From Polynomial IOP and Commitments to Non-malleable zkSNARKs

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

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

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

1 Introduction

Non-interactive succinct zero-knowledge arguments of knowledge (zkSNARKs) [44], are the new Swiss army knife of blockchain scalability and privacy. They effectively deliver the twin dream of probabilistically checkable proofs (PCP) [3] and zero-knowledge proofs (ZKP) [33] while also being non-interactive, short, and efficiently verifiable. These features make zkSNARKs of high practical and theoretical relevance. They are an active area of research that has seen rapid progress in multiple aspects, such as efficiency [7,32,34,35], security and versatility of their setups [6,36], and proof composition [13,15].

Simulation-extractable zkSNARKs. *Knowledge-soundness* is the basic security notion of zkSNARKs: informally speaking, it guarantees that, in isolation, a prover producing a valid proof must know the corresponding witness. In contrast, there exist real-world deployments and cryptographic applications of zkSNARKs that require a stronger property called *simulation extractability* (SE, for brevity) [37,47]. Intuitively, this notion considers attackers that can see proofs for some statements and may use this information in order to produce a proof for some other statement without knowing the witness. Interestingly,

simulation extractability implies that proofs are *non-malleable* [23], a relevant property in practical applications. Most zkSNARKs in the literature are only proven to be knowledge-sound. In some cases, this is due to the fact that their proofs may indeed be malleable, e.g., as in [35] (see also [4]). In other cases, the lack of SE security proof is because it is challenging and may require more investigation.

From polynomial commitments to SNARKs. The design of modern zkSNARKs follows the common cryptographic approach of starting with protocols that achieve information-theoretic security in idealized models and then compiling them into efficient protocols by employing a smaller computationally secure primitive. In the world of SNARKs, the corresponding concepts are (polynomial) interactive oracle proofs F-IOP [16,17,20,27,48] and (polynomial) functional commitments F-COM [12,39,40]. An F-IOP employs two (idealized) oracles that share their state: the prover calls the first oracle *to commit* to functions $f \in F$ and the verifier calls the second *to query* the committed functions. Concretely, the F-IOP to SNARK compiler uses F-COM to replace oracles with commitments, opening proofs, and query proofs. As this only removes reliance on idealized function oracles but not interaction, the compiler additionally employs the usual Fiat-Shamir transformation for public-coin protocols to obtain the final zkSNARK. The benefits of this compilation paradigm are modularity and separation of concerns: once the compiler is proven, a line of research can address the problem of improving F-IOPs while another research line can tackle the problem of realizing F-COM schemes (e.g., with better efficiency, from different assumptions, etc.): this approach has been successfully adopted to construct several recent zkSNARKs. All this recent work, though, only shows that schemes obtained via this paradigm are knowledge-sound.

1.1 Our work

We study the simulation extractability of a broad class of zkSNARKs built through this "natural" compilation approach. In particular, our primary goal includes showing that not only *existing* zkSNARKs but also any future zkSNARK following this, by now standard, construction framework, provide simulation extractability. This goal has a twofold motivation. On the theoretical side, we are interested in understanding sufficient conditions on F-COM to compile an F-IOP into a simulation-extractable zkSNARK. On the practical side, by capturing existing compilers we can show that existing schemes that are under deployment, e.g., Plonk [27], have already this strong security property.[5] For this reason, in our work we focus on the popular case of the compiler where the F-IOP is a polynomial IOP (i.e., the oracle functions F are low-degree polynomials), and F-COM is a polynomial commitment. Furthermore, in terms of instantiations we

[5] In fact as Mahak Pancholi and Akira Takahashi recently informed us of a flaw in the trapdoor-less zero-knowledge simulation of [28] this is arguably the first proof of simulation extractability of unmodified Plonk, i.e., Plonk with deterministic KZG commitments.

are interested to cover the celebrated polynomial commitment scheme of Kate, Zaverucha and Goldberg [39] (KZG, shortly) and on a polynomial IOP framework that is flexible enough to include recent constructions, e.g., [17,20,27,42,46].

The main contributions of our work are: (i) to introduce a relaxed notion of simulation extractability for polynomial commitments; (ii) to prove that the KZG scheme satisfies our relaxed SE notion in the algebraic group model (AGM) and random oracle model (ROM); and (iii) to prove that our notion is sufficient to compile a polynomial IOP into a (full-fledged) simulation-extractable zkSNARK, using the usual compilation approach. By combining these results we obtain a simulation extractability proof for Plonk [27], Basilisk [46], and a slight variation of Marlin [20] and Lunar [17].

Below, we elaborate more on our results and the challenges that we had to overcome along the way.

1.2 Our techniques

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

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

– The PIOP could be arbitrary. For example, consider a PIOP obtained by the sequential composition of two PIOP protocols for two independent statements. Very likely, the set of queries to the polynomials made by the two

[6] PIOPs can flexibly capture under the same hat all the most recent protocols based on the notions of [16], AHP [20], and ILDP [27].

sub-protocols are independent and (unless the PIOP specifies it explicitly) the evaluation queries of the first sub-protocol may be chosen based on the verifier's random challenges sent *before* the second sub-protocol even starts. The simulation extractability of the zkSNARK compiled from this protocol might be affected because one could strip off the second set of evaluation proofs and replace them with those for another statement[7].
- Secondly, one needs to prove that existing, efficient, and practically deployable instantiations of polynomial commitment schemes are simulation-extractable.

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

For the second challenge, unfortunately, the issue is that the most obvious candidate, the efficient and widely deployed KZG polynomial commitment scheme [39], is not simulation-extractable. Using bracket notation, KZG commitments are of the form $[p(s)]_1$ for a trapdoor secret s encoded in the parameters $([s^i]_1)_{i\in[0..d]}, [1, s]_2$, while evaluation proofs for an input x and output y are of the form $[\frac{p(s)-y}{s-x}]_1$. KZG is malleable, for example, given a commitment to p anyone can compute $[p(s) + \Delta]_1$ and open it using the same proof to $(x, y + \Delta)$.

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

To formalize this important relaxation, we introduce the notion of policy-based simulation extractability (Φ-SE, w.r.t. a policy parameter Φ). In the standard simulation extractability experiment, the adversary can ask the simulator to generate proofs for statements of its choice and, eventually, must produce a

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

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

new valid proof without knowing the witness. In Φ-SE, we consider a relaxation of the SE game in which all the simulation queries of the adversary must satisfy a predicate specified in Φ; similarly, Φ can constrain the winning condition of the adversary. For this reason, we refer to Φ as the *policy*. One can see that Φ-SE is a generalization of existing SE notions such as true-simulation extractability (where the adversary can only see simulated proofs on true statements) [22], or weak simulation extractability (where the adversary only wins if it provides a proof for a new statement and, contrary to (strong) SE, loses if it provides a new proof for a statement previously asked to the simulation oracle).

Once having defined this framework, we analyze which policies Φ are strong enough to achieve simulation extractability in the compiled zkSNARK, while at the same time being weak enough for instantiation by KZG under plausible assumptions (in the AGM [26] and RO). Specifically, we isolate the (simulation) extractability properties needed for the compiler and verify it for KZG in the AGM. This is the only part of our results where we need the AGM. Given the broad applications of KZG in the field of practical zkSNARKs and beyond, the characterization of its (non-)malleability is interesting in its own right. In fact, our policy highlights some malleability attacks that we discovered and that we needed to handle. Finally, for our Φ we prove that KZG is Φ-SE in the AGM and ROM. This proof turned out to be highly non-trivial and is one of the main technical contributions of our work.

1.3 Related work

It is hard to be exhaustive, or even representative, in discussing related work on SNARKs. For the sake of our paper, we focus on related work on simulation extractability notions. Groth and Maller [37] give a simulation-extractable zkSNARK that consists of only 3 group elements. Their construction is neither universal nor updatable. The recent work of Ganesh, Orlandi, Pancholi, Takahashi and Tschudi [30] shows that Bulletproofs [14] are non-malleable in AGM. More recently, Dao and Grubbs show that Spartan and Bulletproofs are non-malleable even without AGM [21]. Both Ganesh et al. and Dao et al. work extend the framework introduced by Faust, Kohlweiss, Marson and Venturi in [24] to Fiat-Shamir applied to multi-round interactive arguments. On a similar path, the work of Ganesh, Khoshakhlagh, Kohlweiss, Nitulescu and Zajac [28] shows non-malleability for Plonk, Sonic and Marlin. Both [28,30] show that interactive *arguments* can be simulation-extractable after applying the Fiat-Shamir transform. In particular, their approach consists into defining new properties, like trapdoor-less zero-knowledge[9] and unique response[10] that *need to be proven on a case-by-case basis*. Namely, for each candidate SNARK (even if resulting from a generic compiler) one needs additional effort to show that it is simulation extractable. This is arguably more challenging and less generic than our approach.

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

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

Thanks to our result, once having a Φ-SE polynomial commitment, one only needs to check a very simple property on the polynomial IOP. Furthermore, [28] relies on rewinding-based soundness, which is a non-standard notion of soundness, and rewinding which make the simulator-extractability extractor success dependent on the probability of the adversary returning an acceptable proof; and [30] relies on the AGM. Contrary to these results, our paper's result does not directly rely on the AGM, does not require the protocol to be trapdoor-less zero knowledge or have the unique-response property and uses more standard notion of soundness, i.e. the state restoration soundness.

The work of Abdolmaleki, Ramacher and Slamanig [2] shows a generic compiler to simulation-extractable SNARKs which requires key-homomorphic signatures. Their compiler produces universally-composable SNARKs (UC-SNARKs), which they prove through black-box straight-line extractor. To obtain a black-box straight-line extractor, they append to the SNARK proof an encryption of the witness, thus achieving a relaxed succinctness w.r.t. the size of the circuit describing the relation. The recent work of Ganesh, Kondi, Orlandi, Pancholi and Takahashi [29] shows how to regain full succinctness in UC-SNARKs in the ROM through Fischlin's transform [25].

1.4 Open problems

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

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

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

work when the degree of the polynomial in the proof depends on the number of evaluation points in the proved statement.

1.5 Organization of the paper

We define the framework of policy-based simulation extractability in Section 3, we proceed with the analysis of the simulation extractability of KZG in Section 4 and we give our generic compiler for strong simulation-extractable Universal SNARKs from (simulation-extractable) polynomial commitment and PIOP in Section 5. The thesis of the theorem on KZG polynomial commitment and the hypothesis for the theorem of the Universal SNARKs compiler do not quite match. Indeed, they could be even considered as two independent and (almost) self-contained results. We show how to fill the gaps and connect the two results in Section 5.4.

2 Preliminaries

A function f is negligible in λ (we write $f \in \mathsf{negl}(\lambda)$) if it approaches zero faster than the reciprocal of any polynomial: i.e., for every $c \in \mathbb{N}$ there is an integer λ_c such that $f(\lambda) \leq \lambda^{-c}$ for all $\lambda \geq \lambda_c$. For an integer $n \geq 1$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. Vectors and matrices are denoted in boldface. Calligraphic letters denote sets, while set sizes are written as $|\mathcal{X}|$. Lists are represented as ordered tuples, e.g. $L := (L_i)_{i\in[n]}$ is a shortcut for the list of n elements $(L_1, \ldots, L_n)$. To get a specific value from a list, we also use the "dot" notation; e.g., we use $L.b$ to access the second element of the list $L = (a, b, c)$. The difference between lists and vectors is that elements of vectors are of the same type. An asymmetric bilinear group $\mathbb{G}$ is a tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q, the elements P_1, P_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently-computable non-degenerate bilinear map, and there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Let $\mathsf{GroupGen}$ be some probabilistic polynomial-time (PPT) algorithm which on input 1^λ, where λ is the security parameter, returns a description of a bilinear group $\mathcal{G}$. Elements in $\mathbb{G}_i, i \in \{1, 2, T\}$ are denoted in implicit notation as $[a]_i := aP_i$, where $P_T := e(P_1, P_2)$. Every element in $\mathbb{G}_i$ can be written as $[a]_i$ for some $a \in \mathbb{Z}_q$, but note that, given $[a]_i$, it is in general hard to compute a (discrete logarithm problem). Given $a, b \in \mathbb{Z}_q$ we distinguish between $[ab]_i$, namely the group element whose discrete logarithm base $\mathcal{P}_i$ is ab, and $[a]_i \cdot b$, namely the execution of the multiplication of $[a]_i$ and b, and $[a]_1 \cdot [b]_2 = [a \cdot b]_T$, namely the execution of a pairing between $[a]_1$ and $[b]_2$. We do not use the implicit notation for variables, e.g. $\mathsf{c} = [a]_1$ indicates that c is a variable name for the group element whose logarithm is a.

Definition 1 (Algebraic algorithm, [26]). *An algorithm $\mathcal{A}$ is* algebraic *if for all group elements z that $\mathcal{A}$ outputs (either as returned by $\mathcal{A}$ or by invoking an oracle), it additionally provides the representation of z relative to all previously*

received group elements. That is, if elems *is the list of group elements that* $\mathcal{A}$ *has received so far, then* $\mathcal{A}$ *must also provide a vector* $\boldsymbol{r}$ *such that* $\mathsf{z} = \langle \boldsymbol{r}, \mathsf{elems} \rangle$.

Definition 2 (Commitment scheme). *A commitment scheme with message space* $\mathcal{M}$ *(and group parameters* $\mathsf{GroupGen}*) is a tuple of algorithms* $\mathsf{CS} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{VerCom})$ *that works as follows:*

$\mathsf{KGen}(\mathsf{pp}_{\mathbb{G}}) \to \mathsf{ck}$ *takes as input group parameters* $\mathsf{pp}_{\mathbb{G}} \leftarrow\!\!\$\ \mathsf{GroupGen}(1^{\lambda})$ *and outputs a commitment key* ck.

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

$\mathsf{VerCom}(\mathsf{ck}, c, m, o) \to b$ *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* $(b = 0)$.

A commitment scheme CS satisfies correctness, binding and hiding properties.

Definition 3 (Polynomial Commitment). *A polynomial commitment* PC *is a commitment scheme (Definition 2) in which the message space* $\mathcal{M}$ *is* $\mathbb{F}_{\leq d}[X]$*, the set of low degree polynomials over a finite field* $\mathbb{F}$ *with degree bound* $d \in \mathbb{N}$*. The key generation algorithm might take as additional input the degree* d.

Finally, we state these assumptions on distributions.

Definition 4 (Witness Sampleability, [38]). *A distribution* $\mathcal{D}$ *is witness sampleable if there exists a PPT algorithm* $\tilde{\mathcal{D}}$ *s.t. for any* $\mathsf{pp}_{\mathbb{G}}$*, the random variables* $\boldsymbol{A} \leftarrow\!\!\$\ \mathcal{D}(\mathsf{pp}_{\mathbb{G}})$ *and* $[\tilde{\boldsymbol{A}}]_1$*, where* $\tilde{\boldsymbol{A}} \leftarrow\!\!\$\ \tilde{\mathcal{D}}(\mathsf{pp}_{\mathbb{G}})$*, are equivalently distributed.*

Definition 5 ($\mathcal{D}_{\ell,k}$-Aff-MDH assumption). *Given a matrix distribution* $\mathcal{D}_{\ell,k}$*, the Affine Diffie-Hellman Problem is: given* $\boldsymbol{A} \in \mathbb{G}_1^{\ell \times k}$*, with* $\boldsymbol{A} \leftarrow\!\!\$\ \mathcal{D}_{\ell,k}$*, find a nonzero vector* $\boldsymbol{x} \in \mathbb{Z}_q^{\ell}$ *and a vector* $\boldsymbol{y} \in \mathbb{Z}_q^{k}$ *such that* $[\boldsymbol{x}^{\top}\boldsymbol{A}]_1 = [\boldsymbol{y}]_1$.

The Aff-MDH Assumption could be seen as an extension of the Kernel-MDH Assumption introduced by Morillo, Ràfols, Villar [45] since the Ker-MDH assumes $\boldsymbol{y} = \boldsymbol{0}$; Our assumption is incomparable because we also require the adversary to give $\boldsymbol{x} \in \mathbb{Z}_q^{\ell}$ "in the exponent".

Definition 6 ((d, d')-Power Polynomial in the Exponent). *The* (d, d')*-PEA Assumption holds for a bilinear group generator* $\mathsf{GroupGen}$ *if for every PPT adversary* $\mathcal{A}$ *that receives as input* $([1, \ldots, s^{d}]_1, [1, \ldots, s^{d'}]_2)$ *and outputs a polynomial* $p(X)$ *of degree at most* $\max\{d, d'\}$*, and a value* y*, the probability that* $p(s) = y$ *is negligible. When* $d = d'$ *we use the shortcut* d*-PEA.*

Definition 7 (d-Power Discrete Logarithm [41]). *Given a degree bound* $d \in \mathbb{N}$*, the* d*-Power Discrete Logarithm (*d*-DL) assumption holds for a bilinear group generator* $\mathsf{GroupGen}$ *if for every PPT adversary* $\mathcal{A}$ *that receives as input* $([1, \ldots, s^{d}]_1, [1, \ldots, s^{d}]_2)$*, and outputs the value* s'*, the probability that* $s = s'$ *is negligible. We also use DL as a shortcut for* 1*-DL.*

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

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

The Aff-MDH Assumption could be seen as an extension of the Kernel-MDH Assumption introduced by Morillo, Ràfols, Villar [45] since the Ker-MDH assumes $\boldsymbol{y} = \boldsymbol{0}$; however, our assumption is incomparable because we also require the adversary to give $\boldsymbol{x} \in \mathbb{Z}_q^\ell$ "in the exponent".

3 Policy-based Simulation-Extractable NIZKs

We start by defining the basic syntax and properties for a Non-Interactive Zero-Knowledge Argument of Knowledge. Following Groth et al. [36], we define a PT relation $\mathcal{R}$ verifying triple $(\mathsf{pp}, \mathbb{x}, \mathbb{w})$. We say that $\mathbb{w}$ is a witness to the instance $\mathbb{x}$ being in the relation defined by the parameters pp when $(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$ (equivalently, we sometimes write $\mathcal{R}(\mathsf{pp}, \mathbb{x}, \mathbb{w}) = 1$). For example, pp could be the description of a bilinear group or additionally contain a commitment key for a commitment scheme or a common reference string. A NIZK for a relation $\mathcal{R}$ (and group generator $\mathsf{GroupGen}$) is a tuple of algorithms $\Pi = (\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify})$ that work as described below.

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

Basic notions for a NIZK are completeness, (knowledge) soundness and zero-knowledge.

zkSNARKs. We consider the notion of zero-knowledge succinct argument of knowledge (zkSNARKs) which are NIZKs that are knowledge sound and succinct as defined below.

Definition 8 (Succinctness). *A NIZK Π is said succinct if the running time of* Verify *is* $\mathsf{poly}(\lambda + |\mathbb{x}| + \log |\mathbb{w}|)$ *and the proof size is* $\mathsf{poly}(\lambda + \log |\mathbb{w}|)$.

CP-SNARKs. Commit-and-Prove SNARKs (CP-SNARKs) are zkSNARKs in which the relations verify predicates over commitments (see Campanelli, Fiore and Querol [18]). We consider the following syntax. Briefly speaking, we refer to a CP-SNARK for a relation $\mathcal{R}$ and a commitment scheme CS as a tuple of algorithms $\mathsf{CP} = (\mathsf{KGen}, \mathsf{Prove}, \mathsf{Verify})$ where:

- $\mathsf{KGen}(\mathsf{ck}) \to \mathsf{srs}$ is a probabilistic (or deterministic) algorithm that takes as input a commitment key ck for CS and outputs $\mathsf{srs} := (\mathsf{ek}, \mathsf{vk}, \mathsf{pp})$, where ek is the evaluation key, vk is the verification key, and pp are the parameters for the relation $\mathcal{R}$ (which include the commitment key ck).

Moreover, if we consider the key generation algorithm KGen' that upon group parameters $\mathsf{pp}_{\mathbb{G}}$ first runs $\mathsf{ck} \leftarrow\!\!\$\ \mathsf{CS.KGen}(\mathsf{pp}_{\mathbb{G}})$, runs $\mathsf{srs} \leftarrow\!\!\$\ \mathsf{CP.KGen}(\mathsf{ck})$ and outputs srs; then the tuple $(\mathsf{KGen}', \mathsf{Prove}, \mathsf{Verify})$ defines a SNARK.

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

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

The state $\mathsf{st}_{\mathcal{S}}$ is updated after each operation. Similarly to [24,30], we define the following wrappers.

Definition 9 (Wrappers for NIZK Simulator). *The following algorithms are stateful and share their state* $\mathsf{st} = (\mathsf{st}_{\mathcal{S}}, \mathsf{coms}, \mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}, \mathcal{Q}_{\mathsf{aux}})$ *where* $\mathsf{st}_{\mathcal{S}}$ *is initially set to be the empty string, and* $\mathcal{Q}_{\mathsf{sim}}, \mathcal{Q}_{\mathsf{RO}}$ *and* $\mathcal{Q}_{\mathsf{aux}}$ *are initially set to be the empty sets.*

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

Almost all the oracles in our definitions can take auxiliary information as additional input. We use this auxiliary information in a rather liberal form. For example, in the definition above, the auxiliary information for $\mathcal{S}_1$ refers to the (optional) leakage required by the simulator to work in some cases (see more in

[11] More often, simulators need only the first three inputs, see Definition 10; abusing notation, we assume that such simulators simply ignore the auxiliary input aux.

Definition 11), while the auxiliary information for $\mathcal{S}_2$ can contain, for example, the algebraic representations of the group elements in s (when we restrict to algebraic adversaries) or other information the security experiments might need.

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

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

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

We also introduce a weaker notion of zero-knowledge. A NIZK is F-leaky zero-knowledge if its proofs may leak some information, namely a proof leaks $F(\mathbb{x},\mathbb{w})$, where $(\mathbb{x},\mathbb{w}) \in \mathcal{R}$. We formalize this by giving the zero-knowledge simulator the value $F(\mathbb{x},\mathbb{w})$, which should be interpreted as a hint for the simulation of proofs. This notion could be seen as an extension of the bounded leaky zero-knowledge property defined in [17] and tailored for CP-SNARKs. Our notion is a special case of the leakage-resilient zero-knowledge framework of Garg, Jain and Sahai [31] where the leakage of the simulator is known ahead of time.

Definition 11 (Leaky Zero-Knowledge). *A NIZK* NIZK *is* F*-leaky zero-knowledge if there exists a PPT simulator* $\mathcal{S}$ *such that for all adversaries* $\mathcal{A}$*:*

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

3.1 Policy-Based Simulation Extractability

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

In our work, we also consider policies that constrain the behavior of the zero-knowledge simulator. For example, we could consider the policy that checks that the queried instances belong to the relation, thus obtaining a notion similar to true-simulation extractability (see Dodis et al. [22]). Looking ahead, contrary to the true-simulation extractability notion in [22], our policy-based version of the true-simulation extractability rather than disallowing certain queries, punishes

the adversary at extraction time. It is not hard to see that the two definitional flavors, namely disallowing illegal queries versus punishing an adversary that made an illegal query are equivalent in the context of simulation extractability, because the adversary's goal is computational[12].

Extraction policies. We define an extraction policy as a tuple $\Phi = (\Phi_0, \Phi_1)$ of PPT algorithms. This is used to define Φ-simulation extractability as follows. The security experiment starts by running the extraction policy algorithm Φ_0, which generates public information pp_Φ. The public information may contain, for example, random values that define the constraints later checked by Φ_1. Therefore, we feed pp_Φ to the adversary. In the case of commit-and-prove proof systems, the public information may contain commitments for which the adversary does not know openings (but on which it can still query simulated proofs). After receiving a forgery from the adversary, the security experiment runs the extraction policy Φ_1. The policy Φ_1 is a predicate that takes as input: (i) The public parameter pp_Φ; (ii) The forged instance and proof $(\mathbb{x}, \pi)$; (iii) The view of the experiment, denoted view. Such a view contains the public parameters, the set of simulated instances and proofs $\mathcal{Q}_{\mathsf{sim}}$, and the set $\mathcal{Q}_{\mathsf{RO}}$ of queries and answers to the random oracle[13]; (iv) Auxiliary information aux_Φ which might come along with the forged instance. We use aux_Φ to provide the adversary an interface with the policy[14].

Definition 12 (Simulation extractability). *Let Π be a NIZK for a relation $\mathcal{R}$ whose wrappers are $\mathcal{S}_1, \mathcal{S}_2$, as defined in Definition 9 Consider the experiment in Fig. 1. Π is Φ-simulation-extractable (or simply Φ-SE) if for every PPT adversary $\mathcal{A}$ there exists an efficient extractor $\mathcal{E}$ such that the following advantage is negligible in λ:*

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

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

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

[13] Even if the given NIZK is not in the random oracle (namely neither the prover nor the verifier algorithms make random oracle queries) it still makes sense to assume the existence of the set $\mathcal{Q}_{\mathsf{RO}}$. This is useful to model security for NIZK protocols that eventually are used as sub-protocols in ROM-based protocols (as Universal zkSNARKs based on Polynomial Commitments, see Section 5)

[14] For example, looking ahead, in the policy in Section 5.4 the adversary that forges a "weak" proof of opening for a commitment, additionally provides a certificate (different than the proof itself) that the commitment is indeed extractable. In this case, we require the extractor only to work for those commitments that come along with valid certificates.

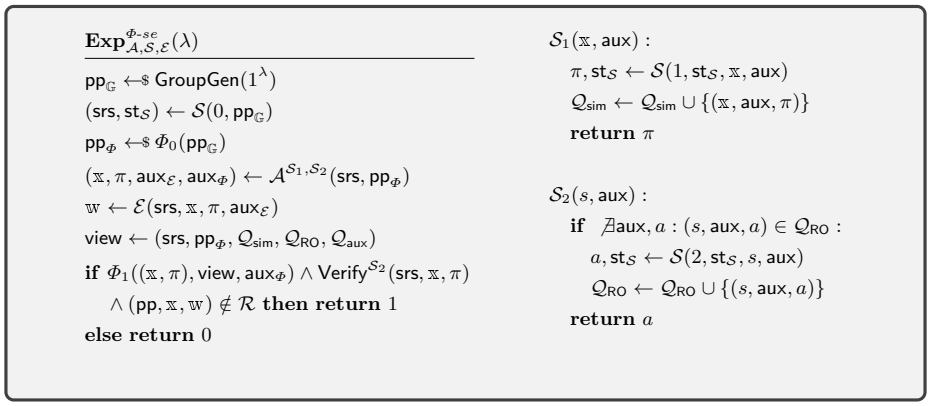

Fig. 1. The Φ-simulation extractability experiments in ROM. The extraction policy Φ takes as input the public view of the adversary view (namely, all the inputs received and all the queries and answers to its oracles). The set $\mathcal{Q}_{\mathsf{sim}}$ is the set of queries and answers to the simulation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

Definition 13 (Simulation extractability in the AGM). *Let Π be a NIZK for a relation $\mathcal{R}$ with a simulator $\mathcal{S}$. Π is Φ-simulation-extractable (or simply Φ-SE) if there exists an efficient extractor $\mathcal{E}$ such that for every PPT algebraic adversary $\mathcal{A}$, the advantage $\mathbf{Adv}^{\Phi\text{-}se}_{\Pi,\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda)$ (defined in Definition 12) is negligible in λ.*

4 Simulation extractability of KZG in AGM

We recap the polynomial commitment scheme of Kate, Zaverucha and Goldberg [39] (KZG, shortly), highlighting in blue the parts related to hiding

Definition 14 (KZG [39]). *KZG [39] is a Polynomial Commitment scheme (see Definition 3) defined over bilinear groups $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, that consists of the following algorithms:*

$\mathsf{KGen}(1^\lambda, d)$ *on input the security parameter 1^λ, and a maximum degree bound $d \in \mathbb{N}$, outputs $\mathsf{ck} := ((\llbracket s^j \rrbracket_1)_{j\in[0,d]}, (\llbracket \alpha s^j \rrbracket_1)_{j\in[0,d]}, [1, s]_2)$, for a random secret $s, \alpha \leftarrow\!\$\ \mathbb{F}_q$.*

$\mathsf{Com}(\mathsf{ck}, f(X), r(X))$ *on input ck, a polynomial $f(X)$, and a masking polynomial $r(X)$, outputs a commitment $c := [f(s)+\alpha r(s)]_1$.*

$\mathsf{VerCom}(\mathsf{ck}, c, f(X))$ *outputs 1 if $c = [f(s)+\alpha r(s)]_1$.*

The above scheme is (standard) binding under the d-DL assumption (see Groth [34]), in fact, given two polynomials f and f' that evaluate to the same value on the secret point s, we can find s among the roots of the (non-zero) polynomial $f - f'$.

4.1 CP-SNARK for polynomial evaluation in AGM

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

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

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

$\mathsf{Verify}_{\mathtt{evl}}(\mathsf{vk}, \mathbb{x} = (\mathsf{c},x,y), (\pi,y'))$**:** output 1 if and only if:

$$e(\mathsf{c} - [y]_1 - [\alpha y']_1, [1]_2) = e(\pi, [s-x]_2).$$

The above CP-SNARK is knowledge extractable in the AGM [20]. The original work of [39] proves a weaker notion of security, called evaluation binding, which states that an adversary cannot find two distinct instances (with relative valid proofs) of the form $\mathbb{x} = (\mathsf{c},x,y)$ and $\mathbb{x}' = (\mathsf{c},x,y')$. The CP-SNARK supports a bounded (by $\mathtt{deg}(r)$) number of evaluation proofs for a given commitment. One may argue that giving more than $\mathtt{deg}(f)$ evaluations of a polynomial f on distinct points should reveal the polynomial and, thus, the zero-knowledge property would not be needed. However, there are applications in which we could give more than $\mathtt{deg}(f)$ evaluation proofs concerning f without necessarily revealing the evaluation values: e.g., this is achieved when we only show the evaluations of linear combinations of multiple committed polynomials to known constants. Since the technique of [39] would leak information on the random masking polynomials and would therefore be usable only a limited number of times, one may use the "full-fledged" CP-SNARK of Lunar [17] for proving an unbounded number of evaluations of committed polynomials in zero-knowledge, at the cost of two additional pairings at verification time.

The scheme that we describe above is zero-knowledge for non-hiding commitments and leaky zero-knowledge (Definition 11) for hiding commitments. For the latters, given a commitment $\mathsf{c} = [f(s) + \alpha r(s)]_1$, a proof for $\mathbb{x} = (\mathsf{c},x,y)$ leaks $y' = r(x)$. Thus, we prove that $\mathsf{CP}_{\mathtt{evl}}$ achieves F-leaky zero-knowledge where $F(\mathbb{x} = (\mathsf{c},x,y), \mathbb{w} = (f,r)) \coloneqq r(x)$.

We define the simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$, where $\mathcal{S}_0$ outputs the trapdoor information s, α together with the srs, and $\mathcal{S}_1$ simulates proofs for $\mathbb{x} = (\mathsf{c},x,y)$ and leakage y' outputting $\pi = ((\mathsf{c} - [y]_1 - [\alpha y']_1)(s-x)^{-1}, y')$.

Remark 1. Consider an attacker that receives a single simulated commitment c and queries the ZK-simulator twice on the same evaluation point x_j with two different evaluation values y_1 and y_2. These two queries form a linear system without solutions because we have only one variable but two linearly independent equations. These simulation queries lead to the two malleability attacks on KZG

that we describe below. For sake of simplicity, we show the attacks in the non-hiding setting.

(Arbitrary Evaluation Point) Let w_1, w_2 and c be such that $\pi_i = [w_i]_1$ for $i \in [2]$, and $\mathsf{c}^* = [c]_1$. We observe that:

$$(w_1 - w_2)(s - x_j) = y_2 - y_1. \tag{1}$$

For an arbitrary point $x^* \neq x_j$, the adversary sets $\mathsf{c}^* \leftarrow (\pi_2 - \pi_1)$, $\pi^* \leftarrow \frac{\pi_1 - \pi_2}{x^* - x_j}$ and $y^* \leftarrow \frac{y_1 - y_2}{x^* - x_j}$. The tuple $(\mathsf{c}^*, x^*, y^*, \pi^*)$ is a valid forgery. Let w^*, c^* be such that $[w^*]_1 = \pi^*$ and $[c^*]_1 = \mathsf{c}^*$. Then:

$$\begin{aligned} w^*(s - x^*) &= \frac{w_1 - w_2}{x^* - x_j}(s - x^*) = \frac{w_2 - w_1}{x^* - x_j}(x^* - x_j) + \frac{w_1 - w_2}{x^* - x_j}(s - x_j) \\ &= w_2 - w_1 - \frac{y_1 - y_2}{x^* - x_j} = c^* - y^*. \end{aligned}$$

The second equation comes directly from the definition of the elements involved and the third equation follows because of Eq. (1).

(Same Evaluation Point) For $x^* = x_j$, instead, by the homomorphic property of KZG we have $((\alpha + \beta)\mathsf{c}^*, x_j, \alpha y_1 + \beta y_2)$ is a valid forgery; we can forge a proof for c^* by setting, for example, $\alpha = 2$ and $\beta = -1$.

The attacks described in Remark 1 can be mounted because of the relation between two simulation queries obtained and can be generalized under the notion of attacks that violate the "Algebraic Consistency", as we explain hereafter. Since we focus on CP-SNARKs in which, given a proof π for a statement $\mathbb{x}$, and a view view, it is possible to derive a linear system of polynomial equations $\{p_i(x_j) = y_{i,j}\}_{i,j}$, we introduce the following definition to make the policies easier to describe.

Definition 15 (Algebraic Consistency). *A view* view *(that might contain a set of simulated instances and proofs for* CP*) satisfies the algebraic consistency for a CP-SNARK* CP *if it is possible to derive (in a way that depends on* CP*) a linear system of polynomial equations that admits a solution.*

The extraction policy for $\mathsf{CP}_{\mathtt{evl}}$. We define $\boldsymbol{\Phi}^{\mathtt{s\text{-}adpt}}_{\mathtt{evl}} = \{\Phi_{\mathcal{D}}\}_{\mathcal{D}}$ as the family (indexed by a sampler $\mathcal{D}$) of *semi-adaptive* extraction policies for the KZG-based $\mathsf{CP}_{\mathtt{evl}}$ CP-SNARK. Indeed, as we show below, the evaluation points x_j for the instances for which the adversary can see simulated proofs are selectively chosen independently of the commitment key, while the evaluation values y can be adaptively chosen by the adversary. Each policy $\Phi_{\mathcal{D}}$ is a tuple of the form $(\Phi_0^{\mathcal{D}}, \Phi_1)$, as defined in Section 3.1, where $\Phi_0^{\mathcal{D}}$ outputs the parameters pp_Φ while Φ_1 outputs a virdict bit. In particular, $\Phi_0^{\mathcal{D}}$ on input group parameters $\mathsf{pp}_{\mathbb{G}}$ outputs $\mathsf{pp}_\Phi := (\mathsf{coms}, \mathcal{Q}_x)$, where coms is a vector of commitments sampled from $\mathcal{D}$, and $\mathcal{Q}_x$ is a set of evaluation points.

For sake of clarity, we define the policy Φ_1 as the logical conjunction of a "simulator" policy Φ_{sim} and an "extractor" policy Φ_{ext}, i.e. $\Phi_1 = \Phi_{\mathsf{sim}} \wedge \Phi_{\mathsf{ext}}$. The

first policy defines rules under which we can classify a simulation query *legal*, while the second one defines rules under which the extractor must be able to extract a meaningful witness. We highlight the parts needed only for the hiding setting.

Definition 16. *Let Φ_{sim} be the policy that returns 1 if and only if:*

1. **Points check:** *let $(\mathbb{x}_i, \mathsf{aux}_i, \pi_i)_i$ be all the entries of $\mathcal{Q}_{\mathsf{sim}}$. Recall that an instance $\mathbb{x}$ can be parsed as (c, x, y). Check that $\forall i : \mathbb{x}_i.x \in \mathcal{Q}_x$.*
2. **Commitment Check:** *For all $i \in [Q_{\mathsf{sim}}]$, parse aux_i as the leakage value y'_i and the representation vectors for $\mathbb{x}_i.\mathsf{c}$ and π_i such that $\boldsymbol{r}_i = \boldsymbol{f}_i \| \boldsymbol{v}_i \| \boldsymbol{c}_i$ is the algebraic representation of the commitment $\mathbb{x}_i.\mathsf{c}$. For any i check that $\langle \boldsymbol{f}_i \| \boldsymbol{v}_i, \mathsf{ek} \rangle + \langle \boldsymbol{c}_i, \mathsf{coms} \rangle = \mathbb{x}_i.\mathsf{c}$.*
3. **Algebraic Consistency:** *The simulation queries satisfy the algebraic consistency for $\mathsf{CP}_{\mathsf{evl}}$. Let $\mathcal{I}_j := \{i : \mathbb{x}_i.x = x_j\}$ and let $\boldsymbol{R}_j := (\boldsymbol{c}_i)_{i \in \mathcal{I}_j}$. Check that $\forall j$: (i) the system of linear equations $\boldsymbol{R}_j \cdot \boldsymbol{z} = \boldsymbol{y}_j$ has at least a solution, where $\boldsymbol{z}$ are the variables and $\boldsymbol{y}_j = (\mathbb{x}_i.y - \langle \boldsymbol{f}_i, (1, x_j, \ldots, x_j^d) \rangle)_{i \in \mathcal{I}_j}$, and (ii) the system of linear equations $\boldsymbol{R}_j \boldsymbol{z}' = \boldsymbol{y}'_j$ has at least a solution, where $\boldsymbol{z}'$ are the variables and $\boldsymbol{y}'_j = (y'_i - \langle \boldsymbol{v}_i, (1, x_j, \ldots, x_j^d) \rangle)_{i \in \mathcal{I}_j}$.*

In more intuitive terms, for every simulation query (c, x, y) made by the adversary: (1) ensures that x is in the set $\mathcal{Q}_x$ chosen at the beginning of the experiment (this is the semi-adaptive restriction); (2) ensures that c is computed as a linear combination of the simulated commitments and the $\mathbb{G}_1$ elements of the SRS, but *not of simulated proofs*; (3) ensures that overall the queried statements are plausibly true (e.g., the adversary does not ask to open the same (c, x) at two different $y \neq y'$).

For the sake of concreteness, we explicitly define the algebraic consistency check in the previous definition. Notice the matrix $\boldsymbol{R}_j$ defines linear constraints that must hold for each of the polynomials for which the adversary asks evaluations. In the case of hiding commitments, some of the constraints are obtained by parsing adequately the inputs of the adversary to the simulation oracle.

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

Definition 17. *Let $\Phi_{\mathsf{ext}}, \Phi^{rnd}_{\mathsf{ext}}$ and $\Phi^{der}_{\mathsf{ext}}$ be predicates that, parsing the forgery instance $\mathbb{x}^* = (\mathsf{c}^*, x^*, y^*)$, are defined as follows:*

- *$\Phi^{rnd}_{\mathsf{ext}}$ returns 1 if and only if there exist a query (s, aux, a) to the random oracle and aux contains a non-constant polynomial $h(X)$ such that the following conditions are satisfied:*
  1. **Hashing check:** *$(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$, note that $\mathcal{Q}_{\mathsf{RO}}$ is contained in view,*
  2. **Decoding check:** *$g_c(c^*, s) = 1$.*
  3. **Polynomial check:** *$g_h(h, \mathsf{aux}) = 1$, where $g_h \colon \mathbb{F}[X] \times \{0,1\}^* \to \{0,1\}$ is a function that on input a polynomial $h(X)$ and a string aux outputs 1 iff $h(X)$ is encoded in aux.*

4. **Computation check:** $h(a) = x^*$.

– $\Phi_{\mathsf{ext}}^{der}$ *returns 1 iff* $\exists(\mathbb{x}, \cdot, \pi) \in \mathcal{Q}_{\mathsf{sim}}$ *s.t.* $\mathbb{x} \coloneqq (\mathsf{c}^*, x^*, y')$ *and* $(y', \pi) \neq (y^*, \pi^*)$.

– Φ_{ext} *returns logical disjunction of* $\Phi_{\mathsf{ext}}^{rnd}$ *and* $\Phi_{\mathsf{ext}}^{der}$.

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

In the following theorem, we focus on the non-hiding version of KZG, we show in Section 4.2 how to handle the hiding setting.

Theorem 1. *For any witness sampleable distribution* $\mathcal{D}$ *that is* $\mathcal{D}$*-Aff-MDH-secure (see Definition 5), any bilinear-group generator* $\mathsf{GroupGen}$ *that samples the generator of the group* $\mathbb{G}_1$ *uniformly at random,* $\forall \Phi_{\mathcal{D}} \in \boldsymbol{\Phi}_{\mathtt{evl}}^{\mathtt{s\text{-}adpt}}$*, the non-hiding KZG is* $\Phi_{\mathcal{D}}$*-simulation-extractable in the AGM. In particular, there exists* $\mathcal{E}$ *such that for any algebraic adversary* $\mathcal{A}$*:*

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

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

Before giving the proof of Theorem 1, we recall that when $\mathcal{D}$ is the distribution $\mathcal{U}_\ell$ that outputs ℓ uniformly random group elements of $\mathbb{G}_1$ we could reduce the $\mathcal{D}_\ell$-Aff-MDH Assumption to the Discrete Log (see Lemma 2). Hence we can state the following corollary, whose proof follows from the fact that $\epsilon_{\mathsf{DL}} \leq \epsilon_{d\text{-}\mathsf{DL}}, \forall d \geq 1$.

Corollary 1. *For any algebraic adversary* $\mathcal{A}$*, for any* $\ell \in \mathbb{N}$*, and for any distribution* $\mathcal{U}_\ell$ *that outputs* ℓ *uniformly random group elements:*

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

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

simplifies the objects involved in our analysis, from rational polynomials (with the formal variable being the trapdoor) to standard polynomials.

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

For the second case, we can reduce a $\Phi^{\mathrm{der}}_{\mathsf{ext}}$ forgery to a $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ forgery. In fact, such a forgery together with the simulated proofs set an algebraic inconsistency, a sub-case of the condition avoided by Item 3 of Definition 16, thus enabling an attack. In more detail, given a $\Phi^{\mathrm{der}}_{\mathsf{ext}}$-forgery $(\mathsf{c}, x, y), \pi$ and let $((\mathsf{c}, x, y'), \pi') \in \mathcal{Q}_{\mathsf{sim}}$ we can define a new $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$-forgery $(\mathsf{c}^*, x^*, y^*), \pi^*$ where $\mathsf{c}^* = (\pi' - \pi)$, $x^* = \mathsf{RO}(\mathsf{c}^*)$ and $\pi^* = \frac{\pi - \pi'}{x^* - x}$ and $y^* = \frac{y - y'}{x^* - x}$. We can prove that the verification equation holds noticing that $(\pi - \pi')(s - x) = [y - y']_1$ and by simple algebraic manipulations.

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

$\mathsf{coms} = ([c_i]_1)_{i\in[Q_c]}$, and $\boldsymbol{o}_\tau$ is the vector of coefficients associated to the group elements of the simulated proofs proofs. Namely, we have:

$$\mathsf{c}^* = \langle \boldsymbol{f}_{c^*}, \mathsf{ek}\rangle + \langle \boldsymbol{c}_{c^*}, \mathsf{coms}\rangle + \langle \boldsymbol{o}_{c^*}, \mathsf{proofs}\rangle \tag{2}$$

$$\pi^* = \langle \boldsymbol{f}_{\pi^*}, \mathsf{ek}\rangle + \langle \boldsymbol{c}_{\pi^*}, \mathsf{coms}\rangle + \langle \boldsymbol{o}_{\pi^*}, \mathsf{proofs}\rangle \tag{3}$$

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

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

We define our extractor to be the *canonical* extractor that returns the polynomial $f(X) \leftarrow \langle \boldsymbol{f}_{c^*}, (1, X, \ldots, X^d)\rangle$.

We start by proving that for any algebraic adversary $\mathcal{A}$ whose forgery satisfies the predicate $\Phi^{\mathrm{der}}_{\mathsf{ext}}$, there exists an algebraic adversary $\mathcal{B}$ whose forgery satisfies the predicate $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$. Let $\{\Phi'_{\mathcal{D}}\}_{\mathcal{D}}$ be the family of policies defined exactly as $\boldsymbol{\Phi}^{\texttt{s-adpt}}_{\mathsf{evl}}$ with the difference that the extracion policy Φ_{ext} is equal to $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$ (i.e., there is no logical disjunction with $\Phi^{\mathrm{der}}_{\mathsf{ext}}$).

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

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

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

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

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

which satisfies the verification equation (cf. Remark 1), and the extraction predicate $\Phi_{\mathsf{ext}}^{\mathrm{rnd}}$ when $(\hat{c}, h, a) \in \mathcal{Q}_{\mathsf{RO}}$. □

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

Hybrid $\mathbf{H}_1$. Recall that $\mathcal{D}$ is witness sampleable, thus according to Definition 4 there exists a PPT algorithm $\tilde{\mathcal{D}}$ associated with the sampler $\mathcal{D}$. The hybrid experiment $\mathbf{H}_1$ is identical to the previous one, but the group elements in coms are "sampled at exponent", i.e. we use $\tilde{\mathcal{D}}$ to generate the field elements $\boldsymbol{\gamma}$, and we let $\mathsf{coms} \leftarrow [\boldsymbol{\gamma}]_1$; we also add $\boldsymbol{\gamma}$ to $\mathsf{st}_{\mathcal{S}}$. By the witness sampleability of $\mathcal{D}$, $\mathbf{H}_0$ and $\mathbf{H}_1$ are perfectly indistinguishable, thus $\epsilon_1 = \epsilon_0$.

Hybrid $\mathbf{H}_2$. In this hybrid, we change the way we generate the SRS srs and the way in which $\mathcal{S}_1$ simulates the proofs.

Let $\left((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), [1]_1, [1]_2\right) \leftarrow\!\$\ \mathsf{GroupGen}(1^\lambda)$, sample $s \leftarrow\!\$\ \mathbb{F}$ and compute $\left[s, \ldots, s^{D+d}\right]_1, [1, s]_2$, where $D \leftarrow Q_x + 1$. Let $x_r \leftarrow\!\$\ \mathbb{F}$, and let $p(X)$ be the vanishing polynomial in $\mathcal{Q}_x \cup \{x_r\}$, namely:

$$p(X) \coloneqq (X - x_r) \prod_{x \in \mathcal{Q}_x} (X - x).$$

Let also $p_j(X) \coloneqq p(X)(X - x_j)^{-1}$, for $j \in [Q_x]$. In $\mathbf{H}_2$ we have that:

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

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

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

Lemma 4. $\epsilon_2 = \epsilon_1$.

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

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

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

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

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

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

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

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

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

Lemma 5. $\epsilon_3 = \epsilon_2$.

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

$$\mathsf{c}^* \coloneqq \mathsf{c}' - [f(s)p(s)]_1 \,, \quad \pi^* \coloneqq \pi' - [q(s)p(s)]_1 \,, \quad y^* \coloneqq y' - f(x^*).$$

By the verification equation of the forgery of $\mathcal{A}_{\mathsf{c}}$ we have:

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

where the last equation holds since $q(X)(X - x') = (f(X) - f(x'))$ and $x^* = x'$.

Finally, notice that a forgery is valid for $\mathcal{A}$ if it provides a string s that satisfies the "hash check" of Φ_{ext}. We have that there exist s, aux, a, and $h(X)$ such that: (i) $g_c(\mathsf{c}^*, s) = 1$, (ii) $g_h(h, \mathsf{aux}) = 1$, (iii) $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO},\mathcal{A}}$, and (iv) $x^* = h(a)$ for the forgery of $\mathcal{A}$.

The way $\mathcal{A}_{\mathsf{c}}$ simulates the RO queries ensures that for the query s of $\mathcal{A}$ to the RO, the core adversary sent the "core" RO query s_{c} that encodes both c' and c^*, thus we have that (i) $g_c(\mathsf{c}^*, s_{\mathsf{c}}) = 1$, (ii) $g_h(h, \mathsf{aux}_{\mathsf{c}}) = 1$, (iii) $(s_{\mathsf{c}}, \mathsf{aux}_{\mathsf{c}}, a) \in \mathcal{Q}_{\mathsf{RO}}$, and (iv) $x^* = h(a)$ for the forgery of $\mathcal{A}_{\mathsf{c}}$.

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

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

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

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

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

Consider the following polynomials:

$$c^*(X) \coloneqq \sum_{i \in [Q_c]} c_{c^*,i} \cdot \gamma_i p(X) + \sum_{\boldsymbol{r},j,k} o_{c^*,\boldsymbol{r},j,k} \cdot \pi_{\boldsymbol{r},j,k}(X)$$

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

$$v(X) \coloneqq c^*(X) - y^* p(X) - (X - x^*)\pi^*(X)$$

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

Next, we show that when the forgery of the adversary is valid the probability of $\boldsymbol{f}_{\pi^*} \neq \mathbf{0}$ or $\boldsymbol{c}_{\pi^*} \neq \mathbf{0}$ is bounded by $\epsilon_{(Q_x+d+1)\text{-DL}}$.

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

$$c^*(X) - y^* p(X) - (X - x^*)\pi^*(X) = v(X) = 0. \tag{4}$$

By the guarantees of the AGM, the polynomial $\pi^*(X)$ is a linear combination of elements that depend on $X^{i-1}p(X)$ for $i \in [d+1]$ and $p_j(X)$ for $j \in [Q_x]$. However, when the verification equation holds, the degree of $\pi^*(X)$ must be

strictly less than the degree of $p(X)$, because, by Eq. (4), $v(X)$ would contain a non-zero coefficient of degree Q_x+d+1 which in particular implies that $v(X) \not\equiv 0$. Then it must be the case that the forged proof $\pi^*(s)$ is a linear combination of the simulated proofs only, thus both $\boldsymbol{f}_{\pi^*}$ and $\boldsymbol{c}_{\pi^*}$ are null. □

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

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

$$\langle \boldsymbol{o}_\tau, \mathsf{proofs} \rangle = \langle \boldsymbol{o}''_\tau, ([p_j(s)]_1)_j \rangle. \tag{5}$$

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

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

Proof. We begin by showing that the hybrid can compute such alternative representations efficiently. We proceed in steps.

Let us parse the simulated proofs $\mathsf{proofs} \coloneqq (\pi_{j,\ell})_{j,\ell}$ such that $\pi_{j,\ell}$ is the ℓ-th simulated proof obtained by $\mathcal{S}_1$ on a query involving the j-th point x_j, i.e., $((x_j, \hat{\mathsf{c}}_{j,\ell}, y_{j,\ell}), \mathsf{aux}_{j,\ell})$. Also, let $\boldsymbol{c}_{j,\ell}$ be the algebraic representation for the group element $\hat{\mathsf{c}}_{j,\ell}$ in $\mathsf{aux}_{j,\ell}$. For every $j \in [Q_x]$, we define $\boldsymbol{R}_j$ as the $Q_c \times Q_c$ matrix whose ℓ-th column is $\boldsymbol{c}_{j,\ell}$.

By construction of $\mathcal{S}_1$ in this hybrid we have that for every $j \in [Q_x]$ it holds

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

and thus $\boldsymbol{\pi}_j \coloneqq \left[(\boldsymbol{R}_j^\top \boldsymbol{\gamma} - \boldsymbol{y}_j) p_j(s)\right]_1$ with $\boldsymbol{y}_j \coloneqq (y_{j,\ell})_\ell$.

Without loss of generality, we assume that for each x_j the adversary makes the maximum number of simulation queries (i.e., $\ell \in [Q_c]$); therefore $\boldsymbol{R}_j$ is a full rank matrix (this follows from the fact that the simulation queries of the adversary satisfy the policy Φ_{sim}, and in particular the algebraic consistency of the policy, see Item 3).

Given any vector $\boldsymbol{o}_\tau$ with $\tau \in \{\mathsf{c}^*, \pi^*\}$, its m-th entry $o_{\tau,m}$ corresponds to the m-th simulated proof in proofs. Therefore, similarly to above, we denote by $o_{\tau,j,\ell}$ the entry corresponding to proof $\pi_{j,\ell}$ and we define $\boldsymbol{o}_{\tau,j} \coloneqq (o_{\tau,j,\ell})_\ell$.

Then, for every $j \in [Q_x]$ we define

$$\boldsymbol{o}'_{\tau,j} \leftarrow \boldsymbol{R}_j \cdot \boldsymbol{o}_{\tau,j} \tag{6}$$

$$\boldsymbol{\pi}'_j \leftarrow (\boldsymbol{R}_j^\top)^{-1} \cdot \boldsymbol{\pi}_j \tag{7}$$

from which we derive that for any τ:

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

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

For all $j \in [Q_x]$ let $\boldsymbol{z}_j \coloneqq (\boldsymbol{R}_j^\top)^{-1} \cdot \boldsymbol{y}_j$, and note that

$$\boldsymbol{\pi}'_j = [(\boldsymbol{\gamma} - \boldsymbol{z}_j) p_j(s)]_1 ,$$

namely $\pi'_{j,i}$ is a valid proof for the instance $(\mathsf{c}_i, x_j, z_{j,i})$ w.r.t. the simulated SRS.

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

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

which proves the first part of the lemma, i.e., computing $\boldsymbol{o}''_{\tau,j}$ satisfying Eq. (5).

In what follows, we prove that if the event that $\mathbf{H}_5$ outputs 0 but $\mathbf{H}_4$ would output 1, namely that all the conditions of $\mathbf{H}_4$ hold but $\boldsymbol{o}_{\mathsf{c}^*} \neq \mathbf{0} \wedge \boldsymbol{o}''_{\mathsf{c}^*} = \mathbf{0}$, then we can break the Aff-MDH assumption.

First, notice that for any j $\boldsymbol{o}_{\mathsf{c}^*,j} \neq \mathbf{0}$ implies that $\boldsymbol{o}'_{\mathsf{c}^*,j} \neq \mathbf{0}$, because the linear transformation applied to compute $\boldsymbol{o}'_{\mathsf{c}^*,j}$ is full rank. Second, take an index j^* such that $\boldsymbol{o}_{\mathsf{c}^*,j^*} \neq \mathbf{0}$ and set $\boldsymbol{A} \leftarrow \boldsymbol{o}'_{\mathsf{c}^*,j^*}$ and $\zeta \leftarrow \langle \boldsymbol{z}_{j^*}, \boldsymbol{o}'_{\mathsf{c}^*,j^*} \rangle$.

By the above definition of the values o''_{c^*,j^*} and our assumption that the "bad event" of this hybrid is $\boldsymbol{o}''_{\mathsf{c}^*} = \mathbf{0}$, we have that:

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

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

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

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

Proof. We bound the probability that the adversary loses in $\mathbf{H}_6$ but not in $\mathbf{H}_5$, namely, the probability that $\boldsymbol{r}_c^* \neq \mathbf{0}$ but the conditions of $\mathbf{H}_5$ hold. We show a reduction $\mathcal{B}$ to the Aff-MDH when this event happens.

First of all, we can assume that the core adversary outputs coefficients $\boldsymbol{f}_{c^*} = \boldsymbol{f}_{\pi^*} = \boldsymbol{c}_{\pi^*} = \mathbf{0}$, i.e. the adversary only makes use of previous commitments $\mathsf{c}_i \in \mathsf{coms}$ and simulated proofs $\pi_{\boldsymbol{r},j,k} \in \mathsf{proofs}$ to represent c^*, and only uses the simulated proofs to represent the proof π^*.

The reduction $\mathcal{B}$ takes as input a distribution $[\boldsymbol{\gamma}]_1$ and runs the experiment as in $\mathbf{H}_5$. $\mathcal{B}$ aborts if the forgery $(\mathsf{c}^*, x^*, y^*, \pi^*)$ returned by the adversary is not valid (i.e. either the extraction predicate or the verification equation is not satisfied) or $\boldsymbol{r}_{c^*} = \mathbf{0}$. Otherwise, we have that:

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

where $\boldsymbol{r}_{c^*} \neq \mathbf{0}$ if $\boldsymbol{o}_{c^*} \neq \mathbf{0} \vee \boldsymbol{c}_{c^*} \neq \mathbf{0}$.

We can then rewrite the commitment and the proof of forgery of the core adversary as a function of the coefficients $\boldsymbol{o}''_{c^*}$ and $\boldsymbol{o}''_{\pi^*}$ (as computed in the $\mathbf{H}_5$):

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

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

$$\sum_{i \in [Q_c]} c_{c^*,i} \gamma_i p(s) + \sum_{j \in [Q_x]} o''_{c^*,j} p_j(s) - p(s) y^* = \sum_{j \in [Q_x]} o''_{\pi^*,j} p_j(s)(s - x^*) \tag{8}$$

For all $j \in [Q_x]$, we define $\delta_j \coloneqq x_j - x^*$. We can rewrite the r.h.s. of Eq. (8) as:

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

In Eq. (8), we group all the terms that depend on $p(s)$ on the left side, and we move all the terms that depend on $p_j(s)$ to the right side, thus obtaining:

$$\underbrace{\Big(\sum_{i \in [Q_c]} c_{c^*,i} \gamma_i - \sum_{j \in [Q_x]} o''_{w^*,j} - y^* \Big)}_{A} p(s) = \sum_{j \in [Q_x]} \underbrace{\Big(o''_{w^*,j} \delta_j - o''_{c^*,j} \Big)}_{B_j} p_j(s) \tag{9}$$

Let $f(X) \coloneqq Ap(X) - \sum_{j \in [Q_x]} B_j p_j(X)$. Notice that because of Eq. (9) we have $f(s) = 0$, thus we can assume $f(X) \equiv 0$, as otherwise we can reduce, by Lemma 1, to the $(Q_x + d + 1)$-DL assumption. It must be the case that both $\sum_{j \in [Q_x]} B_j p_j(s) = 0$ and $A = 0$ because the degree of $p(X)$ and of $p_j(X)$ for any j are different. Moreover, the polynomials $p_j(X)$ are linearly independent,

namely the only linear combination $\sum_j a_j p_j(X) = 0$ is the trivial one where the coefficients $a_j = 0$[15], thus $B_j = 0$ for all j. We have that $o''_{w^*,j}\delta_j - o''_{c^*,j} = 0, \forall j$. Thus we can rewrite the coefficients $o''_{\pi^*,j} = \frac{o''_{c^*,j}}{\delta_j}$, $\forall j$. Since A must be 0:

$$\sum_{i\in[Q_c]} c_{c^*,i}\gamma_i - \sum_{j\in[Q_x]} \frac{o''_{c^*,j}}{\delta_j} - y^* = 0. \tag{10}$$

$\mathcal{B}$ can plug the definition of the coefficients $o''_{c^*,j}$ in Eq. (10) and derive:

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

Above, the second equation follows from the distributive property of the sum, while in the last step we have grouped the terms depending on γ_i. In particular, the last equation shows that $\mathcal{B}$ can make a forgery in the Aff-MDH game since it knows $z \coloneqq y^* - \sum_{i,j} \frac{o'_{c^*,i,j} z_{j_i}}{\delta_j}$ and coefficients $A_i \coloneqq c_{c^*,i} - \sum_j \frac{o'_{c^*,i,j}}{\delta_j}$ such that:

$$\sum_{i\in[Q_c]} A_i \left[\gamma_i\right]_1 = \left[z\right]_1 .$$

For this to be a valid solution in the Aff-MDH game, we need the existence of at least an index i such that $A_i \neq 0$. We show that this occurs with all but negligible probability, i.e., $\Pr[\exists i \in [Q_c] : A_i \neq 0] \geq 1 - \mathsf{negl}(\lambda)$.

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

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

Below, we argue that $\Pr[A_\mu = 0]$ is negligible based on the randomness of x^* which is chosen by the random oracle after defining A_μ, and we make use of the assumption that $\boldsymbol{r}_{\mathsf{c}^*} \neq 0$.

We claim that the value $A_\mu = c_{\mathsf{c}^*,\mu} - \sum_j \frac{o'_{\mathsf{c}^*,j,\mu}}{(x^* - x_j)}$ can be fixed before the random oracle query $\mathbb{x}^*$ is made. To this end, we start by showing that $\boldsymbol{o}'_{\mathsf{c}^*,j}$ does not depend on x^*. Let $B(j) \subseteq [Q_c]$ be the subset of indices of the simulation queries that involve x_j and that occurred before the random oracle query that

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

returned x^*. We observe that for every $\eta \in B(j)$ it must be $o_{\mathsf{c}^*,j,\eta} = 0$ since the simulated proof $\pi_{j,\eta}$ is not in the view of the adversary. Therefore:

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

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

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

Notice that, once the coefficients $\boldsymbol{c}_{\mathsf{c}^*}$ and $\boldsymbol{o}'_{\mathsf{c}^*,j}$ are fixed, the coefficient A_μ can be seen as function of $x^* \in \mathbb{Z}_q$, i.e. $A_\mu = A_\mu(x^*)$, where:

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

Notice that $A_\mu(X)(\prod_j (X - x_j))$ vanishes in at most Q_x points in $\mathbb{F} \setminus \mathcal{Q}_x$ and vanishes in the set of points $\mathcal{Q}_x$. Let $\mathcal{R}$ be the set of the roots of such a polynomial, since $\forall i \in [n]$, h_i is defined before x^* is computed, and by union bound:

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

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

Hybrid $\mathbf{H}_7$. The hybrid $\mathbf{H}_7$ additionally checks that $y^* \neq 0$, and if the condition holds the adversary $\mathcal{A}_{\mathsf{c}}$ loses the game.

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

Proof. We reduce to the evaluation binding of KZG polynomial commitment for polynomials of maximum degree $Q_x + 1 + d$, which, in turn, can be reduced to $(Q_x + 1 + d)$-strong Discrete Log assumption. Let $\mathcal{B}$ be the reduction that upon input $\mathsf{pp}_{\mathbb{G}}, \mathsf{ck} = \left[1, s, \ldots, s^{Q_x+d+1}\right]_1, [1, s]_2$ simulates experiment $\mathbf{H}_4$ for the adversary $\mathcal{A}_{\mathsf{c}}$. Eventually, $\mathcal{A}_{\mathsf{c}}$ outputs its forgery $(\mathsf{c}^*, x^*, y^*, \pi^*)$, and $\mathcal{B}$ aborts if $y^* = 0$. The reduction sets $\tilde{f}(X) \coloneqq -y^* p(X)$, sets $y \coloneqq \tilde{f}(x^*)$, and computes π to be a valid KZG-proof for $(\left[\tilde{f}(s)\right]_1, x^*, y)$. The forgery against evaluation biding of the reduction is set to be (y, π) and $(0, \pi^*)$ for the commitment $\left[\tilde{f}(s)\right]_1$ on the point x^*. We need to show that:

1. $(\left[\tilde{f}(s)\right]_1, x^*, 0, \pi^*)$ satisfies the verification equation of KZG commitment where the commitment key is set to ck
2. $y \neq 0$

For the first item notice that, by the definition of core adversary, we have that $\boldsymbol{r}_{c^*} = \mathbf{0}$ thus $c^* = 0$. Therefore, by the verification equation:

$$e([0]_1 - y^* [p(s)]_1, [1]_2) = e(\left[\tilde{f}(s)\right]_1 - 0\,[1]_1, [1]_2) = e(\pi^*, [s]_2 - x^* [1]_2).$$

For the second item, notice that $\tilde{f}(x^*) = 0$ if and only if x^* is a root of $p(X)$, i.e. $x^* \in \mathcal{Q}_x$ or $x^* = x_r$. Thus, similarly to the previous lemma, by the assumption on the polynomials h_i and by union bound:

$$\Pr[\exists i : h_i(\mathsf{RO}(s)) \in \mathcal{Q}_x \cup \{x_r\}] \leq n Q_{\mathsf{RO}}(Q_x + 1) \frac{\max_i \deg(h_i)}{q}.$$

□

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

4.2 Simulation extractability of Hiding KZG

We extend the result of Theorem 1 to the case in which the $\mathsf{CP}_{\mathtt{evl}}$ based on KZG uses hiding commitments. In particular, we show a reduction to the simulation extractability of non-hiding KZG for uniform distributions $\mathcal{U}_\ell$ of the commitments.

Theorem 2. *Let $\mathcal{U}_\ell$ be the distribution that outputs ℓ uniformly random group elements in $\mathbb{G}_1$. $\forall \ell \in \mathbb{N}$, the hiding KZG $\mathsf{CP}_{\mathtt{evl}}$ is $\Phi_{\mathcal{U}_\ell}$-simulation-extractable in the AGM. In particular, there exists $\mathcal{E}$ such that for any algebraic adversary $\mathcal{A}$:*

$$\mathbf{Adv}^{\Phi\text{-}se}_{\mathsf{CP}_{\mathtt{evl}},\mathcal{A},\mathcal{S},\mathcal{E},\mathcal{U}_\ell}(\lambda) \leq O(\ell \epsilon_{q\text{-}\mathsf{DL}}(\lambda)) + \mathsf{poly}(\lambda)\epsilon_h(\lambda).$$

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

Proof. We reduce an adversary $\mathcal{A}$ for the simulation extractability of $\Pi_{\mathtt{evl}}$ with hiding commitments to an adversary $\mathcal{B}$ for the simulation extractability with non-hiding commitments and a uniform distribution that generates twice as many commitments. $\mathcal{B}$ receives as input the description of a bilinear group $\mathsf{pp}_{\mathbb{G}}$ and an SRS $\mathsf{srs} \coloneqq (\mathsf{ek}, \mathsf{vk})$ where $\mathsf{ek} = \left[1, s, \ldots, s^d\right]_1$ and $\mathsf{vk} = [1, s]_2$; $\mathcal{B}$ samples $\alpha \leftarrow\!\!\$\, \mathbb{Z}_q$ and computes $\mathsf{ek}_\alpha \leftarrow \left[\alpha, \alpha s, \ldots, \alpha s^d\right]_1$ and forwards to $\mathcal{A}$ the SRS $\mathsf{srs}' \coloneqq ((\mathsf{ek}, \mathsf{ek}_\alpha), \mathsf{vk})$. Finally, $\mathcal{B}$ parses the list of commitments coms as two lists of equal size $(\mathsf{coms}_1, \mathsf{coms}_2)$ and gives $\mathcal{A}$ the commitements $\mathsf{coms}' = \mathsf{coms}_1 + \alpha\mathsf{coms}_2$. Every time $\mathcal{A}$ comes up with some group element c, attaches a representation vector $\boldsymbol{r}_c$ that we can parse as $\boldsymbol{f}_c \| \boldsymbol{v}_c \| \boldsymbol{c}_c \| \boldsymbol{o}_c$ where $\boldsymbol{f}_c$ (resp. $\boldsymbol{v}_c$) is the vector of coefficients associated to group elements ek (resp. ek_α), $\boldsymbol{c}_c$ is the vector of coefficients associated to group elements coms', and $\boldsymbol{o}_c$ is the vector of coefficients associated to the group elements of the simulated proofs proofs'. The strategy of $\mathcal{B}$ is to forward the simulation queries of $\mathcal{A}$ whose representation vectors are given w.r.t. srs', attaching the representation vectors w.r.t. srs.

On input a simulation query of the form $\mathbb{x} = (\mathsf{c}, x_j, y)$ and leakage y', with $\boldsymbol{r}_c = \boldsymbol{f}_c \| \boldsymbol{v}_c \| \boldsymbol{c}_c$ as representation vector for c, $\mathcal{B}$ invokes the simulation oracle $\mathcal{S}_1$ first on input $\mathbb{x}_1 = (\langle \boldsymbol{f}_c + \alpha \boldsymbol{v}_c, \mathsf{ek} \rangle + \langle \boldsymbol{c}_c \| \boldsymbol{0}, \mathsf{coms} \rangle, x_j, y)$ and then on $\mathbb{x}_2 = (\langle \boldsymbol{0} \| \boldsymbol{c}_c, \mathsf{coms} \rangle, x_j, y')$, obtaining the two proofs π_1 and π_2. Finally returns to $\mathcal{A}$ the proof $\pi \coloneqq \pi_1 + \alpha\pi_2$ which satisfies the verification equation:

$$e(\mathsf{c} - [y]_1 - [\alpha y']_1, [1]_2) = e(\pi, [s - x_j]_2)$$

and adds π to proofs'. When $\mathcal{A}$ makes the forgery $(\mathsf{c}^*, x^*, y^*, y'^*, \pi^*)$, with associated representation vectors $\boldsymbol{r}'_\tau = \boldsymbol{f}_\tau \| \boldsymbol{v}_\tau \| \boldsymbol{c}_\tau \| \boldsymbol{o}_\tau$, for $\tau \in \{c^*, \pi\}$, $\mathcal{B}$ forwards the forgery $(\mathsf{c}, x^*, y^* + \alpha y'^*, \pi^*)$. $\mathcal{B}$ needs to attach representation vectors for the group elements c^*, π^* w.r.t. $\mathsf{srs}, \mathsf{coms}$ and proofs; to do that, $\mathcal{B}$ applies the following transformation and computes $\boldsymbol{r}_\tau = \boldsymbol{f}_\tau + \alpha \boldsymbol{v}_\tau \| \boldsymbol{c}_\tau \| \alpha \boldsymbol{c}_\tau \| (o_{\tau,i} \| \alpha o_{\tau,i})_i$. This choice, indeed, is such that:

$$\langle \boldsymbol{r}'_\tau, (\mathsf{ek} \| \mathsf{ek}_\alpha \| \mathsf{coms}' \| \mathsf{proofs}') \rangle = \langle \boldsymbol{r}_\tau, (\mathsf{ek} \| \mathsf{coms} \| \mathsf{proofs}) \rangle$$

If the forgery of $\mathcal{A}$ is valid, so is the forgery of $\mathcal{B}$ and, unless with negligible probability, the canonical extractor $\mathcal{E}$ extracts the valid witness $f(X) + \alpha v(X)$.

5 Simulation-Extractable Universal zkSNARKs

In this section we show our compiler for universal SNARKs based on polynomial IOPs.

5.1 Polynomial Holographic Interactive Oracle Proofs

Definition 18 (Polynomial Holographic IOP). *Let $\mathcal{F}$ be a family of finite fields and let $\mathcal{R}$ be an indexed relation. A (public-coin non-adaptive) Polynomial Holographic IOP over $\mathcal{F}$ for $\mathcal{R}$ is a tuple $\mathsf{PIOP} = (r, n, m, D, n_e, \mathsf{RE}, \mathcal{P}, \mathcal{V})$ where $r, n, m, D, n_e \colon \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions, and $\mathsf{RE}, \mathcal{P}, \mathcal{V}$ are three algorithms for the encoder, prover and verifier respectively, that work as follows.*

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

Online phase: *The prover* $\mathcal{P}(\mathbb{F}, \mathtt{i}, \mathtt{x}, \mathtt{w})$ *and the verifier* $\mathcal{V}^{\mathsf{RE}(\mathbb{F},\mathtt{i})}(\mathbb{F}, \mathtt{x})$ *are executed for* $r(|\mathtt{i}|)$ *rounds; the prover has a tuple* $(\mathbb{F}, \mathtt{i}, \mathtt{x}, \mathtt{w}) \in \mathcal{R}$ *and the verifier has an instance* $\mathtt{x}$ *and oracle access to the polynomials encoding* $\mathtt{i}$*.*
In the i*-th round,* $\mathcal{P}$ *sends* $m(i)$ *messages* $\{\pi_{i,j} \in \mathbb{F}\}_{j\in[m(i)]}$*, and* $n(i)$ *oracle polynomials* $\{p_{i,j} \in \mathbb{F}[X]\}_{j\in[n(i)]}$ *of degree at most* $D := D(|\mathtt{i}|)$*, while* $\mathcal{V}$ *replies (except for the last round) with a uniformly random message* $\rho_i \in \mathbb{F}$*.*

Decision phase: *After the* $r := r(|\mathtt{i}|)$*-th round, let* $n_e := n_e(|\mathtt{i}|)$*, the verifier* $\mathcal{V}(\mathbb{F}, \mathtt{x}, \boldsymbol{\rho})$*, on input the description of the field* $\mathbb{F}$*, the input* $\mathtt{x}$ *and all the random messages of the verifier* $\boldsymbol{\rho} := (\rho_1, \ldots, \rho_{r-1})$*, outputs tuples* $(G^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})_{k\in[n_e]}$ *which define the following algebraic checks. Let* $n := \sum_{k=0}^{r} n(k)$*, let* $m := \sum_{k=1}^{r(|\mathtt{i}|)} m(k)$*, and denote by* $(p_1, ..., p_n)$ *all the oracle polynomials (including the* $n(0)$ *ones from the encoder) and by* $(\pi_1, ..., \pi_m)$ *all the messages sent by the prover. For any* $k \in [n_e], j \in [n]$ *we have* $G^{(k)} \in \mathbb{F}[X, X_1, ..., X_n, Y_1, ..., Y_m]$ *and* $v_j^{(k)} \in \mathbb{F}[X]$*. A tuple* $(G^{(k)}, v_1^{(k)}, ..., v_n^{(k)})$ *is satisfied if and only if* $F^{(k)}(X) \equiv 0$ *where:*

$$F^{(k)}(X) := G^{(k)}(X, \{p_i(v_i^{(k)}(X))\}_{i\in[n]}, \{\pi_i\}_{i\in[m]}) \tag{11}$$

The verifier accepts if and only if all the checks are satisfied.

In our work, we use PIOPs with some slight refinements.[16] The first one, called *(non-adaptive) algebraic verifiers* (see Definition 19), says that the above polynomials $v_j^{(k)}$ do not depend on the instance and can be expressed as polynomial functions of $\mathcal{V}$'s random coins, i.e., $v_j^{(k)}(X) = \tilde{v}_j^{(k)}(X, \boldsymbol{\rho})$ for some instance-independent $\tilde{v}_j^{(k)}$.

Definition 19 (Non-adaptive Algebraic Verifier). *A PIOP* PIOP *is* (non-adaptive) algebraic verifier *if there exists an alternative deterministic PT algorithm* $\tilde{\mathcal{V}}$ *such for any* $\mathtt{i}$ *and* $\mathtt{x}$ *we have* $(\tilde{v}_1^{(k)}, \ldots, \tilde{v}_n^{(k)})_{k\in[n_e]} \leftarrow \tilde{\mathcal{V}}(\mathbb{F}, |\mathtt{i}|)$ *where for any* $j \in [n]$ *and* $k \in [n_e]$ *we have* $\tilde{v}_j^{(k)} \in \mathbb{F}[X, X_1, \ldots, X_{r-1}]$*. Also, for any field elements* $\rho_1, \ldots, \rho_{r-1}$*, let* $(G^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})_{k\in[n_e]} \leftarrow \mathcal{V}(\mathbb{F}, \mathtt{i}, \mathtt{x}, \rho_1, \ldots, \rho_{r-1})$*, we have for any* $i \in [r(|\mathtt{i}|) - 1], j \in [n_{i-1}, n_i], k \in [n_e]$*:*

$$\tilde{v}_j^{(k)}(X, \rho_1, \ldots, \rho_{r-1}) = v_j^{(k)}(X)$$

The second one is a more restrictive[17] concept of soundness called *state-restoration straight-line knowledge soundness* (see Definition 20). This notion combines the notion of state-restoration soundness from [9] with the concept

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

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

of straight-line extractability from [17]. For further clarification, the malicious prover $\tilde{\mathcal{P}}$ engages in a game with the honest verifier $\mathcal{V}$ and has the additional ability to *roll back* the interaction with the verifier to a previous state. At some point, the interaction may reach a final state. The prover is considered successful if it produces an accepting transcript, while the extractor, given such a transcript that includes all the oracle polynomials, fails to produce a valid witness.

Let $\mathbf{Exp}^{sr}_{\tilde{\mathcal{P}},\mathsf{PIOP},\mathcal{E}}(\mathbb{F})$ be the following experiment:

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

Definition 20 (State-restoration (straight-line) proof of knowledge). *A PIOP* PIOP *is* state-restoration (straight-line) proof of knowledge *if there exists an extractor* $\mathcal{E}$ *such that for any* $\tilde{\mathcal{P}}$ *and any* $\mathbb{F}$*:*

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

The third one is that we we do not explicitly check the degree of the polynomials. This is mainly for simplicity in the presentation of the compiler since, for each poly p_i where we check degree d_i the prover $\mathcal{P}$ additionally sends $p'_i(X) = p(X) \cdot X^{D-d_i}$ where D is the maximum degree, and $\mathcal{V}$ additionally checks the equation $\sum \rho^i \cdot (X^{D-d_i} p_i(X) - p'_i(X)) \equiv 0$, for a randomizer ρ.

Similarly to previous work, we use the notion of bounded zero-knowledge of [20,17].

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

Definition 21 ($\boldsymbol{b}$-bounded Zero-Knowledge). *A PIOP* PIOP *is* $\boldsymbol{b}$*-Zero-Knowledge if there exists a checker* C *such that* $\Pr[C(i, x) = 0] \leq \mathsf{negl}(|\mathbb{F}|)$ *over*

random x such that for every index $\mathtt{i}$, and $(\mathsf{pp}, \mathtt{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$, and every $(\boldsymbol{b}, C)$-bounded list $\mathcal{L}$, the following random variables are within ϵ statistical distance:

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

where $p_1, \ldots, p_n$ are the polynomials returned by the prover $\mathcal{P}$ and $\mathcal{S}$ is a PPT simulator. Moreover, PIOP *is* special honest-verifier $\boldsymbol{b}$-bounded zero-knowledge *if $\mathcal{S}$ first samples uniformly at random the verifier's messages $\rho_1, \ldots, \rho_{r-1}$ and then computes the full transcript. Namely, $\mathcal{S}$ can take as additional input the verifier's messages.* PIOP *is* independent leakage *if $\mathcal{S}$ can be divided in two algorithms $(\mathcal{S}_0, \mathcal{S}_1)$ where $\mathcal{S}_0$ outputs the simulated transcript, and $\mathcal{S}_1$ the leakage. Namely $\mathcal{S}(\mathbb{F}, \mathtt{i}, \mathbb{x}, \mathcal{L}; r) = \mathcal{S}_0(\mathbb{F}, \mathtt{i}, \mathbb{x}; r), \mathcal{S}_1(\mathbb{F}, \mathtt{i}, \mathbb{x}, \mathcal{L}; r)$.*

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

Definition 22 (Compiler-safe PIOP). *A PIOP* PIOP *is* compiler-safe *if for any $\mathtt{i}, \mathbb{x}$ and $\boldsymbol{\rho} \coloneqq \rho_1, \ldots, \rho_{r-1}$ and any tuple $(G^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})_{j\in[n_e]} \leftarrow \mathcal{V}(\mathbb{F}, \mathtt{i}, \mathbb{x}, \boldsymbol{\rho})$ there exists an index k such that for all j the polynomials $v_j^{(k)}$ are of degree at least one.*

We notice that some PIOPs have this technical condition (an example is [27]) while other PIOPs that internally run different sub-protocols might not have this property. However, at PIOP level we can always obtain this property by adding an extra degree of randomness to the polynomials (obtaining $\boldsymbol{b} + \mathbf{1}$-bounded zero-knowledge) and adding an extra trivial equation over the polynomials with the v_j set to be the identity function.

Commitments Simulator. We abstract how our compiler handles the simulation of the commitments for the oracles sent during a PIOP protocol's execution.

Definition 23 (Commitment Simulator for PIOP). *Let* PIOP *be a PIOP with $\boldsymbol{b}$-Zero-Knowledge simulator $\mathcal{S}$ and checker C and consider a two-stage PPT algorithm where $\mathcal{S}_{\mathsf{Com}}(0, \mathsf{ck})$ outputs a vector* coms *of commitments, and $\mathcal{S}_{\mathsf{Com}}(1, |\mathtt{i}|, \mathbb{x})$ outputs a matrix $\boldsymbol{M}_{\mathtt{i},\mathbb{x}}$ of linearly independent rows. $\mathcal{S}_{\mathsf{Com}}$ is a* commitment simulator *for a* CP *(defined over the same commitment scheme) and* PIOP *if for every $(\boldsymbol{b}, C)$-bounded list $\mathcal{L}$, $\mathbb{F}, \mathtt{i}$ and $\mathbb{x}$ the following distributions are computationally indistinguishable:*

$$\begin{aligned}&\big(\mathsf{ck}, \mathsf{view}\big(\mathcal{P}(\mathbb{F}, \mathtt{i}, \mathbb{x}, \mathbb{w}), \mathcal{V}\big), (p_j(x))_{(j,x)\in\mathcal{L}}, (\mathsf{Com}(\mathsf{ck}, p_i))_{i\in[n]}\big) \approx_c \\ &(\mathsf{ck}, \mathcal{S}(\mathbb{F}, \mathtt{i}, \mathbb{x}, \mathcal{L}), (\boldsymbol{M}_{\mathtt{i},\mathbb{x}} \cdot \mathsf{coms} \| \mathsf{ck}))\end{aligned}$$

Moreover, we have that the view $(\mathcal{L}, \mathcal{S}(\mathbb{F}, \mathtt{i}, \mathbb{x}, \mathcal{L}), \mathbb{x}, \boldsymbol{M}_{\mathtt{i},\mathbb{x}})$ satisfies the algebraic consistency for the CP-SNARK CP *(as in Definition 15).*

The definition is in two stages so to fit our result of Theorem 1. For KZG the first part handles the generation of instance-independent commitments according to a $\mathcal{D}_k$-Aff-MDH Assumption, while the second part acts over the formerly sampled commitments adapting them to the instance at hand. If the commitment scheme is hiding then the task of defining the commitment simulator $\mathcal{S}_{\mathsf{Com}}$ for an arbitrary PIOP is trivial (the distribution of $\mathcal{S}_{\mathsf{Com}}(0, \mathsf{ck})$ is uniformly over the commitment space), thus this property is interesting for the case of non-hiding commitments. Similarly, this notion can be applied to commitment schemes that are not homomorphic: in that case, we must require $\boldsymbol{M}_{\mathtt{i},\mathbb{x}}$ to be the identity.

5.2 The Compilation-Ready CP-SNARK

Instead of compiling directly a PIOP through a polynomial commitment in its simplest form (i.e., an evaluation proof for each polynomial queried in the PIOP), we take an alternative road similar to [17]. Namely, we assume the existence of a CP-SNARK that, w.r.t. a tuple of commitments $(\mathsf{c}_j)_{j\in[n]}$, is capable of proving either knowledge of polynomials $(p_j)_{j\in[n]}$ opening these commitments, or that the committed polynomials satisfy a statement like the one in Eq. (11) (i.e., that the oracles committed in $(\mathsf{c}_j)_{j\in[n]}$ would make the PIOP verifier accept)[18]. We call this building block a *compilation-ready* CP-SNARK (CP, shortly), and informally we refer to the former type of statements as "proof of knowledge" and to the latter as "PIOP verifier". While our compilation strategy follows previous work, our novel contribution is to properly define the properties that this CP-SNARK must satisfy in order to argue that the result of the compiler is simulation-extractable, and not only knowledge-sound. These properties are mainly three. The first one is that the CP prover can "append" arbitrary messages to the proven instances. Looking ahead to our compiler, this feature is used so that prover and verifier can append the (hash of the) protocol's transcript to the proven instance, in such a way that a CP proof acts as a *signature of knowledge for the transcript*. Note, this hashing of the transcript already happens in the standard PIOP compiler due to the application of the Fiat-Shamir transform; here, we highlight it explicitly as it plays an important role in the proof of simulation extractability. The second property, referred to as the *commitment simulator for PIOP* (see Definition 23), intuitively requires the existence of a strategy to simulate commitments such that: adding them to the view preserves zero-knowledge, and the simulation respects the "commitment check" constraint in Item 2 of Definition 16. This is a very mild property that is trivially satisfied when employing hiding commitments, and is met by existing simulation strategies based on deterministic commitments to randomized polynomials [27,17]. The third property of CP is that it must be *simulation-extractable* w.r.t. a policy $\hat{\mathbf{\Phi}}$ such that:

[18] The reason to assume a single CP-SNARK for both kind of statements, instead of one for polynomial equations and one for openings, has to do with the security guarantees when we compose protocols in the AGM [1].

– The adversary can ask simulated proofs for "PIOP verifier" statements where all the $v_j^{(k)}$ of Eq. (11) are fixed at the beginning of the experiment.
– If the forgery of the adversary is a "proof of knowledge" for commitments $\mathbf{c}^*$, then the adversary must return as auxiliary output yet another forgery for a "PIOP verifier" statement such that: (1) All the commitments $\mathbf{c}^*$ appear in the second forgery, (2) the second forgery is valid according to the extractor policy described next.
– If the forgery of the adversary is for the "PIOP verifier" statement, then the statement-proof pair returned by the adversary must not be in the list of simulated statements-proofs, and (similarly to Definition 22) there exists a k such that for all j the polynomial $v_j^{(k)}$ has degree at least 1.

Let $\mathcal{R}_{\mathtt{poly}}$ be the relation that upon relation parameters the commitment key ck and an instance $\mathbb{x}_{\mathtt{poly}} \coloneqq \big(\mathtt{msg}, \mathbf{c}, (\bar{G}^{(k)}, \boldsymbol{v}^{(k)})_{k\in[n_e]}\big)$, where $\mathtt{msg}$ is an arbitrary string and $\mathbf{c} \coloneqq (\mathsf{c}_j)_{j\in[n]}$, and whose witness is $\mathbb{w}_{\mathtt{poly}} \coloneqq (\boldsymbol{p}, \boldsymbol{o})$, outputs 1 if and only if:

$$\begin{aligned}&\forall k \in [n_e] : \ \bar{G}^{(k)}(X, p_1(v_1^{(k)}(X)), \ldots, p_n(v_n^{(k)}(X))) \equiv 0 \quad \wedge \\ &\forall j : \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p_j, o_j) = 1\end{aligned}$$

Notice that a simulation-extractable CP-SNARK for the relation $\mathcal{R}_{\mathtt{poly}}$ forms a *signature of knowledge* [19] for an instance $\big(\mathbf{c}, (\bar{G}^{(k)}, \boldsymbol{v}^{(k)})_{k\in[n_e]}\big)$ and the message $\mathtt{msg}$. Consider the relation for multi-instance opening of commitments defined below:

$$\mathcal{R}_{\mathtt{opn}}(\mathsf{ck}, \mathbb{x}_{\mathtt{opn}} = \mathbf{c}, \mathbb{w}_{\mathtt{opn}} = (\boldsymbol{p}, \boldsymbol{o})) \coloneqq (\forall j \in [|\mathbf{c}|] : \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p_j, o_j)) \, .$$

Our compiler needs a CP-SNARK that can simultaneously prove the polynomial evaluation relation and the knowledge of the openings. Let $\hat{\mathcal{R}}$ be the joint relation of $\mathcal{R}_{\mathtt{poly}}$ and $\mathcal{R}_{\mathtt{opn}}$, i.e.:

$$\hat{\mathcal{R}} \stackrel{\text{def}}{=} \{\mathsf{ck}, (\mathtt{poly}, \mathbb{x}), \mathbb{w} : \mathcal{R}_{\mathtt{poly}}(\mathsf{ck}, \mathbb{x}, \mathbb{w})\} \cup \{\mathsf{ck}, (\mathtt{opn}, \mathbf{c}), \mathbb{w} : \mathcal{R}_{\mathtt{opn}}(\mathsf{ck}, \mathbf{c}, \mathbb{w})\} \quad (12)$$

We say that a statement of the form $\mathbb{x} \coloneqq (\mathtt{poly}, \mathbb{x}_{\mathtt{poly}})$ (resp. $\mathbb{x} \coloneqq (\mathtt{opn}, \mathbb{x}_{\mathtt{opn}})$) is a $\mathtt{poly}$-instance (resp. $\mathtt{opn}$-instance).

Notice that if we have a CP-SNARK for $\mathcal{R}_{\mathtt{poly}}$ and a CP-SNARK for $\mathcal{R}_{\mathtt{opn}}$ we can easily define a single proof system that proves the relation $\hat{\mathcal{R}}$. In fact, the relation $\hat{\mathcal{R}}$ could be seen as a join of the two relations plus some syntactic sugar. The reason to use a single CP-SNARK for $\hat{\mathcal{R}}$ (instead of one for each relation) is rather technical and it has to do with the security guarantees when we compose protocols in the AGM [1]. In fact, we need a CP-SNARK for $\mathcal{R}_{\mathtt{poly}}$ that is simulation-extractable in the AGM even in presence of the simulated proofs for a CP-SNARK for $\mathcal{R}_{\mathtt{opn}}$. In particular, the simulated proofs for the latter CP-SNARK could contain group elements that might interfere with the security proved for the former CP-SNARK (and vice versa). In other words, even if the first CP-SNARK is simulation-extractable (in the AGM), it could potentially be

insecure when we use it in combination with the second CP-SNARK (if we are using the same group to instantiate the proof of opening).

Let now introduce the following set of policies $\hat{\mathbf{\Phi}}$.

Definition 24. *A policy $\hat{\Phi} := (\hat{\Phi}_0, \hat{\Phi}_1) \in \hat{\mathbf{\Phi}}$. $\Phi_0(\mathsf{pp}_{\mathbb{G}})$ outputs parameters pp_Φ that contain a set of vectors of polynomials $\mathcal{Q}_v = \{\boldsymbol{v}^{(i)} = (v_1^{(i)}, \ldots, v_n^{(i)})\}_{i \in \mathsf{poly}(\lambda)}$ and a list $\mathsf{coms} := (\mathbf{c}^{(i)})_{i \in [q]}$ sampled from a distribution $\mathcal{D}$ where the $\mathcal{D}$-Aff-MDH assumption holds, while $\hat{\Phi}_1$ is defined as follows.*

- **Semi-adaptive w.r.t. `poly`-queries.** *Given the set of simulation queries $\mathcal{Q}_{\mathsf{sim}}$, define the projections of the set to the `poly`-simulation queries and `opn`-simulation queries. Let the i-th `poly`-query to the simulation oracle be $(\mathbb{x}, \mathsf{aux}, \pi) \in \mathcal{Q}_{\mathsf{sim}}$, and parse $\mathbb{x}$ as $(\mathtt{msg}, \mathbf{c}, (G^{(k)}, \boldsymbol{v}^{(k)})_{k \in [n_e]})$.*
  1. *$\forall k \in [n_e] : \boldsymbol{v}^{(k)} \in \mathcal{Q}_v$*
  2. *parse aux as two matrices $\boldsymbol{C}, \boldsymbol{F}$; the tuple $(\mathbf{c}, \boldsymbol{C}, \boldsymbol{F})$ satisfies the commitment check, namely $\mathbf{c} = \boldsymbol{C} \cdot \mathbf{c}^{(i)} + \boldsymbol{F} \cdot \mathsf{ck}$*
  3. *the view defined by set of all `poly`-queries satisfies the algebraic consistency for CP (cf. Definition 15).*
- **Extractor policy for `opn`-forgery.** *If the forgery of the adversary $(\mathbb{x}^*, \pi^*)$ is of the form $\mathbb{x}^* := (\mathtt{opn}, \mathbf{c}^*)$, parse aux_Φ as (yet another) forgery $\tilde{\mathbb{x}} := (\mathtt{poly}, \mathtt{msg}, \mathbf{c}, (G^{(k)}, \boldsymbol{v}^{(k)})_{i \in [n_e]}), \tilde{\pi}$ and check that:*
  1. *All the commitments $\mathbf{c}^*$ are in the vector of commitments $\mathbf{c}$,*
  2. *The proof $\tilde{\pi}$ is valid according to the extractor policy for `poly`-forgeries below.*
- **Extractor policy for `poly`-forgery.** *If the forgery of the adversary $(\mathbb{x}^*, \pi^*)$ is of the form $\mathbb{x}^* := (\mathtt{poly}, \mathbb{x}')$, then check that $(\mathbb{x}', \pi^*) \notin \mathcal{Q}_{\mathsf{sim}}$, i.e., π^* is "fresh" and not produced by the simulation oracle on input $\mathbb{x}'$. Moreover, there exists $k \in [n_e]$ such that for all j the polynomial $v_j^{(k)}$ has degree at least 1 (and degree $\mathsf{poly}(\lambda)$).*

Intuitively, the extractor policy for `opn`-forgery means that a proof of opening for a commitment `c` can be extracted (given some auxiliary information) only when the adversary can exhibit a proof of evaluation that involves such a commitment.

Definition 25 (Compilation-Ready CP-SNARK). *A Compilation-Ready CP-SNARK is a $\hat{\mathbf{\Phi}}$-simulation-extractable CP-SNARK for $\hat{\mathcal{R}}$.*

In Section 5.4 we describe a simple and unoptimized CP-SNARK for $\hat{\mathcal{R}}$ for KZG, whose security analysis is given in the AGM. The CP-SNARK uses classical random-point evaluation for the polynomial equations and a *vacuous proof of opening* that can be extracted given the algebraic representation. The extraction of the latter is successful, namely the representation depends only on the element in the commitment key, only if the adversary exhibits a proof of polynomial equation. Both the proof of polynomial equation and of opening are extracted through the algebraic representations. Thus, unless the binding property is broken, when the proof of evaluation holds the proof of opening can be extracted correctly.

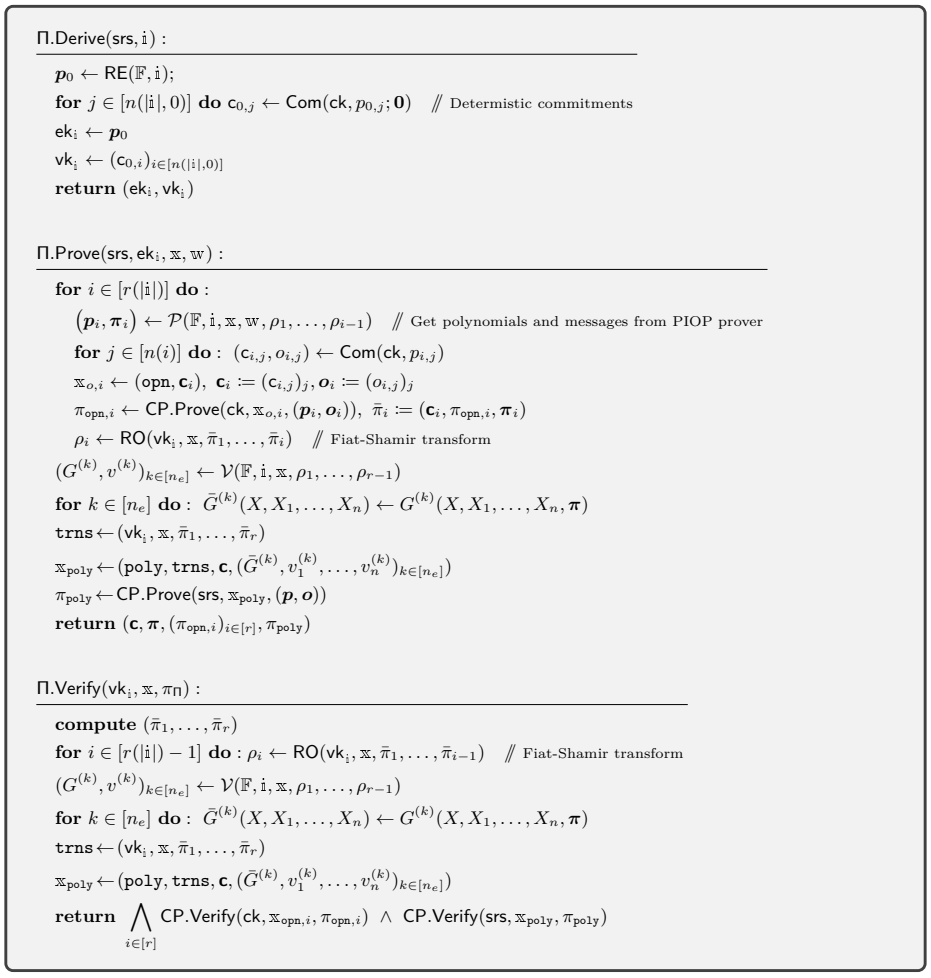

Fig. 2. The Compiler to Universal zkSNARKs.

5.3 The Universal zkSNARK

Let $\Phi_{\mathtt{SE}}$ be the standard (strong) simulation extractability policy. Namely, the policy checks that the forgery of the adversary is a tuple $(\mathbb{x}_\Pi, \pi_\Pi) \notin \mathcal{Q}_{\mathsf{sim}}$.

Theorem 3. *Let* CP *be a compilation-ready CP-SNARK (cf. Definition 25). Let* PIOP *be a PIOP for an indexed relation* $\mathcal{R}$ *that is state-restoration straight-line extractable (cf. Definition 20), and bounded special honest-verifier zero-knowledge, where the technical condition of Definition 22 holds and equipped with a commitment simulator for* CP *(cf. Definition 23). Let* Π *be the zkSNARK compiled from* PIOP *using the compiler from Fig. 2. Then* Π *is* $\Phi_{\mathtt{SE}}$*-simulation-extractable.*

We follow the classical compilation strategy where: for each of the r rounds, the zkSNARK prover sends commitments of the PIOP oracle polynomials (along with a proof of knowledge) and then computes the PIOP verifier's challenges using Fiat-Shamir; in the last round, the prover sends a CP proof that the PIOP verifier accepts, i.e., Eq. (11) holds w.r.t. all the commitments sent earlier. Notably, this CP proof is produced using the statement and the hash of the transcript as "message" for the signature of knowledge.

We briefly discuss how the properties of PIOP and CP play a role in the security of the compiled zkSNARK Π. We recall that in the simulation-extractability experiment, we have an adversary $\mathcal{A}$ who makes simulation queries for statements of its choice and eventually comes up with a forgery, which is a statement-proof that is new and valid. The goal is to show that for such adversary there is an extractor that outputs a valid witness with overwhelming probability. Roughly speaking, we build this extractor by first extracting the committed oracle polynomials from the CP "proof of knowledge" in the random oracle query of $\mathcal{A}$ in each round,[19] and then by running the PIOP extractor to obtain the witness.

For this extraction strategy to work, we need two conditions: (A) The "proof of knowledge" extraction must be valid. (B) The zkSNARK extractor feeds the PIOP extractor with polynomials that pass the PIOP verification equations. A technicality about relying on CP extraction for (A) and (B) is that we actually have to make a reduction to the its *policy-based simulation-extractability*. In particular, this means that we have to turn $\mathcal{A}$ into CP adversaries that comply with the policy $\hat{\mathbf{\Phi}}$.

To obtain (A), we use the second property of $\hat{\mathbf{\Phi}}$ mentioned above, which ensures a valid extraction if the adversary later provides a valid proof of polynomial evaluation. This is however the case for us, since a successful adversary must provide such proof.

For (B), we rely on the following observations. If $\mathcal{A}$ produces a forgery for a new statement of Π then the CP proof (aka signature of knowledge) must use a new message, and thus we can build a CP adversary returning a new statement-proof pair. If $\mathcal{A}$ produces a forgery for a statement queried to the simulation oracle, then by strong simulation extractability the proof must be new, which means that: either the commitments in the transcript are different, or the commitments are all the same but the "PIOP verifier" proof is different. In the former case, we get a different transcript, which yields a CP forgery with a new message, as in the previous case. In the latter case, the transcript is the same and we get a CP forgery with the same message but fresh proof. Notably, all the cases, the CP forgeries respect the degree-1 condition thanks to the compiler-safe property of the PIOP. Finally, the reduction CP adversaries that we build satisfy the first property of $\hat{\mathbf{\Phi}}$ thanks to the algebraic verifier property of PIOP, which allows us to precompute the instance-independent polynomials $\tilde{v}_j^{(k)}$, and to the programmability of the random oracle that allows us to presample the verifier's

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

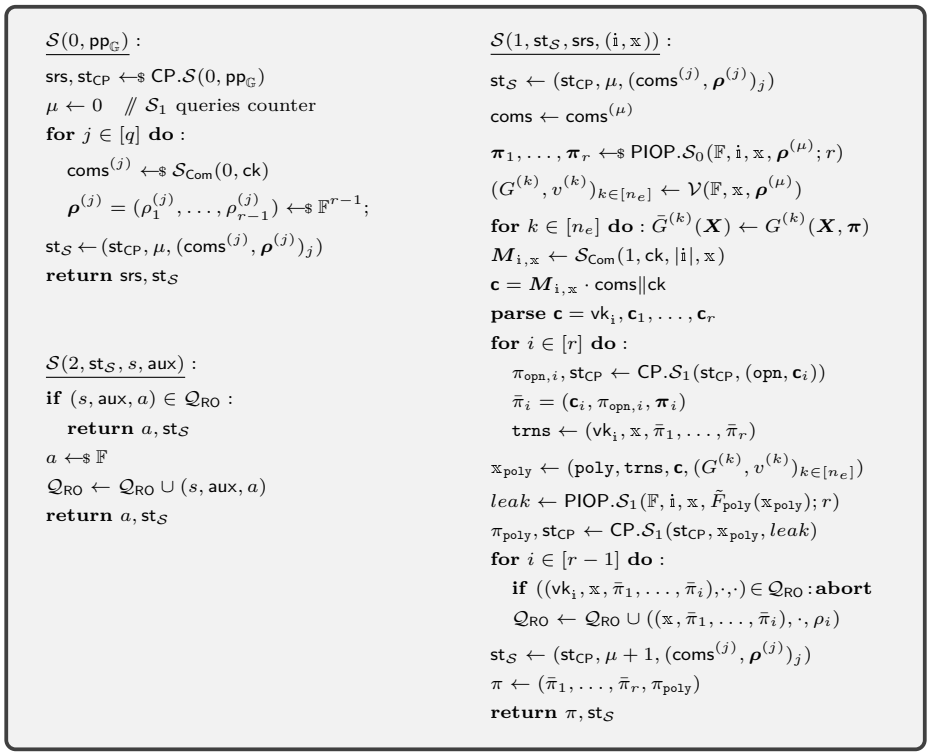

Fig. 3. The simulator $\mathcal{S}$ for Π.

challenges $\boldsymbol{\rho}$, define $v_j^{(k)}(X) = \tilde{v}_j^{(k)}(X, \boldsymbol{\rho})$, and later program the random oracle to use these coins $\boldsymbol{\rho}$.

Proof. We prove the theorem assuming the commitments are not hiding. The adaption to hiding commitments is straightforward. We start showing the zero-knowledge simulator for Π. The simulator is in Fig. 3. For the description of the simulator, we assume we can couple the leakage function $F_{\mathtt{poly}}$, which defines the $F_{\mathtt{poly}}$-leaky zero-knowledge of the CP-SNARK, with a function $\tilde{F}_{\mathtt{poly}}$ that upon input the instance outputs the set $\mathcal{L}_{\mathbb{x}}$ of evaluation points necessary to compute a simulated proof. The zero-knowledge guarantees of the simulator come from the zero-knowledge property of the PIOP, the zero-knowledge property of the CP and the simulator commitment property of $\mathcal{S}_{\mathsf{Com}}$ (see Definition 23). Notice that $\mathcal{S}_1$ might abort if $(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i)$ for some i was already queried to the random oracle. However, by simply assuming that the first message of $\mathcal{P}$ has $\omega(\log |\mathbb{F}|)$ min-entropy then the event that the simulator aborts happens with negligible probability.

We define the extractor for Π for a given adversary $\mathcal{A}_{\Pi}$. We make some simplifying assumptions on the behavior of $\mathcal{A}_{\Pi}$: (1) the adversary always queries first the RO on a string that can be parsed as $(\mathbb{i}, \mathbb{x})$ before querying the simulation

oracle on the same string, (2) the auxiliary string $\mathsf{aux}_{\mathcal{E}}$ output by $\mathcal{A}_{\Pi}$ can be parsed as a list of strings $(s_i, \mathsf{aux}_i, \mathsf{st}_i)_i$ and a string $\mathsf{aux}'_{\mathcal{E}}$ where for any i we have $(s_i, \mathsf{aux}_i, \mathsf{st}_i)$ string is identical to the auxiliary input output at the i-th query of the adversary and (3) for any i the auxiliary input of the i-th random oracle query of the adversary contains the full internal state of the adversary. These assumptions are w.l.g. In fact, given an adversary $\mathcal{A}_{\Pi}$ that does not respect these rules we can always define another adversary that runs internally $\mathcal{A}_{\Pi}$, collects all the necessary information to comply with (2) and (3), and moreover follows the rule (1)

Let $\mathcal{B}_i$ be a reduction to the simulation extractability of CP that runs $\mathcal{A}_{\Pi}$ (simulating the oracles for $\mathcal{A}_{\Pi}$ using its own oracles and the code defined in Fig. 3) and sets its output to be the i-th random oracle query of $\mathcal{A}_{\Pi}$. For any $i \in [q]$ let $\mathcal{E}_i$ be the extractor associated to $\mathcal{B}_i$ (thanks to the $\hat{\Phi}$-simulation extractability of CP we can associate to $\mathcal{B}_i$ an extractor, in Lemma 11 we describe $\mathcal{B}_i$ with more details and we show that it complies with the $\hat{\Phi}$ policy). Similarly, let $\mathcal{E}_{\mathtt{poly}}$ be the extractor for the adversary $\mathcal{B}_{\mathtt{poly}}$ that runs $\mathcal{A}_{\Pi}$ (simulating the oracles of $\mathcal{A}_{\Pi}$ using its own oracles and the code defined in Fig. 3) until completion, and isolates the instance $\mathbb{x}_{\mathtt{poly}} := (\mathtt{poly}, \mathtt{trns}, \mathbf{c}, (G^{(k)}, \boldsymbol{v}^{(k)})_{k\in[n_e]})$ and the proof $\pi_{\mathtt{poly}}$ in the forgery of $\mathcal{A}_{\Pi}$, and outputs all the auxiliary outputs that $\mathcal{A}_{\Pi}$ does.

We first set some notation:

- Let $\mathcal{P}_i$ be the indexes of the polynomials sent at the i-th round by the PIOP.
- Given a proof π for Π we define *the RO-queries set of* π be the set of string $\{(\mathsf{vk}_{\mathbb{i}}, \mathbb{x}), \ldots, (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_r)\}$.
- We say that a proof π *shares a simulated commitment* c of proof π' simulated by $\mathcal{S}$ if in $\mathsf{st}_{\mathcal{S}}$ one can find $\boldsymbol{m}_1 \neq \mathbf{0}$ and $\boldsymbol{m}_2$ such that $\mathsf{c} = \boldsymbol{m}_1 \| \boldsymbol{m}_2 \cdot \mathsf{coms} \| \mathsf{ck}$.

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

1. Parse $\mathsf{aux}_{\mathcal{E}}$ as the concatenation of a list $(s_i, \mathsf{aux}_i, \mathsf{st}_i)_i$ and $\mathsf{aux}'_{\mathcal{E}}$, where $(s_i, \mathsf{aux}_i, \mathsf{st}_i)$ is the output of $\mathcal{A}_{\Pi}$ at the i-th query to the ROM and $\mathsf{aux}'_{\mathcal{E}}$ the remaining auxiliary information given by the adversary (namely, the auxiliary information associated with its last output).
2. From π_{Π} derive the messages $\bar{\pi}_1, \ldots, \bar{\pi}_r$ and find the indexes q_i such that $s_{q_i} = (\mathsf{vk}_{\mathbb{i}}, \mathbb{x}, \bar{\pi}_1, \ldots, \bar{\pi}_i)$.
3. If π_{Π} shares a simulated commitment with one of the simulated proofs then return $\perp$.
4. For $i \in [r]$ run $\mathbb{w}_{i,\mathtt{opn}} \leftarrow \mathcal{E}_{q_i}(\mathsf{srs}, \mathbf{c}_i, \pi_{\mathtt{opn},i}, \mathsf{aux}_i)$ where $\bar{\pi}_i$ contains both the instance and the proof of opening and $\mathbb{w}_{i,\mathtt{opn}} = (p'_j)_{j\in\mathcal{P}_i}$.
5. Run $(p_j)_{j\in[n]} \leftarrow \mathcal{E}_{\mathtt{poly}}(\mathsf{srs}, \mathbb{x}_{\mathtt{poly}}, \pi_{\mathtt{poly}}, \mathsf{aux}')$.
6. If $\exists i : p'_i \neq p_i$ then return $\perp$.
7. If $\exists i \in [n]$ and $j \in \mathcal{P}_i : \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p'_j) \neq 1$ then return $\perp$.
8. If $\exists k : G^{(k)}(X, p_1(v_1^{(k)}(X)), \ldots, p_n(v_n^{(k)}(X)), \pi_1, \ldots, \pi_r) \not\equiv 0$ then return $\perp$.
9. Return $\mathcal{E}_{\mathsf{PIOP}}(\mathbb{i}, \mathbb{x}_{\Pi}, (p_j)_{j\in[n]})$.

To analyze the success of the extractor we define a series of hybrid games. We start from the first hybrid that is the $\mathbf{Exp}^{sr}_{\mathcal{A}_{\mathsf{PIOP}},\mathsf{PIOP}}(\mathbb{F})$ experiment for PIOP (see Definition 20) for an adversary $\mathcal{A}_{\mathsf{PIOP}}$ that we define next.

$\mathcal{A}_{\mathsf{PIOP}}$:

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

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

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

Consider $\mathbf{H}_1$ that additionally extracts $(p_j)_{j\in[n]} \leftarrow \mathcal{E}_{\mathtt{poly}}(\mathsf{srs}, \mathtt{x}_{\mathtt{poly}}, \pi_{\mathtt{poly}}, \mathsf{aux}')$ and returns 1 if $\exists k : G^{(k)}(X, p_1(v_1^{(k)}(X)), \ldots, p_n(v_n^{(k)}(X)), \pi_1, \ldots, \pi_r) \not\equiv 0$ or $\exists j : \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p_j) = 0$.

Lemma 10. $\Pr[\boldsymbol{H}_1] \leq \Pr[\boldsymbol{H}_0] + \epsilon_{\mathsf{CP}}$

Proof. We reduce to $\hat{\Phi}$-simulation extractability of CP. We define with more details the adversary $\mathcal{B}_{\mathtt{poly}}$ that runs $\mathcal{A}_{\Pi}$, and we define a policy $\hat{\Phi}_0$ that samples parameters for $\mathcal{B}_{\mathtt{poly}}$.

Policy $\hat{\Phi}_0(\mathsf{pp}_{\mathbb{G}})$

1. Run $\mathcal{S}(0, \mathsf{pp}_{\mathbb{G}})$
2. Parse $\mathsf{st}_{\mathcal{S}} = (\mathsf{st}_{\mathsf{CP}}, 0, (\mathsf{coms}^{(j)})_j, (\boldsymbol{\rho}^{(j)})_j)$.
3. Let $\mathcal{Q}_v$ be an empty set. Let D be the maximum degree of the polynomials in srs. For $d \in [D]$:
   (a) let $(\tilde{v}_j^{(k)})_{j,k} \leftarrow \tilde{\mathcal{V}}(\mathbb{F}, d)$ be the polynomials defined by the (non-adaptive) algebraic verifier of PIOP (see Definition 19).

(b) add to $\mathcal{Q}_v$ the polynomials $\tilde{v}_j^{(k)}(X, \boldsymbol{\rho}^i) : i \in [q], j \in [n], k \in [n_e]$

Next, we define the reduction $\mathcal{B}_{\mathtt{poly}}$.

Reduction $\mathcal{B}_{\mathtt{poly}}(\mathsf{srs}, \mathsf{pp}_{\hat{\Phi}})$

1. Run $\mathcal{A}_{\Pi}(\mathsf{srs})$.
2. Upon query $((\mathtt{i}, \mathbb{x}), \mathsf{aux})$ to the simulation oracle, run the same strategy of $\mathcal{S}_1$ defined in Fig. 3 where the calls to $\mathsf{CP}.\mathcal{S}_1$ are forwarded to the simulation oracle of $\mathcal{B}_{\mathtt{poly}}$.
3. Given the forgery $((\mathtt{i}, \mathbb{x}_{\Pi}), \pi_{\Pi})$ output by $\mathcal{A}_{\Pi}$, define the instance $\mathbb{x}_{\mathtt{poly}} = (\mathtt{poly}, \mathtt{trns}, \mathbf{c}, (G^{(k)}, \boldsymbol{v}^{(k)})_{k \in [n_e]})$ and the proof $\pi_{\mathtt{poly}}$.
4. Return the forgery $(\mathbb{x}_{\mathtt{poly}}, \pi_{\mathtt{poly}})$ and set the auxiliary input $\mathsf{aux}_{\mathcal{E}}$ as the adversary $\mathcal{A}_{\Pi}$ does.

By inspection, if the forgery of $\mathcal{A}_{\Pi}$ passes the verification then forgery of $\mathcal{B}_{\mathtt{poly}}$ passes the verification too. Moreover, by the (strong) simulation extractability game of $\mathcal{A}_{\Pi}$ we have that $((\mathtt{i}, \mathbb{x}_{\Pi}), \pi_{\Pi})$ is not in the set of simulation proofs $\mathcal{Q}_{\mathsf{sim},\Pi}$ of $\mathcal{A}_{\Pi}$, thus the pair $(\mathbb{x}_{\mathtt{poly}}, \pi_{\mathtt{poly}})$ is not in the simulation proofs $\mathcal{Q}_{\mathsf{sim},\mathtt{poly}}$ of $\mathcal{B}_{\mathtt{poly}}$. In particular, there are three cases:

1. If $\mathcal{A}$ never queried $(\mathtt{i}, \mathbb{x}_{\Pi})$ to its simulation oracle, i.e. $\mathbb{x}_{\Pi}$ is a "fresh" instance for Π, then $\mathcal{B}_{\mathtt{poly}}$ never queried the simulator with $\mathbb{x}_{\mathtt{poly}}$ (notice that $\mathbb{x}_{\mathtt{poly}}$ contains $\mathbb{x}_{\Pi}$ in the message $\mathtt{trns}$).
2. Otherwise, $\mathcal{A}$ queried $(\mathtt{i}, \mathbb{x}_{\Pi})$ to its simulation oracle:
   (a) For any simulated proof π'_{Π} for $(\mathtt{i}, \mathbb{x}_{\Pi})$ the transcript of π_{Π} is different from the simulated one. This implies that $\mathbb{x}_{\mathtt{poly}} \neq \mathbb{x}'_{\mathtt{poly}}$ (for the same reason of above).
   (b) There exists a simulated proof π'_{Π} for $(\mathtt{i}, \mathbb{x}_{\Pi})$ such that the transcripts of the forgery and of the simulated proof are equal. Since the forgery of $\mathcal{A}_{\Pi}$ is not in $\mathcal{Q}_{\mathsf{sim},\Pi}$ then the proofs $\pi_{\mathtt{poly}} \neq \pi'_{\mathtt{poly}}$.

By the property of the PIOP we have that there exists an index k such that for all j we have $v_j^{(k)}$ is not constant polynomial, thus the forgery of $\mathcal{B}_{\mathtt{poly}}$ meets the extractor policy.

Finally, by the definition of the simulator $\mathcal{S}$ in Fig. 3, the query to the simulation oracle of $\mathcal{B}_{\mathtt{poly}}$ respects the simulator policy of $\hat{\Phi}_1$. In fact, the simulator policy of $\hat{\Phi}$ matches the constraint in Definition 23. Thus the reduction $\mathcal{B}_{\mathtt{poly}}$ follows the policy $\hat{\Phi}$. On the other hand, the distinguish event implies that the extractor fails, thus this would break the $\hat{\Phi}$-simulation extractability of CP. □

Let $\mathbf{H}_2$ additionally return 1 if $\exists i, j : j \in \mathcal{P}_i \wedge \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p'_j) \neq 1$, where q_i is the index of the random oracle query such that $(s_{q_i}, \mathsf{aux}_{q_i}, \rho_i) \in \mathcal{Q}_{\mathsf{RO}}$, and $\mathtt{w}_{q_i} = (p'_j)_{j \in \mathcal{P}_i}$ are the polynomials extracted by $\mathcal{E}_{q_i}$.

Lemma 11. $\Pr[\boldsymbol{H}_2] \leq \Pr[\boldsymbol{H}_1] + r \cdot \epsilon_{\mathsf{CP}}$

Proof. We prove through a series of r hybrids. Let $\mathbf{H}_{1,0}$ be the same as $\mathbf{H}_1$; moreover, let $\mathbf{H}_{1,i}$ be the same as $\mathbf{H}_{1,i-1}$ but that additionally returns 1 if $\exists j \in \mathcal{P}_i : \mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p'_j) \neq 1$, where q_i is the index of the random oracle query such that $(s_{q_i}, \mathsf{aux}_{q_i}, \rho_i) \in \mathcal{Q}_{\mathsf{RO}}$, and $\mathbb{w}_{q_i} = (p'_j)_{j \in \mathcal{P}_i}$ are the polynomials extracted by $\mathcal{E}_{q_i}$. Clearly, $\mathbf{H}_{1,r} \equiv \mathbf{H}_2$ (where r is the number of rounds of the PIOP).

We reduce again to $\hat{\Phi}$-simulation extractability of CP. We define the adversary $\mathcal{B}$ that runs $\mathcal{A}_\Pi$. The adversary $\mathcal{B}$ corresponds to the reduction $\mathcal{B}_{q_i}$ mentioned before. Additionally, we need to define a policy $\hat{\Phi}_0$ that samples the parameters for $\mathcal{B}$, we set the policy identical to the one used in Lemma 10.

Reduction $\mathcal{B}(\mathsf{srs}, \mathsf{pp}_{\hat{\Phi}})$

1. Run $\mathcal{A}_\Pi(\mathsf{srs})$.
2. Upon query $((\mathbb{i}, \mathbb{x}), \mathsf{aux})$ to simulation oracle, run the same strategy of $\mathcal{S}_1$ defined in Fig. 3 with the only difference that instead of calling $\mathsf{CP}.\mathcal{S}_1$ it makes a query to its simulator.
3. Parse the q_i-th query $(s_{q_i}, \mathsf{aux}_{q_i})$ to the RO of $\mathcal{A}_\Pi$ as a partial transcript where $\bar{\pi}_i := (\mathbf{c}_i, \pi_{\mathtt{opn},i}, \boldsymbol{\pi}_i)$.
4. Continue running the adversary, parse the forgery output by $\mathcal{A}_\Pi$ as $(\bar{\pi}'_1, \ldots, \bar{\pi}'_r, \pi_{\mathtt{poly}})$. Set $\tilde{\mathbb{x}} := (\mathtt{poly}, \mathtt{trns}, \mathbf{c}, (G^{(k)}, \boldsymbol{v}^{(k)})_{k \in [n_\epsilon]})$ as the verifier would do.
5. Return the forgery $((\mathtt{opn}, \mathbf{c}_i), \pi_{\mathtt{opn},i})$, using as auxiliary information $\mathsf{aux}_{\mathcal{E}} \leftarrow \mathsf{aux}_{q_i}$ and $\mathsf{aux}_{\Phi} \leftarrow (\tilde{\mathbb{x}}, \pi_{\mathtt{poly}})$.

By inspection, if the forgery of $\mathcal{A}_\Pi$ passes the verification (both for $\mathtt{poly}$-instance in the final proof and for the $\mathtt{opn}$-instance in the q_i-th query) then $\mathcal{B}$'s forgery passes the verification too. Moreover, by the (strong) simulation extractability game of $\mathcal{A}_\Pi$ we have that $((\mathbb{i}, \mathbb{x}_\Pi), \pi_\Pi)$ is not in the set of simulations $\mathcal{Q}_{\mathsf{sim}}$ of $\mathcal{A}_\Pi$, thus the pair $(\tilde{\mathbb{x}}, \pi_{\mathtt{poly}})$ is not in the simulations of $\mathcal{B}$ (cf. Lemma 10). By the property of the PIOP, there exists an index k such that for all j we have $v_j^{(k)}$ is not constant polynomial, thus the forgery of $\mathcal{B}$ meets the extractor policy.

Finally, by the definition of the simulator $\mathcal{S}$ in Fig. 3, the query to the simulation oracle of $\mathcal{B}$ respects the simulator policy of $\hat{\Phi}_1$. In fact, the simulator policy of $\hat{\Phi}$ matches with the constraint in Definition 23. Notice we are running the experiment for $\mathcal{B}$ with the same extractor of $\mathcal{B}_{q_i}$. However, the first three outputs of $\mathcal{A}$ and $\mathcal{B}$ are distributed equivalently ($\mathcal{B}$ additionally outputs aux_Φ), thus the same extractor $\mathcal{E}_{q_i}$ works either for $\mathcal{B}_{q_i}$ or for $\mathcal{B}$. □

Let $\mathbf{H}_3$ additionally return 1 if $\exists i : p'_i \neq p_i$ where $(p'_j)_{j \in \mathcal{P}_i}$ are the polynomials extracted by $\mathcal{E}_{q_i}$ and $(p_j)_{j \in [n]}$ are the polynomials extracted by $\mathcal{E}_{\mathtt{poly}}$.

Lemma 12. $\Pr[\boldsymbol{H}_3] \leq \Pr[\boldsymbol{H}_2] + \epsilon_{\mathsf{CP}}$.

Proof. The distinguishing event implies that we can define an adversary $\mathcal{B}$ that runs the $\hat{\Phi}$-simulation extractability experiment for CP and finds a commitment c and two distinct polynomials p and p' such that $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, p) =$

$\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}, p') = 1$[20].Let D be the maximum degree allowed by the commitment key, let $x \in \mathbb{F}$ such that $p(x) \neq p'(x)$ and consider the following (false) statement for $\mathcal{R}_{\texttt{poly}}$ (and in turn of $\hat{\mathcal{R}}$):

- For $j \in [D+1]$ set $G^{(j)}(X, X_1, X_j) = (X_1 - p(j)) - (X - j)X_j$. Namely, the polynomial associated with formal variable X_1 evaluates on $p(j)$ at point j, and X_j is the *witness*.
- Set $G^{(D+2)}(X, X_1, X_{D+2}) = (X_1 - p'(x)) - (X - x)X_{D+2}$. Namely, the polynomial associated to formal variable X_1 evaluates on $p'(x)$ at point x, and X_{D+2} is its *witness*.

We can create a valid proof for such a statement using the prover of CP: the first $d+1$ equations are proved using p and the polynomials $p_j(X) = (p(X) - p(j))/(X - j)$, while to prove the last equation we use the witness p'. Here, we use the hypothesis that CP uses internally a CP-SNARK for $\mathcal{R}_{\texttt{m-evl}}$. The proof is for a statement that is not in the language, thus we break the simulation extractability of CP.

Let $\mathbf{H}_4$ additionally return 1 if the forged proof shares a simulated commitment with a simulated proof in $\mathcal{Q}_{\mathsf{sim}}$.

Lemma 13. $\Pr[\boldsymbol{H}_4] \leq \Pr[\boldsymbol{H}_3] + \epsilon_{\mathsf{owf}}$

Proof. We need to bound the probability that $\mathcal{A}_{\Pi}$ returns a forgery that shares a simulated commitment with a proof in $\mathcal{Q}_{\mathsf{sim}}$.

Let $\mathcal{B}$ be the same reduction described in Lemma 10: as shown before, if $\mathcal{A}_{\Pi}$ satisfies the simulation-extractability policy then the reduction $\mathcal{B}$ satisfies the policy $\hat{\Phi}$. We notice that the extractor of $\mathcal{B}$ extracts a witness for all the commitments in $\mathbf{c}$ in the forgery. Let assume that the forgery shares a commitment with the i-th simulated proof for some $i \in [q]$ and $\mathsf{coms} \leftarrow \boldsymbol{M}_{\mathbb{i},\mathbb{x}} \cdot (\mathsf{coms}^{(i)} \| \mathsf{ck})$ (in case the commitments are not homomorphic we assume that $\boldsymbol{M}_{\mathbb{i},\mathbb{x}}$ is the identity matrix). Since $\mathsf{coms} \leftarrow \boldsymbol{M}_{\mathbb{i},\mathbb{x}} \cdot (\mathsf{coms}^{(i)} \| \mathsf{ck})$ there exists a row $\boldsymbol{m}$ of the matrix $\boldsymbol{M}_{\mathbb{i},\mathbb{x}}$ such that $\mathsf{c}_j = \boldsymbol{m} \cdot \mathsf{coms}^{(i)}$. Moreover, we have a valid opening for c_j so $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p)$. Notice that a commitment function defines a one-way function; moreover, the simulated commitment is sampled from a distribution that is computationally indistinguishable from a distribution that samples *random* commitments. Thus, such an extractor would break the one-way property of the commitment function. □

Lemma 14. $\Pr[\boldsymbol{H}_4] = \mathbf{Adv}^{\Phi_{\mathtt{SE}}\text{-}se}_{\mathcal{A}_{\Pi},\mathcal{S},\mathcal{E}_{\Pi}}(\lambda)$

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

[20] Intuitively, this breaks the binding property of the commitment scheme, however, the binding property does not assume that the adversary can see simulated proofs thus we cannot reduce to it. Here, instead, we show how forge a proof for an instance that is not in the language.

– π_{Π} cannot share a simulated commitment with one of the simulated proofs (see Item 3) because of the check introduced in Lemma 13. Thus all the RO queries of $\mathcal{A}_{\Pi}$ that constitute π_{Π} (namely the queries $q_1, \ldots, q_r$) were forwarded to the challenger in Item 3b (in particular, any of the query was already answerd by the programming of the RO by the simulator $\mathcal{S}_1$). This implies that the complete transcript sent by $\mathcal{A}_{\mathsf{PIOP}}$ is in the list $\mathsf{SeenStates}$. On the other hand, the extractor $\mathcal{E}_{\Pi}$ does not abort at Item 3.
– $\forall j \in [n]$, the polynomial p_j extracted by $\mathcal{E}_{q_i}$ is equal to p'_j extracted from $\mathcal{E}_{\mathtt{poly}}$ and $\mathsf{VerCom}(\mathsf{ck}, \mathsf{c}_j, p_j) = 1$; this is ensured by the checks introduced in Lemma 11 and Lemma 12. This implies that the extractor $\mathcal{E}_{\mathsf{PIOP}}$ in the $\mathbf{Exp}^{sr}_{\mathcal{A}_{\mathsf{PIOP}},\mathsf{PIOP}}$ in $\mathbf{H}_3$ is fed with the same polynomials extracted by $\mathcal{E}_{\Pi}$.
– Finally, the polynomial check on $\mathbb{x}_{\mathtt{poly}}$ must be satisfied by the extracted polynomials p_j because of the check introduced in Lemma 10. Thus, if the proof π_{PIOP} is valid, the decision bit in the state-restoration game is 1.

5.4 The Compilation-Ready CP-SNARK in the AGM

To connect together Section 4 and the results of this section, we show a simple compilation-ready CP-SNARK in the ROM based on batched KZG evaluation proofs (cf. Section 5.4). Upon $\mathtt{opn}$-instances $\mathbb{x} := (\mathtt{opn}, \mathbb{x})$, the prover simply outputs an empty string (the verifier is the obvious algorithm). Upon $\mathtt{poly}$-instances $\mathbb{x} := (\mathtt{poly}, \mathbb{x})$, where $\mathbb{x}' := (\mathtt{msg}, \mathbf{c}, (\bar{G}^{(k)}, v_1^{(k)}, \ldots, v_n^{(k)})_{k \in [n_e]})$:

– $\mathsf{Prove}(\mathsf{ck}, \mathbb{x} = (\mathtt{poly}, \mathbb{x}'), \mathbb{w} = (\boldsymbol{p}, \boldsymbol{o}))$:
  1. Compute $x^* \leftarrow \mathsf{RO}(\mathbb{x}')$.
  2. Compute for any j, k the value $x_j^{(k)} \leftarrow v_j^{(k)}(x^*)$.
  3. Let $\mathcal{X} := \{x_1^*, \ldots, x_m^*\}$ be the set of the points $x_j^{(k)}$ for all k and j computed at the previous step, and let $\mathcal{P}_i := \{j : v_j^{(k)}(x^*) = x_i^*\}$.
  4. For $i \in [m]$, compute $\pi_{\mathtt{m\text{-}evl},i} \leftarrow \mathcal{P}_{\mathtt{m\text{-}evl}}(\mathsf{ck}, \mathbb{x}_{\mathtt{m\text{-}evl}}, (p_j, o_j)_{j \in \mathcal{P}_i})$ for the instance $\mathbb{x}_{\mathtt{m\text{-}evl}} := (x_i^*, (\mathsf{c}_j, p_j(x_i^*))_{j \in \mathcal{P}_i})$.
  5. Output $\left(\pi_{\mathtt{m\text{-}evl},i}, (p_j(x_i^*))_{j \in \mathcal{P}_i}\right)_{i \in [m]}$.
– $\mathsf{Verify}(\mathsf{ck}, \mathbb{x} = (\mathtt{poly}, \mathbb{x}'), \pi)$
  1. Compute $x^* \leftarrow \mathsf{RO}(\mathbb{x}')$.
  2. Compute for any j, k the value $x_j^{(k)} \leftarrow v_j^{(k)}(x^*)$.
  3. Let $\mathcal{X} := \{x_1^*, \ldots, x_m^*\}$ be the set of the points $x_j^{(k)}$ computed at the previous step, and let $\mathcal{P}_i := \{j : v_j^{(k)}(x^*) = x_i^*\}$.
  4. Parse π as $\left(\pi_{\mathtt{m\text{-}evl},i}, (y_{i,j})_{j \in \mathcal{P}_i}\right)_{i \in [m]}$. For any j, k let $\bar{y}_j^{(k)}$ be the (claimed) evaluation of the polynomial committed in c_j on point $v_j^{(k)}(x^*) \in \mathcal{X}$, this value is equivalent to $y_{i,j}$ for the index i such that $v_j^{(k)}(x^*) = x_i^*$.
  5. Output 1 iff:
     (a) $\forall i \in [m] : \mathcal{V}_{\mathtt{m\text{-}evl}}(\mathsf{ck}, \mathbb{x}_{\mathtt{m\text{-}evl},i} = (x_i^*, (\mathsf{c}_j, y_{i,j})_{j \in \mathcal{P}_i}), \pi_{\mathtt{m\text{-}evl},i}) = 1$
     (b) $\forall k \in [n_e] : \bar{G}^{(k)}(x^*, \bar{y}_1^{(k)}, \ldots, \bar{y}_n^{(k)}) = 0$.

For a "PIOP verifier" statement, the prover RO-hashes the instance and obtains a random point ξ, evaluates the polynomials $v_j^{(k)}(\xi)$ for any j and outputs the evaluations $p_j(v_j^{(k)}(\xi))$ together with a batch evaluation proof for all of them. For a "proof of knowledge" statement, the prover does not output an explicit proof element (we call this a *vacuous proof*) , and we rely on the AGM to argue its extractability. The idea is that, for an algebraic adversary that produces an alleged commitment c and its algebraic representation, we can find a way to *open* c, under some circumstances. For example, consider the adversary that, during the simulation-extractability experiment, hashes (i.e., makes a random oracle query) the commitment c, and later includes c in a "PIOP verifier" instance. Then the algebraic representation of c returned at hashing time must coincide with the same polynomial extracted at forgery time, otherwise one can break the standard binding of the commitment. Crucially, this scenario fits exactly the second part of the policy $\hat{\mathbf{\Phi}}$.

As for the third part of the policy, we notice that an attack similar to the mix-and-match malleability attack mentioned in the introduction applies for our compilation-ready CP-SNARK. For example, the adversary could ask a simulation for an instance that tests two (fake) commitments on constant values defined by the $v_j^{(k)}$, and then it can produce a forgery which includes one of the commitments by copying part of the simulated proof. Intuitively, this is why we require that the $v_j^{(k)}$ have degree at least 1: when evaluated on a fresh random point ξ, a valid proof for $p_j(v_j^{(k)}(\xi))$ intuitively ensures that the prover knows p_j

Definition 26 (Compilation-Ready Leakage Function). *Let* $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ *be* $F_{\mathtt{m\text{-}evl}}$*-leaky zero-knowledge. We define the "Compilation-Ready Leakage Function"* F *as the function that on input* $\mathtt{opn}$*-instances leaks no information, while on input* $\mathtt{poly}$*-instances* $\mathbb{x} := (\mathtt{poly}, \mathbb{x}')$ *and witness* $\mathbb{w} := (\boldsymbol{p}, \boldsymbol{o})$ *does the following:*

1. *Leak* $\{(j, p_j(x_j^{(k)}))\}_{j,k}$
2. *For any* $i \in [m]$ *compute* $\mathbb{x}_{\mathtt{m\text{-}evl},i}$ *from* $\mathbb{x}'$ *and the leaked points (as the honest prover would do), leak points* $F_{\mathtt{m\text{-}evl}}(\mathbb{x}_{\mathtt{m\text{-}evl},i}, (p_j, o_j)_{i \in \mathcal{P}_i})$.

Theorem 4. *Let* CP *be the CP-SNARK presented above. If* $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ *is* $F_{\mathtt{m\text{-}evl}}$*-leaky zero-knowledge then* CP *is* F*-leaky zero-knowledge (see Definition 26).*

Proof. We restrict our attention to $\mathtt{poly}$-instances of the form $\mathbb{x} := (\mathtt{poly}, \mathbb{x}_{\mathtt{poly}})$, and with associated witness $\mathbb{w}_{\mathtt{poly}}$, since the $\mathtt{opn}$-instances can be trivially simulated.

We rely on the leaky zero knowledge simulator $\mathcal{S}' = (\mathcal{S}'_0, \mathcal{S}'_1, \mathcal{S}'_2)$ of the scheme $\mathsf{CP}_{\mathtt{m\text{-}evl}}$. In particular, we define the simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$, where $\mathcal{S}_0$ (resp. $\mathcal{S}_2$) invokes $\mathcal{S}'_0$ (resp. $\mathcal{S}'_2$) on input $\mathbb{x}_{\mathtt{poly}}$. Upon input the statement $\mathbb{x}$ and the leakage $(j, \tilde{y}_{i,j})_{i \in [m], j \in \mathcal{P}_i}, y'_1, \ldots, y'_m \leftarrow F_{\mathtt{poly}}(\mathbb{x}_{\mathtt{poly}}, \mathbb{w}_{\mathtt{poly}})$, $\mathcal{S}_1$ derives, for any i, the statement $\mathbb{x}_{\mathtt{m\text{-}evl},i}$ from $\mathbb{x}_{\mathtt{poly}}$ and $(j, \tilde{y}_{i,j})_{i \in [m]}$, then it runs $\mathcal{S}'_1$ on the derived statement and leakage y'_i. The indistinguishability of the simulated view from the real view can be proved with a hybrid argument where at each step we use the leaky zero-knowledge of $\mathsf{CP}_{\mathtt{m\text{-}evl}}$. □

Theorem 5. *Let $\epsilon_{\texttt{m-evl}}$ be the maximum winning probability of a PT adversary against the $\Phi_{\texttt{m-evl}}$-simulation extractability. Let $\mathcal{E}$ be the canonical extractor in the AGM, let $\mathcal{S}$ be the F-leaky ZK simulators for CP (see Definition 26). For every $\hat{\Phi} \in \hat{\mathbf{\Phi}}$ (see Definition 24), for every adversary $\mathcal{A}$ that makes at most Q_{sim} simulation queries:*

$$\mathbf{Adv}^{\hat{\Phi}\text{-}se}_{\mathsf{CP},\mathcal{S},\mathcal{E},\mathcal{A}}(\lambda) \leq (Q_{\mathsf{sim}}+1)\epsilon_{\texttt{m-evl}} + \epsilon_{\textit{Aff-MDH}} + \frac{\mathsf{poly}(d,\lambda)}{|\mathbb{F}|}$$

Proof. We consider the *canonical* extractor given by the AGM guarantees, i.e., the extractor that parses and outputs the polynomials in the instance as derived by the coefficients in the algebraic representations.

We start by proving that for any algebraic adversary $\mathcal{A}$ whose forgery satisfies the extractor policy $\texttt{opn}$-forgery of $\hat{\Phi}$, there exists an algebraic adversary $\mathcal{B}$ whose forgery satisfies the extractor policy $\texttt{poly}$-forgery of $\hat{\Phi}$.

Lemma 15. *For any algebraic adversary $\mathcal{A}$ there exists an algebraic adversary $\mathcal{B}$ such that:*

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

Proof. The redudction $\mathcal{B}$ forwards all the simulation queries of $\mathcal{A}$. When $\mathcal{A}$ outputs as a forgery an $\texttt{opn}$-instance, $\mathcal{B}$ outputs as forgery the instance $\mathbb{x}$ and valid proof π contained in aux_{Φ}.

We can assume that the representations of the commitments in the $\texttt{opn}$-forgery and the representations of the same commitments in the auxiliary output aux_{Φ} are the same; otherwise, if we have two distinct representations for the same commitment, we break the binding of KZG. Thus, the canonical extractor would output the same opening both when extracting the $\texttt{opn}$-forgery (using the representations in $\mathsf{aux}_{\mathcal{E}}$) and when extracting from aux_{Φ}. □

From now on, we parse $\mathbb{x}^*$ as $(\texttt{poly}, \mathbb{x}^*_{\texttt{poly}})$, and we define $\pi^*_{\texttt{poly}} \coloneqq \pi^*$.

We define a hybrid experiment $\mathbf{H}_1$ that is equivalent to the $\hat{\Phi}$-simulation extractability experiment but additionally the hybrid experiment outputs 0 if the instance $\mathbb{x}^*_{\texttt{poly}}$ in the forgery of the adversary is in the set of simulation queries and the canonical extractor fails to extract a valid witness.

Lemma 16. $\mathbf{Adv}^{\hat{\Phi}\text{-}se}_{\mathsf{CP},\mathcal{S},\mathcal{E},\mathcal{A}}(\lambda) \leq \Pr[\boldsymbol{H}_1] + \epsilon_{\texttt{m-evl}} + \epsilon_{\textit{Aff-MDH}}$

Proof. The distinguishing event between the original experiment and $\mathbf{H}_1$ is that $\mathcal{A}$ returns a valid "fresh" proof for a statement for which has seen a simulated proof which the extractor cannot extract.

We reduce to $\Phi_{\texttt{m-evl}}$-simulation extractability showing an adversary that produces a *fresh* proof for a statement $\mathbb{x}_{\texttt{m-evl}}$ for which has seen a simulated proof.

Consider the policy $\Phi_{\texttt{m-evl}} = (\bar{\Phi}_0, \bar{\Phi}_1) \in \mathbf{\Phi}_{\texttt{m-evl}}$ where $\bar{\Phi}_0(\mathsf{pp}_{\mathbb{G}})$ does the following:

1. Runs the policy $\Phi_0(\mathsf{pp}_{\mathbb{G}})$ and obtains the set of vectors $\mathcal{Q}_v$.
2. Samples random $\tilde{x}_1, \ldots, \tilde{x}_q$ from $\mathbb{F}$ and computes $\mathcal{Q}_x \coloneqq \{v_j(\tilde{x}_i) : \boldsymbol{v} \in \mathcal{Q}_v\}_{i,j}$.

3. Runs for $i \in [Q_{\mathsf{sim}}]$ the sampler $\mathbf{c}^{(i)} \leftarrow\!\!\$\ \mathcal{D}(\mathsf{pp}_{\mathbb{G}})$, where Q_{sim} is the maximum number of simulation queries made by $\mathcal{A}$, and defines $\mathsf{coms} := (\mathbf{c}^{(i)})_i$.
4. Outputs $(\mathcal{Q}_x, \mathsf{coms})$.

We can assume w.l.g. that $\mathcal{A}$ queries the random oracle on $\mathbb{x}$ before querying the simulation oracle on such an instance.

Consider the adversary $\mathcal{B}$ for the $\Phi_{\texttt{m-evl}}$-simulation-extractability game that:

1. Run the adversary $\mathcal{A}$ on parameters ck and $\mathsf{pp}_{\Phi} := (\mathcal{Q}_v, \mathsf{coms})$
2. Parse coms as $(\mathbf{c}^{(i)})_{i\in[Q_{\mathsf{sim}}]}$ and keep a list $\mathcal{Q}'_{\mathsf{RO}}$ (initially empty) of the random oracle call of $\mathcal{A}$.
3. At the i-th random-oracle query on input (s, aux), store $(s, \mathsf{aux}, \tilde{x}_i)$ in $\mathcal{Q}'_{\mathsf{RO}}$, and forward $\tilde{x}_i$ to the adversary $\mathcal{A}$.
4. Upon ($\texttt{poly}$-instance) simulation query with a tuple $(\mathbb{x}_{\texttt{poly}}, \mathsf{aux}_{\texttt{poly}})$ from $\mathcal{A}$:
   - Find in $\mathsf{aux}_{\texttt{poly}}$ the leakage-input for the simulator $((y_{i,j})_{j\in\mathcal{P}_i}, y'_i)_{i\in[m]}$.
   - Following the specification of the prover, query the simulation oracle with instances $\mathbb{x}_{\texttt{m-evl},i} = ((\mathbf{c}_j)_{j\in\mathcal{P}_i}, x^*_i, (y_{i,j})_{i,j\in\mathcal{P}_i})$, using y'_i as the leakage for the i-th instance. Parse them as a proof for $\texttt{poly}$.
5. Upon forgery $(\mathbb{x}^*_{\texttt{poly}}, \mathsf{aux}^*_{\mathcal{E}}, \pi^* = (\pi^*_{\texttt{m-evl},i}, (y_{i,j})_j)_i)$:
   (a) **Abort** if $\mathcal{A}$ never queried $\mathcal{S}_2$ with $\mathbb{x}^*_{\texttt{poly}}$. .
   (b) Let $\tilde{\pi}_{\texttt{poly}} := (\tilde{\pi}_{\texttt{m-evl},i}, (\tilde{y}_{i,j})_j)_i$ be the first simulated proof for $\mathbb{x}^*_{\texttt{poly}}$ and let i^* be the index such that either $(y_{i^*,j}) \neq (\tilde{y}_{i^*,j})$ or $\tilde{\pi}_{\texttt{m-evl},i} \neq \pi_{\texttt{m-evl},i}$, return the forgery $(\mathbb{x}^*_{\texttt{m-evl},i^*}, \mathsf{aux}^*_{\mathcal{E}}, \pi^*_{\texttt{m-evl},i^*})$.

We need to show that if $\hat{\Phi}$ holds for $\mathcal{A}$ then the policy $\Phi_{\texttt{m-evl}}$ holds for $\mathcal{B}$. Notice that, by condition (1) of the semi-adaptive simulation queries property of $\hat{\Phi}$ and the definition of Φ_0 the reduction $\mathcal{B}$ calls its own simulator on points in $\mathcal{Q}_x$, moreover conditions (2,3) of the semi-adaptive simulation queries property of $\hat{\Phi}$ easily imply respectively the Commitment Check and the Algebraic Consistency of $\Phi_{\texttt{m-evl}}$ (cf. Definition 27).

By inspection on the forgery, if the reduction $\mathcal{B}$ does not abort and there exists only one $\tilde{\pi}_{\texttt{poly}}$ associated with the forged instance then $\mathcal{B}$ matches the predicate $\Phi^{\mathrm{der}}_{\mathsf{ext}}$ of the $\Phi_{\texttt{m-evl}}$ policy (cf. Definition 28). We show that, because of condition (2) and the algebraic consistency check of the semi-adaptive simulation queries property of $\hat{\Phi}$, there exists indeed only one simulated proof $\tilde{\pi}_{\texttt{poly}}$ for each queried instance. In fact, from an adversary that asks twice the same instance to the simulator, let say at query i and i', we can derive that $\boldsymbol{C}^{(i)} \cdot \mathbf{c}^{(i)} + \boldsymbol{F}^{(i)} \cdot \mathsf{ck} = \boldsymbol{C}^{(i')} \cdot \mathbf{c}^{(i')} + \boldsymbol{F}^{(i')} \cdot \mathsf{ck}$. If the matrices $\boldsymbol{C}^{(i)}$ or $\boldsymbol{C}^{(i')}$ are non-zero then we can break the $\mathcal{D}$-Aff-MDDH assumption. On the other hand, if both matrices are the zero matrix, we have that the commitments are fully defined given the ck and thus by the algebraic consistency check the leakage for the simulated proofs must be the same (and thus the proofs are the same). □

The next hybrid experiment $\mathbf{H}_2$ is equivalent to $\mathbf{H}_1$ but additionally it outputs 0 if the forgery of the adversary is valid and the canonical extractor fails to extract valid witnesses for the $\texttt{m-evl}$-instances derived by $\mathbb{x}^*_{\texttt{poly}}$.

Lemma 17. $\Pr[\boldsymbol{H}_1] \leq \Pr[\boldsymbol{H}_2] + \frac{\epsilon_{\texttt{m-evl}}}{Q_{\mathsf{sim}}}$

Proof. We reduce again to $\Phi_{\texttt{m-evl}}$-simulation-extractability We consider the same $\bar{\Phi}_0(\mathsf{pp}_{\mathbb{G}})$ as in the previous lemma.

Consider the adversary $\mathcal{B}$ that:

1. Sample an index $q^* \leftarrow\!\!\$\ [Q_{\mathsf{sim}}]$.
2. Run the adversary $\mathcal{A}$ on parameters ck and $\mathsf{pp}_{\Phi} := (\mathcal{Q}_v, \mathsf{coms})$
3. Parse coms as $(\mathbf{c}^{(i)})_{i\in[Q_{\mathsf{sim}}]}$ and keep a list $\mathcal{Q}'_{\mathsf{RO}}$ (initially empty) of the random oracle call of $\mathcal{A}$.
4. At the i-th random-oracle query on input (s, aux):
   - if $i \neq q^*$ then store $(s, \mathsf{aux}, \tilde{x}_i)$ in $\mathcal{Q}'_{\mathsf{RO}}$, and forward $\tilde{x}_i$ to $\mathcal{A}$
   - else parse s as $(\mathtt{msg}, \mathbf{c}, (G^{(k)}, \boldsymbol{v}^{(k)})_k)$, find k^* such that $\forall j : \mathsf{deg}(v_j^{(k^*)}) \geq 1$, set $\mathsf{aux}' := (\mathsf{aux}, (v_j^{(k^*)})_{j\in[n]})$, and send the query (s, aux') to $\mathcal{S}_2$, and forward a to the adversary $\mathcal{A}$
5. Upon simulation query with a tuple $(\mathbb{x}_{\texttt{poly}}, \mathsf{aux}_{\texttt{poly}})$ from $\mathcal{A}$ (as in the previous lemma):
   - find in $\mathcal{Q}'_{\mathsf{RO}}$ the tuple $(\mathbb{x}_{\texttt{poly}}, \cdot, \tilde{x})$
   - find in $\mathsf{aux}_{\texttt{poly}}$ the leakage-input for the simulator $((y_{i,j})_{j\in\mathcal{P}_i}, (y'_i))_{i\in[m]}$
   - following the specification of the prover, query the simulation oracle with instances $\mathbb{x}_{\texttt{m-evl},i} = ((\mathbf{c}_j)_{j\in\mathcal{P}_i}, x_i^*, (y_{i,j})_{i,j\in\mathcal{P}_i})$, using y'_i as the leakage for the i-th instance
6. Upon forgery $(\mathbb{x}^*_{\texttt{poly}}, \mathsf{aux}^*_{\mathcal{E}}, \pi^* = (\pi^*_{\texttt{m-evl},i}, (y_{i,j})_j)_i)$
   (a) **abort** if $\mathbb{x}^*_{\texttt{poly}}$ was not queried at the q^*-th random oracle query by $\mathcal{A}$
   (b) **abort** if $\forall j \in [n]$ c_j can be extracted as $p_j(X)$ and $\forall i : p_j(x_i^*) = y_{i,j}$,
   (c) Let j^*, i be such that $p_{j^*}(x_i^*)$ the previous check does not hold. Return the forgery $(\mathbb{x}^*_{\texttt{m-evl},i^*}, \mathsf{aux}^*_{\mathcal{E}}, \pi^*_{\texttt{m-evl},i^*})$.

Let Abort be the event that $\mathcal{B}$ aborts. We notice that $\mathcal{B}$ could abort if one of two distinct events happens. The first event Abort_1 in Item 6a, and the second event Abort_2 is the condition in Item 6b. Notice that the distinguishing event between the two hybrids implies that Abort_2 does not happen. Moreover, notice that when the reduction does not abort, it returns a valid proof that either the canonical extractor cannot extract or such that the extracted polynomial $p_j(x_i^*) \neq y_{i^*,j^*}$; thus, if the policy $\Phi_{\texttt{m-evl}}$ is valid the reduction wins the $\Phi_{\texttt{m-evl}}$-simulation extractability experiment. Namely,

$$\Pr\left[\bar{\Phi}_1 \wedge \neg\mathsf{Abort} | b = 0\right] \leq \epsilon_{\texttt{m-evl}}$$

The event Abort_1 only depends on the uniformly random index q^* which is independent of the view of the adversary $\mathcal{A}$, i.e., $\Pr[\neg\mathsf{Abort}_1] = \frac{1}{Q_{\mathsf{sim}}}$. Notice that $\mathcal{A}$ wins the $\Phi_{\texttt{poly}}$-simulation extractability experiment when $\neg\mathsf{Abort}_2$ happens and Φ_1 holds. Thus, we only need to show that, conditioned on $\neg\mathsf{Abort}$, when the policy Φ_1 holds w.r.t. the view of $\mathcal{A}$ then the policy $\bar{\Phi}_1$ holds w.r.t. the view of $\mathcal{B}$. By construction of $\mathcal{Q}_x$ and the answers to the random oracle queries to $\mathcal{A}$, if the simulation query of $\mathcal{A}$ has $\boldsymbol{v}^{(k)} \in \mathcal{Q}_v$ then the evaluation points derived are in $\mathcal{Q}_x$. Moreover, it is not hard to see that the consistency check of the two policies match: in particular, the conditions on matrix $\boldsymbol{C}_i$ (resp $\boldsymbol{F}_i$) are

equivalent to the consistent opening check described in Item 3 of the extraction policy (the adversary is considered valid if its queries form a solvable linear system of equations defined by the coefficients of the linear combinations of commitments and the evaluation values). As for the forgery, in case of $\neg\mathsf{Abort}_1$, the entry $(\mathbb{x}^*_{\mathtt{poly}}, \mathsf{aux}', a^*)$ is indeed in the list of random oracle checked by the `m-evl`-policy: in fact, it is the only query. And, by construction, aux' contains the polynomial $v_{j^*}^{(k^*)}$. Putting things together we have the statement of the lemma. □

The next hybrid experiment $\mathbf{H}_3$ is equivalent to $\mathbf{H}_2$ but additionally it outputs 0 if the instance in the forgery of the adversary is valid and the canonical extractor fails to extract valid witness, namely:

$$\exists k : \bar{G}^{(k)}(X, p_1(v_n^{(k)}(X)), \ldots, p_n(v_n^{(k)}(X))) \not\equiv 0$$

Lemma 18. $\Pr[\boldsymbol{H}_3] \leq \frac{\mathsf{poly}(d,\lambda)}{|\mathbb{F}|}$.

Proof. First notice that by the changes introduced in the previous hybrid the canonical extractor must extract valid polynomials and $p_j(x_i^*) = y_{i,j}$ for any i and $j \in \mathcal{P}_i$. Also x^* is sampled after all the polynomials extracted by the canonical extractor are defined.

Let d be the degree of $\bar{G}^{(k)}(X, p_1(v_n^{(k)}(X)), \ldots, p_n(v_n^{(k)}(X)))$; such value is polynomial in the security parameter and the maximum degree for the commitments. If the verification passes we have:

$$\bar{G}^{(k)}(x^*, p_1(v_n^{(k)}(x^*), \ldots, p_n(v_n^{(k)}(x^*))) = 0$$

which, by Swartz-Zippel lemma happens with probability $\frac{d}{\mathbb{F}}$. □

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

References

1. M. Abdalla, M. Barbosa, J. Katz, J. Loss, and J. Xu. Algebraic adversaries in the universal composability framework. In M. Tibouchi and H. Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 311–341. Springer, Heidelberg, Dec. 2021.
2. B. Abdolmaleki, S. Ramacher, and D. Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 2020*, pages 1987–2005. ACM Press, Nov. 2020.

3. S. Arora and S. Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd FOCS*, pages 2–13. IEEE Computer Society Press, Oct. 1992.
4. K. Baghery, M. Kohlweiss, J. Siim, and M. Volkhov. Another look at extraction and randomization of groth's zk-snark. In N. Borisov and C. Díaz, editors, *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 457–475. Springer, 2021.
5. M. Bellare, D. Hofheinz, and E. Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? *Journal of Cryptology*, 28(1):29–48, Jan. 2015.
6. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable zero knowledge with no trusted setup. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732. Springer, Heidelberg, Aug. 2019.
7. E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, Aug. 2013.
8. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.
9. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, Oct. / Nov. 2016.
10. E. Ben-Sasson, L. Goldberg, S. Kopparty, and S. Saraf. DEEP-FRI: Sampling outside the box improves soundness. In T. Vidick, editor, *ITCS 2020*, volume 151, pages 5:1–5:32. LIPIcs, Jan. 2020.
11. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Heidelberg, May 2004.
12. D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081, 2020. https://eprint.iacr.org/2020/081.
13. D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Halo infinite: Proof-carrying data from additive polynomial commitments. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 649–680, Virtual Event, Aug. 2021. Springer, Heidelberg.
14. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.
15. B. Bünz, A. Chiesa, P. Mishra, and N. Spooner. Recursive proof composition from accumulation schemes. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 1–18. Springer, Heidelberg, Nov. 2020.
16. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020.
17. M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In M. Tibouchi and H. Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Heidelberg, Dec. 2021.

18. M. Campanelli, D. Fiore, and A. Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, Nov. 2019.
19. M. Chase and A. Lysyanskaya. On signatures of knowledge. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, Aug. 2006.
20. A. Chiesa, Y. Hu, M. Maller, P. Mishra, P. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.
21. Q. Dao and P. Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In C. Hazay and M. Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 531–562. Springer, Heidelberg, Apr. 2023.
22. Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631. Springer, Heidelberg, Dec. 2010.
23. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991.
24. S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In S. D. Galbraith and M. Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Heidelberg, Dec. 2012.
25. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, Aug. 2005.
26. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, Aug. 2018.
27. A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.
28. C. Ganesh, H. Khoshakhlagh, M. Kohlweiss, A. Nitulescu, and M. Zajac. What makes fiat-shamir zksnarks (updatable SRS) simulation extractable? In C. Galdi and S. Jarecki, editors, *Security and Cryptography for Networks, SCN 2022*, volume 13409 of *Lecture Notes in Computer Science*, pages 735–760. Springer, 2022.
29. C. Ganesh, Y. Kondi, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Witness-succinct universally-composable snarks. Cryptology ePrint Archive, Paper 2022/1618, 2022. https://eprint.iacr.org/2022/1618.
30. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In O. Dunkelman and S. Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 397–426. Springer, Heidelberg, May / June 2022.
31. S. Garg, A. Jain, and A. Sahai. Leakage-resilient zero knowledge. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 297–315. Springer, Heidelberg, Aug. 2011.
32. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

33. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.
34. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, Dec. 2010.
35. J. Groth. On the size of pairing-based non-interactive arguments. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.
36. J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, Aug. 2018.
37. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, Aug. 2017.
38. C. S. Jutla and A. Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, Dec. 2013.
39. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, Dec. 2010.
40. J. Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In K. Nissim and B. Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 1–34. Springer, Heidelberg, Nov. 2021.
41. H. Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, Mar. 2012.
42. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, Nov. 2019.
43. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updateable structured reference strings. Cryptology ePrint Archive, Report 2019/099, 2019. https://eprint.iacr.org/2019/099.
44. S. Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, Nov. 1994.
45. P. Morillo, C. Ràfols, and J. L. Villar. The kernel matrix Diffie-Hellman assumption. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Heidelberg, Dec. 2016.
46. C. Ràfols and A. Zapico. An algebraic framework for universal and updatable SNARKs. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, Aug. 2021. Springer, Heidelberg.
47. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, Oct. 1999.
48. A. Szepieniec. Polynomial IOPs for linear algebra relations. Cryptology ePrint Archive, Report 2020/1022, 2020. https://eprint.iacr.org/2020/1022.

49. A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In C. Galdi and V. Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 45–64. Springer, Heidelberg, Sept. 2020.
50. A. Zapico, V. Buterin, D. Khovratovich, M. Maller, A. Nitulescu, and M. Simkin. Caulk: Lookup arguments in sublinear time. In H. Yin, A. Stavrou, C. Cremers, and E. Shi, editors, *ACM CCS 2022*, pages 3121–3134. ACM Press, Nov. 2022.

A CP-SNARK for evaluation of multiple polynomials in AGM

Here we generalize the scheme described in Section 4.1 to the batched setting, highlighting the parts of the construction that are not needed if hiding is not desired. In particular, this scheme $\mathsf{CPm\text{-}evl}$ allows one to prove that for all i the polynomial f_i committed in c_i evaluates to y_i on the point x. This batched version, which is given in the ROM, follows from [27,43] and relies on the linearity of the polynomials and the homomorphic properties of KZG. Our contribution is to prove its policy-based simulation extractability.

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

$\mathsf{Prove}_{\mathtt{m\text{-}evl}}(\mathsf{ek}, \mathbb{x} = (x,(\mathsf{c}_i,y_i)_i), \mathbb{w} = (f_i,r_i)_i)$**:** for all i compute the polynomials $\pi_i(X)$ such that $\pi_i(X)(X-x) \equiv f_i(X) - y_i$, the polynomials $\pi'_i(X)$ such that $\pi'_i(X)(X-x) \equiv r_i(X) - r(x)$, $\rho \leftarrow \mathsf{RO}(\mathsf{batch}\|\mathbb{x})$, and output

$$\left(\sum_i \rho^{i-1} \left[\pi_i(s)+\alpha\pi'(s)\right]_1, \sum_i \rho^{i-1} r_i(x)\right)$$

$\mathsf{Verify}_{\mathtt{m\text{-}evl}}(\mathsf{vk}, \mathbb{x} = (x,(\mathsf{c}_i,y_i)_i), (\pi,y'))$**:** compute $\rho \leftarrow \mathsf{RO}(\mathsf{batch}\|\mathbb{x})$ and output 1 iff

$$e(\sum_i \rho^{i-1}\mathsf{c}_i - \left[\sum_i \rho^{i-1} y_i\right]_1 - [\alpha y']_1, [1]_2) = e(\pi, [s-x]_2).$$

Similarly to CPevl, we can prove that $\mathsf{CPm\text{-}evl}$ achieves F-leaky zero-knowledge (see Definition 11) where $F(\mathbb{x} = (x,(\mathsf{c}_i,y_i)_i), \mathbb{w} = (f_i,r_i)_i) = \sum_i \rho^{i-1} r_i(x)$ and $\rho = \mathsf{RO}(\mathsf{batch}\|\mathbb{x})$.

We define the simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$, where $\mathcal{S}_0$ outputs the trapdoor information s, α together with the srs, $\mathcal{S}_2$ simulates the random oracle on any input via lazy sampling, and $\mathcal{S}_1$ simulates proofs for $\mathbb{x} = (x,(\mathsf{c}_i,y_i)_i)$ and leakage y' by outputting $\pi = ((\mathsf{c} - [y]_1 - [\alpha y']_1)(s-x)^{-1}, y')$, where $\mathsf{c} \leftarrow \sum_i \rho^{i-1}\mathsf{c}_i$, $y \leftarrow \sum_i \rho^{i-1} y_i$ for $\rho \leftarrow \mathcal{S}_2(\mathsf{batch}\|\mathbb{x})$.

The extraction policy. The extraction policy $\boldsymbol{\Phi}^{\mathtt{s\text{-}adpt}}_{\mathtt{m\text{-}evl}}$ is naturally extended from the single point case of CPevl. The evaluation points x_j for the instances for which the adversary can see simulated proofs are selectively chosen independently of the commitment key, while the evaluation values $(y_i)_i$ can be arbitrarily chosen by the adversary. Each policy $\Phi_{\mathcal{D}}$ in the family is a tuple of the form $(\Phi_0^{\mathcal{D}}, \Phi_1)$. Each policy $\Phi_{\mathcal{D}}$ is a tuple of the form $(\Phi_0^{\mathcal{D}}, \Phi_1)$, as defined in Section 3.1, where $\Phi_0^{\mathcal{D}}$ outputs the parameters pp_Φ while Φ_1 outputs a virdict bit. In particular, $\Phi_0^{\mathcal{D}}$ on input group parameters $\mathsf{pp}_{\mathbb{G}}$ outputs $\mathsf{pp}_\Phi := (\mathsf{coms}, \mathcal{Q}_x)$, where coms is a vector of commitments sampled from $\mathcal{D}$, and $\mathcal{Q}_x$ is a set of $\mathcal{Q}_x$ evaluation points.

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

the second defines rules under which the extractor must be able to extract a meaningful witness. We highlight the parts needed only for the hiding setting.

Definition 27. *Let Φ_{sim} be the policy that returns 1 if and only if:*

1. **Points check:** *let $(\mathbb{x}_i, \mathsf{aux}_i, \pi_i)_i$ be all the entries of $\mathcal{Q}_{\mathsf{sim}}$. Recall that an instance $\mathbb{x}$ can be parsed as $(x, (\boldsymbol{c}, \boldsymbol{y}))$, check that $\forall i : \mathbb{x}_i.x \in \mathcal{Q}_x$.*
2. **Commitment Check:** *For any $i \in [Q_{\mathsf{sim}}]$, parse aux_i as the leakage value y'_i and the representation vectors for $\mathbb{x}_i.\boldsymbol{c}$ and π_i; in particular, let $\boldsymbol{M}_i = \boldsymbol{F}_i \| \boldsymbol{V}_i \| \boldsymbol{C}_i$ be the algebraic representation of the commitments $\mathbb{x}_i.\boldsymbol{c}$. For any i check that $\left(\boldsymbol{F}_i \| \boldsymbol{V}_i\right) \cdot \mathsf{ek} + \boldsymbol{C}_i \cdot \mathsf{coms} = \mathbb{x}_i.\boldsymbol{c}$ (namely, that the commitments in the instance $\mathbb{x}_i$ are computed as a linear combination of the simulated commitments and the elements of $\mathbb{G}_1$ of the SRS)*
3. **Algebraic Consistency:** *The simulation queries satisfy the algebraic consistency for $\mathsf{CP}_{\mathtt{m\text{-}evl}}$. Namely, let $\mathcal{I}_J = \{i : \mathbb{x}_i.x = x_j\}$ and let $\boldsymbol{R}_j = (\boldsymbol{C}_i)_{i \in \mathcal{I}_J}$. Check that for all j: (i) the system of linear equations $\boldsymbol{R}_j \cdot \boldsymbol{z} = \boldsymbol{y}_j$ has at least a solution, where $\boldsymbol{z}$ are the variables and $\boldsymbol{y}_j = (\mathbb{x}_i.\boldsymbol{y} + \boldsymbol{F}_i \cdot (1, x_j, \ldots, x_j^d)^\top)_{i \in \mathcal{I}_J}$, and (ii) the system of linear equations $\boldsymbol{R}_j \cdot \boldsymbol{z}' = \boldsymbol{y}'_j$ has at least a solution, where $\boldsymbol{z}'$ are the variables and $\boldsymbol{y}'_j = (y'_i + (1, \rho_i, \ldots, \rho_i^m) \cdot \boldsymbol{V}_i \cdot (1, x_j, \ldots, x_j^d)^\top)_{i \in \mathcal{I}_J}$, and $\rho_i = \mathsf{RO}(\mathsf{batch} \| \mathbb{x}_i)$.*

Next, we define the policy Φ_{ext} as the logical disjunction of two policies. We recall and extend the notation introduced in Section 4.1: let $g_c \colon \mathbb{G}_1^* \times \{0,1\}^* \to \{0,1\}$ be a function that on inputs a list of group elements c_i and a string s, that can be parsed as a list of group elements $\hat{\mathsf{c}}_i$ followed by a second string $\tilde{s}$, outputs 1 iff $\forall i, \exists j : \mathsf{c}_i = \hat{\mathsf{c}}_j$.

Definition 28. *Let $\Phi_{\mathsf{ext}}, \Phi_{\mathsf{ext}}^{rnd}$ and $\Phi_{\mathsf{ext}}^{der}$ be predicates that parse the forgery instance $\mathbb{x}^* = (x^*, \boldsymbol{c}^*, \boldsymbol{y}^*)$.*

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

Theorem 6. *For any witness sampleable distribution $\mathcal{D}$ that is $\mathcal{D}$-Aff-MDH-secure (see Definition 5), any bilinear-group generator $\mathsf{GroupGen}$ that samples the generator of the group $\mathbb{G}_1$ uniformly at random, $\forall \Phi_{\mathcal{D}} \in \boldsymbol{\Phi}_{\mathtt{m\text{-}evl}}^{\mathtt{s\text{-}adpt}}$, the scheme $\mathsf{CP}_{\mathtt{m\text{-}evl}}$ is $\Phi_{\mathcal{D}}$-simulation-extractable in the AGM. In particular, there exists $\mathcal{E}$ such that for any algebraic adversary $\mathcal{A}$:*

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

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

Proof. The goal of the proof is to show that for any algebraic adversary $\mathcal{A}$ there exists an extractor $\mathcal{E}$ such that $\Pr[\mathbf{Exp}^{\Phi_{\mathcal{D}}\text{-}\mathsf{se}}_{\Pi_{\mathtt{m\text{-}evl}},\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda)=1]$ is negligible.

Since $\mathcal{A}$ is algebraic (cf. Definition 1), for each group element g that it produces, it additionally attaches a representation vector $\boldsymbol{r}_{\mathsf{g}} := \boldsymbol{f}_{\mathsf{g}} \| \boldsymbol{c}_{\mathsf{g}} \| \boldsymbol{o}_{\mathsf{g}}$ such that:

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

where ek are all the group elements of the SRS, $\mathsf{coms} = ([c_i]_1)_{i\in[Q_c]}$ are the simulated commitments, and proofs are the simulated proofs.

The algebraic assumption applies to each query $(\mathbb{x} = (x, (\mathsf{c}_i, y_i)_i), \mathsf{aux})$ to the oracle $\mathcal{S}_1$, in which case we can parse aux as $((\boldsymbol{r}_{\mathsf{c}_i})_i, \mathsf{aux}')$ such that $\boldsymbol{r}_{\mathsf{c}_i}$ is the representation of c_i, as well as to each query (s, aux) to $\mathcal{S}_2$, in which case aux includes a valid representation for all the group elements g_i encoded in s, i.e., such that $g_c(\mathsf{g}_i, s) = 1$.

Furthermore, when returning its forgery $\mathbb{x}^* = (x^*, (\mathsf{c}_i^*, y_i)_i), [\pi]_1^*$, the algebraic adversary encodes a polynomial $h(X)$ in aux_ϕ, and stores in $\mathsf{aux}_{\mathcal{E}}$ the representation vectors $\{\boldsymbol{r}_{c_i^*}\}_i$ and $\boldsymbol{r}_{\pi^*}$ for the group elements $(\mathsf{c}_i^*)_i$ and π^*.

We can assume w.l.o.g. that all the simulation queries and the forgery of the adversary $\mathcal{A}$ agree with the policy $\Phi_{\mathcal{D}}$, as otherwise the adversary would automatically lose the experiment. In particular, we assume that the forgery satisfies the extraction predicate $\Phi^{\mathrm{rnd}}_{\mathsf{ext}}$: we show in Lemma 28 why this is without loss of generality. For every query $(\mathbb{x} = (x, (\mathsf{c}_i, y_i)_i), \mathsf{aux})$ to the oracle $\mathcal{S}_1$, recall that this means that $\boldsymbol{o}_{\mathsf{c}_i} = \mathbf{0}$ for all i.

We define the extractor $\mathcal{E}$ to be the *canonical* extractor that, given $\mathsf{aux}_{\mathcal{E}}$, returns the polynomials $(f_i(X))_i$ where:

$$f_i(X) = \langle \boldsymbol{f}_{c_i^*}, (1, X, \ldots, X^d) \rangle$$

We let $\mathbf{H}_0(\mathcal{A})$ be the $\mathbf{Exp}^{\Phi_{\mathcal{D}}\text{-}\mathsf{se}}_{\Pi_{\mathtt{m\text{-}evl}},\mathcal{A},\mathcal{S},\mathcal{E}}(\lambda)$ experiment (executed with adversary $\mathcal{A}$).

Hybrid $\mathbf{H}_1$. This is a modification of $\mathbf{H}_0$ in which the adversary is allowed to only do queries to $\mathcal{S}_1$ of the form $(\mathbb{x} = (x, \mathsf{c}, y), \mathsf{aux})$ (i.e., for single evaluations) and such that $\boldsymbol{f}_{\mathsf{c}} = \mathbf{0}$. This means that an adversary not meeting this requirement loses, i.e., the hybrid returns 0. As we show in the following lemma, we can perfectly simulate adversaries in $\mathbf{H}_0$ using an adversary in $\mathcal{A}_1$. Such a change is useful as it allows us to later consider "simpler" adversaries.

Lemma 19. *For any $\mathcal{A}_0$ there exists $\mathcal{A}_1$ such that* $\Pr[\boldsymbol{H}_1(\mathcal{A}_1)] = \Pr[\boldsymbol{H}_0(\mathcal{A}_0)]$.

Proof. We build $\mathcal{A}_1$ runs $\mathcal{A}_0$ providing its own inputs and forwarding all the random oracle queries, except for $\mathcal{S}_1$ queries that are handled as follows. Whenever $\mathcal{A}_0$ makes a simulation query $(\mathbb{x} = (x, (\mathsf{c}_i, y_i)_i), \mathsf{aux})$, $\mathcal{A}_1$ proceeds as described below.

– For every i, parse $\boldsymbol{r}_{\mathsf{c}_i} \coloneqq \boldsymbol{f}_{\mathsf{c}_i} \| \boldsymbol{c}_{\mathsf{c}_i} \| \boldsymbol{o}_{\mathsf{c}_i}$ and define $f_i(X) \coloneqq \langle \boldsymbol{f}_{\mathsf{c}_i}, (1, X, \ldots, X^d)$.
– Compute $y'_i \leftarrow f_i(x)$, $\pi'_i(X) \leftarrow (f_i(X) - y'_i)/(X - x)$ and $\pi'_i \leftarrow [\pi'_i(s)]_1$, which is essentially an "honest" evaluation proof w.r.t. a commitment to the polynomial $f_i(X)$, input x and output y'_i.
– For every i, ask the query $(x, \mathsf{c}_i - [f_i(s)]_1, y_i - y'_i)$ to its simulation oracle. Notice that the queried commitments satisfy the condition $\boldsymbol{f} = 0$. Let $\tilde{\pi}_i$ be the obtained proofs.
– "Merge" these proofs by computing $\pi_i \leftarrow \pi'_i + \tilde{\pi}_i$. One can verify that π_i is a valid evaluation proof for (x, c_i, y_i).
– Aggregate all the proofs by computing $\pi \leftarrow \sum_i \rho^{i-1} \pi_i$, where $\rho \leftarrow \mathcal{S}_2(\mathbb{x})$.

Hybrid $\mathbf{H}_2$. Recall that $\mathcal{D}$ is witness sampleable, thus according to Definition 4 there exists a PPT algorithm $\tilde{\mathcal{D}}$ associated with the sampler $\mathcal{D}$. The hybrid experiment $\mathbf{H}_2$ is identical to the previous one, but the group elements in coms are "sampled at the exponent", i.e., we use $\tilde{\mathcal{D}}$ to first generate the field elements $\boldsymbol{\gamma}$, we compute $\mathsf{coms} \leftarrow [\boldsymbol{\gamma}]_1$, and we also add $\boldsymbol{\gamma}$ to $\mathsf{st}_{\mathcal{S}}$. By the witness sampleability of $\mathcal{D}$, $\mathbf{H}_1$ and $\mathbf{H}_2$ are perfectly indistinguishable, i.e., for any $\mathcal{A}$ $\Pr[\mathbf{H}_2(\mathcal{A})] = \Pr[\mathbf{H}_1(\mathcal{A})]$.

Hybrid $\mathbf{H}_3$. In this hybrid, we change the way we generate the SRS srs with $\mathcal{S}_0$ and the way in which $\mathcal{S}_1$ simulates proofs.

Let $\left((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), [1]_1, [1]_2\right) \leftarrow\!\$\ \mathsf{GroupGen}(1^\lambda)$, sample $s \leftarrow\!\$\ \mathbb{F}$ and compute $\left[s, \ldots, s^{D+d}\right]_1, [1, s]_2$, where $D \leftarrow Q_x + 1$. Let $x_r \leftarrow\!\$\ \mathbb{F}$, and let $p(X)$ be the vanishing polynomial in $\mathcal{Q}_x \cup \{x_r\}$, namely:

$$p(X) \coloneqq (X - x_r) \prod_{x \in \mathcal{Q}_x} (X - x).$$

Let also $p_j(X) \coloneqq p(X)(X - x_j)^{-1}$, for $j \in [Q_x]$. In this game $\mathcal{S}_0$ computes:

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

Upon a query of the form $(\mathbb{x} = (x, \mathsf{c}, y), \mathsf{aux} = (\boldsymbol{r}_{\mathsf{c}}, \mathsf{aux}'))$ to $\mathcal{S}_1$ such that $x = x_j \in \mathcal{Q}_x$, we proceed as follows.

– Parse $\boldsymbol{r}_{\mathsf{c}} \coloneqq \boldsymbol{f}_{\mathsf{c}} \| \boldsymbol{c}_{\mathsf{c}} \| \boldsymbol{o}_{\mathsf{c}}$ (recall, $\boldsymbol{o}_{\mathsf{c}} = \mathbf{0}$ by simulation policy and $\boldsymbol{f}_{\mathsf{c}} = \mathbf{0}$ by the change introduced earlier).
– Compute $\pi \leftarrow [(\langle \boldsymbol{c}_{\mathsf{c}}, \boldsymbol{\gamma} \rangle - y) \cdot p_j(s)]_1$, which is a valid simulated proof w.r.t. commitment $[\langle \boldsymbol{c}_{\mathsf{c}}, \boldsymbol{\gamma} \rangle p(s)]_1$, input x and output y.

We now show that the view offered to the adversary $\mathcal{A}$ in the two experiments is statistically close.

Lemma 20. *For any $\mathcal{A}$* $\Pr[\boldsymbol{H}_3(\mathcal{A})] \leq \Pr[\boldsymbol{H}_2(\mathcal{A})] + \frac{Q_x + 1 + d}{q}$.

Proof. Let us first look at the distribution of the input provided to $\mathcal{A}$. Notice that in $\mathbf{H}_3$ we sample from $\mathsf{GroupGen}$ the description of the group, and then we set the generator of $\mathbb{G}_1$ to be $[p(s)]_1$ which, thanks to the random root x_r, is distributed uniformly at random even given the value s. This however holds except with the negligible probability $\frac{Q_x+1+d}{p}$ that s is one of the roots of $p(X)$.

Second, it is not hard to verify that the simulated proofs generated in the hybrid $\mathbf{H}_3$ pass the verification equations. Since the proofs are unique, given the SRS and the statements, the simulated proofs created in $\mathbf{H}_3$ are distributed identically to the simulated proofs generated by the simulator $\mathcal{S}_1$ in $\mathbf{H}_2$. □

Hybrid $\mathbf{H}_4$ and core adversaries. This is a modification of $\mathbf{H}_3$ in which we provide the adversary with a slightly different random oracle interface and in which the adversary loses if in its forgery there is a commitment c_i^* and associated representation vector such that $\boldsymbol{f}_{\mathsf{c}_i^*} \neq \mathbf{0}$. The random oracle is changed only in the sense that each input queried to the random oracle in $\mathbf{H}_4$ may contain more elements than the same query in $\mathbf{H}_3$. We call an adversary that satisfies this additional requirement of $\mathbf{H}_4$, i.e., $\boldsymbol{f}_{\mathsf{c}_i^*} = \mathbf{0}$ for all i, a *core* adversary.

To continue the proof, we show how to perfectly simulate any adversary $\mathcal{A}$ playing in $\mathbf{H}_3$ using a core adversary $\mathcal{A}_\mathsf{c}$ that runs in hybrid $\mathbf{H}_4$.

Lemma 21. *For any $\mathcal{A}$ there exists $\mathcal{A}_\mathsf{c}$ such that* $\Pr[\boldsymbol{H}_4(\mathcal{A}_\mathsf{c})] = \Pr[\boldsymbol{H}_3(\mathcal{A})]$.

Proof. We define $\mathcal{A}_\mathsf{c}$ as follows. Whenever the adversary $\mathcal{A}$ outputs a group element g it provides a representation vector $\boldsymbol{r}_\mathsf{g} := \boldsymbol{f}_\mathsf{g} \| \boldsymbol{c}_\mathsf{g} \| \boldsymbol{o}_\mathsf{g}$ for g such that:

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

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

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

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

$$
\begin{aligned}
&\mathsf{c}^*_i \leftarrow \mathsf{c}'_i - \mathsf{Com}(\mathsf{ck}, f_i(X)) = \mathsf{c}'_i - [f_i(s)p(s)]_1 \\
&x^* \leftarrow x' \\
&y^*_i \leftarrow y'_i - y_i = y'_i - f_i(x') \\
&\pi^* \leftarrow \pi' - \sum_i \rho^{i-1} \mathsf{Com}(\mathsf{ck}, q_i(X)) = \pi' - \left[p(s) \sum_i \rho^{i-1} q_i(s) \right]_1
\end{aligned}
$$

where ρ is the value obtained by $\mathcal{A}$ as the result of the random oracle query on $\mathsf{batch}\|(x', (\mathsf{c}_i, y'_i)_i)$. $\mathcal{A}_{\mathsf{c}}$ inserts into aux_{Φ} the (correct) algebraic representations $(\mathbf{0}\|\boldsymbol{c}_{c'_i}\|\boldsymbol{o}_{c'_i})$ for each c^*_i and $((\boldsymbol{f}_{\pi'} - \sum_i \rho^{i-1}\boldsymbol{f}_{q_i})\|\boldsymbol{c}_{\pi'}\|\boldsymbol{o}_{\pi'})$ for π^*.

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

$$
\mathsf{c}^*_i \coloneqq \mathsf{c}'_i - [f_i(s)p(s)]_1 \,, \quad \pi^* \coloneqq \pi' - \left[p(s) \sum_i \rho^{i-1} q_i(s) \right]_1 \,, \quad y'_i \coloneqq y^*_i - f_i(x^*).
$$

By the verification equation of the forgery of $\mathcal{A}_{\mathsf{c}}$ we have:

$$
\begin{aligned}
& e\left(\sum_i \rho^{i-1} \mathsf{c}^*_i - \sum_i \rho^{i-1} [y^*_i]_1 \,, [1]_2 \right) - e(\pi^*, [s - x^*]_2) && = \\
& e\left(\sum_i \rho^{i-1} \mathsf{c}'_i - \sum_i \rho^{i-1} [f_i(s)p(s)]_1 - [y'_i - f_i(x')]_1 \,, [1]_2 \right) \\
& \qquad - e\left(\pi' - \sum_i \rho^{i-1} [q_i(s)p(s)]_1 \,, [s - x^*]_2 \right) && = \\
& e\left(\sum_i \rho^{i-1} \mathsf{c}'_i - \sum_i \rho^{i-1} [y'_i]_1 \,, [1]_2 \right) - e(\pi', [s - x']_2) \\
& \qquad - \left[p(s) \sum_i \rho^{i-1} (f_i(s) - f_i(x') - q_i(s)(s - x^*)) \right]_T && = \\
& e\left(\sum_i \rho^{i-1} \mathsf{c}'_i - \sum_i \rho^{i-1} [y'_i]_1 \,, [1]_2 \right) - e(\pi', [s - x']_2)
\end{aligned}
$$

where the last equation holds because for every i we have $q_i(X)(X - x') = (f_i(X) - f_i(x'))$ and $x^* = x'$ by definition.

Finally, notice that a forgery is valid for $\mathcal{A}$ if it provides a string s that satisfies the "hash check" of Φ_{ext}. We have that there exist s, aux, a, and $h(X)$ such that: (i) $g_c((\mathsf{c}_i^*)_i, s) = 1$, (ii) $g_h(h, \mathsf{aux}) = 1$, (iii) $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO},\mathcal{A}}$, and (iv) $x^* = h(a)$ for the forgery of $\mathcal{A}$.

The way $\mathcal{A}_{\mathsf{c}}$ simulates the RO queries ensures that for the query s of $\mathcal{A}$ to the RO, the core adversary sent the "core" RO query s_{c} that encodes both vectors $(\mathsf{c}'_i)_i$ and $(\mathsf{c}_i^*)_i$, thus we have that (i) $g_c((\mathsf{c}_i^*)_i, s_{\mathsf{c}}) = 1$, (ii) $g_h(h, \mathsf{aux}_{\mathsf{c}}) = 1$, (iii) $(s_{\mathsf{c}}, \mathsf{aux}_{\mathsf{c}}, a) \in \mathcal{Q}_{\mathsf{RO}}$, and (iv) $x^* = h(a)$ for the forgery of $\mathcal{A}_{\mathsf{c}}$. □

Notice that if we run the canonical extractor on the outputs of a core adversary $\mathcal{A}_{\mathsf{c}}$ in $\mathbf{H}_4$, the extractor sets the extracted witness to be the zero polynomials.

Hybrid $\mathbf{H}_5$. The hybrid $\mathbf{H}_5$ additionally checks that $\boldsymbol{f}_{\pi^*} \neq \mathbf{0} \vee \boldsymbol{c}_{\pi^*} \neq \mathbf{0}$, and if the condition holds the adversary $\mathcal{A}_{\mathsf{c}}$ loses the game.

Lemma 22. *For any $\mathcal{A}_{\mathsf{c}}$ it holds* $\Pr[\boldsymbol{H}_5(\mathcal{A}_{\mathsf{c}})] \leq \Pr[\boldsymbol{H}_4(\mathcal{A}_{\mathsf{c}})] + \epsilon_{(Q_x+d+1)\text{-DL}}$

Proof. Recall that from the definition of the experiment, upon the ℓ-th query $(\mathbb{x}, \mathsf{aux})$ from $\mathcal{A}_{\mathsf{c}}$ to the simulation oracle $\mathcal{S}_1$ of the form $\mathbb{x} = (x_j, \mathsf{c}, y)$ and $\mathsf{aux} = \boldsymbol{r}_{\mathsf{c}} = (\mathbf{0} \| \boldsymbol{c}_{\mathsf{c}} \| \mathbf{0})$ such that $\mathsf{c} = \langle \boldsymbol{c}_{\mathsf{c}}, \mathsf{coms} \rangle$, the adversary receives the proof $[\pi_{\mathbb{x}}(s)]_1$ where:

$$\pi_\ell(X) \coloneqq (\langle \boldsymbol{c}_{\mathsf{c}}, \boldsymbol{\gamma} \rangle - y) p_j(X).$$

Given the forgery, consider the following polynomials:

$$c_i^*(X) \coloneqq \langle \boldsymbol{c}_{\mathsf{c}_i^*}, \boldsymbol{\gamma} \rangle p(X) + \sum_{\ell=1}^{|\mathcal{Q}_{\mathsf{sim}}|} o_{c_i^*,\ell} \cdot \pi_\ell(X)$$

$$\pi^*(X) \coloneqq \langle \boldsymbol{f}_{\pi^*}, (1, X, \ldots, X^d) \rangle p(X) + \langle \boldsymbol{c}_{\pi^*}, \boldsymbol{\gamma} \rangle p(X) + \sum_{\ell=1}^{|\mathcal{Q}_{\mathsf{sim}}|} o_{\pi^*,\ell} \cdot \pi_\ell(X)$$

$$v(X) \coloneqq \sum_i \rho^{i-1} (c_i^*(X) - y_i^* p(X)) - (X - x^*) \pi^*(X)$$

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

Next, we show that when the forgery of the adversary is valid the probability of $\boldsymbol{f}_{\pi^*} \neq \mathbf{0}$ or $\boldsymbol{c}_{\pi^*} \neq \mathbf{0}$ is bounded by $\epsilon_{(Q_x+d+1)\text{-DL}}$.

First, notice that if the verification equation for $\mathcal{A}_{\mathsf{c}}$ holds then the polynomial $v(X)$ must be equivalent to the zero polynomial with overwhelming probability. In fact, if $v(X) \not\equiv 0$ and $v(s) = 0$ then, by Lemma 1, we can reduce $\mathcal{A}_{\mathsf{c}}$ to an adversary against the $(Q_x + d + 1)$-DL assumption. To conclude the proof we show that, whenever $\boldsymbol{f}_{\pi^*} \neq \mathbf{0}$ or $\boldsymbol{c}_{\pi^*} \neq \mathbf{0}$, we have $v(X) \not\equiv 0$.

Assume by contradiction that:

$$\sum_i \rho^{i-1} (c_i^*(X) - y_i^* p(X)) - (X - x^*) \pi^*(X) = v(X) = 0. \tag{13}$$

For this equation to hold, it must be the case that the degree of $(X - x^*)\pi^*(X)$ is less than or equal to the degree of $\sum_i \rho^{i-1}(c_i^*(X) - y_i^* p(X))$. However, by the definition of $\pi^*(X)$ above, this cannot occur if $\boldsymbol{f}_{\pi^*} \neq \mathbf{0}$ or $\boldsymbol{c}_{\pi^*} \neq \mathbf{0}$ as in such a case the degree of $(X - x^*)\pi^*(X)$ would be $> Q_x + 1$ while the degree of $\sum_i \rho^{i-1}(c_i^*(X) - y_i^* p(X))$ is $\leq Q_x + 1$. Therefore we can conclude that the forged proof $\pi^*(s)$ is a linear combination of the simulated proofs only, i.e., both $\boldsymbol{f}_{\pi^*}$ and $\boldsymbol{c}_{\pi^*}$ are null. □

The representation of the commitments $(\mathsf{c}_i^*)_i$ and the proof π^* computed by the adversary in the forgery (possibly) depends on the simulated proofs proofs returned by $\mathcal{S}_1$. This dependence is characterized by the vector of coefficients $\boldsymbol{o}_\tau$ for $\tau \in \{\mathsf{c}_i^*\}_i \cup \{\pi^*\}$. However, to proceed with the proof it is more convenient to obtain an equivalent representation that depends on the polynomials $p_j(X)$. This motivates the definition of our next hybrid.

Hybrid $\mathbf{H}_6$. The hybrid $\mathbf{H}_6$ finds coefficients $\boldsymbol{o}''_\tau$, for $\tau \in \{\mathsf{c}_i^*\}_i \cup \{\pi^*\}$ such that:

$$\langle \boldsymbol{o}_\tau, \mathsf{proofs} \rangle = \langle \boldsymbol{o}''_\tau, ([p_j(s)]_1)_j \rangle. \tag{14}$$

Moreover, if there is an index k such that $\boldsymbol{o}_{\mathsf{c}_k^*} \neq \mathbf{0}$ but $\boldsymbol{o}''_{\mathsf{c}_k^*} = \mathbf{0}$ the adversary loses the game.

Lemma 23. *For any PPT $\mathcal{A}_\mathsf{c}$ we have* $\Pr[\boldsymbol{H}_6(\mathcal{A}_\mathsf{c})] \leq \Pr[\boldsymbol{H}_5(\mathcal{A}_\mathsf{c})] + \epsilon_{\mathit{Aff\text{-}MDH}}$

Proof. We begin by showing that the hybrid can compute such alternative representations efficiently. We proceed in steps.

Let us parse the simulated proofs $\mathsf{proofs} \coloneqq (\pi_{j,\ell})_{j,\ell}$ such that $\pi_{j,\ell}$ is the ℓ-th simulated proof obtained by $\mathcal{S}_1$ on a query involving the j-th point x_j, i.e., $((x_j, \hat{\mathsf{c}}_{j,\ell}, y_{j,\ell}), \mathsf{aux}_{j,\ell})$. Also, let $\boldsymbol{c}_{j,\ell}$ be the algebraic representation for the group element $\hat{\mathsf{c}}_{j,\ell}$ in $\mathsf{aux}_{j,\ell}$. For every $j \in [Q_x]$, we define $\boldsymbol{\Gamma}_j$ as the $Q_c \times Q_c$ matrix whose ℓ-th column is $\boldsymbol{c}_{j,\ell}$.

By construction of $\mathcal{S}_1$ in this hybrid we have that for every $j \in [Q_x]$ it holds

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

and thus $\boldsymbol{\pi}_j \coloneqq \left[(\boldsymbol{\Gamma}_j^\top \boldsymbol{\gamma} - \boldsymbol{y}_j) p_j(s)\right]_1$ with $\boldsymbol{y}_j \coloneqq (y_{j,\ell})_\ell$.

Without loss of generality, we assume that for each x_j the adversary makes the maximum number of simulation queries (i.e., $\ell \in [Q_c]$); therefore $\boldsymbol{\Gamma}_j$ is a full rank matrix (this follows from the fact that the simulation queries of the adversary satisfy the policy Φ_{sim}, and in particular the *consistent opening checks* of the policy, see Item 3).

Given any vector $\boldsymbol{o}_\tau$ with $\tau \in \{\mathsf{c}_i^*\}_i \cup \{\pi^*\}$, its m-th entry $o_{\tau,m}$ corresponds to the m-th simulated proof in proofs. Therefore, similarly to above, we denote by $o_{\tau,j,\ell}$ the entry corresponding to proof $\pi_{j,\ell}$ and we define $\boldsymbol{o}_{\tau,j} \coloneqq (o_{\tau,j,\ell})_\ell$.

Then, for every $j \in [Q_x]$ we define

$$\boldsymbol{o}'_{\tau,j} \leftarrow \boldsymbol{\Gamma}_j \cdot \boldsymbol{o}_{\tau,j} \tag{15}$$

$$\boldsymbol{\pi}'_j \leftarrow (\boldsymbol{\Gamma}_j^\top)^{-1} \cdot \boldsymbol{\pi}_j \tag{16}$$

from which we derive that for any τ:

$$\begin{aligned}\sum_j \langle \boldsymbol{o}'_{\tau,j}, \boldsymbol{\pi}'_j \rangle &= \sum_j \langle \boldsymbol{\Gamma}_j \cdot \boldsymbol{o}_{\tau,j}, (\boldsymbol{\Gamma}_j^\top)^{-1} \cdot \boldsymbol{\pi}_j \rangle \\ &= \sum_j \boldsymbol{o}_{\tau,j}^\top \boldsymbol{\Gamma}_j^\top (\boldsymbol{\Gamma}_j^\top)^{-1} \cdot \boldsymbol{\pi}_j \\ &= \sum_j \langle \boldsymbol{o}_{\tau,j}, \boldsymbol{\pi}_j \rangle.\end{aligned}$$

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

For all $j \in [Q_x]$ let $\boldsymbol{z}_j \coloneqq (\boldsymbol{\Gamma}_j^\top)^{-1} \cdot \boldsymbol{y}_j$, and note that

$$\boldsymbol{\pi}'_j = [(\boldsymbol{\gamma} - \boldsymbol{z}_j) p_j(s)]_1$$

namely $\pi'_{j,i}$ is a valid proof for the instance $(\mathsf{c}_i, x_j, z_{j,i})$ w.r.t. the simulated SRS.

The hybrid computes $o''_{\tau,j} \leftarrow \langle \boldsymbol{o}'_{\tau,j}, (\boldsymbol{\gamma} - \boldsymbol{z}_j) \rangle$ and sets $\boldsymbol{o}''_\tau \coloneqq (o''_{\tau,j})_{j \in [Q_x]}$. By construction, it holds:

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

which proves the first part of the lemma, i.e., computing $\boldsymbol{o}''_{\tau,j}$ satisfying Eq. (14).

In what follows, we prove that if the event that $\mathbf{H}_6$ outputs 0 but $\mathbf{H}_5$ would output 1, namely that all the conditions of $\mathbf{H}_5$ hold but there is an index k such that $\boldsymbol{o}_{\mathsf{c}_k^*} \neq \mathbf{0} \wedge \boldsymbol{o}''_{\mathsf{c}_k^*} = \mathbf{0}$, then we can break the Aff-MDH assumption.

First, notice that for any j and k $\boldsymbol{o}_{\mathsf{c}_k^*,j} \neq \mathbf{0}$ implies that $\boldsymbol{o}'_{\mathsf{c}_k^*,j} \neq \mathbf{0}$, because the linear transformation applied to compute $\boldsymbol{o}'_{\mathsf{c}_k^*,j}$ is full rank.

Second, take an index j^* such that $\boldsymbol{o}_{\mathsf{c}_k^*,j^*} \neq \mathbf{0}$ and set $\boldsymbol{A} \leftarrow \boldsymbol{o}'_{\mathsf{c}_k^*,j^*}$ and $\zeta \leftarrow \langle \boldsymbol{z}_{j^*}, \boldsymbol{o}'_{\mathsf{c}_k^*,j^*} \rangle$.

By the above definition of the values o''_{c^*,j^*} and our assumption that the "bad event" of this hybrid is $\boldsymbol{o}''_{\mathsf{c}_k^*,} = \mathbf{0}$, we have that:

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

The reduction $\mathcal{B}$ to the $\mathcal{D}$-Aff-MDH Assumption takes as input a distribution $[\boldsymbol{\gamma}]_1$ and runs the experiment as in $\mathbf{H}_5$ (it perfectly emulates $\mathbf{H}_5$, and in particular the simulation oracle, because it knows the trapdoor s "at the exponent"). Then $\mathcal{B}$ computes the coefficients $(A_i)_{i \in [Q_c]}$ and the value ζ as described above, which is a valid $\mathcal{D}$-Aff-MDH solution. □

Hybrid $\mathbf{H}_7$. This hybrid proceeds as the previous one except that we abort in the case the representation vectors of the commitments in the forgery are different from the representations provided for the same commitments when querying

the batching random oracle. More precisely, let $\mathbb{x}^*$ be the statement returned by the adversary at the end of its execution and along with the representation vectors $\{\boldsymbol{r}_{c_i^*}\}_i$ of the commitments $(\mathsf{c}_i^*)_i$. Look for the tuple $(\mathsf{batch}\|\mathbb{x}^*, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$ (this tuple must exist w.l.o.g. since otherwise we can always define another adversary that makes this query at the very last) and let $\tilde{\boldsymbol{r}}_{c_i^*}$ be the representation vectors of the commitments in aux. If there exists an index i such that $\boldsymbol{r}_{c_i^*} \neq \tilde{\boldsymbol{r}}_{c_i^*}$ the adversary loses the game.

Lemma 24. *For any PPT* $\mathcal{A}_{\mathsf{c}}$, $\Pr[\boldsymbol{H}_7(\mathcal{A}_{\mathsf{c}})] \leq \Pr[\boldsymbol{H}_6(\mathcal{A}_{\mathsf{c}})] + \epsilon_{(Q_x+d+1)\text{-}\mathsf{DL}}(\lambda) + \epsilon_{\textit{Aff-MDH}}$.

Proof. Let c^* be a commitment of the forgery with two different representations $\boldsymbol{r}_{\mathsf{c}^*} \neq \tilde{\boldsymbol{r}}_{\mathsf{c}^*}$. By definition of core adversary we have that $\boldsymbol{f}_{c^*} = \tilde{\boldsymbol{f}}_{c^*} = 0$. Thus we have

$$\langle (\boldsymbol{c}_{\mathsf{c}^*} - \tilde{\boldsymbol{c}}_{\mathsf{c}^*}), \boldsymbol{\gamma} p(s) \rangle + \langle (\boldsymbol{o}_{\mathsf{c}^*} - \tilde{\boldsymbol{o}}_{\mathsf{c}^*}), (p_j(s))_j \rangle = \langle \boldsymbol{\Delta}_c, \boldsymbol{\gamma} p(s) \rangle + \langle \boldsymbol{\Delta}_0, (p_j(s))_j \rangle = 0$$

Similarly to previous cases, under the $(Q_x + d + 1)$-DL assumption we can write the equation above symbolically as

$$\langle \boldsymbol{\Delta}_c, \boldsymbol{\gamma} \rangle p(X) + \langle \boldsymbol{\Delta}_0, (p_j(X))_j \rangle = 0$$

which, by looking at the degree of $p(X)$ and $p_j(X)$ and the linear independence of the latter, implies that $\langle \boldsymbol{\Delta}_c, \boldsymbol{\gamma} \rangle = 0$ and $\boldsymbol{\Delta}_o = \boldsymbol{0}$. However, the event that $\langle \boldsymbol{\Delta}_c, \boldsymbol{\gamma} \rangle = 0$ for a nonzero $\boldsymbol{\Delta}_c$ can be reduced to the Aff-MDH assumption.

Hybrid $\mathbf{H}_8$. The hybrid $\mathbf{H}_8$ additionally checks if there exists an index $k \in [m]$ such that $\boldsymbol{r}_{\mathsf{c}_k^*} \neq \boldsymbol{0}$, and if the condition holds the adversary $\mathcal{A}_{\mathsf{c}}$ loses the game.

Lemma 25. *For any PPT* $\mathcal{A}_{\mathsf{c}}$,

$$\Pr[\boldsymbol{H}_8(\mathcal{A}_{\mathsf{c}})] \leq \Pr[\boldsymbol{H}_7(\mathcal{A}_{\mathsf{c}})] + \epsilon_{\textit{Aff-MDH}} + 2\epsilon_{(Q_x+1+d)\text{-}\mathsf{DL}} + \mathsf{poly}(\lambda)\frac{\mathtt{deg}(h)}{q}.$$

Proof. We bound the probability that the adversary loses in $\mathbf{H}_8$ but not in $\mathbf{H}_7$, namely, the probability that $\exists k : \boldsymbol{r}_{\mathsf{c}_k^*} \neq \boldsymbol{0}$ but the conditions of $\mathbf{H}_7$ hold. The main idea is to do a reduction $\mathcal{B}$ to the Aff-MDH when this event happens.

First of all, notice that by the changes introduced earlier we can assume that the core adversary outputs commitments $(\mathsf{c}_\ell^*)_\ell$ and proof π whose representations are such that $\boldsymbol{f}_{\mathsf{c}_\ell^*} = \boldsymbol{f}_{\pi^*} = \boldsymbol{c}_{\pi^*} = \boldsymbol{0}$, i.e. the adversary only makes use of previous commitments in coms and simulated proofs in proofs to represent each c_ℓ^*, and only uses the simulated proofs to represent the proof π^*.

The reduction $\mathcal{B}$ takes as input a distribution $[\boldsymbol{\gamma}]_1$ and runs the experiment as in $\mathbf{H}_7$. $\mathcal{B}$ aborts if the forgery $\mathbb{x}^* = (x^*, (\mathsf{c}_\ell^*, y_\ell^*)_{\ell \in [m]}), \pi^*$ returned by the adversary is not valid (i.e. either the extraction predicate or the verification equation is not satisfied) or if $\boldsymbol{r}_{\mathsf{c}_\ell^*} = \boldsymbol{0}$ for all ℓ. Otherwise, we have that:

$$e(\sum_\ell \rho^{\ell-1} \mathsf{c}_\ell^* - \left[\sum_i \rho^{\ell-1} y_\ell^*\right]_1, [1]_2) = e(\pi^*, [s - x^*]_2) \text{ and } \exists k : \boldsymbol{r}_{\mathsf{c}_k^*} \neq \boldsymbol{0}$$

where $\boldsymbol{r}_{\mathsf{c}^*_k} \neq \mathbf{0}$ if $\boldsymbol{o}_{\mathsf{c}^*_k} \neq \mathbf{0} \vee \boldsymbol{c}_{\mathsf{c}^*_k} \neq \mathbf{0}$.

We can rewrite each commitment and the proof in the forgery as functions of the coefficients $\boldsymbol{o}''_{\mathsf{c}^*_\ell}$ and $\boldsymbol{o}''_{\pi^*}$ (as computed in the previous hybrid):

$$\begin{aligned}\mathsf{c}^*_\ell &\coloneqq \left[\langle \boldsymbol{c}_{\mathsf{c}^*_\ell}, \boldsymbol{\gamma} p(s)\rangle + \langle \boldsymbol{o}''_{\mathsf{c}^*_\ell}, (p_j(s))_j\rangle\right]_1 \\ \pi^* &\coloneqq \left[\langle \boldsymbol{o}''_{\pi^*}, (p_j(s))_j\rangle\right]_1\end{aligned}$$

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

$$\begin{aligned}\langle \sum_\ell \rho^{\ell-1} \cdot \boldsymbol{c}_{\mathsf{c}^*_\ell}, \boldsymbol{\gamma} p(s)\rangle + \langle \sum_\ell \rho^{\ell-1} \cdot \boldsymbol{o}''_{\mathsf{c}^*_\ell}, (p_j(s))_j\rangle - p(s) \sum_\ell \rho^{\ell-1} \cdot y^*_\ell \\ = \langle \boldsymbol{o}''_{\pi^*}, (p_j(s))_j\rangle (s - x^*) \qquad (17)\end{aligned}$$

For all $j \in [Q_x]$, let $\delta_j \coloneqq x_j - x^* \neq 0$. We can rewrite the r.h.s. of Eq. (17) as:

$$\begin{aligned}\langle \boldsymbol{o}''_{\pi^*}, (p_j(s)(s - x^*))_j\rangle &= \langle \boldsymbol{o}''_{\pi^*}, (p_j(s)((s - x_j) + \delta_j))_j\rangle \\ &= \langle \boldsymbol{o}''_{\pi^*}, (p(s) + p_j(s)\delta_j))_j\rangle\end{aligned}$$

In Eq. (17), we group all the terms that depend on $p(s)$ on the left side, and we move all the terms that depend on $p_j(s)$ to the right side, thus obtaining:

$$H \cdot p(s) = \sum_{j \in [Q_x]} L_j \cdot p_j(s) \qquad (18)$$

where

$$\begin{aligned}H &\coloneqq \langle \sum_\ell \rho^{\ell-1} \cdot \boldsymbol{c}_{\mathsf{c}^*_\ell}, \boldsymbol{\gamma}\rangle - \sum_j o''_{\pi^*,j} - \sum_\ell \rho^{\ell-1} \cdot y^*_\ell \\ L_j &\coloneqq o''_{\pi^*,j}\, \delta_j - \sum_\ell \rho^{\ell-1} o''_{\mathsf{c}^*_\ell,j}.\end{aligned}$$

Let $v(X) \coloneqq Hp(X) - \sum_{j \in [Q_x]} L_j p_j(X)$. Notice that because of Eq. (18) we have $v(s) = 0$, thus we can assume $v(X) \equiv 0$, as otherwise we can reduce, by Lemma 1, to the $(Q_x + d + 1)$-DL assumption. It must be the case that both $\sum_{j \in [Q_x]} L_j p_j(X) = 0$ and $H = 0$ because the degrees of $p(X)$ and $p_j(X)$, for all j, are different. Moreover, the polynomials $p_j(X)$ are linearly independent, namely the only linear combination $\sum_j a_j p_j(X) = 0$ is the trivial one where the coefficients $a_j = 0$[21], thus $L_j = 0$ for all j. From this, we have that:

$$\forall j: \quad o''_{\pi^*,j} = \frac{-\sum_\ell \rho^{\ell-1} o''_{\mathsf{c}^*_\ell,j}}{\delta_j}.$$

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

Since H must be 0, we have that:

$$\sum_{\ell} \rho^{\ell-1} \sum_{i\in[Q_c]} c_{\mathsf{c}^*_\ell,i}\gamma_i - \sum_{j\in[Q_x]} \frac{\sum_\ell \rho^{\ell-1} o''_{\mathsf{c}^*_\ell,j}}{\delta_j} - \sum_{\ell} \rho^{\ell-1}\cdot y^*_\ell = 0. \tag{19}$$

$\mathcal{B}$ can derive from Eq. (19) that:

$$\begin{aligned}
0 &= \sum_{i\in[Q_c]} \sum_{\ell} \rho^{\ell-1} c_{\mathsf{c}^*_\ell,i}\gamma_i - \sum_{j\in[Q_x]} \frac{\sum_\ell \rho^{\ell-1} o''_{\mathsf{c}^*_\ell,j}}{\delta_j} - \sum_{\ell} \rho^{\ell-1} y^*_\ell \\
&= \sum_{i\in[Q_c]} \sum_{\ell} \rho^{\ell-1} c_{\mathsf{c}^*_\ell,i}\gamma_i - \sum_{i,j,\ell} \frac{\rho^{\ell-1} o'_{\mathsf{c}^*_\ell,j,i}(\gamma_i - z_{j,i})}{\delta_j} - \sum_{\ell} \rho^{\ell-1} y^*_\ell \\
&= \sum_{i\in[Q_c]} \underbrace{\left(\sum_{\ell} \rho^{\ell-1} \underbrace{(c_{\mathsf{c}^*_\ell,i} - \sum_{j} \frac{o'_{\mathsf{c}^*_\ell,j,i}}{\delta_j})}_{A_{i,\ell}} \right)}_{A_i} \gamma_i + \underbrace{\sum_{i,j,\ell} \frac{\rho^{\ell-1} o'_{\mathsf{c}^*_\ell,i,j} z_{j,i}}{\delta_j} - \rho^{\ell-1} y^*_\ell}_{-z}
\end{aligned}$$

Above, the second equation comes directly from the definition of the coefficients $o''_{\tau,j}$, and in the last step we have grouped the terms depending on γ_i. In particular, the last equation shows that $\mathcal{B}$ can make a forgery in the Aff-MDH game since it knows z and all the coefficients A_i such that:

$$\sum_{i\in[Q_c]} A_i \left[\gamma_i\right]_1 = \left[z\right]_1 .$$

For this to be a valid solution in the Aff-MDH game we need the existence of at least an index i such that $A_i \neq 0$. We show that this occurs with all but negligible probability, i.e., there is a negligible value ν such that $\Pr[\exists i \in [Q_c] : A_i \neq 0] \geq 1-\nu$.

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

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

Next, let k be the index such that $\boldsymbol{r}_{\mathsf{c}^*_k} \neq 0$. We have

$$\Pr[A_\mu = 0] \leq \Pr[A_\mu = 0 | A_{\mu,k} \neq 0] + \Pr[A_{\mu,k} = 0]$$

Below, we argue that $\Pr[A_\mu = 0 | A_{\mu,k} \neq 0]$ is negligible based on the randomness of ρ, which can be shown to be chosen by the random oracle after all the coefficients $A_{\mu,\ell}$ are defined, and that $\Pr[A_{\mu,k} = 0]$ is negligible based on the randomness of x^*, which also in this case is chosen by the random oracle after defining $A_{\mu,k}$. Furthermore, for the latter, we make use of the assumption that $\boldsymbol{r}_{\mathsf{c}^*_k} \neq 0$.

For every ℓ, we claim that the values $A_{\mu,\ell} = c_{\mathsf{c}^*_\ell,\mu} - \sum_j \frac{o'_{\mathsf{c}^*_\ell,j,\mu}}{(x^* - x_j)}$ can be fixed before the random oracle query $\mathsf{batch}\|\mathbb{x}^*$ is made. To this end, we start by showing that $\boldsymbol{o}'_{\mathsf{c}^*_\ell,j}$ does not depend on x^*. Let $B(j) \subseteq [Q_c]$ be the subset of indices of the simulation queries that involve x_j and that occurred before the random oracle query that returned x^*. We observe that for every $\eta \in B(j)$ it must be $o_{\mathsf{c}^*_\ell,j,\eta} = 0$ since the simulated proof $\pi_{j,\eta}$ is not in the view of the adversary. Therefore, we have

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

and observe that all the rows of $\boldsymbol{\Gamma}_j$ belonging to $B(j)$ can all be defined before x^* is sampled.

Hence, we have that each $A_{\mu,\ell}$ depends on the values $\boldsymbol{c}_{\mathsf{c}^*_\ell}, x^*, \{x_j\}_j$, and $\boldsymbol{o}_{\mathsf{c}^*_\ell,j}$ which can all be defined before the random oracle query $\mathsf{batch}\|\mathbb{x}^*$ is made.

Therefore, assuming the existence of at least a nonzero $A_{\mu,k} \neq 0$ we have that $A_\mu = \sum_{\ell=1}^{m} \rho^{\ell-1} A_{\mu,\ell} = 0$ is at most m/q over the random choice of ρ.

Next, we bound $\Pr[A_{\mu,k} = 0]$. Recall that, since the extractor policy Φ_{ext} holds true, we have that $x^* = h(a)$ and $(s, \mathsf{aux}, a) \in \mathcal{Q}_{\mathsf{RO}}$ where $g_c(\mathsf{c}^*_{\mathsf{k}}, s) = 1$ and the function h is the polynomial encoded in aux_ϕ. Moreover, by the AGM, since $\mathcal{A}_{\mathsf{c}}$ sends a query s (where c^*_k is encoded in s) to the random oracle it also defines coefficients for $\mathsf{c}^*{}_k$ before the value a, and therefore $x^* = h(a)$, is defined. Also, it is not hard to see that the representation vector of c^*_k defined by $\mathcal{A}_{\mathsf{c}}$ when querying the random oracle must be the same representation vector used for the forgery. As otherwise we would break the $(Q_x + d + 1)$-DL assumption. Thus the coefficients $\boldsymbol{c}_{\mathsf{c}^*_k}$ and $\boldsymbol{o}'_{\mathsf{c}^*_k,j}$ are defined by the adversary before seeing the random value x^*.

Notice that, once the coefficients $\boldsymbol{c}_{\mathsf{c}^*_k}$ and $\boldsymbol{o}'_{\mathsf{c}^*_k,j}$ are fixed, the coefficient $A_{\mu,k}$ can be seen as function of $x^* \in \mathbb{Z}_q$, i.e. $A_{\mu,k} = A_{\mu,k}(x^*)$, where

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

Notice that $A_{\mu,k}(X)(\prod_j (X - x_j))$ vanishes in at most Q_x points in $\mathbb{F} \setminus \mathcal{Q}_x$ and vanishes in the set of points $\mathcal{Q}_x$. Let $\mathcal{R}$ be the set of the roots of such a polynomial, since h is defined before x^* is computed, and by union bound:

$$\Pr[h(\mathsf{RO}(s)) \in \mathcal{R}] \leq \sum_{r \in \mathcal{R}} \Pr[h(\mathsf{RO}(s)) = r] \leq Q_x \tfrac{\mathsf{deg}(h)}{q}$$

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

Hybrid $\mathbf{H}_9$. This hybrid proceeds as the previous one except that the adversary loses if the forgery triggers the following event:

$$\sum_{\ell \in [m]} \rho^{\ell-1} y^*_\ell = 0 \text{ and } \exists k : y^*_k \neq 0$$

Lemma 26. *For any PPT $\mathcal{A}_{\mathsf{c}}$ we have* $\Pr[\boldsymbol{H}_9(\mathcal{A}_{\mathsf{c}})] \leq \Pr[\boldsymbol{H}_8(\mathcal{A}_{\mathsf{c}})] + m/q$.

The proof follows by observing that, since all the outputs $(y^*_\ell)_\ell$ are fixed in the input of the query $(\mathsf{batch} \| \mathbb{x}^*)$ that returns ρ, we can apply the Schwartz-Zippel lemma.

Hybrid $\mathbf{H}_{10}$. The hybrid $\mathbf{H}_{10}$ additionally checks if there exists k such that $y^*_k \neq 0$, and if the condition holds the adversary $\mathcal{A}_{\mathsf{c}}$ loses the game.

Lemma 27. *For any PPT adversary $\mathcal{A}_{\mathsf{c}}$ we have:*

$$\Pr[\boldsymbol{H}_{10}(\mathcal{A}_{\mathsf{c}})] \leq \Pr[\boldsymbol{H}_9(\mathcal{A}_{\mathsf{c}})] + \epsilon_{(Q_x+1+d)\text{-}\mathsf{DL}} + \mathsf{poly}(\lambda)\frac{\mathsf{deg}(d)}{q}$$

Proof. We reduce to the evaluation binding of KZG polynomial commitment for polynomials of maximum degree $Q_x + 1 + d$, which, in turn, can be reduced to $(Q_x + 1 + d)$-strong Discrete Log assumption. Let $\mathcal{B}$ be the reduction that upon input $\mathsf{pp}_{\mathbb{G}}, \mathsf{ck} = \left[1, s, \ldots, s^{Q_x+d+2}\right]_1, [1, s]_2$ simulates experiment $\mathbf{H}_9$ for the adversary $\mathcal{A}_{\mathsf{c}}$. Eventually, $\mathcal{A}_{\mathsf{c}}$ outputs its forgery $(x^*, (\mathsf{c}^*_\ell, y^*_\ell)_\ell, \pi^*)$, and $\mathcal{B}$ aborts if $y^*_\ell = 0$ for all ℓ. Else, the reduction sets $y^* \leftarrow \sum_{\ell \in [m]} \rho^{\ell-1} y^*_\ell$, $\tilde{f}(X) \coloneqq -y^* p(X)$, $y \coloneqq \tilde{f}(x^*)$, and computes π to be a valid KZG-proof for $(\left[\tilde{f}(s)\right]_1, x^*, y)$. The forgery against evaluation biding of the reduction is set to be (y, π) and $(0, \pi^*)$ for the commitment $\left[\tilde{f}(s)\right]_1$ on the point x^*.

We need to show that (1) $(\left[\tilde{f}(s)\right]_1, x^*, 0, \pi^*)$ passes the verification equation of KZG commitment where the commitment key is set to ck and that (2) $y \neq 0$. For the first item notice that, by the definition of core adversary, we have that $\boldsymbol{r}_{c^*_k} = \mathbf{0}$ thus $c^* = [0]_1$. Therefore, by the verification equation:

$$e([0]_1 - y^* [p(s)]_1, [1]_2) = e(\left[\tilde{f}(s)\right]_1 - 0\,[1]_1, [1]_2) = e(\pi^*, [s]_2 - x^* [1]_2).$$

For the second item, first notice that by the change in the previous hybrid we have that $y^* \neq 0$. Second, notice that $\tilde{f}(x^*) = 0$ if and only if x^* is a root of $p(X)$, i.e. $x^* \in \mathcal{Q}_x$ or $x^* = x_r$. Thus, similarly to the previous lemma, by the assumption on h and by union bound:

$$\Pr[h(\mathsf{RO}(s)) \in \mathcal{Q}_x \cup \{x_r\}] \leq Q_{\mathsf{RO}}(Q_x + 1)\frac{\mathsf{deg}(d)}{q}.$$

□

Finally, we have that the probability that the adversary wins in $\mathbf{H}_{10}$ is null. Indeed, the canonical extractor $\mathcal{E}$ outputs the 0 polynomial, moreover because of the condition introduced in $\mathbf{H}_8$, we have $\mathsf{c}^* = [0]_1$, and because of the condition introduced in $\mathbf{H}_{10}$ we have $y^* = 0$, thus the witness extracted is valid for the instance $\mathbb{x}^* = (x^*, ([\mathbf{0}]_1, \mathbf{0}))$. □

Finally, we prove that for any algebraic adversary $\mathcal{A}$ whose forgery satisfies the predicate $\Phi^{\text{der}}_{\mathsf{ext}}$, there exists an algebraic adversary $\mathcal{B}$ whose forgery satisfies the predicate $\Phi^{\text{rnd}}_{\mathsf{ext}}$.

Lemma 28. *For any algebraic adversary $\mathcal{A}$ there exists an algebraic adversary $\mathcal{B}$ such that:*

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

Proof. The reduction $\mathcal{B}$ internally runs $\mathcal{A}$ forwarding all the queries, up to the forgery $(\mathbb{x}^*, \pi^*)$, where $\mathbb{x}^* = (x^*, (\mathsf{c}_i^*, y_i^*)_i)$. If the simulation queries and/or the forgery of the adversary $\mathcal{A}$ do not agree with the policy $\Phi_{\mathcal{D}}$, i.e. $\mathcal{A}$ automatically loses its game, $\mathcal{B}$ aborts. Otherwise, it must be true that the forgery of $\mathcal{A}$ either (i) satisfies the extraction predicate $\Phi^{\text{rnd}}_{\mathsf{ext}}$ or (ii) satisfies the extraction predicate $\Phi^{\text{der}}_{\mathsf{ext}}$. Both cases can be efficiently checked by $\mathcal{B}$. In case (i) $\mathcal{B}$ would simply forward the forgery of $\mathcal{A}$ retaining the same advantage of $\mathcal{A}$. Otherwise, before submitting the forgery, $\mathcal{B}$ retrieves from $\mathcal{Q}_{\mathsf{sim}}$ the statement $\mathbb{x} = (x^*, (\mathsf{c}_i^*, y_i)_i)$, where there exists an index i such that $y_i \neq y_i^*$. Let π_i be a proof for the "single-eval" statement $\mathbb{x}_i = (x, (c_i, y_i))$: these additional simulation queries submitted to $\mathcal{S}_1$ do not alter the consistency check, because they are consistent with the simulation for the statement $\mathbb{x}$. Also, let $\rho \leftarrow \mathsf{RO}(\mathsf{batch} \| \mathbb{x}^*)$ be the coefficient used for batch-verify the proof π^*, and let $\pi \coloneqq \sum_i \rho^{i-1}\pi_i$. We can define:

$$\begin{aligned}
\hat{\mathsf{c}} &\leftarrow \pi^* - \pi \\
\hat{x} &\leftarrow h(a) \\
\hat{y} &\leftarrow \frac{\sum_i \rho^{i-1}(y_i - y_i^*)}{\hat{x} - x} \\
\hat{\pi} &\leftarrow \frac{\pi - \pi^*}{\hat{x} - x}
\end{aligned}$$

where $(\hat{c}, h, a) \in \mathcal{Q}_{\mathsf{RO}}$. We need to prove that $\hat{\mathbb{x}} \coloneqq (\hat{x}, \hat{c}, \hat{y})$ and $\hat{\pi}$ satisfy the (single-eval) verification equation:

$$\hat{\mathsf{c}} - [\hat{y}]_1 = \hat{\pi}([s - x]_2)$$

We notice that the probability that $\exists i : y_i \neq y_i^*$ but $\hat{y}_1 \coloneqq \sum_i \rho^{i-1}(y_i - y_i^*) = 0$ is only negligible; this is because ρ is chosen uniformly at random and after the values y_i^* and the values y_i are determined by the simulation queries made up to the forgery. Then, the correctness of the above forgery follows from Remark 1, where for the same commitment $\sum_i \rho^{i-1}\mathsf{c}_i$ we have two valid openings $(\sum_i \rho^{i-1} y_i, \pi)$ and $(\sum_i \rho^{i-1} y_i^*, \pi^*)$ on the point x^*.