

As mentioned in Section 1, when designing a new scheme, it is easier to describe the PIOP by specifying a list of polynomial equations between the polynomial oracles as, for example, in [GWC19,CFF+21,RZ21,FFK+23]. In this section we formalize this class of PIOPs using the oracle relation $\hat{\mathcal{R}}_{\mathtt{poly}}$ that upon the instance $\mathbb{x}_{\mathtt{poly}} := ((\llbracket p_j \rrbracket)_{j\in[n]}, F, (v_j)_{j\in[n]})$, outputs 1 if and only if:

$$F(p_1(v_1(X)), \ldots, p_n(v_n(X)), X) \equiv 0$$

where $v_j \in \mathbb{F}[X], \forall j$ and $F \in \mathbb{F}[X_1, \ldots, X_n, X]$.

We consider $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOPs that are *structured* as described below.

Definition 27 (Structured $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP). *An r-rounds $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP IP is* structured *if there exists a deterministic PT algorithm $\tilde{\mathsf{V}}$ such that for any $\mathtt{i}$ and $\mathbb{x}$, for any $\boldsymbol{\pi}$ and for any $\boldsymbol{\rho}$, $j \in [n]$,$k \in [n_e]$ we have that:*

$$\tilde{v}_{j,k}((\rho_i)_{i\in[r-2]}, X) = \mathbb{x}_k.v_j(X)$$

where $(\tilde{v}_{j,k})_{j,k} \leftarrow \tilde{\mathsf{V}}(\mathbb{F}, \mathtt{i}, \mathbb{x})$ and $\{\hat{\mathbb{x}}_k\}_k \leftarrow \mathsf{V}^{\mathsf{I}(\mathbb{F},\mathtt{i})}(\mathbb{F}, \mathbb{x}, \boldsymbol{\pi}, \boldsymbol{\rho})$.
Moreover, an $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP with a delegation phase $\mathsf{IP}_1 \| \mathsf{IP}_2$ is structured if IP_1 and IP_2 are both structured.

We extend the *compiler-safe* definition of [FFK+23] to capture $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOPs with delegation phase. We require that for each polynomial sent by the prover in the first $r-1$ rounds there must be an equation that involves evaluating it on a (non-constant function of) the last random coin sent by the verifier. Crucially, we require that the prover does not send any polynomial in the last round. We do not make any restriction on the index polynomials. Our notion of compiler-safe is more inclusive than in [FFK+23], as it holds for PIOPs such like Marlin [CHM+20] and Lunar [CFF+21] without any changes.

Definition 28 (Compiler-safe $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP). *An r-rounds $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP IP is* compiler-safe *if for any $\mathtt{i}$ and any $\mathbb{x}$, $\forall \boldsymbol{\pi}$, $n(r) = 0$ and:*

$$\forall j \in [n] \setminus [n(0)] : \exists k \ s.t \ \mathtt{deg}_{X_{r-1}}(\tilde{v}_{j,k}) \geq 1$$

where $(\tilde{v}_{j,k})_{j,k} \leftarrow \tilde{\mathsf{V}}(\mathbb{F}, \mathtt{i}, \mathbb{x})$.
Moreover an $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP with a delegation $\mathsf{IP}_1 \| \mathsf{IP}_2$ is compiler-safe if IP_1 and IP_2 are both compiler-safe.

6.3 From $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP to $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP

Arguably, the notion of compiler safe for $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP is very easy to check. In particular, it is much easier to check than the same notion for $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP. In this section, we show that we can transform a compiler-safe $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP into a compiler-safe $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP. We believe this can give an easy-to-follow recipe when designing new KZG-based SE-zkSNARKs from PIOPs because the cryptographer needs only to focus on designing compiler-safe $\hat{\mathcal{R}}_{\mathtt{poly}}$-PIOP.

Our transformation is similar to the general strategy proposed by [GWC19], but it explicitly makes sure that the derived checks result into a compiler-safe $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP. We start by giving the transform for the case in which there is one[20] polynomial check G and all the $(v_i)_{i\in[n]}$ are equal to some non-constant polynomial v. Let $\mathcal{J}_0$ be the set of the index polynomials, we need to find a subset $\mathcal{I} \subset [n]$ of minimal size $m < n$, and polynomials $(G_i)_{i\in\bar{\mathcal{I}}}$, where $\bar{\mathcal{I}} = [n] \setminus \mathcal{I}$, such that:

- $F(X_1, \ldots, X_n, X) = \sum_{i\in\bar{\mathcal{I}}} G_i((X_j)_{j\in\mathcal{I}}, X) \cdot X_i$,
- $(G_i((p_j(v(X)))_{j\in\mathcal{I}}, X))_{i\notin\mathcal{J}_0}$ are 1-independent.

[20] When there are multiple checks with the same non-constant v we can simply batch together the equations in one single equation.

We define the instance $\hat{\mathfrak{x}}_{\mathsf{lin}} := ((\llbracket p_i \rrbracket)_{i\in\mathcal{I}}, (\llbracket p_i \rrbracket)_{i\in\bar{\mathcal{I}}}, (G_i)_{i\in\bar{\mathcal{I}}}, v(\boldsymbol{\rho}), 0)$. This transform minimizes the number of *core* polynomial oracles, which results in minimizing the size of the proof of Π_{lin}.

When there are distinct polynomials v_i the optimization problem gets more complex. In this case, assume that the number of distinct polynomials v_i is equal to ν, then the transformation needs to find sets $\mathcal{I}_1$ and $\mathcal{I}_2$ and minimizes the size of $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2$, such that it can decompose F into $\nu - 1$ instances for batch-evaluation[21] that check $p_i(v_i(\boldsymbol{\rho})) = y_i$ for all $i \in \mathcal{I}_1$ and for values y_i that are sent as part of the last message of the $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP prover, and one polynomial $F'((X_i)_{i\in\bar{\mathcal{I}}_1}, X) = F((y_i)_{i\in\mathcal{I}_1}, (X_i)_{i\in\bar{\mathcal{I}}_1}, X)$ where, similar to the previous case, F' can be decomposed as:

- $F'((X_i)_{i\in\bar{\mathcal{I}}_1}, X) = \sum_{i\in\mathcal{I}_2} G_i((X_j)_{j\in\bar{\mathcal{I}}}, X) \cdot X_i$,
- $(G_i((p_j(v_j(X)))_{j\in\bar{\mathcal{I}}}, X))_{i\notin\mathcal{J}_0\cup\mathcal{I}}$ are ν-independent.

Finally, when the PIOP has a delegation phase, we can just apply the transform both to IP_1 and IP_2, this works because the checks involve two disjoint sets of polynomial oracles (excluding the index polynomials, that however are shared among the two phases).

6.4 Notable PIOPs

In what follows, we show how to express in $\hat{\mathcal{R}}_{\mathsf{lin}}$ form the PIOPs underlying PLONK and Marlin (with all optimizations); it is easy to extend this analysis to Lunar and Basilisk that are very similar to Marlin and PLONK, respectively. **PLONK.** We show in Fig. 4 how PLONK [GWC19] can be written as a 4-rounds

$\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP in which the verifier outputs two checks. The maximum number of distinct points for which the verifier evaluates a non-index polynomial is 2 since the oracle polynomial $\llbracket z \rrbracket$ is evaluated on $\mathfrak{z}$ and $\mathfrak{z}\omega$. In $\hat{\mathfrak{x}}_1$ the verifier tests that $z(\mathfrak{z}\omega)$ equals the field element $\bar{z}_\omega$ sent in the last round by the prover. Moreover, all but $\llbracket t_{lo} \rrbracket, \llbracket t_{mid} \rrbracket, \llbracket t_{hi} \rrbracket$ of the *right* oracle polynomials of $\hat{\mathfrak{x}}_2$ are part of the index or are extracted from $\hat{\mathfrak{x}}_1$. However, the corresponding *left* oracle polynomials, that we highlight in the figure, are linearly independent w.r.t. $\mathbb{F}_{\leq 2}[X]$, which results in a compiler-safe PIOP according to Definition 26.

Marlin. We show in Fig. 5 how Marlin [CHM+20,ark21] can be written as a 3-rounds $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP with a Delegation Phase.

7 Revisiting the PIOP-to-zkSNARK compiler

We show how to turn compiler-safe $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOPs into simulation-extractable zkSNARKs. We stress that, although the formalism we adopt differs from previous work, the resulting compiler's construction is the usual one with the linearization trick optimization.

Definition 29. *We say that Π is* strong simulation-extractable in the algebraic group model with oblivious sampling *if and only if Π is $\boldsymbol{\Phi}_{\mathsf{sse}}$-simulation-extractable where for any (Φ_0, Φ_1) in the family of policies $\boldsymbol{\Phi}_{\mathsf{sse}}$ we have that Φ_0 outputs group elements $\mathsf{coms} = (\mathsf{c}_i)_i$ from an Aff-MDH secure and witness sampleable distribution and Φ_1 checks that the forgery $(\mathbb{x}^*, \pi^*) \notin \mathcal{Q}_{\mathsf{sim}}$.*

Theorem 4. *Let Π_{lin} be the CP-SNARK for $\mathcal{R}_{\mathsf{lin}}$ defined in Section 5.2. Let IP be a compiler-safe $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP for relation $\mathcal{R}$ that is state-restoration straightline extractable, bounded zero-knowledge, and with simulation-friendly polynomial oracles. Let Π be the zkSNARK compiled from IP using the compiler in Fig. 6. Then Π is zero-knowledge and strong simulation-extractable in the AGM. Furthermore, if $\mathcal{R}$ is an oracle relation, then Π is a CP-SNARK.*

[21] We notice that a $\hat{\mathcal{R}}_{\mathsf{lin}}$-instance can be trivially reduced to a batch-evaluation by having an empty set of right polynomial oracles.

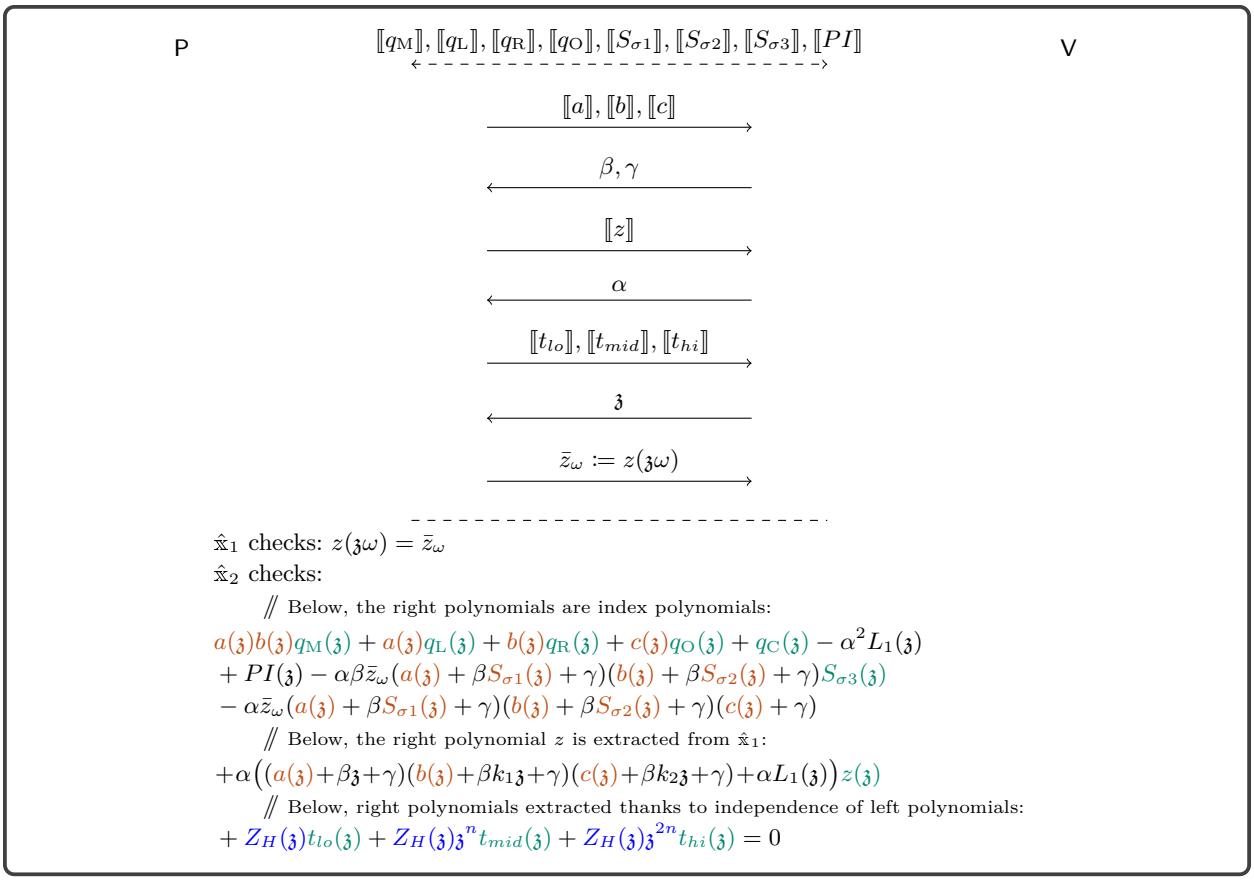

Fig. 4. The $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP PLONK. For $\hat{\mathbb{x}}_2$, we highlight the *core* polynomials, the *left* polynomials that are linearly independent, and the *right* polynomials.

Proof. In what follows, we assume that IP is in fact an $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP with a delegation phase, i.e., the prover interacts with the verifier for $r+1$ rounds.

Zero-Knowledge. We start showing the zero-knowledge simulator for Π and an adversary submitting Q queries. The simulator is in Fig. 7. We define the bounded (leakage) list $\mathsf{Leak}_{\hat{\mathbb{x}}}$ as the list of all tuples $(i, \hat{\mathbb{x}}.x)$ such that $i \in \mathsf{indexes}(\hat{\mathbb{x}})$ and $[\![p_i]\!]$ is a left oracle polynomial of $\hat{\mathbb{x}}$. We recall that $\mathsf{indexes}(\hat{\mathbb{x}})$ is the list of indexes of the polynomial oracles sent by the prover that are involved in the instance $\hat{\mathbb{x}}$.

The zero-knowledge guarantees of the simulator come from the simulation-friendly polynomial oracles of IP (see Definition 22), the bounded zero-knowledge property of the IP, and the L_{lin}-leaky zero-knowledge property of Π_{lin}, where $L_{\mathsf{lin}}(\mathbb{x}, \mathbb{w}) := (\mathbb{w}.C_j(\mathbb{x}.x))_j$ (cf. Section 5.2).

We can show this with a simple hybrid argument. Let $\mathbf{G}_0$ be the *real-world* experiment where $\mathcal{A}$ interacts with a real prover and the random oracle.

- The first hybrid $\mathbf{G}_1$ is the same as $\mathbf{G}_0$ but where, at every call to the prover, we additionally compute the bounded (leakage) list Leak computed by the real prover. This hybrid is obliviously identically distributed to the previous one.
- The second hybrid $\mathbf{G}_2$ is the same as $\mathbf{G}_1$ but where the SRS is generated using $\Pi_{\mathsf{lin}}.\mathcal{S}_0$ and the proofs of $\mathcal{R}_{\mathsf{lin}}$ for the instances in $\mathcal{K}_1$ are generated using the simulator $\Pi_{\mathsf{lin}}.\mathcal{S}_1$. Notice that, since Π_{lin} is only leaky-zero-knowledge, to generate such proofs, the simulator additionally needs the leakage which we can compute using the bounded list $\mathsf{Leak}_{\hat{\mathbb{x}}}$ over the polynomials computed by the prover. The proof of

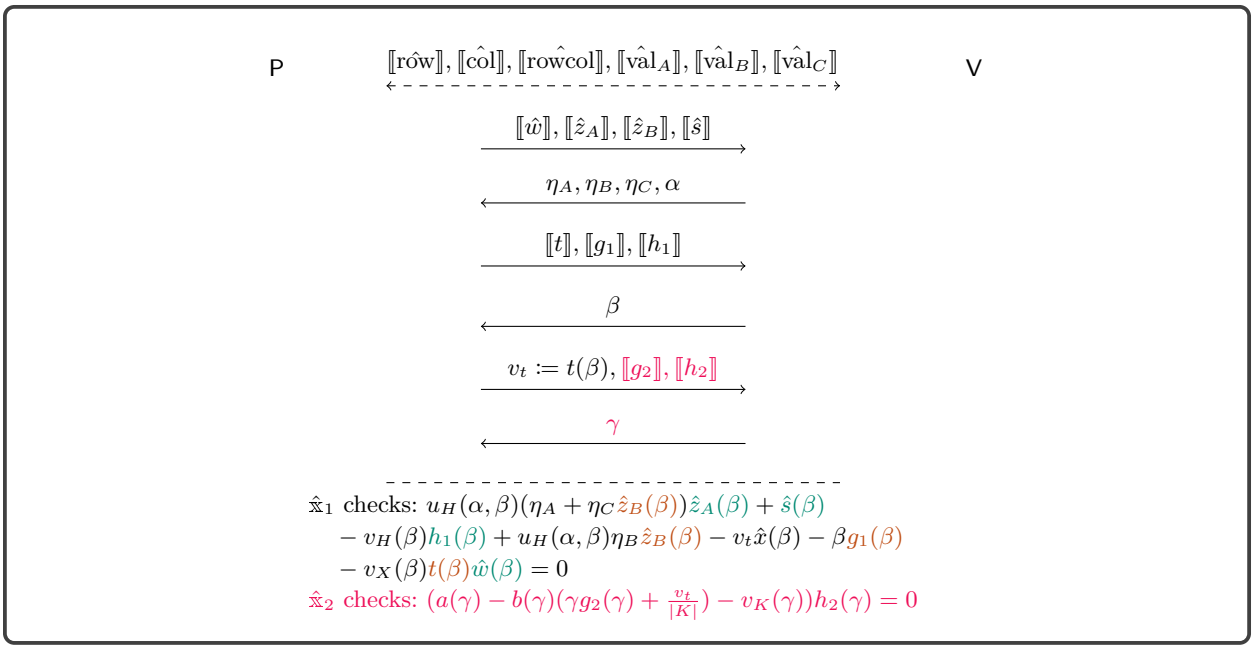

Fig. 5. The $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP Marlin, where: v_H (resp. v_K, v_X) denotes the vanishing polynomial of the subgroup H (resp. K, X) of $\mathbb{F}$; u_H is the formal derivative of v_H; the polynomials a and b are computed using the index polynomials and the coins α and β. We highlight the delegation phase, the *core* and *right* polynomials of $\hat{\mathbb{x}}_1$.

indistinguishability between the two hybrids follows easily from the leaky zero-knowledge property of Π_{lin}.

- The third hybrid $\mathbf{G}_3$ is the same as $\mathbf{G}_2$ but where, at every call to the prover, we sample the commitments as uniformly random $\mathbb{G}_1$-group elements. For this step we use that IP (or IP_1, if $\mathsf{IP} = \mathsf{IP}_1 \| \mathsf{IP}_2$ has a delegation phase) has simulation-friendly polynomial oracles (cf. Definition 22).
- The last hybrid $\mathbf{G}_4$ is the same a $\mathbf{G}_3$ but where (1) we finally switch to use the zero-knowledge of the $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP and (2) we use the leakage using the simulator. In particular, we need to use the simulator for an (honest) verifier that samples its messages by computing the random oracle on the transcript so far[22]. This allows showing that the simulator is in the non-programmable random oracle. The last hybrid is identically distributed to the *ideal-world* experiment where $\mathcal{A}$ interacts with the simulator and the random oracle.

Finally, we show that, independently of the strategy of the adversary $\mathcal{A}$, the view at the end of the simulation extractability experiment for the compiled zkSNARK and with the simulator described in Fig. 7 is algebraic consistent.

When the relation $\mathcal{R}$ is not an oracle relation, namely when the instances do not contain any commitment, we can easily show that the view is algebraic consistent by noticing that the simulator, at the q-th query, produces n_e simulated proofs on the commitments $\mathsf{coms}^{(q)}$, moreover, since the PIOP prover can only prove algebraic consistent statements then also the simulator can only compute algebraic consistent statements, otherwise we would have a distinguisher for the zero-knowledge of the PIOP.

We need a more careful analysis when the relation $\mathcal{R}$ is an oracle relation, in which case Π is a CP-SNARK. In fact, assume that, at the q-th simulation query, the adversary includes a simulated commitment $\tilde{\mathsf{c}}$ in the queried instance $\mathbb{x}$, namely either $\tilde{\mathsf{c}} \in \mathsf{coms}^{(j)}$ for $j < q$ or $\tilde{\mathsf{c}} \in \mathsf{coms}'$. We observe that the simulator trivially preserves the algebraic consistency across multiple proofs for *focal* checks since they involve evaluation on

[22] Technically, we can hardcode the full description of the random oracle inside the verifier and rely on the statistical zero-knowledge property.

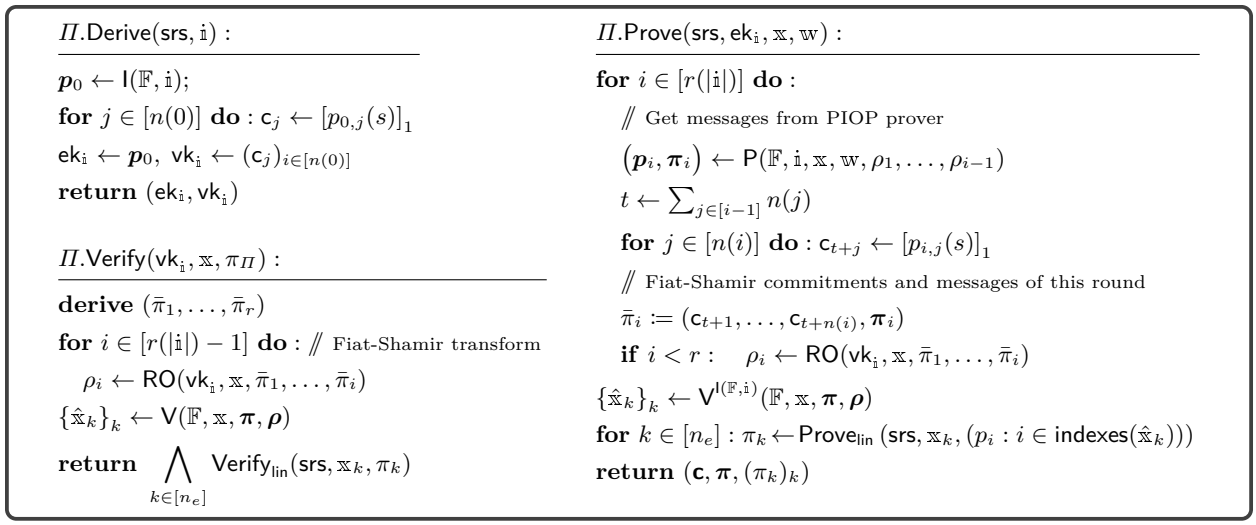

Fig. 6. The compiler based from $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOPs and KZG commitment scheme to Universal zkSNARKs. We associate to the instance $\hat{\mathbb{x}}_k$ for the oracle relation $\hat{\mathcal{R}}_{\mathsf{lin}}$ the instance $\mathbb{x}_k$ for the commit-and-prove relation $\mathcal{R}_{\mathsf{lin}}$, the latter instance is identical to $\hat{\mathbb{x}}_k$ but where any oracle $[\![p]\!] \in \mathsf{oracles}(\hat{\mathbb{x}}_k)$ is substituted with the commitment $[p(s)]_1$.

random coins, thus on evaluation points that were not queried before. As for non-focal checks, notice that, because of the $\boldsymbol{b}$-bounded zero-knowledge of the PIOP we have that $\tilde{\mathsf{c}}$ (or better say the oracle associated to it) must be a right polynomial oracle in all the $\hat{\mathcal{R}}_{\mathsf{lin}}$ instances where it appears. Otherwise, the PIOP would not be able to support leakage on this oracle. Moreover, in all the non-focal instances where it appears as a left polynomial either the instance contains another left polynomial that is sent by the prover or the instance evaluates to a constant value y, again, because of the bounded zero-knowledge property of the PIOP. Notice, in the first case the algebraic consistency holds because $\tilde{\mathsf{c}}$ is evaluated together with a simulated commitment from $\mathsf{coms}^{(q)}$. In the second case, algebraic consistency holds because $\tilde{\mathsf{c}}$ is evaluated on an evaluation point and to a constant value, thus consistently with the previous simulated proofs.

Simulation-Extractability. We define the extractor for Π for a given adversary $\mathcal{A}_\Pi$. We make some simplifying assumptions on the behavior of $\mathcal{A}_\Pi$: (1) the adversary always queries first the RO on a string that can be parsed as $(\mathbb{i}, \mathbb{x})$ before querying the simulation oracle on the same string, (2) the auxiliary string $\mathsf{aux}_{\mathcal{E}}$ output by $\mathcal{A}_\Pi$ can be parsed as a list of strings $(s_i, \mathsf{aux}_i, \mathsf{st}_i)_i$ and a string $\mathsf{aux}'_{\mathcal{E}}$ where for any i we have $(s_i, \mathsf{aux}_i, \mathsf{st}_i)$ string is identical to the auxiliary input output at the i-th query of the adversary. These assumptions are w.l.g.. In fact, given an adversary $\mathcal{A}_\Pi$ that does not respect these rules we can always define another adversary that runs internally $\mathcal{A}_\Pi$, collects all the necessary information to comply with (2) and moreover follows the rule (1).

Finally, we emphasize that in our proof strategy we need to extract the witness polynomials from the proofs according to some order, thus we would first need to sort the proofs of the adversary and then proceed. However, w.l.o.g., we assume that the focal checks output by the verifier are already sorted so to match the compilation-safeness property (Definition 26), and such that all the "delegation checks", namely $(\hat{\mathbb{x}}_k)_{k \in \mathcal{K}_2}$, come before the others: notice that it is always possible to define such sorting since the non-delegation checks and the delegation checks involve two disjoint sets of polynomials (excluding the index polynomials that however are already in $\mathcal{J}_0$).

Before proceeding, we set some notation:

– Let $\mathsf{oracles}_b(\hat{\mathbb{x}})$ and $\mathsf{oracles}_c(\hat{\mathbb{x}})$ be respectively the right and the core oracles of $\hat{\mathbb{x}}$. Similarly, let $\mathsf{indexes}_b(\hat{\mathbb{x}})$ and $\mathsf{indexes}_c(\hat{\mathbb{x}})$ be respectively the indexes of the right and the core oracles of $\hat{\mathbb{x}}$.

$\mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$

$\mathsf{srs}, \mathsf{st}_{\Pi} \leftarrow\!\$\ \Pi_{\mathsf{lin}}.\mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$

$\mu \leftarrow 0$ // $\mathcal{S}_1$ queries counter

for $j \in [Q]$ **do** :

$\quad \mathsf{coms}^{(j)} \leftarrow\!\$\ \mathbb{G}_1^{n-n(r)}$

$\mathsf{coms}' \xleftarrow{\$} \Phi_0(\mathsf{pp}_{\mathbb{G}})$

$\mathsf{st}_{\mathcal{S}} \leftarrow (\mathsf{st}_{\Pi}, \mu, (\mathsf{coms}^{(j)})_j, \mathsf{coms}')$

return $\mathsf{srs}, \mathsf{st}_{\mathcal{S}}$

$\mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s, \mathsf{aux})$

if $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$:

$\quad$ **return** $a, \mathsf{st}_{\mathcal{S}}$

$a \leftarrow\!\$\ \mathbb{F}$

$\mathcal{Q}_{\mathsf{RO}} \leftarrow \mathcal{Q}_{\mathsf{RO}} \cup (s, \mathsf{aux}, a)$

return $a, \mathsf{st}_{\mathcal{S}}$

$\mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathsf{srs}, (\mathbb{i}, \mathbb{x}))$

$\mathsf{st}_{\mathcal{S}} \leftarrow (\mathsf{st}_{\Pi}, \mu, (\mathsf{coms}^{(j)})_j, \mathsf{coms}')$

$\mathsf{c}_1, \ldots \mathsf{c}_{n-n(r)} \leftarrow \mathsf{coms}^{(\mu)}$

for $i \in [r-1]$ **do** :

$\quad \boldsymbol{\pi}_i \leftarrow \mathsf{IP}.\mathcal{S}_0(\mathbb{F}, \mathbb{i}, \mathbb{x}, \rho_1, \ldots, \rho_{i-1})$

$\quad t \leftarrow \sum_{j \in [i-1]} n(j)$

$\quad \bar{\pi}_i = (\mathsf{c}_t, \ldots, \mathsf{c}_{t+n(i)}, \boldsymbol{\pi}_i)$

$\quad \rho_i \leftarrow \mathsf{RO}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i)$

$(\boldsymbol{p}_r, \boldsymbol{\pi}_r) \leftarrow \mathsf{P}_2(\mathbb{F}, \mathbb{i}, \mathbb{x}, \boldsymbol{\pi}, \rho_1, \ldots, \rho_{r-1}) \| \mathsf{IP}.\mathcal{S}_0(\mathbb{F}, \mathbb{i}, \mathbb{x}, \rho_1, \ldots, \rho_{r-1})$

$(\mathsf{c}_{n-n(r)+1}, \ldots, \mathsf{c}_n) \leftarrow ([p_{r,j}(s)]_1)_{j \in [n(r)]}$

$\bar{\pi}_r = (\mathsf{c}_{n-n(r)+1}, \ldots, \mathsf{c}_n, \boldsymbol{\pi}_r)$

$\rho_r \leftarrow \mathsf{RO}(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_r)$

$\boldsymbol{\pi}_{r+1} \leftarrow \mathsf{P}_2(\mathbb{F}, \mathbb{i}, \mathbb{x}, \boldsymbol{\pi}, \boldsymbol{\rho})$

$\{\hat{\mathbb{x}}_k\}_{k \in \mathcal{K}_1}, \{\hat{\mathbb{x}}_k\}_{k \in \mathcal{K}_2} \leftarrow \mathsf{V}^{\mathsf{I}(\mathbb{F}, \mathbb{i})}(\mathbb{F}, \mathbb{x}, \boldsymbol{\pi}, \boldsymbol{\rho})$

for $k \in \mathcal{K}_1$:

$\quad \mathsf{leak} \leftarrow \mathsf{IP}.\mathcal{S}_1(\mathbb{F}, \mathbb{i}, \hat{\mathbb{x}}_k, \mathsf{Leak}_{\hat{\mathbb{x}}_k})$

$\quad \pi_k \leftarrow \Pi_{\mathsf{lin}}.\mathcal{S}_1(\mathsf{st}_{\Pi}, \mathbb{x}_k, \mathsf{leak})$

for $k \in \mathcal{K}_2$:

$\quad \pi_k \leftarrow \Pi.\mathsf{Prove}(\mathsf{srs}, \mathbb{x}_k, (p_i : i \in \mathsf{indexes}(\hat{\mathbb{x}}_k)))$

$\pi \leftarrow (\mathbf{c}, \boldsymbol{\pi}, (\pi_k)_k)$

$\mathsf{st}_{\mathcal{S}} \leftarrow (\mathsf{st}_{\Pi}, \mu + 1, (\mathsf{coms}^{(j)})_j, \mathsf{coms}')$

return $\pi, \mathsf{st}_{\mathcal{S}}$

Fig. 7. The simulator $\mathcal{S}$ for a Π compiled from an r-rounds $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP with a delegation phase. We highlight the parts needed only for the delegation setting.

- For all k, let $\mathcal{CI}(\hat{\mathbb{x}}_k) := \mathsf{oracles}_b(\hat{\mathbb{x}}_k) \cap \mathcal{J}_k$ be the *compiler-safe index set*, namely the set of the indexes of the polynomials sent by the prover that are either part of the index or may be extracted from $\hat{\mathbb{x}}_{k'}$, for $k' < k$.
- For all k, we define $\{\gamma_{j,k}\}_j := \mathsf{indexes}_c(\hat{\mathbb{x}}_k)$ and $\{\beta_{i,k}\}_i := \mathsf{indexes}_b(\hat{\mathbb{x}}_k)$. Whenever it is clear from the context, we may omit the index k.
- Let $\mathcal{P}_i$ be the indexes of the polynomials sent at the i-th round by the prover.
- Given a proof π for Π we define *the RO-queries of* π the list of strings $((\mathsf{vk}_{\mathbb{i}}, \mathbb{x}), \ldots, (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_r))$.
- We say that the adversary *copied up to round* i (the transcript of) its proof π_{Π} from a simulated proof π'_{Π} if the first $i+1$ entries of their RO-queries are equal.
- We say that a proof π_{Π} *uses a simulated element* if there is a non-zero coefficient depending on the simulated elements provided by $\mathcal{S}$ in the algebraic representation of any of the commitments in π_{Π}.
- We say that a coin is *fresh* if it does not appear in any of the simulated transcripts of the proofs output by $\mathcal{S}$.
- We use $\mathcal{K}_1$ and $\mathcal{K}_2$ to denote the indexes of the checks output in IP_1 and IP_2 respectively, where $\mathsf{IP} := \mathsf{IP}_1 \| \mathsf{IP}_2$
- We let $\mathcal{E}_{\mathsf{Com}}$ be the canonical AGM extractor of the KZG polynomial commitment, namely the one that parses the algebraic representation of a commitment c as $(f, \boldsymbol{r})$ and returns the polynomial $f(X)$. Notice the extractor fails when $\boldsymbol{r} \neq \mathbf{0}$.

The extractor $\mathcal{E}_{\Pi}(\mathbb{x}_{\Pi}, \pi_{\Pi}, \mathsf{aux}_{\mathcal{E}})$:

1. Parse $\mathsf{aux}_{\mathcal{E}}$ as the concatenation of a list $(s_i, \mathsf{aux}_i, \mathsf{st}_i)_i$ and $\mathsf{aux}'_{\mathcal{E}}$, where $(s_i, \mathsf{aux}_i, \mathsf{st}_i)$ is the output of $\mathcal{A}_{\Pi}$ at the i-th query to the ROM and $\mathsf{aux}'_{\mathcal{E}}$ the remaining auxiliary information given by the adversary (namely, the auxiliary information associated with its last output).
2. Take the commitments $(\mathsf{c}_i)_{i\in[n]}$ in both $\mathbb{x}_{\Pi}$ (if it is a commit-and-prove relation) and π_{Π}; from π_{Π} derive the messages $\bar{\pi}_1, \ldots, \bar{\pi}_r$ and find the indexes q_i such that $s_{q_i} = (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i)$.
3. Return $\perp$ if π_{Π} uses a simulated element of one of the proofs.
4. For $i \in [n]$, let $p_i \leftarrow \mathcal{E}_{\mathsf{Com}}(\mathsf{c}_i)$.
5. Return $\perp$ if for some i, $\exists j \in \mathcal{P}_i : \mathsf{c}_j \neq [p_i(s)]_1$.
6. Let $\hat{\mathbb{x}}_k = ((\mathsf{c}_{\gamma_j})_{j\in[m]}, (\mathsf{c}_{\beta_i})_{i\in[n]}, (G_i)_{i\in[n]}, x, y)$. If $\exists k$: $\sum_i G_i(p_{\gamma_1}(x), \ldots, p_{\gamma_m}(x), x)p_{\beta_i}(x) \neq y$, return $\perp$.
7. Return $\mathcal{E}_{\mathsf{PIOP}}(\mathbb{i}, \mathbb{x}_{\Pi}, (p_j)_j)$

To analyze the success of the extractor we define a series of hybrid games. We start from the first hybrid that is the $\mathbf{Exp}^{sr}_{\mathcal{A}_{\mathsf{IP}},\mathsf{IP}}(\mathbb{F})$ experiment for IP (see Definition 23) for an adversary $\mathcal{A}_{\mathsf{IP}}$ that we define next.

The adversary $\mathcal{A}_{\mathsf{IP}}$:

1. Run simulator $\mathsf{srs}, \mathsf{st}_{\mathcal{S}} \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$ and set $\mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathsf{sim}}$ empty sets.
2. Run $\mathcal{A}_{\Pi}(\mathsf{srs})$ and answer all the simulation queries of $\mathcal{A}_{\Pi}$ with $\mathcal{S}_1$.
3. Upon i-th query (s_i, aux_i) from $\mathcal{A}_{\Pi}$ to $\mathcal{S}_2$:
   (a) if s_i is in the RO-queries of a simulated proof in $\mathcal{Q}_{\mathsf{sim}}$ then run $\mathcal{S}_2$ on input s_i.
   (b) Else parse s_i as a (partial) transcript $\mathtt{trns} = (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_{r'})$; parse $\bar{\pi}_j$ as $(\mathbf{c}_j, \boldsymbol{\pi}_j)$; find the witness polynomials $\mathbb{w} := (f_{r',j})_j$ that are the algebraic representation (depending on ck) of $\mathbf{c}_{r'}$; find in $\mathsf{SeenStates}$ the state $\mathsf{cvs} = (\mathbb{i}, \mathbb{x}, \boldsymbol{\pi}_1, \{p_{1,i}\}_i \| \rho_1 \| \ldots \| \boldsymbol{\pi}_{r'-1}, \{p_{r'-1,i}\}_i \| \rho_{r'-1})$, set the verifier state to cvs, and send the message $(\mathbb{w}, \boldsymbol{\pi}_{r'})$ to the PIOP verifier. Receive the challenge $\rho_{r'}$ from the verifier. Finally, program the random oracle adding $(s_i, \mathsf{aux}_i, \rho_{r'})$ to $\mathcal{Q}_{\mathsf{RO}}$.
4. Eventually the adversary outputs a valid forgery $(\mathbb{x}_{\Pi}, \pi_{\Pi})$. From π_{Π} derive the (full) PIOP transcript $\mathtt{trns} := (\mathbb{i}, \mathbb{x}, \bar{\pi}_1, \rho_1, \ldots, \bar{\pi}_r)$. Let i be the index of RO query of the partial transcript $(\mathbb{i}, \mathbb{x}, \bar{\pi}_1, \rho_1, \ldots, \bar{\pi}_{r-1})$; as described in the previous step, find the cvs in $\mathsf{SeenStates}$ associated with s_i, set the verifier state to cvs, extract the (last) witness polynomials $\mathbb{w}$ and send $(\mathbb{w}, \boldsymbol{\pi}_r)$ to the verifier. The state cvs and the last messages $(\mathbb{w}, \boldsymbol{\pi}_r)$ define a full transcript: this would trigger the verifier to perform the decision phase of the PIOP and set the decision bit d of the game.

Let $\mathbf{H}_0$ be the $\mathbf{Exp}^{sr}_{\mathcal{A}_{\mathsf{IP}},\mathsf{IP}}(\mathbb{F})$. By the state-restoration knowledge extractability of IP:

$$\Pr[\mathbf{H}_0] \in \mathsf{negl}(|\mathbb{F}|)$$

Consider $\mathbf{H}_1$ that additionally returns 1 if the adversary $\mathcal{A}_{\Pi}$ copies a simulated transcript up to and including the last round from a simulated proof π'_{Π}.

Lemma 7. $\Pr[\boldsymbol{H}_1] \leq \Pr[\boldsymbol{H}_0] + \epsilon_{\mathsf{lin}}$

Proof. We have that the forged proof π_{Π} and a simulated proof π'_{Π} share the same commitments $\mathbf{c}$, the same messages $\boldsymbol{\pi}$, and therefore the same set of instances $(\hat{\mathbb{x}}_k)_{k\in[n_e]}$.

We parse $\pi_{\Pi} = (\mathbf{c}, \boldsymbol{\pi}, (\pi_k)_{k\in[n_e]})$ and the simulated proof as $\pi'_{\Pi} = (\mathbf{c}, \boldsymbol{\pi}, (\pi'_k)_{k\in[n_e]})$, if the two hybrids diverge then there must be an index $k \in [n_e]$ such that the proof $\pi_k \neq \pi'_k$, where, depending on the index k,π'_k was either honestly generated (for $k \in \mathcal{K}_1$) or was simulated using $\Pi_{\mathsf{lin}}.\mathcal{S}_1$ (for $k \in \mathcal{K}_0$).

Let $\pi_k = (\bar{\pi}, (y_j)_j)$ and $\pi'_k = (\bar{\pi}', (y'_j)_j)$, by case analysis, either $\exists j : y_j \neq y'_j$ or $\forall j : y_j = y'_j$ and $\bar{\pi} \neq \bar{\pi}'$.

The latter case cannot happen by the uniqueness of KZG proofs, notice the uniqueness of KZG proofs holds even when the trapdoor is known by the adversary. The former case is covered by the $\Phi^{\mathcal{J},\nu}_{\mathsf{lin}_+}$-simulation extractability of Π_{lin}. We describe below a reduction.

Reduction $\mathcal{B}(\mathsf{srs}, \mathsf{pp}_{\Phi_{\mathsf{lin}}})$

1. Run $\mathcal{A}_{\Pi}(\mathsf{srs})$.
2. Upon query $((\mathtt{i}, \mathbb{x}), \mathsf{aux})$ to the simulation oracle, run the same strategy of $\mathcal{S}_1$ defined in Fig. 7, namely using $\Pi_{\mathsf{lin}}.\mathcal{S}_1$.
3. Given the forgery $((\mathtt{i}, \mathbb{x}_{\Pi}), \pi_{\Pi})$ output by $\mathcal{A}_{\Pi}$, find the instance $\hat{\mathbb{x}}_k$ and the corresponding proof π_k; submit the forgery $(\hat{\mathbb{x}}_k, \pi_k)$

We recall that the policy $\Phi^{\mathcal{J},\nu}_{\mathsf{lin}+}$, since the instance for π_k and π'_k is the same, holds when $\exists j : y_j \neq y'_j$. Thus, by the Theorem 3, we bound the probability that the $\mathcal{B}$ wins to ϵ_{lin}.

Consider the hybrid $\mathbf{H}_2$ that, for all $i \in [n]$, computes $p_i \leftarrow \mathcal{E}_{\mathsf{Com}}(\mathsf{c}_i)$ and, additionally, returns 1 if

$$\exists k \in \mathcal{K}_2 : \sum_i G_i(p_{\gamma_1}(x), \dots p_{\gamma_m}(x), x) \cdot p_{\beta_i}(x) \neq y \quad \vee$$
$$\exists j \in \mathcal{P}_r : \mathsf{c}_j \neq [p_j(s)]_1$$

where $\hat{\mathbb{x}}_k = \big((\mathsf{c}_{\gamma_j})_j, (\mathsf{c}_{\beta_i})_i, (G_i)_i, x, y\big)$.

Lemma 8. $\Pr[\boldsymbol{H}_2] \leq \Pr[\boldsymbol{H}_1] + |\mathcal{K}_2| \cdot \epsilon_{\mathsf{lin}}$

Proof. Notice that, because of the winning condition added in $\mathbf{H}_1$, we can focus on the case in which the adversary $\mathcal{A}_{\Pi}$ does not copy a simulated transcript (up to and including the last round) from a simulated proof π'_{Π}. We also notice that, by our simplifying assumption on the order of the verifier's queries, we have $\mathcal{K}_2 = \{1, \dots, |\mathcal{K}_2|\}$, namely, the indexes in $\mathcal{K}_2$ are consecutive numbers.

We prove this lemma through a series of hybrids. Let $\mathbf{H}_{1,0} \equiv \mathbf{H}_1$. For any $k > 1$, let $\mathbf{H}_{1,k}$ be the same as $\mathbf{H}_{1,k}$ but that additionally returns 1 if:

$$\sum_i G_i(p_{\gamma_1}(x), \dots, p_{\gamma_m}(x), x) p_{\beta_i}(x) \neq y \quad \vee$$
$$\exists i : \mathsf{c}_{\beta_i} \neq [p_{\beta_i}(s)]_1 \vee \exists j : \mathsf{c}_{\gamma_j} \neq \big[p_{\gamma_j}(s)\big]_1$$

where $\hat{\mathbb{x}}_k = \big((\mathsf{c}_{\gamma_j})_j, (\mathsf{c}_{\beta_i})_i, (G_i)_i, x, y\big)$. Finally, we have that $\mathbf{H}_2 \equiv \mathbf{H}_{1,|\mathcal{K}_2|}$.

Let $\mathcal{I} \coloneqq \mathcal{CI}(\hat{\mathbb{x}}_k)$ be the *compiler-safe index set* of $\hat{\mathbb{x}}_k$ (namely, the set of polynomials that the PIOP's specification guarantees we can extract from the instance $\hat{\mathbb{x}}_k$, see Section 7). We reduce to the $(\boldsymbol{\Phi}^{\mathcal{I},\nu}_{\mathsf{lin}}, \mathcal{F}_{\mathcal{I},\nu})$-simulation extractability of Π_{lin}. We define the reduction $\mathcal{B}_{\mathsf{lin},k}$.

Reduction $\mathcal{B}_{\mathsf{lin},k}(\mathsf{srs}, \mathsf{pp}_{\Phi_{\mathsf{lin}}})$

1. Run $\mathcal{A}_{\Pi}(\mathsf{srs})$.
2. Upon query $((\mathtt{i}, \mathbb{x}), \mathsf{aux})$ to the simulation oracle, run the same strategy of $\mathcal{S}_1$ defined in Fig. 7 and use oracle access to $\Pi_{\mathsf{lin}}.\mathcal{S}_1$.
3. Upon query to $\mathcal{S}_2$:
   (a) If it can be parsed as a partial transcript $\mathtt{trns} = (\mathsf{vk}_{\mathtt{i}}, \mathbb{x}, \bar{\pi}_1, \dots, \bar{\pi}_r)$, derive the list of single-variable polynomials $(\tilde{v}_{k'})_{k'} \leftarrow \tilde{\mathsf{V}}_2(\mathbb{F}, \mathtt{i}, \mathbb{x})$ and forward the query to $\Pi_{\mathsf{lin}}.\mathcal{S}_2$ adding the polynomial $(\tilde{v}_k(X))$ to the auxiliary information.
   (b) Otherwise, simply forward the query to the simulator
4. Given the forgery $((\mathtt{i}, \mathbb{x}_{\Pi}), \pi_{\Pi})$ output by $\mathcal{A}_{\Pi}$, define the instance $\hat{\mathbb{x}}_k$ and the corresponding proof $\hat{\pi}_k$.
5. For $i \in \mathsf{indexes}(\mathcal{J}_{k-1})$, run $p_i \leftarrow \mathcal{E}_{\mathsf{Com}}(\mathsf{c}_i)$
6. Return the forgery $(\hat{\mathbb{x}}_k, \hat{\pi}_k)$ and set the auxiliary input $\mathsf{aux}_{\mathcal{E}}$ as the adversary $\mathcal{A}_{\Pi}$ does and include the polynomials $(p_i)_i$ extracted at the previous step.

By inspection, if the forgery of $\mathcal{A}_{\Pi}$ passes the verification equation, then the forgery of $\mathcal{B}_{\mathsf{lin},k}$ passes the verification equation too.

Since the adversary $\mathcal{A}_{\Pi}$ has not fully copied the transcript from any simulated proof, the random coin ρ_r computed by the verifier to verify the proof π_{Π} is, with overwhelming probability, a *fresh* coin, which

implies that the simulator has never output a proof on it. Moreover, since the check $\hat{\mathbb{x}}_k$ is *focal*, the point $\hat{\mathbb{x}}_k.x$ is equal to $\tilde{v}_k(\rho_r)$ and $\mathsf{deg}_X(\tilde{v}) \geq 1$. The forgery of the reduction satisfies the Hash Check of Φ_{lin} because the reduction adds $\tilde{v}_k(X)$ to the auxiliary information of the RO query including all the commitments of $\hat{\mathbb{x}}_k$ at Item 3a. Moreover, when the distinguishing event between the two consecutive hybrids happens, the Partial-Extraction Check of the policy is satisfied:

- For $k = 1$ since, by definition, the index polynomials $(p_i)_{i \in n(0)}$ are honestly generated and therefore equal to what $\mathcal{E}_{\mathsf{Com}}$ extracts.
- For $k > 1$, because of the changes introduced in the hybrids $\mathbf{H}_{1,1}, \dots \mathbf{H}_{1,k-1}$, the list of polynomials extracted in Item 5 are correctly extracted and included in the auxiliary information.

Finally, because of the compilation-safeness property, the linear independence check between the *left* polynomials of $\hat{\mathbb{x}}_k$ allows us to conclude that the forgery of $\mathcal{B}_{\mathsf{lin},k}$ matches the policy, while the distinguishing event asserts that the extractor fails. This bounds the probability of the distinguishing event to be at most ϵ_{lin}. □

Consider $\mathbf{H}_3$ that additionally returns 1 if the adversary $\mathcal{A}_{\Pi}$ copies a simulated transcript up to r-th round from a simulated proof π'_{Π}.

Lemma 9. $\Pr[\boldsymbol{H}_3] \leq \Pr[\boldsymbol{H}_2] + \epsilon_{\mathsf{del}}$

Proof. In this case, we have that π_{Π} and π'_{Π} use the same commitments $\mathbf{c}$, the same messages $\boldsymbol{\pi}$ in the first r rounds, and there must be at least one commitment or a message in the last round that is different. Because of the change introduced in the previous hybrid, the distinguishing event focuses on the case in which the commitments sent in the last round can be extracted using $\mathcal{E}_{\mathsf{Com}}$, and moreover they satisfy the *focal* checks $\mathbb{x}_k$, for $k \in \mathcal{K}_2$.

However, by the uniqueness of PIOP with delegation phase property (see Definition 24), we have that the probability of the distinguishing event, which implies two different IP_2-transcripts for the same instance that differ on the first prover message (and polynomials) is $\epsilon_{\mathsf{del}} \in \mathsf{negl}(\log|\mathbb{F}|)$. □

Consider $\mathbf{H}_4$ that (similarly to $\mathbf{H}_2$) additionally returns 1 if

$$\exists k \in \mathcal{K}_1 : \sum_i G_i(p_{\gamma_1}(x), \dots p_{\gamma_m}(x), x) \cdot p_{\beta_i}(x) \neq y \quad \vee$$
$$\exists j : \mathsf{c}_j \neq [p_j(s)]_1$$

where $\hat{\mathbb{x}}_k = \big((\mathsf{c}_{\gamma_j})_j, (\mathsf{c}_{\beta_i})_i, (G_i)_i, x, y\big)$.

Lemma 10. $\Pr[\boldsymbol{H}_4] \leq \Pr[\boldsymbol{H}_3] + |\mathcal{K}_1| \cdot \epsilon_{\mathsf{lin}}$

Proof. The proof of this lemma proceeds almost identically to Lemma 7. The main difference is that, by the definition of compilation-safeness, a focal check $\hat{\mathbb{x}}_k$ with $k \in \mathcal{K}_1$ queries the polynomials at point $x = \tilde{v}_k(\rho_1, \dots, \rho_{r-1})$ where $\mathsf{deg}_{X_{r-1}}(\tilde{v}_k) \geq 1$, thus we need to use the change introduced in $\mathbf{H}_3$ to make sure that the challenge ρ_{r-1} is different than in all the simulation proofs, and thus the evaluation point x is *fresh*, namely that the simulator of Π_{lin} has never simulated a proof with x as evaluation point.

For completeness, we give the full proof of lemma.

We start noticing that, by our simplifying assumption on the order of the verifier's queries, we have that the indexes in $\mathcal{K}_1$ are consecutive numbers, in particular $\mathcal{K}_1 \coloneqq \{|\mathcal{K}_2| + 1, \dots, |\mathcal{K}_2| + |\mathcal{K}_1|\}$.

We prove this lemma through a series of hybrids. Let $\mathbf{H}_{3,|\mathcal{K}_2|} \equiv \mathbf{H}_3$. For any $k > |\mathcal{K}_2|$, let $\mathbf{H}_{3,k}$ be the same as $\mathbf{H}_{3,k}$ but that additionally returns 1 if:

$$\sum_i G_i(p_{\gamma_1}(x), \dots, p_{\gamma_m}(x), x) p_{\beta_i}(x) \neq y \quad \vee$$
$$\exists i : \mathsf{c}_{\beta_i} \neq [p_{\beta_i}(s)]_1 \vee \exists j : \mathsf{c}_{\gamma_j} \neq \left[p_{\gamma_j}(s)\right]_1$$

where $\hat{\mathbb{x}}_k = \big((\mathsf{c}_{\gamma_j})_j, (\mathsf{c}_{\beta_i})_i, (G_i)_i, x, y\big)$. Finally, we have that $\mathbf{H}_4 \equiv \mathbf{H}_{3,|\mathcal{K}_2|+|\mathcal{K}_1|}$.

Let $\mathcal{I} \coloneqq \mathcal{CI}(\hat{\mathbb{x}}_k)$ be the *compiler-safe index set* of $\mathbb{x}_k$. We reduce to the $(\boldsymbol{\Phi}_{\mathsf{lin}}^{\mathcal{I},\nu}, \mathcal{F}_{\mathcal{I},\nu})$-simulation extractability of Π_{lin}. We make use of a reduction similar to the one used in the proof of Lemma 7.

Reduction $\mathcal{B}_{\mathsf{lin},k}(\mathsf{srs}, \mathsf{pp}_{\Phi_{\mathsf{lin}}})$

1. Run $\mathcal{A}_{\Pi}(\mathsf{srs})$.
2. Upon query $((\mathbb{i}, \mathbb{x}), \mathsf{aux})$ to the simulation oracle, run the same strategy of $\mathcal{S}_1$ defined in Fig. 7, namely using $\Pi_{\mathsf{lin}}.\mathcal{S}_1$.
3. Upon query to $\mathcal{S}_2$:
   (a) If it can be parsed as a partial transcript $\mathtt{trns} = (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_{r-1})$. derive the list $(\tilde{v}_{k'})_{k'} \leftarrow \tilde{\mathsf{V}}_2(\mathbb{F}, \mathbb{i}, \mathbb{x})$. Forward the query to $\Pi_{\mathsf{lin}}.\mathcal{S}_2$ adding the single-variable polynomial $(\tilde{v}_k(\rho_1, \ldots, \rho_{r-2}, X))_j$ to the auxiliary information.
   (b) Otherwise, simply forward the query to the simulator
4. Given the forgery $((\mathbb{i}, \mathbb{x}_{\Pi}), \pi_{\Pi})$ output by $\mathcal{A}_{\Pi}$, define the instance $\hat{\mathbb{x}}_k$ and the corresponding proof $\hat{\pi}_k$.
5. For $i \in \mathsf{indexes}(\mathcal{J}_{k-1})$, run $p_i \leftarrow \mathcal{E}_{\mathsf{Com}}(\mathsf{c}_i)$
6. Return the forgery $(\hat{\mathbb{x}}_k, \hat{\pi}_k)$ and set the auxiliary input $\mathsf{aux}_{\mathcal{E}}$ as the adversary $\mathcal{A}_{\Pi}$ does and include the polynomials $(p_i)_i$ extracted at the previous step.

Since the adversary $\mathcal{A}_{\Pi}$ has not copied, up to the r-th round, the transcript from any simulated proof, the last random coin ρ_{r-1} computed by the verifier to verify the proof π_{Π} is, with overwhelming probability, a *fresh* coin. Moreover, since the check $\hat{\mathbb{x}}_k$ is *focal*, the point $\hat{\mathbb{x}}_k.x$ is equal to $\tilde{v}_k(\rho_1, \ldots \rho_{r-1})$ and $\mathtt{deg}_{X_{r-1}}(\tilde{v}_k) \geq 1$. The forgery of the reduction satisfies the Hash Check of Φ_{lin} because the polynomial $\tilde{v}_k(\rho_1, \ldots, \rho_{r-2}, X)$ is added to the auxiliary information of the RO query including all the commitments of $\hat{\mathbb{x}}_k$. Similarly to the proof of Lemma 7, the Partial-Extraction Check of the policy is also satisfied.

Finally, because of the compilation-safeness property, the linear independence check between the *left* polynomials of $\hat{\mathbb{x}}_k$ allows us to conclude that the extractor would also extract its *core* and the *right* polynomials. By inspection, the list of polynomials extracted by $\mathcal{E}_{\Pi_{\mathsf{lin}}}$ is equal to $(p_i)_i$ at the step 5 of the reduction. □

Consider $\mathbf{H}_5$ that additionally returns 1 if

$$\exists k \in [n_e] : \sum_i G_i(p_{\gamma_1}(x), \ldots p_{\gamma_m}(x), x) \cdot p_{\beta_i}(x) \neq y \quad \vee$$
$$\exists j : \mathsf{c}_j \neq [p_j(s)]_1$$

where $\hat{\mathbb{x}}_k = \big((\mathsf{c}_{\gamma_j})_j, (\mathsf{c}_{\beta_i})_i, (G_i)_i, x, y\big)$. Namely, if the polynomials extracted above do not satisy the non-focal checks of V.

Lemma 11. $\Pr[\boldsymbol{H}_5] \leq \Pr[\boldsymbol{H}_4] + (n_e - |\mathcal{K}_f|) \cdot \epsilon_{\mathsf{lin}}$

Proof. We can show a reduction to the $\Phi_{\mathsf{lin}+}$-simulation extractability of Π_{lin}. Similarly to the proof of Lemma 7, the reduction isolates the pair $(\hat{\mathbb{x}}_k, \pi_k)$ such that the distinguish event happens. Using the polynomials $p_i \leftarrow \mathcal{E}_{\mathsf{Com}}(\mathsf{c}_i)$ that are a valid witness for the *focal* checks because of the change introduced before, the reduction can create an algebraic inconsistent proof: given the result of Theorem 3, we can be bound the probability of such an event to ϵ_{lin}. □

Finally, we prove that the probability that $\mathcal{A}_{\mathsf{IP}}$ wins in $\mathbf{H}_5$ is equal to the probability that $\mathcal{A}_{\Pi}$ wins the strong simulation-extractibility experiment.

Lemma 12. $\Pr[\boldsymbol{H}_5] = \mathbf{Adv}^{\Phi_{\mathsf{SE}}\text{-}se}_{\mathcal{A}_{\Pi}, \mathcal{S}, \mathcal{E}_{\Pi}}(\lambda)$

Proof. We now show that the probability that $\mathbf{H}_5$ outputs 1 is equal to the probability that the adversary $\mathcal{A}_{\Pi}$ wins the Φ_{SE}-se experiment against $\mathcal{E}_{\Pi}$. First, we notice that srs is generated by $\mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$ in both experiments. Because of the checks introduced in $\mathbf{H}_2$, $\mathbf{H}_4$ and $\mathbf{H}_5$, upon a valid forgery $((\mathbb{i}, \mathbb{x}_{\Pi}), \pi_{\Pi})$, we have that:

– The proof π_{Π} cannot contain a simulated element (see Item 3 of $\mathcal{A}_{\mathsf{IP}}$). In fact, because of $\mathbf{H}_4$, all the polynomials can be extracted, and therefore cannot be simulated. Thus, all the RO queries of $\mathcal{A}_{\Pi}$ that constitute π_{Π} (namely the queries $q_1, \ldots, q_r$) are forwarded to the challenger in Item 3b of $\mathcal{A}_{\mathsf{IP}}$ (in particular, any of the queries is already answered by the programming of the RO by the simulator $\mathcal{S}_1$). This implies that the complete transcript sent by $\mathcal{A}_{\mathsf{IP}}$ is in the list $\mathsf{SeenStates}$.

– The extractor $\mathcal{E}_{\Pi}$ does not abort neither at Item 5 nor at Item 6 because the polynomials $(p_i)_i$ extracted by $\mathcal{E}_{\mathsf{Com}}$ satisfy the linearized checks $\hat{\mathfrak{tx}}_k$, for all $k \in [n_e]$. This implies that the extractor $\mathcal{E}_{\mathsf{IP}}$ in the $\mathbf{Exp}^{sr}_{\mathcal{A}_{\mathsf{IP}},\mathsf{IP}}$ in $\mathbf{H}_5$ is fed with the same polynomials extracted by $\mathcal{E}_{\Pi}$.

Thus, the decision bit in the state-restoration game is 1. □

Having bound ϵ_5 along the hybrids concludes the proof.

Acknowledgements This work has received funding from the MESRI-BMBF French-German joint project named PROPOLIS (ANR-20-CYAL-0004-01), the CHIST-ERA project PATTERN (ANR-23-CHR4-0008), 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) funded by MCIN/AEI/10.13039/501100011033/ and the European Union NextGenerationEU/PRTR, and ESPADA (PID2022-142290OB-I00) funded by MCIN/AEI/10.13039/501100011033/ FEDER, UE.

References

ark21. arkworks. Diagram of marlin's prover and verifier, including optimizations, 2021. https://github.com/arkworks-rs/marlin/blob/master/diagram/diagram.pdf.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, January 2012.

BCF+21. Daniel Benarroch, Matteo Campanelli, Dario Fiore, Kobi Gurkan, and Dimitris Kolonelos. Zero-knowledge proofs for set membership: Efficient, succinct, modular. In Nikita Borisov and Claudia Díaz, editors, *FC 2021, Part I*, volume 12674 of *LNCS*, pages 393–414. Springer, Heidelberg, March 2021.

BCKL08. Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 356–374. Springer, Heidelberg, March 2008.

BCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.

BFHK23. Balthazar Bauer, Pooya Farshim, Patrick Harasser, and Markulf Kohlweiss. The uber-knowledge assumption: A bridge to the AGM. Cryptology ePrint Archive, Paper 2023/1601, 2023. https://eprint.iacr.org/2023/1601.

BFL20. Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 121–151. Springer, Heidelberg, August 2020.

BFS20. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020.

Boy08. Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, Heidelberg, September 2008.

CBBZ23. Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 499–530. Springer, Heidelberg, April 2023.

CFF+21. Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2021.

CFH+22. Matteo Campanelli, Dario Fiore, Semin Han, Jihye Kim, Dimitris Kolonelos, and Hyunok Oh. Succinct zero-knowledge batch proofs for set accumulators. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 455–469. ACM Press, November 2022.

CFQ19. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019.

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

DDO+01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001.

DG23. Quang Dao and Paul Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 531–562. Springer, Heidelberg, April 2023.

FFK+23. Antonio Faonio, Dario Fiore, Markulf Kohlweiss, Luigi Russo, and Michal Zajac. From polynomial IOP and commitments to non-malleable zkSNARKs. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part III*, volume 14371 of *LNCS*, pages 455–485. Springer, Heidelberg, November / December 2023.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

FKMV12. Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Heidelberg, December 2012.

GKK+22. Chaya Ganesh, Hamidreza Khoshakhlagh, Markulf Kohlweiss, Anca Nitulescu, and Michal Zajac. What makes fiat-shamir zksnarks (updatable SRS) simulation extractable? In Clemente Galdi and Stanislaw Jarecki, editors, *Security and Cryptography for Networks, SCN 2022*, volume 13409 of *Lecture Notes in Computer Science*, pages 735–760. Springer, 2022.

GKM+18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018.

GM17. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, August 2017.

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

GOP+22. Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 397–426. Springer, Heidelberg, May / June 2022.

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

Ish19. Yuval Ishai. Efficient zero-knowledge proofs: A modular approach. https://simons.berkeley.edu/talks/tbd-79. Also see https://zkproof.org/2020/08/12/information-theoretic-proof-systems/, 2019.

Ish20. Yuval Ishai. Zero-Knowledge Proofs from Information-Theoretic Proof Systems - Part I. ZKProof.org, Blog entry, August 2020. Also see https://zkproof.org/2020/08/12/information-theoretic-proof-systems/.

JR13. Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013.

KPT23. Markulf Kohlweiss, Mahak Pancholi, and Akira Takahashi. How to compile polynomial IOP into simulation-extractable SNARKs: A modular approach. In Guy N. Rothblum and Hoeteck Wee, editors,

TCC 2023, Part III, volume 14371 of *LNCS*, pages 486–512. Springer, Heidelberg, November / December 2023.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010.

LPS23. Helger Lipmaa, Roberto Parisella, and Janno Siim. Algebraic group model with oblivious sampling. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 363–392. Springer, Heidelberg, November / December 2023.

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

Mic94. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

OL. O1-Labs. Maller's Optimization to Reduce Proof Size - Mina Book. https://o1-labs.github.io/proof-systems/plonk/maller.html.

RS20. Lior Rotem and Gil Segev. Algebraic distinguishers: From discrete logarithms to decisional uber assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 366–389. Springer, Heidelberg, November 2020.

RZ21. Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, August 2021. Springer, Heidelberg.

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

Sze20. Alan Szepieniec. Polynomial IOPs for linear algebra relations. Cryptology ePrint Archive, Report 2020/1022, 2020. https://eprint.iacr.org/2020/1022.

TZ23. Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 691–721. Springer, Heidelberg, April 2023.

Zha22. Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Heidelberg, August 2022.

ZZ21. Mark Zhandry and Cong Zhang. The relationship between idealized models under computationally bounded adversaries. Cryptology ePrint Archive, Report 2021/240, 2021. https://eprint.iacr.org/2021/240.

A The OMSDH Assumption

Definition 30. *The* non-adaptive *n-one-more d-strong DH assumption holds for a bilinear group generator* $\mathsf{GroupGen}$ *if for any set $\mathcal{Q}_x$ of cardinality less or equal to n, and for every PPT adversary whose queried points belongs to $\mathcal{Q}_x$, namely, for the class of adversaries whose query points are chosen independently of the randomness of the experiment, the advantage* $\mathbf{Adv}^{(n,d)\text{-}\mathtt{OMSDH}}_{\mathsf{GroupGen},\mathcal{A}}(\lambda)$ *is negligible.*

The following theorem shows that the non-adaptive OMSDH assumption is equivalent to the SDH assumption.

Theorem 5. *For any* $\mathsf{GroupGen}$*, any $n, d \in \mathbb{N}$ and a bound L, for any $\mathcal{Q}_x \subset \mathbb{F}$ of cardinality* $\mathsf{poly}(\lambda)$ *and for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that:*

$$\mathbf{Adv}^{(n,d)\text{-}\mathtt{OMSDH}}_{\mathsf{GroupGen},\mathcal{A}}(\lambda) = \mathbf{Adv}^{(n^2+d+1)\text{-}\mathtt{SDH}}_{\mathsf{GroupGen},\mathcal{B}}(\lambda).$$

Moreover, let $\mathcal{Q}_{\mathcal{A}}$ the query made by an adversary $\mathcal{A}$. For any $m \in \mathbb{N}$, for any PPT $\mathcal{A}$ such that $\max\{i : (x,i) \in \mathcal{Q}_{\mathcal{A}}\} \leq m$*:*

$$\mathbf{Adv}^{(n,d)\text{-}\mathtt{OMSDH}}_{\mathsf{GroupGen},\mathcal{A}}(\lambda) = \mathbf{Adv}^{(nm+d+1)\text{-}\mathtt{SDH}}_{\mathsf{GroupGen},\mathcal{B}}(\lambda).$$

In the proof, we define a reduction to the SDH assumption and whose simulation strategy works, similarly to [FFK+23,TZ23], only for queries on the points $\mathcal{Q}_x$.

Proof. Let m be (an upper bound to) the maximum power i queried to $\mathcal{O}_s$ and let $q' := nm + d + 1$, notice that the worst case is $m = n$. We show a reduction $\mathcal{B}$ to the q'-SDH assumption.

$\mathcal{B}$ takes as input $\mathsf{srs}' \leftarrow (\left[(s^i)_{i\in[0,q']}\right]_1, [1,s]_2)$ and defines the new SRS $\mathsf{srs} \leftarrow (\left[(p(s)s^i)_{i\in[0,d]}\right]_1, [1,s]_2)$, where $p(X) := (X - x_r)\prod_{x\in\mathcal{Q}_x}(X-x)^n$ and x_r is a random point. In particular, the group generator $[1]_1$ given to the adversary is randomized (and equal to $[p(s)]_1$ in the basis of the reduction).

The reduction can easily answer, for any $x_j \in \mathcal{Q}_x$ and any $i \leq m$, when the adversary queries the oracle $\mathcal{O}_s$ with (x_j, i): it just computes $z \leftarrow \left[p(s)(s-x_j)^{-i}\right]_1$, that is a valid output since $e(z, [s-x_j]_2^i) = [1]_T$.

Finally, notice that the forgery for $\mathcal{A}$ is $[p(s)/(s-x^*)]_1$. If we compute the euclidean division between polynomials we obtain $q(X)$ and r such that:

$$\frac{p(X)}{(X-x^*)} = q(X) + \frac{r}{X-x^*}.$$

Noticing that $x^* \notin \mathcal{Q}_x$ implies, by construction of $p(X)$, that x^* does not divide $p(X)$, then $r \neq 0$. Therefore the forgery of the reduction $\mathcal{B}$ is set to $(\mathsf{y} - [q(s)]_1)r^{-1}$. □

Hereafter, we show that the OMSDH assumption holds in the generic bilinear Maurer's version of the GGM [Mau05,ZZ21], in which an adversary can access elements from the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ only via abstract handles. These are maintained by the challenger in lists L_1, L_2, and L_T, which correspond to the three groups. Similarly to [BFHK23], we consider the variant proposed by Zhandry [Zha22], where the adversary cannot choose the handle where the result is stored nor access handles not explicitly given as an oracle reply.

One-more d-SDH in the Maurer's GGM. The list L_1 initially contains the handles to the elements $1, Z, \ldots, Z^d$, and the list L_2 contains the handle to the elements $1, Z$. The challenger samples $z \leftarrow\!\!\$\ \mathbb{F}_q$ as the solution: the goal is to prove that z remains information-theoretically hidden from the adversary. Since the OMSDH assumption is interactive, we also give the adversary access to the oracle $\mathcal{O}_s$, and the handles returned by this oracle are stored in the list L_s.

The adversary is granted access to three types of oracles:

Group Oracles: for $i \in \{1, 2, T\}$, the oracle $\mathcal{O}_i$ takes as input two handles $h_1, h_2 \in L_i$ for polynomials $P_1(Z)$, $P_2(Z)$ and outputs a handle h for the polynomial $P_1(Z) + P_2(Z)$; L_i is accordingly updated with the handle h.

Paring Oracle: the oracle $\mathcal{O}_e$ on input two handles $h_1 \in L_1$ and $h_2 \in L_2$ for $P_1(Z)$ and $P_2(Z)$, outputs the handle h_T to the element $P_1(Z) \cdot P_2(Z)$ and updates L_T accordingly.

SDH Oracle: The oracle $\mathcal{O}_s$ takes as input a pair (x, i) and returns and handle h_s for $(Z - x)^{-i}$, updating L_s accordingly.

Following the standard argument in the GGM, we need to bound the probability that the so-called *collision event* E occurs, namely that there exist two handles h_1, h_2 that point to two distinct polynomials $P_1(Z)$ and $P_2(Z)$ and such that $P_1(Z) \neq P_2(Z)$, but $P_1(z) = P_2(z)$.

Definition 31. *We say that the n-one-more d-SDH assumption is secure in the Maurer's bilinear GGM if for any no-uniform PT adversary $\mathcal{A}$ with oracle interfaces described above triggers the collision event with almost negligible probability in λ.*

Theorem 6. *The n-one-more d-SDH assumption is secure in the Maurer's bilinear GGM.*

Proof. We notice that any no-uniform PT adversary for the OMSDH assumption in the GGM is inherently non-adaptive. In fact, if we run twice $\mathcal{A}$ then it would output the same queries to its SDH oracle, unless in one of the executions the collision event happens, in that case, the set of queries made by $\mathcal{A}$ in one execution is a subset of the set of queries made by $\mathcal{A}$ in the other execution.

Finally, by the composition lemma of the AGM [FKL18, Lemma 2.2], by Theorem 5 and since the SDH assumption holds in the Maurer's GGM, the theorem follows. □