Real-World Universal zkSNARKs are Non-Malleable

Antonio Faonio*
EURECOM
Biot, France
antonio.faonio@eurecom.fr

Dario Fiore*
IMDEA Software Institute
Madrid, Spain
dario.fiore@imdea.org

Luigi Russo*
EURECOM
Biot, France
russol@eurecom.fr

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.

CCS CONCEPTS

• **Security and privacy → Cryptography**.

KEYWORDS

Polynomial Commitments, Simulation extractability, Zero-knowledge

ACM Reference Format:
Antonio Faonio, Dario Fiore, and Luigi Russo. 2024. Real-World Universal zkSNARKs are Non-Malleable. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3658644.3690351

1 INTRODUCTION

Zero-knowledge proofs [22] 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) [6, 32] 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 property 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 [36], who introduced the concept of simulation soundness, subsequently expanded upon as simulation extractability (SE, for brevity) [16].

For zkSNARKs, the notion of SE was first studied by Groth and Maller [24] 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 [15, 17, 20, 21, 29], motivated by this state of affairs, prove the SE of existing zkSNARKs, such as Bulletproofs [8], Spartan [15], Sonic [31], PLONK [19], Marlin [14], Lunar [10] and Basilisk [34]. We can divide these results into two main categories: those such as [15, 20, 21] that prove the SE of specific zkSNARKs; those like [17, 29] 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.*

*All authors contributed equally to this research.

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690351

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[1] 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) [19, 33] 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 [17, 29].

Delegation phase The second gap is that the aforementioned frameworks of Faonio *et al.* [17] and Kohlweiss *et al.* [29] 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 [17, 29] 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 [17] 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 [17] 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) [30].

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, [17] 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 [17], we provide a more comprehensive explanation of our results in the following section.

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

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 a zkSNARK via the use of a computationally-secure primitive [25, 26]. The most popular instantiation of this approach uses PIOP [9, 10, 14, 19, 38] for the information-theoretic part, and polynomial commitments [28] 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 [9, 14]. Other PIOP variants [10, 19] 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 [13]. Under this definition, Marlin uses an $\mathcal{R}_{\mathsf{evl}}$-PIOP, where $\mathcal{R}_{\mathsf{evl}}$ is the relation that checks that a polynomial oracle evaluates to y at point x, while PLONK [19] and Lunar [10] are $\mathcal{R}_{\mathsf{poly}}$-PIOPs, where $\mathcal{R}_{\mathsf{poly}}$ is the relation that checks polynomial equations over polynomial oracles.

How to compile $\mathcal{R}$-PIOPs? Unfortunately, zkSNARKs obtained by (mechanically) applying compilations from $\mathcal{R}_{\mathsf{evl}}$-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 [28], have. Thus, subsequent optimizations usually accompany, and slightly change, the formally analyzed zkSNARKs. Instead, zkSNARKs compiled from $\mathcal{R}_{\mathsf{poly}}$-PIOPs can defer all the optimizations to the richer and more expressive (sub)-proof systems for $\mathcal{R}_{\mathsf{poly}}$.[2] 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 [19, 33]. This optimization allows reducing the number of field elements in the final proofs.

[2] 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.

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 [19] describes a general recipe to compile an $\mathcal{R}_{\mathsf{poly}}$-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.[3]

On the (in)security of the linearization trick. It turns out that this general recipe is not always sound. In fact, the work of [30] 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 [30], 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 [30], 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 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 [30] 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 [19] 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.[4] 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 4.1). Second, we analyze the sufficient conditions on the A_i polynomials that make the linearization trick simulation extractable (cf. Sections 2.3 and 4.2).

2.2 SE of KZG for linearized commitments

The work of [17] 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.[5] 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. [17] 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 obtain 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 Q_x. This property is called semi-adaptive queries.
(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

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

[4] 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.

[5] 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.

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 [17]. 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[6] $(A_i)_i$ 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 4.3. *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 [30]. 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 2.1. *PLONK and Marlin are knowledge-sound in the AGMOS.*

2.3 SE of the linearization trick

Unfortunately, when the adversary can make simulation oracle calls, 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 to 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:

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

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[7], and so on.

Informal Statement of Theorem 4.5. *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 [17] 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 reuse a simulated proof for the first instance in $\mathcal{R}$ and honestly prove the knowledge for the second instance. To avoid these pathological cases, [17] 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 [14] and other proof systems [10, 34] based on Checkable Subspace Sampling Arguments [34] are not compilation-safe. They have a two-phase algorithm for the prover where the first

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

[7] 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 $(X-x_2)^{-1}$.

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[8]. With this characterization of PIOPs we can prove the following theorem:

Informal Statement of Theorem 6.2. *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 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 the 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 5.3 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 $\{1, 2, \ldots, n\}$. 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 3.1. 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 $\forall (\alpha_j)_j \in \mathbb{F}_{\leq \nu}[X]\colon \sum_j \alpha_j A_j(X) \neq 0$.

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 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.

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

$Q_x \leftarrow \emptyset;\quad \mathsf{pp}_\mathbb{G} \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda);\quad s \leftarrow\!\!\$\ \mathbb{F}_q$

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

return $x^* \notin Q_x \wedge \mathsf{y}^* = \left[(s - x^*)^{-1}\right]_1$

Oracle $O_s(x, i)$

if $i > n$:
  return $\bot$

$Q_x \leftarrow Q_x \cup \{x\}$

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

Figure 1: The OMSDH experiment.

Definition 3.2 ((n, d)-OMSDH). Consider the experiment in Fig. 1. The n-one-more d-strong DH assumption holds for a bilinear group generator 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 [2], OMSDH is a special case of an adaptive Uber Assumption for Rational Fractions. When the set of points 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 [2], that is reducible to discrete log in the AGM. We defer the proof to [18].

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, in our work we consider distributions that are witness samplable [27] and Aff-MDH-secure [17]. We give their formal definition in [18].

3.1 SE CP-SNARKS in the AGM

We define a PT relation $\mathcal{R}$ verifying triple $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ as in [23]. 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,

[8] 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.

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 GroupGen) is a tuple of algorithms $\Pi \coloneqq (\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify})$ where: (1) KGen is a probabilistic algorithm that takes as input the group parameters $\mathsf{pp}_{\mathbb{G}} \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$ and outputs $\mathsf{srs} \coloneqq (\mathsf{ek}, \mathsf{vk}, \mathsf{pp})$, where ek is the evaluation key, vk is the verification key, and pp are the parameters for $\mathcal{R}$. (2) Prove takes as input an evaluation key ek, a statement $\mathbb{x}$, and a witness $\mathbb{w}$ s.t. $\mathcal{R}(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ holds, and returns a proof π. (3) Verify takes as input a verification key vk, 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 CP-SNARK is a stateful PPT algorithm that can operate in three modes: (1) $(\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$ takes care of generating the parameters and the simulation trapdoor (if necessary) (2) $(\pi, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(1, \mathsf{st}_{\mathcal{S}}, \mathbb{x})$ simulates the proof for a statement $\mathbb{x}$ (3) $(a, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(2, \mathsf{st}_{\mathcal{S}}, s)$ answers random oracle queries. The state $\mathsf{st}_{\mathcal{S}}$ is updated after each operation. In the case of *non-programmable* random oracle model, $\mathcal{S}$ is notified of the RO queries but cannot control the answers.

Informally, the (perfect) zero-knowledge property states that no (unbounded) adversary on input the srs can distinguish whether it is interacting with the simulator $\mathcal{S}$ or with the real prover and the random oracle. We defer the formal definition to [18].

Commitment Schemes. A commitment scheme with message space $\mathcal{M}$ (and group parameters GroupGen) is a tuple of algorithms $\mathsf{CS} \coloneqq (\mathsf{KGen}, \mathsf{Com}, \mathsf{VerCom})$ 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. For the basic security properties of binding and hiding see [18].

CP-SNARKs. Commit-and-Prove SNARKs, or simply CP-SNARKs, are proof systems whose relations verify predicates over commitments [12]. We refer to a CP-SNARK for a relation $\mathcal{R}$ and a commitment scheme CS as a tuple of algorithms $\Pi \coloneqq (\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify})$ where $\mathsf{KGen}(\mathsf{ck}) \to \mathsf{srs}$ is an algorithm that takes as input a commitment key ck for CS and outputs $\mathsf{srs} \coloneqq (\mathsf{ek}, \mathsf{vk})$[9]; 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.

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 $(x_1, \ldots x_n; y_1, \ldots, y_m) \to_{\mathsf{RO}} a$ to indicate that there is a list of tuples $(s_1, \mathsf{aux}_1), \ldots, (s_k, \mathsf{aux}_k)$ 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

3.2 Policy-Based Simulation Extractability

We recall the definitional framework of [17]. A policy is a tuple $\Phi \coloneqq (\Phi_0, \Phi_1)$ 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 $\mathsf{coms} \coloneqq (\mathsf{c}_i)_i$ 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_1 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 Q_{sim}, and the set Q_{RO} of queries and answers to the random oracle[10]; (iv) Auxiliary information aux_Φ which might come along with the forged instance.[11]

We extend the definitional framework of [17] to the $\mathcal{F}$-extractability setting, introduced by [3], 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 [17]. We define the policy-based SE game in Fig. 2. In the figure, 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 Q_{sim} is the set of queries and answers to the simulation oracle. The set Q_{RO} is the set of queries and answers to the random oracle. The set Q_{aux} is the set of all the auxiliary information sent by the adversary (depending on the policy, this set might be empty or not). All these sets are initially empty and

[9] 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.

[10] As noted in [17], even if the given NIZK is not in the ROM it still makes sense to assume the existence of the set Q_{RO} (e.g., to model security for NIZK protocols that eventually are used as sub-protocols in ROM-based protocols)

[11] 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.

stored in the state of the simulator. The oracles $\mathcal{S}_1$ and $\mathcal{S}_2$ deal respectively with the simulation queries and the random oracle queries of $\mathcal{A}$.

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

$\mathsf{pp}_{\mathbb{G}} \leftarrow\!\$\ \mathsf{GroupGen}(1^\lambda)$
$\mathsf{pp}_\Phi \leftarrow\!\$\ \Phi_0(\mathsf{pp}_{\mathbb{G}})$
$(\mathsf{srs}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$
$(\mathbb{x}, \pi, \mathsf{aux}_{\mathcal{E}}, \mathsf{aux}_\Phi) \leftarrow \mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}(\mathsf{srs}, \mathsf{pp}_\Phi)$
$\mathbb{w}_{\mathcal{F}} \leftarrow \mathcal{E}(\mathsf{srs}, \mathbb{x}, \pi, \mathsf{aux}_{\mathcal{E}})$
$\mathsf{view} \leftarrow (\mathsf{srs}, \mathsf{pp}_\Phi, Q_{\mathsf{sim}}, Q_{\mathsf{RO}}, Q_{\mathsf{aux}})$
$b_\Phi \leftarrow \Phi_1((\mathbb{x}, \pi), \mathsf{view}, \mathsf{aux}_\Phi)$
$b_V \leftarrow \mathsf{Verify}^{\mathcal{S}_2}(\mathsf{srs}, \mathbb{x}, \pi)$
$b_{\mathcal{E}} \leftarrow \forall\, \mathbb{w} : \mathcal{F}(\mathbb{w}) \neq \mathbb{w}_{\mathcal{F}} \vee (\mathsf{pp}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}$
return $(b_\Phi \wedge b_V \wedge b_{\mathcal{E}})$

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

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

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

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

Figure 2: The $(\Phi, \mathcal{F})$-SE experiments in ROM.

Definition 3.3 (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:

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

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.

Finally, we say that a CP-SNARK Π is $(\Phi, \mathcal{F})$-SE *in the AGM* if the definition above holds when enumerating over all PT algebraic adversaries.

3.3 SE for KZG-based CP-SNARKs

We recall the (non-hiding version of the) commitment scheme of Kate, Zaverucha and Goldberg [28] 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:

- $\mathsf{KGen}(\mathsf{pp}_{\mathbb{G}}, d)$ 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 $\mathsf{c} \coloneqq [f(s)]_1$.
- $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, f(X))$ outputs 1 iff $\mathsf{c} = [f(s)]_1$.

KZG commitment scheme allows for simple and efficient *evaluation proofs* which, in the framework of [12], 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 $\pi \coloneqq [\pi(s)]_1$, 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 [14] and in the AGMOS [30]. The ZK simulator is $\mathcal{S} \coloneqq (\mathcal{S}_0, \mathcal{S}_1)$, where $\mathcal{S}_0$ outputs the trapdoor s together with the srs, and $\mathcal{S}_1$ simulates proofs for $\mathbb{x} \coloneqq (\mathsf{c}, x, y)$ outputting $\pi \coloneqq (\mathsf{c} - [y]_1)(s - x)^{-1}$.

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 3.4 (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.* [17] 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 3.5 (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 $\mathsf{coms} \coloneqq (\mathsf{c}_k)_k$ and $\forall k : \mathsf{c}_k \in \mathbb{G}_1$, and proofs be the list of simulated proofs $\mathsf{proofs} \coloneqq (\pi_k)_k$ (where $\pi_k \coloneqq (\mathsf{q}_{k,j})_j, \boldsymbol{y}_k$ 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 Π_{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 $\mathsf{c} = [f(s)]_1 + \langle \boldsymbol{c}, \mathsf{coms} \rangle + \langle \boldsymbol{o}, \mathsf{proofs} \rangle$.

4 SIMULATION EXTRACTABILITY OF KZG

We give our new results on the simulation extractability of KZG. First, we show that (a batched version of) Π_{evl} is simulation extractable in the presence of *linearized commitments* (cf. Section 4.1), thus extending the result of [17]. Then we show that the linearization trick is simulation-extractable (cf. Section 4.2).

4.1 SE of batched KZG

We consider a batched version of Π_{evl} described in Section 3.3 that we name $\Pi_{\mathtt{m\text{-}evl}}$. This batched version, given in the ROM, follows from [19, 31] and relies on the linearity of the polynomials and the homomorphic properties of KZG.

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

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 Π_{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 Π_{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 4.1 (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 $\mathsf{coms} \coloneqq (\mathsf{c}_j)_j$ (resp. simulated proofs $\mathsf{proofs} \coloneqq (\pi_j)_j$) 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:

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

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

	[17]	$\boldsymbol{\Phi}^{\mathtt{s\text{-}adpt}}_{\mathtt{m\text{-}evl}}$	$\boldsymbol{\Phi}^{\mathtt{adpt}}_{\mathtt{m\text{-}evl}}$
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.

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 [18]). 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 $\mathbb{x}^* \coloneqq (x^*, (\mathsf{c}^*_i, y^*_i)_{i\in[n]})$, 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:

- $\forall i \in [n] : \mathsf{c}^*_i = \sum_r A_{i,r}(x^*)\mathsf{b}_{i,r}$
- $\forall i \in \mathcal{I} : ((\mathsf{b}_{i,r})_r; (A_{i,r})_r, h) \to_{\mathsf{RO}} a$.
- $h(a) = x^*$
- $\forall i \in \mathcal{I} : \left\{A_{i,r}\right\}_r$ are $\bar{\nu}$-independent polynomials, where $\bar{\nu}$ is the maximum nesting level (cf. Definition 4.1)

Definition 4.2. Let $\boldsymbol{\Phi}^{\mathtt{adpt}}_{\mathtt{m\text{-}evl},\mathcal{I}}$ (resp. $\boldsymbol{\Phi}^{\mathtt{s\text{-}adpt}}_{\mathtt{m\text{-}evl},\mathcal{I}}$) 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 $\mathsf{pp}_\Phi \coloneqq \mathsf{coms}$, 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 [17]. 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 4.3. *$\forall \mathcal{I} \subseteq [n]$, $\Pi_{\mathtt{m\text{-}evl}}$ is $(\boldsymbol{\Phi}^{\mathtt{adpt}}_{\mathtt{m\text{-}evl},\mathcal{I}}, \sigma_{\mathcal{I}})$-SE under the OMSDH assumption and is $(\boldsymbol{\Phi}^{\mathtt{s\text{-}adpt}}_{\mathtt{m\text{-}evl},\mathcal{I}}, \sigma_{\mathcal{I}})$-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[12], 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 one $o_j \neq 0$. 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. We defer the proof to [18].

4.2 SE of the Linearization Trick

In this section we formalize the linearization trick for KZG commitments [19, 33] 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 $G_i \in \mathbb{F}[X_1, \ldots, X_m, X]$.

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 $R(X) \coloneqq \sum_i A_i(x)B_i(X)$, $\mathsf{r} \coloneqq \sum_i A_i(x)\mathsf{b}_i$, and $\mathbb{x}_{\mathtt{m\text{-}evl}} \coloneqq (x, (\mathsf{c}_j, C_j(x))_j, (\mathsf{r}, y))$. Output $\pi \coloneqq (\pi_{\mathtt{m\text{-}evl}}, (C_j(x))_j)$

$\mathsf{Verify}_{\mathsf{lin}}(\mathsf{vk}, \mathbb{x}, \pi)$: 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 $\mathbb{x}_{\mathtt{m\text{-}evl}} \coloneqq (x, ((\mathsf{c}_j, y_j)_j, (\mathsf{r}, y)))$

This scheme is not zero-knowledge as the proofs leak information on the witness, however it achieves a weaker form of zero-knowledge called *leaky* zero-knowledge [10], we give more details in [18].

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 $\mathbb{x}^* \coloneqq ((\mathsf{c}_j^*)_j, (\mathsf{b}_i^*)_i, (G_i^*)_i, x^*, y^*)$, return 1 if and only if there exists a polynomial h such that:
- $((\mathsf{c}_j^*)_j, (\mathsf{b}_i^*)_i; (G_i^*)_i, h) \rightarrow_{\mathsf{RO}} a$ 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 4.4. Let $\mathbf{\Phi}_{\mathsf{lin}}^{\mathcal{J},\nu}$ 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 $\mathsf{pp}_\Phi \coloneqq \mathsf{coms}$, 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 [5, 11]) 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 5.3.

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 $A_i(X) \coloneqq G_i((C_j(X))_j, X)$, 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 4.5. *For any $n, \nu \in \mathbb{N}, \mathcal{J} \subseteq [n]$, Π_{lin} is $(\mathbf{\Phi}_{\mathsf{lin}}^{\mathcal{J},\nu}, \mathcal{F}_{\mathcal{J},\nu})$-simulation-extractable in the AGM under the OMSDH assumption.*

Proof Intuition. Thanks to the heavy lifting of Theorem 4.3 the proof of Theorem 4.5 is not much different than a proof of (standard) extractability in the AGMOS [30] 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 $\Pi_{\mathtt{m\text{-}evl}}$

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

extracts $R(X) = \sum A_i(x^*)B_i(X)$ while we need to show how to extract the polynomials $(B_i(X))_i$. For simplicity, assume that the adversary obtains an obliviously sampled element c, thus we can write $\mathrm{b}_i = [B_i(s)]_1 + \bar{B}_i(s) \cdot \mathrm{c}$. We need to show that $\bar{B}_i \equiv 0$, and we can assume, thanks to the SE of $\Pi_{\mathtt{m\text{-}ev1}}$, 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. We defer the proof to [18].

5 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 [13]. 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 5.1 (Oracle Relations, [13]). 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 5.2 ((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 $\hat{\mathcal{R}}$-PIOP over $\mathcal{F}$ for $\mathcal{R}$ is a tuple $\mathsf{IP} \coloneqq (r, n, m, D, \mathsf{I}, \mathsf{P}, \mathsf{V})$ where $r, n, m, D\colon \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions, and $\mathsf{I}, \mathsf{P}, \mathsf{V}$ are three algorithms for the indexer, prover and verifier respectively, that work as follows.

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

Online phase: The prover $\mathsf{P}(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w})$ and the verifier $\mathsf{V}^{\mathsf{I}(\mathbb{F},\mathbb{i})}(\mathbb{F}, \mathbb{x})$ are executed for $r(|\mathbb{i}|)$ rounds; the prover has a tuple $(\mathbb{F}, \mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and the verifier has an instance $\mathbb{x}$ and oracle access to the polynomials encoding $\mathbb{i}$.
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 $D \coloneqq D(|\mathbb{i}|)$, while V replies (except for the last round) with a uniformly random message $\rho_i \in \mathbb{F}$.

Decision phase: Let $r \coloneqq r(|\mathbb{i}|)$, $n \coloneqq \sum_{k=0}^{r} n(k)$, $m \coloneqq \sum_{k=1}^{r} m(k)$. 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 $\boldsymbol{\rho} \coloneqq (\rho_1, \ldots, \rho_{r-1})$, the messages of the prover $\boldsymbol{\pi} \coloneqq (\pi_1, \ldots, \pi_m)$ 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 $\mathsf{indexes}(\hat{\mathbb{x}}) \coloneqq \{\mathsf{index}([\![f]\!]) : [\![f]\!] \in \mathsf{oracles}(\hat{\mathbb{x}})\}$ the indexes in the transcript associated with (the pointers to) the polynomial oracles involved in $\hat{\mathbb{x}}$.

Security Properties for PIOPs. In terms of security, we require $\mathcal{R}$-PIOPs to satisfy the usual properties in line with previous work. In particular, we require *b-bounded zero-knowledge* for the PIOP, a notion that assures zero-knowledge even in the presence of evaluations on random points for some of the polynomial oracles. This notion was originally defined in [14] and generalized in [10]. Additionally, we require *simulation-friendly polynomial oracles*, which essentially means that the commitments to the oracles can be simulated by sampling random group elements. This property is trivial to verify when the commitments are hiding and it can be proved, with some extra work, for non-hiding commitments relying, for example, on Decisional Uber Assumption [7][13] and the zero-knowledge of the PIOP. Finally, as in [17], we require that the PIOP is *state-restoration straight-line knowledge sound*. The notion is defined by a security game where the malicious prover engages with the honest verifier and has the additional ability to *roll back* the interaction with the verifier to a previous state. At some point, the interaction may reach a final state. The prover is considered successful if it produces an accepting transcript, while the extractor, given the oracles in the accepting transcript, fails to produce a valid witness. State-restoration knowledge soundness is implied by standard knowledge soundness [4], although with a loose reduction, and it is, arguably, the correct notion of soundness for multi-round public-coin PIOP compiled through the Fiat-Shamir transform as it avoids *grinding attacks* [37]. We defer the formal definitions of these properties for $\mathcal{R}$-PIOPs to [18].

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 $\hat{\mathbb{x}} \coloneqq (\hat{\mathbb{x}}_k)_k$ 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 natural approach is used by the verifier: for sake of simplicity, we extend Definition 5.2 of an $\hat{\mathcal{R}}$-PIOP and allow the verifier to output n_e checks $(\hat{\mathbb{x}}_k)_k$, and the verifier accepts iff $\hat{\mathbb{x}}_k \in \mathcal{L}_{\hat{\mathcal{R}}}$ 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 $\mathbb{i}$, the instance $\mathbb{x}$ and the witness $\mathbb{w}$, 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 $\mathbb{w}$, interacts with the verifier for

[13] Which reduces to discrete log for algebraic adversaries [35].

only two rounds.[14] 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 [17] (sufficient to prove Simulation Extractability of the compiled SNARK) holds.

Definition 5.3 (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 $\mathbb{i}$ 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 $(\mathbb{F}, \mathbb{i}, (\mathbb{x}, (\boldsymbol{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]}))$ where $\tilde{\mathsf{V}}(\mathbb{F}, \mathbb{i}, \mathbb{x}, (\boldsymbol{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]})) = 1$ 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 $\mathbb{F}, \mathbb{i}, \mathbb{x}, (\boldsymbol{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]}$ the probability, taken over the V_2's message $\rho_r \xleftarrow{\$} \mathbb{F}$, that V_2 on input $(\mathbb{F}, \mathbb{x}, (\boldsymbol{\pi}_j)_{j\in[r]}, (\rho_j)_{j\in[r-1]})$ 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 and only if all the checks are satisfied.[15] Note, we say that a PIOP with delegation has simulation-friendly polynomial oracles if so does IP_1.

5.1 $\hat{\mathcal{R}}_{\mathsf{lin}}$-PIOP

Similarly to Section 4.2, let $\hat{\mathcal{R}}_{\mathsf{lin}}$ be the oracle indexed relation that upon an instance

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

outputs 1 if and only if $\sum_i a_i(x)b_i(x) = y$, where for all i, we have that $a_i(X) \coloneqq G_i(c_1(X), \ldots, c_m(X), X)$. 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.

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

[15] 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.

Definition 5.4 (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 4.4.

Definition 5.5 (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 no 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 $k \notin \mathcal{K}_f$, $\mathcal{J}_k \coloneqq \mathcal{J}_{k-1}$
- for all $k \in \mathcal{K}_f$, $\mathcal{J}_k \coloneqq \mathcal{J}_{k-1} \cup \{\hat{\mathbb{x}}_k.c_j : j \in [m]\} \cup \{\hat{\mathbb{x}}_k.b_i : i \in [n]\}$
- ν is the maximum number of distinct points for which the verifier evaluates a non-index polynomial, i.e.,

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

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*).

5.2 $\hat{\mathcal{R}}_{\mathsf{poly}}$-PIOP

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 [10, 17, 19, 34]. In Section 5.2, we formalize this class of PIOPs using the oracle relation $\hat{\mathcal{R}}_{\mathsf{poly}}$ that upon the instance $\mathbb{x}_{\mathsf{poly}} \coloneqq ((\llbracket p_j \rrbracket)_{j\in[n]}, F, (v_j)_{j\in[n]})$, 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 then show a notion of compiler-safe $\hat{\mathcal{R}}_{\mathsf{poly}}$ which very informally asks that each oracle p_i is queried on a non-constant v_i at least in one of the polynomial equations, and which is, arguably, much easier to verify and enforce than Definition 5.5. In particular our notion of compiler-safe is more inclusive than in [17], as it holds for PIOPs such like Marlin [14] and Lunar [10] without any changes.

In [18] we show that we can transform a compiler-safe $\hat{\mathcal{R}}_{\mathsf{poly}}$-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 $\hat{\mathcal{R}}_{\mathsf{poly}}$-PIOP.

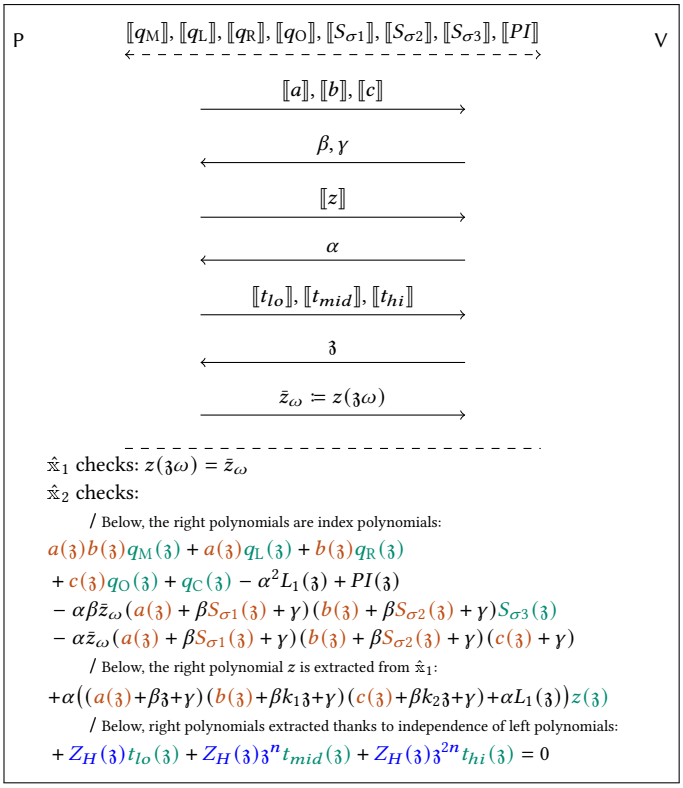

Figure 3: The $\hat{\mathcal{R}}_{\text{lin}}$-PIOP PLONK. For $\hat{\mathbb{x}}_2$, we highlight the *core* polynomials, the *left* polynomials that are linearly independent, and the *right* polynomials.

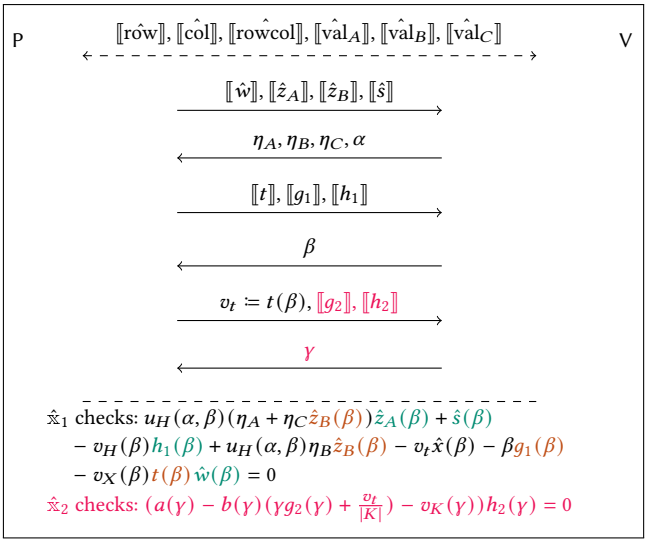

Figure 4: The $\hat{\mathcal{R}}_{\text{lin}}$-PIOP Marlin, where: v_H (resp. v_K, v_X) denotes the vanishing polynomial of the subgroup H (resp. K, X) of $\mathbb{F}$; u_H is the formal derivative of v_H; the polynomials a and b are computed using the index polynomials and the coins α and β. We highlight the delegation phase, the *core* and *right* polynomials of $\hat{\mathbb{x}}_1$.

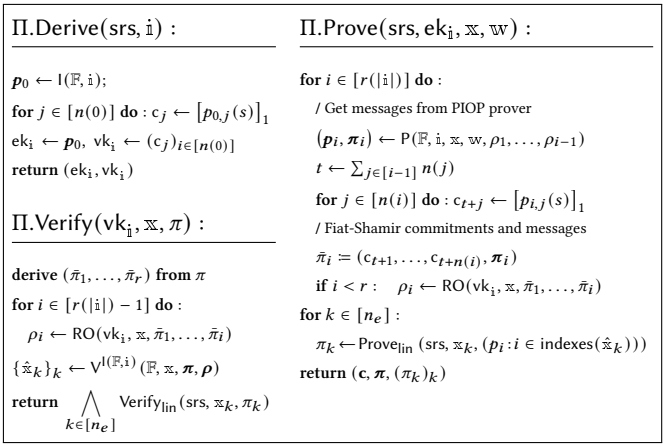

Figure 5: The compiler based from $\hat{\mathcal{R}}_{\text{lin}}$-PIOPs and KZG commitment scheme to Universal zkSNARKs. We associate to the instance $\hat{\mathbb{x}}_k$ for the oracle relation $\hat{\mathcal{R}}_{\text{lin}}$ the instance $\mathbb{x}_k$ for the commit-and-prove relation $\mathcal{R}_{\text{lin}}$, the latter instance is identical to $\hat{\mathbb{x}}_k$ but where any oracle $[\![p]\!] \in \text{oracles}(\hat{\mathbb{x}}_k)$ is substituted with the commitment $[p(s)]_1$.

5.3 Notable PIOPs

In what follows, we show how to express in $\hat{\mathcal{R}}_{\text{lin}}$ 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. 3 how PLONK [19] can be written as a 4-rounds $\hat{\mathcal{R}}_{\text{lin}}$-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 $[\![z]\!]$ is evaluated on $\mathfrak{z}$ and $\mathfrak{z}\omega$. In $\hat{\mathbb{x}}_1$ 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 $[\![t_{lo}]\!], [\![t_{mid}]\!], [\![t_{hi}]\!]$ of the *right* oracle polynomials of $\hat{\mathbb{x}}_2$ are part of the index or are extracted from $\hat{\mathbb{x}}_1$. 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 5.5.

Marlin. We show in Fig. 4 how Marlin [1, 14] can be written as a 3-rounds $\hat{\mathcal{R}}_{\text{lin}}$-PIOP with a Delegation Phase.

6 REVISITING THE PIOP-TO-ZKSNARK COMPILER

We show how to turn compiler-safe $\hat{\mathcal{R}}_{\text{lin}}$-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 6.1. We say that Π is *strong simulation-extractable in the algebraic group model with oblivious sampling* if and only if Π is $\mathbf{\Phi}_{\mathsf{sse}}$-simulation-extractable where for any (Φ_0, Φ_1) in the family of policies $\mathbf{\Phi}_{\mathsf{sse}}$ 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 $(\mathbb{x}^*, \pi^*) \notin Q_{\mathsf{sim}}$.

Theorem 6.2. *Let Π_{lin} be the CP-SNARK for $\mathcal{R}_{\mathsf{lin}}$ defined in Section 4.2. 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 zk-SNARK compiled from* IP *using the compiler in Fig. 5. 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.*

Proof intuition. The proof of zero-knowledge follows rather easily from the bounded zero-knowledge, the simulation-friendly polynomial oracles and the (leaky) zero-knowledge of Π_{lin}. Moreover, the simulation strategy makes sure that the view of the adversary is always algebraic consistent and that the maximum nesting level is at most equal to the number of $\hat{\mathcal{R}}_{\mathsf{lin}}$ instances queried by the verifier in a single proof.

For the proof of simulation extractability, we need to show that for any adversary $\mathcal{A}$, there is an extractor $\mathcal{E}$ who outputs a valid witness whenever $\mathcal{A}$ submits a valid forgery, i.e., a new and valid pair of statement and proof. To do that, we reduce the simulation extractability of Π to the state-restoration knowledge soundness of the PIOP. Our extractor parses the algebraic representation of the commitments sent by the adversary when querying the RO (to compute the next coin of the verifier), and extracts from them the underlying polynomials. The latter polynomials are used as the prover oracles sent in the reduction to the state-restoration knowledge soundness experiment to define a *verifier state* and retrieve the next coins of the verifier. Notice that this reduction would not work if the adversary used simulated elements in (the transcript) of his forgery. We bound the probability of this bad event by reducing to the simulation extractability of Π_{lin}. More in detail, for each focal check we perform one reduction to the Φ_{lin}-SE of Π_{lin} and in the k-th reduction we extract the polynomials $\mathcal{J}_k$ with the knowledge of the polynomials in $\mathcal{J}_{k-1}$ (i.e., we use partial extractability), where the sets $(\mathcal{J}_k)_k$ come from the compiler-safeness (c.f. Definition 5.5).

ACKNOWLEDGMENTS

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

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

[2] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. 2020. A Classification of Computational Assumptions in the Algebraic Group Model. In *CRYPTO 2020, Part II (LNCS, Vol. 12171)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, Heidelberg, 121–151. https://doi.org/10.1007/978-3-030-56880-1_5

[3] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. 2008. P-signatures and Noninteractive Anonymous Credentials. In *TCC 2008 (LNCS, Vol. 4948)*, Ran Canetti (Ed.). Springer, Heidelberg, 356–374. https://doi.org/10.1007/978-3-540-78524-8_20

[4] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. 2016. Interactive Oracle Proofs. In *TCC 2016-B, Part II (LNCS, Vol. 9986)*, Martin Hirt and Adam D. Smith (Eds.). Springer, Heidelberg, 31–60. https://doi.org/10.1007/978-3-662-53644-5_2

[5] Daniel Benarroch, Matteo Campanelli, Dario Fiore, Kobi Gurkan, and Dimitris Kolonelos. 2021. Zero-Knowledge Proofs for Set Membership: Efficient, Succinct, Modular. In *FC 2021, Part I (LNCS, Vol. 12674)*, Nikita Borisov and Claudia Díaz (Eds.). Springer, Heidelberg, 393–414. https://doi.org/10.1007/978-3-662-64322-8_19

[6] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. 2012. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS 2012*, Shafi Goldwasser (Ed.). ACM, 326–349. https://doi.org/10.1145/2090236.2090263

[7] Xavier Boyen. 2008. The Uber-Assumption Family (Invited Talk). In *PAIRING 2008 (LNCS, Vol. 5209)*, Steven D. Galbraith and Kenneth G. Paterson (Eds.). Springer, Heidelberg, 39–56. https://doi.org/10.1007/978-3-540-85538-5_3

[8] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short Proofs for Confidential Transactions and More. In *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 315–334. https://doi.org/10.1109/SP.2018.00020

[9] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. 2020. Transparent SNARKs from DARK Compilers. In *EUROCRYPT 2020, Part I (LNCS, Vol. 12105)*, Anne Canteaut and Yuval Ishai (Eds.). Springer, Heidelberg, 677–706. https://doi.org/10.1007/978-3-030-45721-1_24

[10] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. 2021. Lunar: A Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions. In *ASIACRYPT 2021, Part III (LNCS, Vol. 13092)*, Mehdi Tibouchi and Huaxiong Wang (Eds.). Springer, Heidelberg, 3–33. https://doi.org/10.1007/978-3-030-92078-4_1

[11] Matteo Campanelli, Dario Fiore, Semin Han, Jihye Kim, Dimitris Kolonelos, and Hyunok Oh. 2022. Succinct Zero-Knowledge Batch Proofs for Set Accumulators. In *ACM CCS 2022*, Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.). ACM Press, 455–469. https://doi.org/10.1145/3548606.3560677

[12] Matteo Campanelli, Dario Fiore, and Anaïs Querol. 2019. LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs. In *ACM CCS 2019*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM Press, 2075–2092. https://doi.org/10.1145/3319535.3339820

[13] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. 2023. HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates. In *EUROCRYPT 2023, Part II (LNCS, Vol. 14005)*, Carmit Hazay and Martijn Stam (Eds.). Springer, Heidelberg, 499–530. https://doi.org/10.1007/978-3-031-30617-4_17

[14] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. 2020. Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS. In *EUROCRYPT 2020, Part I (LNCS, Vol. 12105)*, Anne Canteaut and Yuval Ishai (Eds.). Springer, Heidelberg, 738–768. https://doi.org/10.1007/978-3-030-45721-1_26

[15] Quang Dao and Paul Grubbs. 2023. Spartan and Bulletproofs are Simulation-Extractable (for Free!). In *EUROCRYPT 2023, Part II (LNCS, Vol. 14005)*, Carmit Hazay and Martijn Stam (Eds.). Springer, Heidelberg, 531–562. https://doi.org/10.1007/978-3-031-30617-4_18

[16] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. 2001. Robust Non-interactive Zero Knowledge. In *CRYPTO 2001 (LNCS, Vol. 2139)*, Joe Kilian (Ed.). Springer, Heidelberg, 566–598. https://doi.org/10.1007/3-540-44647-8_33

[17] Antonio Faonio, Dario Fiore, Markulf Kohlweiss, Luigi Russo, and Michal Zajac. 2023. From Polynomial IOP and Commitments to Non-malleable zkSNARKs. In *TCC 2023, Part III (LNCS, Vol. 14371)*, Guy N. Rothblum and Hoeteck Wee (Eds.). Springer, Heidelberg, 455–485. https://doi.org/10.1007/978-3-031-48621-0_16

[18] Antonio Faonio, Dario Fiore, and Luigi Russo. 2024. Real-world Universal zkSNARKs are non-malleable. Cryptology ePrint Archive, Paper 2024/721. https://eprint.iacr.org/2024/721 https://eprint.iacr.org/2024/721.

[19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. 2019. PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge. Cryptology ePrint Archive, Report 2019/953. https://eprint.iacr.org/2019/953.

[20] Chaya Ganesh, Hamidreza Khoshakhlagh, Markulf Kohlweiss, Anca Nitulescu, and Michal Zajac. 2022. What Makes Fiat-Shamir zkSNARKs (Updatable SRS)

Simulation Extractable?. In *Security and Cryptography for Networks, SCN 2022 (Lecture Notes in Computer Science, Vol. 13409)*, Clemente Galdi and Stanislaw Jarecki (Eds.). Springer, 735–760. https://doi.org/10.1007/978-3-031-14791-3_32

[21] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. 2022. Fiat-Shamir Bulletproofs are Non-Malleable (in the Algebraic Group Model). In *EUROCRYPT 2022, Part II (LNCS, Vol. 13276)*, Orr Dunkelman and Stefan Dziembowski (Eds.). Springer, Heidelberg, 397–426. https://doi.org/10.1007/978-3-031-07085-3_14

[22] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. 1985. The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract). In *17th ACM STOC*. ACM Press, 291–304. https://doi.org/10.1145/22145.22178

[23] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. 2018. Updatable and Universal Common Reference Strings with Applications to zk-SNARKs. In *CRYPTO 2018, Part III (LNCS, Vol. 10993)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Heidelberg, 698–728. https://doi.org/10.1007/978-3-319-96878-0_24

[24] Jens Groth and Mary Maller. 2017. Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs. In *CRYPTO 2017, Part II (LNCS, Vol. 10402)*, Jonathan Katz and Hovav Shacham (Eds.). Springer, Heidelberg, 581–612. https://doi.org/10.1007/978-3-319-63715-0_20

[25] Yuval Ishai. 2019. Efficient Zero-Knowledge Proofs: A Modular Approach. (2019). https://simons.berkeley.edu/talks/tbd-79. Also see https://zkproof.org/2020/08/12/information-theoretic-proof-systems/.

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

[27] Charanjit S. Jutla and Arnab Roy. 2013. Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces. In *ASIACRYPT 2013, Part I (LNCS, Vol. 8269)*, Kazue Sako and Palash Sarkar (Eds.). Springer, Heidelberg, 1–20. https://doi.org/10.1007/978-3-642-42033-7_1

[28] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. 2010. Constant-Size Commitments to Polynomials and Their Applications. In *ASIACRYPT 2010 (LNCS, Vol. 6477)*, Masayuki Abe (Ed.). Springer, Heidelberg, 177–194. https://doi.org/10.1007/978-3-642-17373-8_11

[29] Markulf Kohlweiss, Mahak Pancholi, and Akira Takahashi. 2023. How to Compile Polynomial IOP into Simulation-Extractable SNARKs: A Modular Approach. In *TCC 2023, Part III (LNCS, Vol. 14371)*, Guy N. Rothblum and Hoeteck Wee (Eds.). Springer, Heidelberg, 486–512. https://doi.org/10.1007/978-3-031-48621-0_17

[30] Helger Lipmaa, Roberto Parisella, and Janno Siim. 2023. Algebraic Group Model with Oblivious Sampling. In *TCC 2023, Part IV (LNCS, Vol. 14372)*, Guy N. Rothblum and Hoeteck Wee (Eds.). Springer, Heidelberg, 363–392. https://doi.org/10.1007/978-3-031-48624-1_14

[31] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. 2019. Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings. In *ACM CCS 2019*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM Press, 2111–2128. https://doi.org/10.1145/3319535.3339817

[32] Silvio Micali. 1994. CS Proofs (Extended Abstracts). In *35th FOCS*. IEEE Computer Society Press, 436–453. https://doi.org/10.1109/SFCS.1994.365746

[33] O1-Labs. [n. d.]. Maller's Optimization to Reduce Proof Size - Mina Book. ([n. d.]). https://o1-labs.github.io/proof-systems/plonk/maller.html.

[34] Carla Ràfols and Arantxa Zapico. 2021. An Algebraic Framework for Universal and Updatable SNARKs. In *CRYPTO 2021, Part I (LNCS, Vol. 12825)*, Tal Malkin and Chris Peikert (Eds.). Springer, Heidelberg, Virtual Event, 774–804. https://doi.org/10.1007/978-3-030-84242-0_27

[35] Lior Rotem and Gil Segev. 2020. Algebraic Distinguishers: From Discrete Logarithms to Decisional Uber Assumptions. In *TCC 2020, Part III (LNCS, Vol. 12552)*, Rafael Pass and Krzysztof Pietrzak (Eds.). Springer, Heidelberg, 366–389. https://doi.org/10.1007/978-3-030-64381-2_13

[36] Amit Sahai. 1999. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In *40th FOCS*. IEEE Computer Society Press, 543–553. https://doi.org/10.1109/SFFCS.1999.814628

[37] StarkWare. 2021. ethSTARK Documentation. Cryptology ePrint Archive, Report 2021/582. https://eprint.iacr.org/2021/582.

[38] Alan Szepieniec. 2020. Polynomial IOPs for Linear Algebra Relations. Cryptology ePrint Archive, Report 2020/1022. https://eprint.iacr.org/2020/1022.