Witness-Succinct Universally-Composable SNARKs[⋆]

Chaya Ganesh[1], Yashvanth Kondi[2], Claudio Orlandi[2], Mahak Pancholi[2], Akira Takahashi[3], and Daniel Tschudi[4]

[1] Indian Institute of Science
chaya@iisc.ac.in
[2] Aarhus University
{ykondi,orlandi,mahakp}@cs.au.dk
[3] University of Edinburgh
takahashi.akira.58s@gmail.com
[4] Concordium
dt@concordium.com

April 26, 2023

Abstract. Zero-knowledge Succinct Non-interactive ARguments of Knowledge (zkSNARKs) are becoming an increasingly fundamental tool in many real-world applications where the proof compactness is of the utmost importance, including blockchains. A proof of security for SNARKs in the Universal Composability (UC) framework (Canetti, FOCS'01) would rule out devastating malleability attacks. To retain security of SNARKs in the UC model, one must show their *simulation-extractability* such that the knowledge extractor is both *black-box* and *straight-line*, which would imply that proofs generated by honest provers are *non-malleable*. However, existing simulation-extractability results on SNARKs either lack some of these properties, or alternatively have to sacrifice *witness succinctness* to prove UC security.

In this paper, we provide a compiler lifting any simulation-extractable NIZKAoK into a UC-secure one in the global random oracle model, importantly, while preserving the same level of witness succinctness. Combining this with existing zkSNARKs, we achieve, to the best of our knowledge, the first zkSNARKs simultaneously achieving UC-security and constant sized proofs.

[⋆] An extended abstract appeared at Eurocrypt 2023. This is the full version.

Table of Contents

1 Introduction

The UC framework and UC Secure NIZKs. The Universal Composability (UC) framework [Can01] allows for the modular design and analysis of complex cryptographic protocols, and guarantees security in the presence of arbitrarily many sessions running concurrently. The *environment* $\mathcal{Z}$ (representing everything that is external to the execution of the protocol of interest) interacts with the protocol, at the conclusion of which it outputs a decision bit, indicating whether it thinks it has interacted with a "real-life" adversary $\mathcal{A}$ and parties running the protocol, or with an "ideal-process" adversary (or *simulator*) Sim and parties accessing the so-called *ideal functionality* $\mathcal{F}$ specifying the ideal outcome of a given protocol.

This paper focuses on non-interactive zero-knowledge proofs (NIZK) [BFM90, BSMP91] in the UC framework. In the standalone setting, security of NIZKs is guaranteed by showing standard properties separately such as completeness, zero-knowledge, and (knowledge) soundness under some setup assumptions, like a common reference string (CRS) or the Random Oracle Model (ROM). However, several restrictions and stronger properties come into play once the NIZK functionality is to be realized in the UC model. A common methodology to design NIZKs in the ROM is to start with an interactive argument which is proven ZK/knowledge sound, and then compile this interactive argument into a non-interactive proof. This means that NIZKs that are proven secure using rewinding (either for ZK or for extraction) are at odds with UC, because the environment $\mathcal{Z}$ is an interactive distinguisher between the real execution protocol and the ideal process, and therefore a simulator Sim in the security proof cannot rewind $\mathcal{Z}$. Thus, *straight-line* simulation and extraction are required for a NIZK to be UC secure. Informally, a proof system is straight-line extractable if one can efficiently extract a valid witness without interacting with any successful prover. On top of extraction being straight-line, by definition, UC simulators must be able to obtain a witness having only *black-box* access to $\mathcal{Z}$, i.e., without knowing the concrete code of $\mathcal{Z}$.

Another important ingredient to realize UC security is *non-malleability (NM)* [DDN91], which is often referred to as *simulation-extractability* in context of UC (NI)ZK [Sah99, DDO$^+$01, PR05, GMY06, JP14, FKMV12]. Essentially, a malleability attack allows an adversary to maul existing proofs observed during the protocol execution, and to forge a proof on some statement for which they do not know the corresponding witness. Preventing such attacks is crucial in the UC model: as $\mathcal{Z}$ may ask uncorrupted provers or simulator to produce proofs on arbitrary statement-witness pairs, the ability to maul such proofs will induce the simulation to fail (i.e., Sim fails to extract a witness) and thus helps $\mathcal{Z}$ distinguish the real execution from the ideal one. The non-malleable NIZK construction of [DDO$^+$01] was shown to be UC secure in [CLOS02]. Subsequently, [GOS06, Gro06] constructed UC secure NIZKs in the presence of adaptive adversaries, and [Gro06] proved that simulation-extractability is necessary for UC. In sum, black-box extraction (BBE), straight-line extraction (SLE) and simulation-extractability (SIMEXT) are the properties a NIZK must satisfy in order to be UC secure.

We now discuss UC security for SNARKs (*succinct* non-interactive arguments[5] of knowledge) where the communication is sublinear (ideally polylogarithmic or constant[6]) in the size of the non-deterministic witness used to verify the relation. A SNARK is *circuit-succinct* if the proof size is sublinear only in the size of the circuit representing the statement; if it is sublinear in the length of the witness too, it is *witness-succinct*. Many SNARK constructions in the literature rely on *knowledge assumptions* to prove witness extraction, i.e. their extractors rely on examining the concrete code of the adversary in order to extract a witness. As discussed earlier, this is a barrier to achieving UC security, as simulation in the UC framework can not depend on the code of the environment.

One simple folklore method to obtain UC-secure circuit-succinct NIZK given a SNARG (a SNARK that only guarantees soundness, not proof-of-knowledge) and a (perfectly correct) public key encryption scheme is the following: a public key pk serves as a common reference string, given which the prover computes a ciphertext ct to encrypt the witness w under randomness r. The prover then computes a SNARG π that proves that the message encrypted by ct is indeed a witness to the statement, and outputs (ct, π). This tuple now constitutes a straight-line extractable NIZK, as the extractor (given sk) can simply decrypt ct to obtain w—intuitively this w must be a valid witness since ct is a perfectly binding commitment to w, and so if w is not a valid witness then π would be proving a false theorem. Notice that this proof additionally inherits the *circuit succinctness* property of the SNARG, as ct is of

[5] Argument systems are proofs where soundness is computational. For proofs to be shorter than the length of the witness, restricting to arguments is necessary [GH98, GVW02].
[6] Polynomial only in the security parameter.

size $O(|w|)$ and π is the SNARG itself. This approach was described by De Santis et al. [DDO+01] in the context of lifting ordinary NIZK to simulation-sound NIZK, and implemented as part of the C∅C∅ framework for circuit-succinct UC NIZK by Kosba et al. [KZM+15], with optimizations for concrete efficiency using the state-of-the-art SNARKs at the time. C∅C∅ further proposed an optimized method to obtain non-malleability, by additionally proving that the encrypted string is a valid signature on the statement. Putting all these features together, C∅C∅ serves as the first *generic* UC lifting compiler preserving circuit succinctness.

A major limitation of this technique is that it is inherently limited to producing proofs that are at least as large as the witness, by virtue of the witness having to be 'decryptable' from the ciphertext. Constructing *witness-succinct* proofs that enjoy black-box straight-line extraction appears to require a fundamentally different approach. Indeed, Kosba et al. remarked that there is "no known UC-secure zero-knowledge proof construction that is circuit *and* witness-succinct, even under non-standard assumptions" [KZM+15, pg. 2], and left open the question of whether such an object is even feasible to construct. Given this, one may ask:

Is it possible to obtain UC-secure witness-succinct NIZKs
under well-studied setup assumptions?

The requirement of "well-studied" setup assumptions is meant to capture those forms of setup that have generally accepted realizations. In this work, we consider the *common reference string* (CRS) model, and the *random oracle model* (ROM) to fall within the scope of well-studied setup. For SNARKs in particular, there is already established infrastructure to generate the CRSs required (via so called "powers of tau" ceremonies implemented by major blockchains such as ZCash, FileCoin, etc. [BGG19]). There are also established heuristics to instantiate the ROM in practice with carefully chosen hash functions, and the ROM itself is arguably amongst the oldest and most comprehensively studied idealized models [BR93].

Models we do *not* consider. Several SNARK constructions are known to be secure with non-black-box extraction under knowledge assumptions, or in idealized models such as the Generic Group Model (GGM) or Algebraic Group Model (AGM). The UC-AGM framework [ABK+21] allows to model the AGM and algebraic adversaries in a composable fashion. However, doing so requires the use of algebraic environments making it incompatible with standard UC. The other related alternative model is considered in [KKK21] where they formally define the concept of *knowledge-respecting distinguishing environments*, enabling the usage of primitives relying on knowledge assumptions in larger protocols. However, their entire formalization is built on top of a different compositional framework [Mau11] than UC. Similar to the UC-AGM framework, distinguishers in their model are globally assumed to explain how they computed each knowledge-implying object they output, making themselves weaker than environments in the standard UC.

Succinct Arguments of Knowledge with a CRS alone. Folklore has long held that NIZKs in the CRS model with black-box straight-line extraction cannot be witness-succinct, as the witness must be 'decryptable' from the proof string as in the simple approach described earlier. Indeed, all pairing based efficient SNARKs that are witness-succinct in the standard model with a CRS (like [GGPR13, PHGR13]) are not black-box extractable[7]. The intuition is that for a language whose witnesses have enough entropy, an argument that is too "short" cannot contain enough information about a witness: this makes extraction impossible for an extractor that does not have any additional power, like access to the prover's randomness (like in non-black-box extractors) or the ability to rewind the prover (like in interactive arguments and resulting NIZKs compiled in the ROM). We refer the reader to the recent work of Campanelli et al. [CGKS22] for a formal treatment of this. Given that black-box extraction is necessary for UC security, we consider it justified to consider UC security in the ROM in light of this impossibility.

Succinct Arguments of Knowledge in the ROM. There are several witness-succinct proof systems in the ROM in the literature such as the classical Probabilistically Checkable Proofs (PCP) based approach of Kilian [Kil92], Micali's CS proofs [Mic00], and the recent works on Interactive Oracle Proofs [BCS16]. However to our knowledge, there are no witness-succinct proof systems in the ROM that have been formally analyzed in the UC framework. While some of these constructions [Mic00, AHIV17, BSBHR18] are black-box straight-line extractable, simulation-extractability of these has not been shown. SNARKs in the ROM that are logarithmic in the statement and wit-

[7] Pairing based constructions like PLONK, Sonic, Marlin are not black-box extractable as well, but they are also in the ROM in addition to requiring a CRS.

Scheme	Assumption	Model	Transparent	BBE	SLE	SIMEXT
STARK [BBHR18]	ROM	ROM	✓	✓	✓	unknown
Aurora [BCR+19]	ROM	ROM	✓	✓	✓	unknown
RedShift [KPV19]	ROM	ROM	✓	✓	✓	unknown
Bulletproofs [BBB+18]	DLOG	ROM	✓	✓	✗	✓ [GOP+23]
SONIC [MBKM19]	AGM & q-DLOG	CRS & ROM	✗	✗	✓	✓ [GKK+22]
PLONK [GWC19]	AGM	CRS & ROM	✗	✗	✓	✓ [GKK+22]
Marlin [CHM+20]	AGM	CRS & ROM	✗	✗	✓	✓ [GKK+22]
Groth16 [Gro16]	GGM	CRS(& ROM for NM)	✗	✗	✓	✓ [BG18]
Groth-Maller [GM17]	XPKE & Poly	CRS	✗	✗	✓	✓
LAMASSU [ARS20]	q-MC & q-MK & BDH & DL	CRS	✗	✗	✓	✓
Ours + [GM17] + [KZG10]	XPKE & Poly & SDH	CRS & GROM	✗	✓	✓	✓
Ours + [ARS20] + [KZG10]	q-MC & q-MK & BDH & DL & SDH	CRS & GROM	✗	✓	✓	✓

Table 1: Known properties of existing (witness-succinct) zkSNARKs compared to example instantiation of our compilation. "BBE" stands for black-box knowledge extraction; "SLE" for straight-line knowledge extractor; "SIMEXT" for simulation-extractability. We say a proof system is "transparent" if no trusted generation of CRS is required. Note that the assumptions for the last row are derived from an example instantiation of [ARS20, Theorem 4] where they adapt [GKM+18] as an underlying SNARK.

ness size are known from conservative computational assumptions such as the hardness of computing discrete logarithms [BCC+16, BBB+18] in the standalone setting. Bulletproofs [BBB+18] are known to be simulation-extractable, but currently only in the AGM+ROM [GOP+22] or in the ROM with rewinding [GOP+23, DG23]. If a CRS is assumed in addition to ROM, then constructions like PLONK, Sonic, and Marlin also provide constant sized (polynomial only in the security parameter) proofs, but their simulation-extractability is only shown in AGM+ROM [GKK+22]. We indicate these properties of existing SNARKs in Table 1. Given this state of affairs, we can refine our earlier question to the following:

Is it possible to obtain UC-secure NIZKs with constant size proofs in the ***random oracle model****?*

1.1 Our Results

In this work, we answer the above question in the affirmative. In particular, we give a compiler (in the ROM) that lifts any SNARK from non-black-box to black-box straight-line extraction, with *constant* (i.e. $O_\lambda(1)$) overhead.

Theorem 1.1. *(Informal) Given a non-black-box simulation-extractable zkSNARK $\Pi_\mathcal{R}$ for a relation $\mathcal{R}$ and a succinct polynomial commitment scheme, there exists a UC-secure, witness-succinct zkSNARK $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ in the (global random oracle ($\mathcal{G}_\mathsf{RO}$), local setup ($\mathcal{F}_\mathsf{Setup}$))-hybrid model, where $\mathcal{G}_\mathsf{RO}$ is observable but non-programmable as in [CJS14] and $\mathcal{F}_\mathsf{Setup}$ models the setup required by the original zkSNARK $\Pi_\mathcal{R}$ (e.g., a trusted CRS generator or the local random oracle).*

Plugging well-known SNARKs such as [GM17, ARS20] into our compiler gives us as a corollary the first constant sized UC NIZKs in the $(\mathcal{G}_\mathsf{RO}, \mathcal{F}_\mathsf{crs})$-hybrid model, from pairings under knowledge assumptions.

Remarks. There are a few qualifications to our main theorem:

– *Knowledge Assumptions:* Any output NIZK produced via our compiler inherits the knowledge assumptions used by the input SNARK. However, as knowledge assumptions cannot be used directly in the UC framework (as simulation cannot depend on the environment), the extraction strategy for our compiled SNARK does not involve invoking the non-black-box extractor of the input SNARK. Intuitively, we only make use of the input SNARK's non-black-box extractor to argue the indistinguishability of intermediate hybrid experiments (which can depend on the environment).

– *Unique Proofs:* Our compiler requires polynomial commitments that support a new 'unique proof' property, i.e. it is hard for an adversary to produce two distinct proofs for the same evaluation point. This is in fact an analogous notion to *unique response* defined for ROM-based NIZK proofs to be simulation-extractable [FKMV12, GOP+22]. Although this is not a standard property in the stand-alone setting, we show that it is a natural feature of common polynomial commitment schemes such as KZG [KZG10].

1.2 Technical Overview

We begin with the observation that most SNARKs already have *straight-line zero-knowledge simulators*—the verifier of a non-interactive object has no secrets and so there is nothing to be gained by rewinding or looking at its code—and therefore simulating an honest prover's SNARK string in the UC context is straightforward. Moreover, a plethora of work suggest that many concretely efficient SNARKs are already *simulation extractable* (see Table 1). The barrier to using existing SNARKs in the UC context is that the only known extractors require either looking into the code of the prover (i.e. non-black-box extraction) or rewinding the prover. Neither of these extraction techniques can be directly used within the UC framework, as the simulator in the UC experiment can not rewind the environment, nor depend on its code.

Previous works have recognized the fact that even though simulation must be straight-line in the UC framework, their proofs of indistinguishability can make use of arguments that involve rewinding the environment [DSW08, CDG+18]. The underlying principle is that even though the environment can not be rewound during simulation for the UC experiment, rewinding the environment can still be helpful as an analytical tool, for example in generating intermediate hybrid distributions between the real and ideal experiments. To our knowledge, this principle has not been applied to the case of *non-black-box* simulation, i.e. generating intermediate hybrid distributions using the code of the environment.

Our insight is that the existence of a non-black-box extractor guarantees that in order to produce a SNARK, the environment must fundamentally 'know' a witness—lifting the SNARK to a UC NIZK is then a matter of forcing the environment to use this knowledge. We describe below how we leverage this insight, by incrementally building upon the simple approach described earlier.

Commitments instead of encryption. Recall that the simple approach—where a proof consists of ciphertext ct and proof π that ct encrypts a witness—is bottlenecked by the ciphertext having to be 'decryptable', which means that $|\mathsf{ct}| \in \Omega(|w|)$. If we relax the decryptability requirement, we can have ct be a *commitment* instead. This is helpful, because commitments can be independent of the size of the message committed, and therefore succinct. Obtaining the witness from ct now becomes a matter of extracting a committed message rather than simply decrypting a ciphertext, and forms the core of the technical challenge.

Core Tool: Succinct, provable, straight-line extractable commitments. Straight-line extractable commitments are typically straightforward to construct in the random oracle model—simply computing $H(w, r)$ to commit to w with randomness r suffices [Pas03, CDG+18]. However H must be a random oracle to enable straight-line extraction, meaning that one cannot prove statements about its input. This is an issue as we need to prove that w committed to in ct is indeed a valid witness. This issue can be solved by assuming that since H is instantiated with a concrete hash function, it will have a circuit representation (as is common in the literature on recursive SNARKs [BCMS20, COS20]) however we wish to avoid such heuristics.

We must therefore construct a 'provable' commitment scheme, i.e. one that has a meaningful circuit representation while also supporting straight-line extraction of the committed message. Our methodology for designing such a commitment involves two parts $(\mathsf{cm}, \pi_{\mathsf{cm}})$, where cm is a commitment string output by a standard model commitment algorithm Com, and π_{cm} is a straight-line extractable proof of knowledge of its opening—notice that now it is meaningful to prove via a SNARK that cm is a commitment to a valid witness, as Com is a standard model algorithm. Since it is straightforward to achieve $|\mathsf{cm}| \in O_\lambda(1)$, we will focus on the design of π_{cm}.

Like much of the SNARK literature, in constructing π_{cm} we leverage the fact that arithmetization is conducive to succinct proofs. In particular, we instruct the prover to encode the witness w as the coefficients of a polynomial $f(x)$, and commit to f within cm (rather than committing to w directly). Assuming a prime $q \in \omega(\mathsf{poly}(\lambda))$ is a parameter of the scheme, and $d \in \mathbb{Z}$ a parameter of the statement, w is interpreted as a vector $w \in \mathbb{F}_q^d$ that characterize the coefficients of the degree[8] $d-1$ polynomial $f \in \mathbb{F}_q[X]$. Our straight-line extraction strategy will be to ensure that the prover queries at least d evaluations of f to the random oracle (i.e. enough to reconstruct w), by having the verifier check a subset of the evaluations. Importantly, this validation of f can be performed succinctly; the verifier need only query $O_\lambda(1)$ evaluations of f, and each evaluation can be authenticated at $O_\lambda(1)$ cost. We sketch our ideas behind these principles below.

[8] We remark that the actual compiler needs to inflate the degree according to the number of revealed evaluations in order to retain zero-knowledge, but we omit this technicality here for ease of exposition.

$O_\lambda(1)$ **Verifier Queries:** The prover first evaluates f at n points and commits to each $\{f(i)\}_{i\in[n]}$. The prover is then instructed to reveal r of the committed evaluations—which are checked for correctness—to guarantee that the commitments contain at least $d-1$ correct evaluations in total, with overwhelming probability. Assuming that $r \in [n]$ is chosen at random, the parameters can be fixed so that $r \in O_\lambda(1)$, due to the following rough analysis: the best adversarial assignment (for a cheating prover) of the n committed evaluations consists of only $d-1$ correct (and $n-d+1$ 'junk') ones, to maximize the number of subsets of size r that will satisfy a verifier—i.e. $\binom{d-1}{r}$. The total number of possible subsets that the verifier could query is $\binom{n}{r}$, which brings the probability of success of the best possible cheating strategy to:

$$\frac{\binom{d-1}{r}}{\binom{n}{r}} \approx \frac{d^r/r!}{n^r/r!} = \left(\frac{d}{n}\right)^r$$

Now if we fix r as say, λ (so that $r \in O_\lambda(1)$), notice that for any $d \in \mathbb{Z}$ the above quantity can be upper bounded by $2^{-\lambda}$ by setting $n \approx 2d$. In general, as long as $r \in \Omega(\lambda/\log\lambda)$, the same upper bound can be achieved with $n \in \mathsf{poly}(d,\lambda)$.

Authenticating Evaluation Openings at $O_\lambda(1)$ Cost via Fischlin's Technique [Fis05]: We framed our description above in a PCP-like model, where the prover writes down n evaluations of f, of which the verifier queries and checks r of them. As n is clearly not witness-succinct, we need a method by which the prover can commit to the n evaluations, and succinctly reveal r of them upon request. In the PCP/IOP literature [Mic00, BCS16], it is common to use Merkle trees for this task; they provide $O_\lambda(1)$ sized commitments with r short ($O_\lambda(\log n)$ sized) openings, and even natively support straight-line extraction. This follows a 'cut-and-choose' paradigm, where the prover commits to n objects, and the verifier checks r of them in order to guarantee that a total of at least d of the committed objects are 'good'. However the $O_\lambda(\log n)$ sized evaluation opening is a deal breaker (in the context of achieving $O_\lambda(1)$-sized proofs) as it grows—albeit slowly—with the witness size, and appears to be a fundamental hurdle with such techniques.

In the context of compiling Σ-protocols to NIZKs with straight-line extraction, Fischlin [Fis05] presented a technique based on *proofs of work* that shed the $O_\lambda(\log n)$ cost of Merkle tree openings when checking the validity of a subset of committed objects. At a very high level, Fischlin's technique emulates the combinatorial properties of the cut-and-choose approach, without the logistics of providing explicit commitments/openings. Fischlin's idea is that rather than challenging the prover to reveal a (randomly chosen) r-sized subset of some committed x_i values, the prover is challenged to provide any r values $\{x_i\}_{i\in[r]}$ such that $H(x_i) = 0$ for each i, where H is a random oracle. This forces the prover to query multiple 'good' x_i values to H before finding r of them that hash to the zero string, and no explicit decommitment information is necessary.

Applying Fischlin's technique to our setting yields a protocol of the following form. Upon fixing cm, for each $i \in [r]$: (1) the prover computes $\pi_{\mathsf{cm}}^{(i)} = (z_i, f(z_i))$ with uniform z_i and the corresponding *evaluation proof* $\pi_{\mathsf{ev}}^{(i)}$ that ensures the polynomial f committed to in cm has been correctly evaluated at z_i, and (2) store $(\pi_{\mathsf{cm}}^{(i)}, \pi_{\mathsf{ev}}^{(i)})$ and go to the next iteration if $H(\mathsf{cm}, i, \pi_{\mathsf{cm}}^{(i)}, \pi_{\mathsf{ev}}^{(i)}) = 0$ for a random oracle H with b-bit outputs, and go to step (1) otherwise. Thanks to the evaluation proof, $\pi_{\mathsf{cm}}^{(i)}$ is tied to a given commitment cm. In practice, succinct evaluation proof can be easily implemented by naively invoking the underlying SNARK prover[9] or by instantiating cm with a dedicated *polynomial commitment scheme* such as [KZG10], which usually minimizes the overhead in prover's work. Computing such a proof is easy for an honest prover, via rejection-sampling with random $(z_i, f(z_i))$ values until r of them that hash to zero are found. As for an adversarial prover P^*, the aim is to produce an accepting proof—by finding r pre-images of 0—with $d-1$ or fewer queries to the random oracle. As a loose upper bound, the probability that P^* finds r successes within $d-1$ queries is at most the probability that for every $i \in [r]$, P^* is able to find $H(\mathsf{cm}, i, \cdot) = 0$ within $d-1$ queries. For any given i, the probability that P^* finds $H(\mathsf{cm}, i, \cdot) = 0$ within d queries is at most $d/2^b$; therefore the probability that P^* finds $H(\mathsf{cm}, i, \cdot) = 0$ within $d-1$ queries for every $i \in [r]$ simultaneously is at most $(d/2^b)^r = 2^{-r(b-\log d)}$. The proof sketch here are implicitly assuming that a valid evaluation proof is determined *uniquely* once cm, z_i, and $f(z_i)$ are fixed. Our formal analysis accounts for this subtlety and we show that [KZG10] indeed satisfies this property.

Assuming that $r = \lambda \in O_\lambda(1)$, the above quantity is bounded by $2^{-\lambda}$ when $b = 1 + \log d \in O_\lambda(\log d)$. The prover's work is in expectation $2^b \cdot r = 2^{1+\log d} \cdot \lambda$ which is in $\mathsf{poly}(\lambda)$ as well as $O_\lambda(d)$, i.e. it scales

[9] For this alternative instantiation, one must use a de-randomized version of the underlying SNARK to obtain the unique proof property, as also required by our main compiler.

linearly in the witness size. Of course better parameters are possible; r can be improved by up to a log factor, as we explore later in the 'succinctness' component of the proof of Theorem 3.1.

Putting it together: The prover produces an $O_\lambda(1)$-sized standard model commitment cm to a degree d polynomial f that encodes the witness, and proves knowledge of its opening via $\boldsymbol{\pi}_{\mathsf{cm}} = (\pi_{\mathsf{cm}}^{(i)})_{i\in[r]}$ – this proof is at the heart of forcing the environment to use the witness within the context of the protocol. The proof $\boldsymbol{\pi}_{\mathsf{cm}}$ requires the prover to 'work' to find $r \in O_\lambda(1)$ pre-images of 0 for random oracle H, where each pre-image is an evaluation of f. The parameters for this proof-of-work are set so that (except with negligible probability) the prover queries more than $d-1$ evaluations of f in its effort to find these r pre-images of zero. Reading these d evaluations of f allows an extractor to reconstruct f—which is an opening to cm. Finally, the prover gives a SNARK π to prove that it knows an opening to cm that is the witness to a public statement (through a suitable witness-polynomial encoding function Enc). If one were to hypothetically run the non-black-box SNARK extractor on the environment at this point, the opening to cm that it finds should be *exactly the same* as the f reconstructed via the extractor of $\boldsymbol{\pi}_{\mathsf{cm}}$; if not, then one would obtain two openings to cm, in contradiction of the binding property of the commitment scheme. Therefore, any knowledge that the environment uses in the production of π—perhaps even outside the protocol—is extracted in a black-box, straight-line fashion via $\boldsymbol{\pi}_{\mathsf{cm}}$ within the context of the protocol.

In sum, a proof generated as per our compiled NIZK consists of the following components (with certain parameters omitted for readability):

– cm: commitment to an (encoded) witness.
– π: Input SNARK to prove knowledge of witness w satisfying $(\mathcal{C}(w) = 1 \ \wedge \ \mathsf{cm} = \mathsf{Com}(f) \ \wedge \ f = \mathsf{Enc}(w))$.
– $\boldsymbol{\pi}_{\mathsf{cm}} = (z_i, y_i)_{i\in[r]}$: Evaluations of the witness-encoding polynomial f on r points.
– $\boldsymbol{\pi}_{\mathsf{ev}} = (\pi_{\mathsf{ev}}^{(i)})_{i\in[r]}$: Proofs of r correct evaluations, guaranteeing $y_i = f(z_i)$ for all $i \in [r]$ *and* $\mathsf{cm} = \mathsf{Com}(f)$.

An important additional constraint is that $H(\mathsf{cm}, i, \pi_{\mathsf{cm}}^{(i)}, \pi_{\mathsf{ev}}^{(i)}) = 0$ for each $i \in [r]$.

1.3 Related Work

Straight-line Extraction. Our UC-lifting technique is inspired by Fischlin's transform [Fis05] based on Proof-of-Work. Kondi and shelat [Ks22] gave an analysis for using Fischlin's transformation for *compressing* proofs in the context of signature aggregation, and showed how randomizing Fischlin's technique is conducive to zero-knowledge. Very recently, Lysyanskaya and Rosenbloom [LR22b, LR22a] present compilers lifting Σ-protocols to UC-secure (adaptive) NIZKPoK in the global ROM, where the straight-line extraction is realized via Fischlin's transform. Canetti, Sarkar, and Wang [CSW20] realized *triply adaptive* UC-secure NIZK using a straight-line extractable commitment in the CRS model. Pass [Pas03] described a generic way to turn Σ-protocols with special soundness into straight-line extractable proof systems using RO-based commitment. The technique is somewhat analogous to the verifiable encryption of Camenisch and Damgård [CD00] where the commitment is instantiated using public-key encryption and thus SLE holds in the CRS model (where the decryption key serves as a private extraction key for the knowledge extractor). The transform of Unruh [Unr15] extended [Pas03] to retain security against an adversary making superposition queries to the RO (the so-called *quantum random oracle model*). Recently, Katsumata [Kat21] showed an efficient SLE transform in the QROM tailored to lattice-based ZK proofs.

Lifting Transformations. Techniques for generically adding black-box simulation extractability to any NIZK were first shown in the works of [Sah99, DDO$^+$01, Gro06], optimized in the C∅C∅ framework [KZM$^+$15], and tailored to Groth16 in [AB19, Bag19]. These techniques augment the relation to an OR language and the trapdoor for one of the OR clauses is used by the ZK simulator. Extractability is obtained by encrypting the witness under a public key that is part of the CRS and additionally proving correct encryption. The LAMASSU [ARS20] framework extends the C∅C∅ lifting technique to work with updatable SNARKs giving a generic compiler from updatable CRS SNARKS to SE SNARKs. TIRAMISU [BS21] builds on these frameworks to additionally lift SNARKs into black-box simulation extractable ones. However, all these lifting transformations yield SNARKs where one of either witness succinctess or blackbox extraction is lost, unlike our compiler. There are works on lifting specific SNARKs into SE; the work of Groth and Maller [GM17] presents an SE SNARK, but the simulation extractability is non-black-box. There is a line of work on analysing the simulation extractability [BG18, BKSV21, BPR20] of Groth16; all of these are in idealized models like GGM/AGM, in addition to ROM.

2 Preliminaries

Notations. For positive integers a and b such that $a < b$ we use the integer interval notation $[a, b]$ to denote $\{a, a+1, \ldots, b\}$. We also use $[b]$ as shorthand for $[1, b]$. If S is a set we write $s \leftarrow\!\!\$\, S$ to indicate sampling s from the uniform distribution defined over S; if $\mathcal{A}$ is a randomized (resp. deterministic) algorithm we write $s \leftarrow \mathcal{A}$ (resp. $s := \mathcal{A}$) to indicate assigning an output from $\mathcal{A}$ to s. The security parameter λ is 1^λ in unary. A function $f(\lambda)$ is said to be *negligible in* λ if for any polynomial $\mathsf{poly}(\lambda)$ it holds that $f(\lambda) < 1/\mathsf{poly}(\lambda)$ for sufficiently large $\lambda > 0$. We write "$f(\lambda) < \mathsf{negl}(\lambda)$" to indicate $f(\lambda)$ is negligible in λ. $\mathbb{F}[X]$ denotes polynomials over a finite field $\mathbb{F}$. For an integer $d \geq 1$, $\mathbb{F}_{<d}[X] \subseteq \mathbb{F}[X]$ denotes polynomials of degree less than d.

2.1 UC Framework

In this work, we use the *Universal Composability* (UC) framework [Can01] for security proofs. UC follows the simulation-based paradigm where the security of a protocol is defined with respect to an ideal world where a trusted party, the functionality $\mathcal{F}$, does the all of the computation. Informally, a protocol securely realizes $\mathcal{F}$ in the real world if for any real world adversary there exist an equivalent ideal world adversary (the simulator). Equivalent meaning that any outside observer (the environment) cannot distinguish between the real protocol execution and the ideal execution. UC's composition theorem ensures that one can safely compose protocols that have been proven UC-secure.

Global Random Oracle. More precisely, we use the generalized UC (GUC) framework [CDPW07] which allows to model global functionalities that are shared between different protocol instances. We consider a hybrid-model where parties have access to a (non-programmable) global random oracle $\mathcal{G}_{\mathsf{RO}}$ as introduced in [CJS14]. We follow the simplified description from [CDG+18]. The $\mathcal{G}_{\mathsf{RO}}$ functionality can be queried by any party and the ideal adversary with two commands: QUERY and OBSERVE. The environment can query $\mathcal{G}_{\mathsf{RO}}$ by spawning additional dummy parties outside the context of the current protocol execution. $\mathcal{G}_{\mathsf{RO}}$ answers all new QUERY command by lazy sampling from the domain and stores them locally in a list $\mathcal{Q}$. A repeated query requires a simple lookup in $\mathcal{Q}$. Some QUERY queries are marked "illegitimate" and can be observed via OBSERVE command. Next we explain which query counts as an illegitimate one. Each party is associated with its party identifier pid and a session identifier sid. When a party queries $\mathcal{G}_{\mathsf{RO}}$ with the command (QUERY, x), the query is parsed as (s, x') where s denotes the session identifier associated with the party. A query is marked as illegitimate if the sid field of the query differs from the sid associated to the party making the query. In other words, these are the queries made outside the context of the current session execution. We formally define the functionality $\mathcal{G}_{\mathsf{RO}}$ in Fig. 1.

Remark 2.1. In [CJS14] the random oracle allows ideal functionalities to obtain the list of illegitimate queries. In order for the adversary to fetch those queries there needs to be a (dummy) functionality that forwards those queries. In [CDG+18] this is simplified by allowing the adversary to directly query the random oracle for illegitimate queries. Thus, functionalities no longer need to forward the illegitimate queries.

Intuitively, these illegitimate queries are required for proving security of our protocols. The ideal adversary (or the simulator) works by observing $\mathcal{G}_{\mathsf{RO}}$ queries made by the corrupt party during the protocol execution. However, the environment can bypass this by querying $\mathcal{G}_{\mathsf{RO}}$ via additional dummy parties outside the current session. The simulator remains oblivious to these additional parties and thus fails in proving security. However, this behavior of the environment is accounted for in [CDG+18] by marking such queries as illegitimate and disclosing them to the simulator via OBSERVE command. Note that any $\mathcal{G}_{\mathsf{RO}}$ query for session id sid made by a party (or the simulator) participating in the session identified by sid will never be marked as illegitimate. Thus, any query made the simulator itself is not recorded by the functionality and hence cannot be observed by anyone. This is crucial for proving indistinguishability between the ideal and the real world and we elaborate in the proof of Theorem 3.1.

Functionality 1: $\mathcal{G}_{\mathsf{RO}}$

$\mathcal{G}_{\mathsf{RO}}$ is parametrized by the output length $\ell(\lambda)$.

- **Query** Upon receiving a query (QUERY, x), from some party $\mathcal{P} = (\mathsf{pid}, \mathsf{sid})$ or from the adversary Sim do:
  - Look up v if there is a pair (x, v) for some $v \in \{0,1\}^{\ell(\lambda)}$ in the (initially empty) list $\mathcal{Q}$ of past queries. Else, choose uniformly $v \in \{0,1\}^{\ell(\lambda)}$ and store the pair (x, v) in $\mathcal{Q}$.
  - Parse x as (s, x'). If $\mathsf{sid} \neq s$ then add (s, x', v) to the (initially empty) list of illegitimate queries for SID s, that is denoted by $\mathcal{Q}_{|s}$.
  - Return v to $\mathcal{P}$.
- **Observe** Upon receiving a request $(\text{OBSERVE}, \mathsf{sid})$ from the adversary Sim, return the list $\mathcal{Q}_{|\mathsf{sid}}$ of illegitimate queries for SID sid to the adversary.

Fig. 1: Functionality for Global Random Oracle $\mathcal{G}_{\mathsf{RO}}$ [CDG+18]

Definition 2.2 (UC Security in the Global ROM [CDPW07, CJS14]). *Let $\mathcal{F}, \mathcal{F}'$ be m-party functionalities and Π be a protocol. We say that Π UC-realizes $\mathcal{F}$ in the $\mathcal{G}_{\mathsf{RO}}, \mathcal{F}'$-hybrid model if for any hybrid-model* PPT *adversary $\mathcal{A}$, there exists an ideal process* PPT *adversary* Sim *such that for every* PPT *environment $\mathcal{Z}$, it holds that:*

$$\{\mathsf{IDEAL}^{\mathcal{G}_{\mathsf{RO}}}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}(\mathbf{x}, \lambda, z)\}_{\mathbf{x},\lambda,z} \approx \{\mathsf{REAL}^{\mathcal{G}_{\mathsf{RO}}}_{\mathcal{F}',\Pi,\mathcal{A},\mathcal{Z}}(\mathbf{x}, \lambda, z)\}_{\mathbf{x},\lambda,z}$$

where REAL *is the outputs of the honest parties and the adversary $\mathcal{A}$ after a real execution of protocol Π with input $\mathbf{x} = (x_1, \ldots, x_m)$ for parties $P_1, \ldots, P_m$ where each $x_i \in \{0,1\}^*$, $z \in \{0,1\}^*$ is the auxiliary input for $\mathcal{A}$ and λ is the security parameter.* IDEAL *is the analogous distribution in an ideal execution with a trusted party that computes $\mathcal{F}$ for the parties and hands the output to the designated players.*

2.2 Succinct Non Interactive Zero-Knowledge Proof

A *non-interactive proof system* for relation $\mathcal{R}$, denoted by $\Pi_{\mathcal{R}}$, consists a tuple of algorithms $(\mathsf{PGen}, \mathcal{O}_{\mathsf{Setup}}, \mathcal{P}, \mathcal{V})$.

- $\mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda)$: Takes as input the security parameter λ and outputs public parameters pp. Once PGen is invoked we assume that all of the following algorithms take pp as an implicit input.
- $\mathsf{out} \leftarrow \mathcal{O}_{\mathsf{Setup}}(\mathsf{in})$: A stateful setup oracle that takes an input string in and outputs out.
- $\pi \leftarrow \mathcal{P}^{\mathcal{O}_{\mathsf{Setup}}}(x, w)$: Takes as input a statement x and witness w, and outputs a proof π if $(x, w) \in \mathcal{R}$.
- $b \leftarrow \mathcal{V}^{\mathcal{O}_{\mathsf{Setup}}}(x, \pi)$: Takes as input a statement x and proof π, and outputs a bit b, indicating "accept" or "reject".

We introduce the setup oracle $\mathcal{O}_{\mathsf{Setup}}$ to the notation of NIZKs to capture the two typical setup assumptions in an abstract manner. That is, if a proof system is instantiated in the CRS model, then $\mathcal{O}_{\mathsf{Setup}}$ internally generates crs upon receiving a query with any input for the first time, and keeps outputting the same crs regardless of the input. When instantiating the RO model, $\mathcal{O}_{\mathsf{Setup}}$ is initialized with an empty query-response table and proceeds as follows. On receiving $\mathsf{in} \in \{0,1\}^*$, if in has never been queried, sample uniform $\mathsf{out} \in \{0,1\}^{\ell(\lambda)}$, store $(\mathsf{in}, \mathsf{out})$ in the table, and return out. Otherwise, look up the table to find out associated with in, and return out.

We define three basic security properties for $\Pi_{\mathcal{R}}$ in the stand-alone setting.

Definition 2.3 (Completeness). *$\Pi_{\mathcal{R}}$ satisfies* completeness *if for every $(x, w) \in \mathcal{R}$, it holds that*

$$\Pr\left[b = 1 \ : \ \mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda); \pi \leftarrow \mathcal{P}^{\mathcal{O}_{\mathsf{Setup}}}(x, w); b \leftarrow \mathcal{V}^{\mathcal{O}_{\mathsf{Setup}}}(x, \pi)\right] = 1.$$

We define zero-knowledge by following the syntax of [FKMV12, GOP+22]. A zero-knowledge simulator $\mathcal{S}$ is defined as a stateful algorithm with initial state $\mathsf{st} = \mathsf{pp}$ that operates in two modes. The first mode, $(\mathsf{out}, \mathsf{st}') \leftarrow \mathcal{S}(1, \mathsf{st}, \mathsf{in})$ takes care of handling calls to the oracle $\mathcal{O}_{\mathsf{Setup}}$ on input in. The second mode, $(\pi, \mathsf{st}') \leftarrow \mathcal{S}(2, \mathsf{st}, x)$ simulates a proof for the input statement x. For convenience we

Functionality 2: $\mathcal{F}_{\mathsf{Setup}}$

$\mathcal{F}_{\mathsf{Setup}}$ is parametrized by a security parameter λ and a degree bound $D > 0$ and runs with parties $\mathsf{P}_1, \ldots, \mathsf{P}_N$ and an ideal process adversary Sim.

- **Parameters** Upon receiving input $(\textsc{genparams}, \mathsf{sid})$ from a party P_i, if no pp has been stored, run $\mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda)$, initialize oracle $\mathcal{O}_{\mathsf{Setup}}$ with pp, and store pp. Send $(\textsc{params}, \mathsf{sid}, \mathsf{pp})$ to P_i.
- **Commitment Key** Upon receiving input $(\textsc{genkey}, \mathsf{sid})$ from a party P_i, if no ck has been stored, run $\mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D)$ and store ck. Send $(\textsc{comkey}, \mathsf{sid}, \mathsf{ck})$ to P_i.
- **Setup** Upon receiving input $(\textsc{setup}, \mathsf{sid}, \mathsf{in})$ from a party P_i, ignore if $\mathcal{O}_{\mathsf{Setup}}$ has not been initialized with pp. Otherwise run $\mathsf{out} \leftarrow \mathcal{O}_{\mathsf{Setup}}(\mathsf{in})$ using the current state of $\mathcal{O}_{\mathsf{Setup}}$ and send $(\textsc{setup}, \mathsf{sid}, \mathsf{out})$ to P_i.

Fig. 2: N-party functionality for setup $\mathcal{F}_{\mathsf{Setup}}$

define three "wrapper" oracles. These oracles are stateful and share the internal state st, which initially contains an empty string.

- $\mathcal{S}_1(\mathsf{in})$ to denote the oracle that returns the first output of $\mathcal{S}(1, \mathsf{st}, \mathsf{in})$;
- $\mathcal{S}_2(x, w)$ that returns the first output of $\mathcal{S}(2, \mathsf{st}, x)$ if $(x, w) \in \mathcal{R}$ and $\perp$ otherwise;
- $\mathcal{S}_2'(x)$ that returns the first output of $\mathcal{S}(2, \mathsf{st}, x)$.

Definition 2.4 ((Unbounded) Zero-Knowledge). *Let $\Pi_{\mathcal{R}} = (\mathsf{PGen}, \mathcal{O}_{\mathsf{Setup}}, \mathcal{P}, \mathcal{V})$ be a non-interactive proof system for relation $\mathcal{R}$. $\Pi_{\mathcal{R}}$ is unbounded* non-interactive zero-knowledge *(NIZK), if there exists a* PPT *simulator $\mathcal{S}$ with wrapper oracles $\mathcal{S}_1$ and $\mathcal{S}_2$ such that for all* PPT *adversaries $\mathcal{A}$ it holds that*

$$\left| \Pr\left[b = 1 \,:\, \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda); \\ b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Setup}}, \mathcal{P}}(\mathsf{pp}) \end{array} \right] - \Pr\left[b = 1 \,:\, \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda); \\ b \leftarrow \mathcal{A}^{\mathcal{S}_1, \mathcal{S}_2}(\mathsf{pp}) \end{array} \right] \right| < \mathsf{negl}(\lambda).$$

Next, we define simulation extractability, which essentially guarantees that proofs are *non-malleable*. We stress that the present definition is weaker than what is necessary for realizing UC security, because the extractor algorithm here is *non-black-box*, i.e., it requires looking into the code of the adversary. The definition is an abstracted version of [GM17] and the schemes satisfying their definition clearly meet the version below by instantiating $\mathcal{S}$ with trapdoor'd CRS generator in mode 1 and ZK simulator in mode 2.[10]

Definition 2.5 ((Non-black-box) Simulation Extractability). *Consider a non-interactive proof system $\Pi_{\mathcal{R}} = (\mathsf{PGen}, \mathcal{O}_{\mathsf{Setup}}, \mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}$ with an* NIZK *simulator $\mathcal{S}$. Let $(\mathcal{S}_1, \mathcal{S}_2')$ be wrapper oracles for $\mathcal{S}$ as defined above. $\Pi_{\mathcal{R}}$ is non-black-box* simulation-extractable *($\mathsf{SIM\text{-}EXT}$) with respect to $\mathcal{S}$, if for any* PPT *adversary $\mathcal{A}$, there exists a* PPT *extractor $\mathcal{E}_{\mathcal{A}}$ such that*

$$\Pr\left[\begin{array}{r} (x, \pi) \notin \mathcal{Q} \wedge (x, w) \notin \mathcal{R} \\ \wedge\, b = 1 \end{array} \,:\, \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda); (x, \pi) \leftarrow \mathcal{A}^{\mathcal{S}_1, \mathcal{S}_2'}(\mathsf{pp}); \\ b \leftarrow \mathcal{V}^{\mathcal{S}_1}(x, \pi); w \leftarrow \mathcal{E}_{\mathcal{A}}(x, \pi, \mathsf{state}_{\mathcal{A}}, \mathsf{st}) \end{array} \right] < \mathsf{negl}(\lambda)$$

where st is the final state of the simulator $\mathcal{S}$, $\mathsf{state}_{\mathcal{A}}$ is a string containing all inputs and outputs of $\mathcal{A}$, including random coins, and $\mathcal{Q}$ is a set of statement-proof pairs (x, π) with x being a statement queried by $\mathcal{A}$ to the proof simulation wrapper oracle $\mathcal{S}_2'$, and π being the corresponding simulated proof, respectively.

The ideal functionality $\mathcal{F}_{\mathsf{Setup}}$ that provides the setup and oracle for non-interactive proof system $\Pi_{\mathcal{R}} = (\mathsf{PGen}, \mathcal{O}_{\mathsf{Setup}}, \mathcal{P}, \mathcal{V})$ is described in Fig. 2.

Our final goal is to compile $\Pi_{\mathcal{R}}$ with the above basic security properties into a UC-secure NIZK protocol $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$. The ideal functionality for Non-interactive Zero-Knowledge $\mathcal{F}_{\mathsf{NIZK}}$ is defined in Fig. 3. The functionality is taken from [GOS12] with a minor difference being that $\mathcal{F}_{\mathsf{NIZK}}$ explicitly informs Sim of the associated session ID.

[10] Following [GM17], we also assume the relation $\mathcal{R}$ (which may potentially contain auxiliary inputs) is *benign*, i.e. it is distributed in such a way that the SNARKs for $\mathcal{R}$ can be simulation extractable.

Functionality 3: $\mathcal{F}_{\mathsf{NIZK}}$

$\mathcal{F}_{\mathsf{NIZK}}$ is parametrized by polynomial-time-decidable relation $\mathcal{R} \in \{0,1\}^* \times \{0,1\}^*$, and runs with parties $\mathsf{P}_1, \ldots, \mathsf{P}_N$ and an ideal process adversary Sim. It stores proof table $\mathcal{Q}$ which is initially empty.

- **Proof** Upon receiving input $(\textsc{prove}, \mathsf{sid}, \mathsf{ssid}, x, w)$ from a party P_i, ignore if $(x, w) \notin \mathcal{R}$. Otherwise, send $(\textsc{prove}, \mathsf{sid}, x)$ to Sim. Upon receiving $(\textsc{proof}, \pi)$ from Sim, store (x, π) in $\mathcal{Q}$ and send $(\textsc{proof}, \mathsf{sid}, \mathsf{ssid}, \pi)$ to P_i.
- **Verification** Upon receiving input $(\textsc{verify}, \mathsf{sid}, \mathsf{ssid}, x, \pi)$ from a party P_i, if (x, π) is not stored in $\mathcal{Q}$, then send $(\textsc{verify}, \mathsf{sid}, x, \pi)$ to Sim. Upon receiving $(\textsc{witness}, w)$ from Sim, if $(x, w) \in \mathcal{R}$, store (x, π) in $\mathcal{Q}$. Finally, return $(\textsc{verification}, \mathsf{sid}, \mathsf{ssid}, (x, \pi) \in_? \mathcal{Q})$ to P_i.

Fig. 3: N-party functionality for non-interactive zero-knowledge $\mathcal{F}_{\mathsf{NIZK}}$

2.3 Succinct Polynomial Commitment Scheme

The following definition is adapted from the full version of [CHM+20]. The difference is that we omit the commitment key trimming algorithm as it is only necessary for concrete optimization.

Definition 2.6 (Polynomial Commitment Scheme). *A polynomial commitment scheme over field $\mathbb{F}$, denoted by PCS, is a tuple of algorithms $(\mathsf{KGen}, \mathsf{Com}, \mathsf{Eval}, \mathsf{Check})$:*

1. $\mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D)$: *Takes as input the security parameter λ and the maximum degree bound D and generates commitment key ck as output.*
2. $c \leftarrow \mathsf{Com}(\mathsf{ck}, f, d; \rho_c)$: *Takes as input ck, the polynomial $f \in \mathbb{F}_{<d}[X]$, the degree bound $d \leq D$, randomness ρ_c and outputs a commitment c. In case the commitment scheme is deterministic $\rho_c = \bot$.*
3. $\pi \leftarrow \mathsf{Eval}(\mathsf{ck}, c, d, z, y, f; \rho_c)$: *Takes as input ck, the commitment c, degree bound $d \leq D$, evaluation point $z \in \mathbb{F}$, claimed polynomial evaluation $y \in \mathbb{F}$, the polynomial f, and outputs a non-interactive proof of evaluation π. The randomness ρ_c must equal the one previously used in Com.*
4. $b \leftarrow \mathsf{Check}(\mathsf{ck}, c, d, z, y, \pi)$: *Takes as input statement $(\mathsf{ck}, c, d, z, y)$ and the proof of evaluation π and outputs a bit b.*

satisfying the following properties:

Completeness. *For any integer $1 \leq d \leq D$, for all polynomials $f \in \mathbb{F}_{<d}[X]$, for all evaluation points $z \in \mathbb{F}$, and any randomness ρ_c*

$$\Pr\left[b = 1 : \begin{array}{c} \mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D); c \leftarrow \mathsf{Com}(\mathsf{ck}, f, d; \rho_c); \\ y := f(z); \pi \leftarrow \mathsf{Eval}(\mathsf{ck}, c, d, z, y, f; \rho_c); \\ b \leftarrow \mathsf{Check}(\mathsf{ck}, c, d, z, y, \pi) \end{array} \right] = 1.$$

Evaluation Binding. *For any integer $1 \leq d \leq D$, for all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} y \neq y' \\ \wedge\ b = 1 \\ \wedge\ b' = 1 \end{array} : \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D); (c, d, z, y, y', \pi, \pi') \leftarrow \mathcal{A}(\mathsf{ck}); \\ b \leftarrow \mathsf{Check}(\mathsf{ck}, c, d, z, y, \pi); \\ b' \leftarrow \mathsf{Check}(\mathsf{ck}, c, d, z, y', \pi') \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Succinctness. *A PCS is said to be* succinct *if both the size of commitment c and evaluation proof π is of size $\mathcal{O}_\lambda(1)$.*

In addition to standard properties above, we need a few more special properties for our compiler to work. In a later section we show that the widely used scheme of [KZG10] indeed satisfy these.

Definition 2.7 (Unique Proof). *For all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} \pi \neq \pi' \\ \wedge\ b = 1 \\ \wedge\ b' = 1 \end{array} : \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D); \\ (c, d, z, y, \pi, \pi') \leftarrow \mathcal{A}(\mathsf{ck}); \\ b \leftarrow \mathsf{Check}(\mathsf{ck}, c, d, z, y, \pi); \\ b' \leftarrow \mathsf{Check}(\mathsf{ck}, c, d, z, y, \pi') \end{array} \right] \leq \mathsf{negl}(\lambda).$$

We define a polynomial encoding scheme, which takes a vector of field elements and outputs an appropriate randomized polynomial. An important property, sometimes referred to as *bounded independence* in the literature [CHM+20, §2.3][11], guarantees that a bounded number of evaluations do not leak any information about the original polynomial.

Definition 2.8 (Polynomial Encoding Scheme). *A* polynomial encoding scheme*, denoted by* PES*, is a tuple of algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *defined over an evaluation domain* $\mathcal{D}_{\mathsf{Enc}}$ *(which also determines the forbidden domain* $\mathcal{S}_{\mathsf{Enc}} = \mathbb{F} \setminus \mathcal{D}_{\mathsf{Enc}}$*).*

- $f \leftarrow \mathsf{Enc}(\mathbf{w}, n, \ell; \boldsymbol{\rho})$*: Takes as inputs* $\mathbf{w} \in \mathbb{F}^n$*, dimension of the vector* $n > 0$*, evaluation bound* $\ell > 0$*, and randomness* $\boldsymbol{\rho} \in \mathbb{F}^\ell$*, and outputs a polynomial* $f \in \mathbb{F}_{<n+\ell}[X]$*.*
- $\mathbf{w}' \leftarrow \mathsf{Dec}(f, n, \ell)$*: Takes as inputs* $f \in \mathbb{F}_{<n+\ell}[X]$*,* $n > 0$*, and* $\ell > 0$*, and deterministically outputs* $\mathbf{w}' \in \mathbb{F}^n$*.*

We say PES *is* correct *if* $\mathbf{w} = \mathsf{Dec}(\mathsf{Enc}(\mathbf{w}, n, \ell; \boldsymbol{\rho}), n, \ell)$ *for any* $n > 0$*,* $\ell > 0$*,* $\mathbf{w} \in \mathbb{F}^n$*, and* $\boldsymbol{\rho} \in \mathbb{F}^\ell$*.* PES *satisfies* bounded independence *if for any* $n > 0$*,* $\ell > 0$*, and* $\mathbf{w} \in \mathbb{F}^n$*, and for* $\boldsymbol{\rho}$ *sampled uniformly from* $\mathbb{F}^\ell$*, any set of* ℓ *evaluations of* $f \leftarrow \mathsf{Enc}(\mathbf{w}, n, \ell; \boldsymbol{\rho})$ *in* $\mathcal{D}_{\mathsf{Enc}}$ *are independently and uniformly distributed in* $\mathbb{F}$*.*

In this work, we only consider polynomial encoding schemes where the size of the evaluation domain is exponential in the security parameter, i.e. $|\mathcal{D}_{\mathsf{Enc}}| \in O(2^\lambda)$. Below we recall some candidate encoding schemes that are implicitly employed in many SNARK constructions.

- **Coefficient Encoding** $\mathsf{PES}_1 = (\mathsf{Enc}_1, \mathsf{Dec}_1)$: PES can be instantiated using simple coefficient encoding as in [MBKM19]. Here $\mathcal{D}_{\mathsf{Enc}} = \mathbb{F} \setminus \{0\}$ and Enc_1 outputs

$$f(X) = \sum_{i=1}^{n} w_i X^{i-1} + \sum_{i=1}^{\ell} \rho_i X^{n+i-1}$$

  where $\mathbf{w} = (w_i)_{i\in[n]}$ and $\boldsymbol{\rho} = (\rho_i)_{i\in[\ell]}$. The decoding algorithm Dec_1 outputs the first n coefficients of f. It satisfies bounded independence because any set of ℓ evaluations of f are independent of the encoded vector.

- **Lagrange Encoding** $\mathsf{PES}_2 = (\mathsf{Enc}_2, \mathsf{Dec}_2)$: This encoding method has been used in e.g. [GWC19, CHM+20, CFF+21]. Suppose a subset $H \subset \mathbb{F}$ of cardinality n and an evaluation domain $\mathcal{D}_{\mathsf{Enc}} = \mathbb{F} \setminus (H \cup \{0\})$. Assume that an input $\mathbf{w} \in \mathbb{F}^n$ is indexed by H, i.e., $\mathbf{w} = (\mathbf{w}(a))_{a\in H}$. Let $L_{a,H} \in \mathbb{F}_{<n}[X]$ for $a \in H$ be the Lagrange polynomials corresponding to H and $Z_H(X) = \prod_{a\in H}(X - a)$ be a *vanishing polynomial of* H. Then using $\boldsymbol{\rho} = (\rho_i)_{i\in[\ell]}$ as randomness, $\mathsf{Enc}_2(\mathbf{w}, n, \ell; \boldsymbol{\rho})$ outputs

$$f(X) = \sum_{a\in H} \mathbf{w}(a) \cdot L_{a,H}(X) + \left(\sum_{i=1}^{\ell} \rho_i X^{i-1}\right) \cdot Z_H(X).$$

  The decoding algorithm Dec_2 outputs $(f(a))_{a\in H}$. On the one hand, PES_2 satisfies correctness since f agrees with $\mathbf{w}$ over the forbidden domain $\mathcal{S}_{\mathsf{Enc}} = H$. On the other hand, up to ℓ evaluations of f in $\mathcal{D}_{\mathsf{Enc}}$ reveal nothing about the encoded vector $\mathbf{w}$. Typically, the evaluation bound ℓ should be set strictly larger than the number of evaluation proofs the prover explicitly reveals, because a commitment to the polynomial itself may leak information about one evaluation (as in the KZG scheme). It turns out that this property helps us show the hiding property below once combined with a suitable polynomial commitment scheme.

Evaluation Hiding. We now define *evaluation hiding.* Note that this is a stronger property than the usual hiding definition (such as the ones in [KZG10, CHM+20]): essentially, evaluation hiding guarantees that the joint distribution of commitment, evaluation proof, and polynomial evaluations leaks nothing about the committed polynomial, whereas the usual PCS hiding property does allow evaluations to be associated with the committed polynomials. Clearly, if Enc is deterministic PCS can never be evaluation hiding. This is why the definition only makes sense with respect to a specific encoding scheme. Recent IOP-based SNARKs such as [GWC19, MBKM19, CHM+20, CFF+21] in fact exploit this property (albeit without formal definition tailored to PCS) to hide evaluations of a polynomial encoding secret witness and thus to retain perfect zero knowledge. The definition is parameterized by a function $\phi : \mathbb{Z}^+ \to \mathbb{Z}^+$ calculating the expansion factor for encoding randomness: given the number

[11] This property is also know as k-knowledge bound in [BCGV16].

of evaluated points $\ell' > 0$, it determines $\ell > \ell'$ the total number of random field elements necessary for hiding the committed polynomial *even after outputting a commitment, ℓ' evaluation proofs, and ℓ' evaluations.*

Definition 2.9 (ϕ-Evaluation Hiding). *Let PCS* = (KGen, Com, Eval, Check) *be a polynomial commitment scheme and PES* = (Enc, Dec) *be a polynomial encoding scheme. We say PCS is* ϕ-evaluation hiding with respect to PES *if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,*

$$\Pr\left[b = b' \ \wedge\ \mathbf{z} \in \mathcal{D}_{\mathsf{Enc}}^{|\mathbf{z}|} : \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D); (\mathbf{w}, \mathbf{z}) \leftarrow \mathcal{A}_1(\mathsf{ck}); \\ n := |\mathbf{w}|; \ell := \phi(|\mathbf{z}|); d := n + \ell; \\ \boldsymbol{\rho}_w \leftarrow\!\!\$\ \mathbb{F}^\ell; b \leftarrow\!\!\$\ \{0,1\}; \\ f \leftarrow \mathsf{Enc}(b \cdot \mathbf{w}, n, \ell; \boldsymbol{\rho}_w); \\ c \leftarrow \mathsf{Com}(\mathsf{ck}, f, d; \rho_c); \\ \mathbf{y} := f(\mathbf{z}); \\ \boldsymbol{\pi} \leftarrow \mathsf{Eval}(\mathsf{ck}, c, d, \mathbf{z}, \mathbf{y}, f; \rho_c); \\ b' \leftarrow \mathcal{A}_2(c, \mathbf{y}, \boldsymbol{\pi}) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}_1, \mathcal{A}_2$ share the internal states, $\mathbf{y} := f(\mathbf{z})$ denotes setting $y_i := f(z_i)$ for all $i \in [|\mathbf{z}|]$, and $\boldsymbol{\pi} \leftarrow \mathsf{Eval}(\mathsf{ck}, c, d, \mathbf{z}, \mathbf{y}, f; \rho_c)$ denotes setting $\pi_i \leftarrow \mathsf{Eval}(\mathsf{ck}, c, d, z_i, y_i, f; \rho_c)$ for all $i \in [|\mathbf{z}|]$.

Non-Extrapolation. We define a new property related to ϕ-evaluation hiding of a PCS scheme with respect to a PES scheme. We require that, given a polynomial commitment and $\ell' > 0$ evaluations and proofs for an encoding of all-zero vector, no adversary can compute a valid proof for a new evaluation point. In other words, it is hard for an adversary to *extrapolate* a new evaluation given ℓ' evaluations *even when* the polynomial is fixed to be the encoding of all-zero vector. Non-extrapolation naturally follows from evaluation hiding and binding for many PCS plus PES schemes for the right choice of ϕ. We show this explicitly for the KZG polynomial commitment scheme in Section 4.

Definition 2.10 (ϕ-Non-Extrapolation). *Let PCS* = (KGen, Com, Eval, Check) *be a polynomial commitment scheme and PES* = (Enc, Dec) *be a polynomial encoding scheme. We say PCS supports* ϕ-non-extrapolation with respect to PES *if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, and*

$$\Pr\left[\begin{array}{c} v = 1 \ \wedge\ \mathbf{z} \in \mathcal{D}_{\mathsf{Enc}}^{|\mathbf{z}|} \\ \wedge\ z^* \in \mathcal{D}_{\mathsf{Enc}} \ \wedge\ z^* \notin \mathbf{z} \end{array} : \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{KGen}(1^\lambda, D); (n, \mathbf{z}) \leftarrow \mathcal{A}_1(\mathsf{ck}); \\ \ell := \phi(|\mathbf{z}|); d := n + \ell; \\ \boldsymbol{\rho}_w \leftarrow\!\!\$\ \mathbb{F}^\ell; \\ f \leftarrow \mathsf{Enc}(0^n, n, \ell; \boldsymbol{\rho}_w); \\ c \leftarrow \mathsf{Com}(\mathsf{ck}, f, d; \rho_c); \\ \mathbf{y} := f(\mathbf{z}); \\ \boldsymbol{\pi} \leftarrow \mathsf{Eval}(\mathsf{ck}, c, d, \mathbf{z}, \mathbf{y}, f; \rho_c); \\ (z^*, y^*, \pi^*) \leftarrow \mathcal{A}_2(c, \mathbf{y}, \boldsymbol{\pi}); \\ v \leftarrow \mathsf{Check}(\mathsf{ck}, c, d, z^*, y^*, \pi^*) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

where $\mathcal{A}_1$ and $\mathcal{A}_2$ share the internal states, $\mathbf{y} := f(\mathbf{z})$, $\boldsymbol{\pi}$ are as before.

3 Succinctness-Preserving UC NIZK Compiler

In this section, we describe a generic, succinctness-preserving compiler that takes as inputs: (1) a SIM-EXT NIZK proof system $\Pi_{\mathcal{R}} = (\mathsf{PGen}, \mathcal{O}_{\mathsf{Setup}}, \mathcal{P}, \mathcal{V})$ for the *arithmetic circuit satisfiability relation* $\mathcal{R} = \{(\mathcal{C}, w) : \mathcal{C}(w) = 1\}$, and (2) a PCS = (KGen, Com, Eval, Check) that is succinct, evaluation binding, unique proof, ϕ-evaluation hiding, and ϕ-non-extrapolation with respect to some encoding scheme PES = (Enc, Dec). The resulting protocol, denoted by $\Pi_{\text{UC-}\mathcal{R}}$, UC-realizes $\mathcal{F}_{\mathsf{NIZK}}$ in the ($\mathcal{G}_{\mathsf{RO}}$, $\mathcal{F}_{\mathsf{Setup}}$)-hybrid model, where $\mathcal{F}_{\mathsf{Setup}}$ is described in Fig. 2.

Protocol 1: $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$

The protocol $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ is parameterized by: security parameter λ, finite field $\mathbb{F}$, evaluation domain $\mathcal{D}_{\mathsf{Enc}}$ for PES, evaluation hiding expansion factor $\phi : \mathbb{Z}^+ \to \mathbb{Z}^+$, number of parallel repetitions $r = r(\lambda) > 0$, proof-of-work parameter $b(\lambda) > 0$, bound $T(\lambda) > 0$, and maximum degree bound $D > 0$ for PCS.

– **Proof** Upon receiving input $(\textsc{prove}, \mathsf{sid}, \mathsf{ssid}, \mathcal{C}, w)$, ignore if $\mathcal{C}(w) \neq 1$. Otherwise, P_i does:
  1. Send $(\textsc{genparams}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{Setup}}$ and wait for answer $(\textsc{params}, \mathsf{sid}, \mathsf{pp})$.
  2. Send $(\textsc{genkey}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{Setup}}$ and wait for answer $(\textsc{comkey}, \mathsf{sid}, \mathsf{ck})$.
  3. Parse $w = \mathbf{w} \in \mathbb{F}^n$. Let $\ell := \phi(r)$ and $d := n + \ell$. If $d > D$, abort by outputting $(\textsc{proof}, \mathsf{sid}, \mathsf{ssid}, \perp)$.
  4. Generate a polynomial encoding of the witness vector: $f \leftarrow \mathsf{Enc}(\mathbf{w}, n, \ell; \boldsymbol{\rho}_w)$, where $\boldsymbol{\rho}_w \leftarrow\!\!\$\ \mathbb{F}^\ell$.
  5. Generate a commitment to the polynomial encoding: $c \leftarrow \mathsf{Com}(\mathsf{ck}, f, d; \rho_c)$, where the randomness ρ_c is sampled uniformly from the domain specified in PCS.
  6. Define the circuit $\mathcal{C}'$ such that it outputs 1 on input $w' = (\mathbf{w}, \boldsymbol{\rho}_w, \rho_c)$ if and only if the following conditions are met:
  $$\mathcal{C}(\mathbf{w}) = 1 \ \wedge \ c = \mathsf{Com}(\mathsf{ck}, \mathsf{Enc}(\mathbf{w}, n, \ell; \boldsymbol{\rho}_w), d; \rho_c)$$
  7. Run $\Pi_{\mathcal{R}}.\mathcal{P}$ on input pp, $\mathcal{C}'$, and w' to obtain a proof π'. Whenever $\mathcal{P}$ makes a call to $\mathcal{O}_{\mathsf{Setup}}$ with input in, send $(\textsc{setup}, \mathsf{sid}, \mathsf{in})$ to receive a response $(\textsc{setup}, \mathsf{sid}, \mathsf{out})$ and forward out to $\mathcal{P}$.
  8. Initialize empty sets $\mathbf{z}$, $\mathbf{y}$, and $\boldsymbol{\pi}_{\mathsf{PCS}}$.
  9. For each iteration $i \in [r]$ do:
     (a) Initialize counter $ctr := 0$ and an empty set of used evaluation points $\mathcal{D}_i$.
     (b) If $ctr = T$, abort by outputting $(\textsc{proof}, \mathsf{sid}, \mathsf{ssid}, \texttt{runout\_eval})$.
     (c) Sample an evaluation point: $z_i \leftarrow\!\!\$\ \mathcal{D}_{\mathsf{Enc}} \setminus \mathcal{D}_i$. Update $ctr := ctr + 1$. Update $\mathcal{D}_i := \mathcal{D}_i \cup \{z_i\}$.
     (d) Compute $y_i = f(z_i)$ and evaluation proof $\pi_i \leftarrow \mathsf{Eval}(\mathsf{ck}, c, d, z_i, y_i, f; \rho_c)$.
     (e) Send $(\textsc{query}, (\mathsf{sid}, (\mathcal{C}', c, z_i, y_i, \pi_i, i)))$ to $\mathcal{G}_{\mathsf{RO}}$. Upon receiving v from $\mathcal{G}_{\mathsf{RO}}$, if the first b bits of v are not 0^b, go to step 9b. Otherwise, store z_i, y_i, and π_i in $\mathbf{z}$, $\mathbf{y}$, and $\boldsymbol{\pi}_{\mathsf{PCS}}$, respectively.
  10. Output $(\textsc{proof}, \mathsf{sid}, \mathsf{ssid}, \varpi)$, where $\varpi := (\pi', c, \mathbf{z}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCS}})$.

– **Verification** Upon receiving input $(\textsc{verify}, \mathsf{sid}, \mathsf{ssid}, \mathcal{C}, \varpi)$, P_i does:
  1 Send $(\textsc{genparams}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{Setup}}$ and wait for answer $(\textsc{params}, \mathsf{sid}, \mathsf{pp})$.
  2 Send $(\textsc{genkey}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{Setup}}$ and wait for answer $(\textsc{comkey}, \mathsf{sid}, \mathsf{ck})$.
  3 Parse $\varpi = (\pi', c, \mathbf{z}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCS}})$. Derive the witness size n from the description of $\mathcal{C}$. Compute ℓ and d as **Proof** would and if $d > D$ abort by outputting $(\textsc{verification}, \mathsf{sid}, \mathsf{ssid}, 0)$.
  4 Define the circuit $\mathcal{C}'$ as **Proof** would.
  5 Parse $\mathbf{z} = (z_i)_{i \in [r]}$, $\mathbf{y} = (y_i)_{i \in [r]}$, and $\boldsymbol{\pi}_{\mathsf{PCS}} = (\pi_i)_{i \in [r]}$.
  6 Output $(\textsc{verification}, \mathsf{sid}, \mathsf{ssid}, 1)$ if all of the following checks pass, otherwise output $(\textsc{verification}, \mathsf{sid}, \mathsf{ssid}, 0)$:
     (a) $\Pi_{\mathcal{R}}.\mathcal{V}$ on input pp, $\mathcal{C}'$, and π' outputs 1. Calls to $\mathcal{O}_{\mathsf{Setup}}$ by $\mathcal{V}$ are handled similar to the above.
     (b) For all $i \in [r]$: $1 = \mathsf{Check}(\mathsf{ck}, c, d, z_i, y_i, \pi_i)$.
     (c) For all $i \in [r]$: send $(\textsc{query}, (\mathsf{sid}, (\mathcal{C}', c, z_i, y_i, \pi_i, i)))$ to $\mathcal{G}_{\mathsf{RO}}$, and the first b bits of the return value v_i are 0^b .

Fig. 4: Protocol for UC-secure non-interactive proof in the $(\mathcal{G}_{\mathsf{RO}}, \mathcal{F}_{\mathsf{Setup}})$-hybrid model. $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ internally runs $\Pi_{\mathcal{R}}$, PCS, and PES.

3.1 Security Proof

Theorem 3.1. *Let $\mathit{\Pi}_{\mathcal{R}}$ be a $\mathsf{SIM\text{-}EXT\ NIZK}$ proof system (Definition 2.5), for the arithmetic circuit satisfiability relation $\mathcal{R}$, with $\mathcal{O}_\lambda(1)$ size proofs. Let PCS be a polynomial commitment scheme with $\mathcal{O}_\lambda(1)$ size commitments and evaluation proofs, evaluation binding, unique proofs (Definition 2.6), ϕ-evaluation hiding (Definition 2.9), and supports ϕ-non-extrapolation (Definition 2.10) with respect to the encoding scheme $\mathsf{PES} = (\mathsf{Enc}, \mathsf{Dec})$ (Definition 2.8). Then, $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ described in Fig. 4 UC-realizes $\mathcal{F}_{\mathsf{NIZK}}$ in the $(\mathcal{G}_{\mathsf{RO}}, \mathcal{F}_{\mathsf{Setup}})$-hybrid model for relation $\mathcal{R}$ and has proofs of size $\mathcal{O}_\lambda(1)$.*

Proof. We prove the following properties.

Completeness. For a given commitment c and circuit $\mathcal{C}'$, an honest prover fails to generate a valid proof if, after trying at most T distinct evaluation points z_i's $\in \mathcal{D}_{\mathsf{Enc}}$, it fails to find any preimage such that it hashes to 0^b. As we will see, T is required to be only polynomially big in λ and so the prover is guaranteed to stop in polynomial time. For each iteration i, after fixing $c, \mathcal{C}', z_i$, the values $y_i = f(z_i)$ and π_i are derived uniquely. Thus, the honest prover fails in this iteration only if for all the T number of evaluation points $\mathcal{G}_{\mathsf{RO}}(\textsc{query}, (\mathsf{sid}, (\mathcal{C}', c, z_i, y_i, \pi_i, i))) \neq 0^b$. The prover fails overall if it fails in at least one of the iterations. Let the event of failing in iteration i be denoted by fail_i. For $T = (\lambda + \log(r)) \cdot 2^b$, the probability of the honest prover failing can be bounded as below.

$$\Pr[\mathsf{fail}] \leq \sum_{i=1}^{r} \Pr[\mathsf{fail}_i] = r \cdot \left(1 - \frac{1}{2^b}\right)^T \approx e^{\log(r)} \cdot \frac{1}{e^{(\lambda + \log(r))}} \leq 2^{-\lambda}$$

Thus, an honest prover manages to find a preimage of 0^b in polynomial time except with probability $2^{-\lambda}$.

Remark 3.2. Notice that the completeness error increases only additively even if the underlying proof system $\mathit{\Pi}_{\mathcal{R}}$ is statistically complete.[12] This is because the proof-of-work part is only executed after the NIZK proof π' is generated, and therefore an imperfect NIZK prover does not interfere with it. Concretely, if the probability that $\mathit{\Pi}_{\mathcal{R}}.\mathcal{P}$ fails to produce a valid proof is $\varepsilon(\lambda)$, the overall probability that $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ prover fails is at most $\varepsilon(\lambda) + \sum_{i=1}^{r} \Pr[\mathsf{fail}_i]$ by the union bound.

Simulation. We begin by sketching the overall simulation strategy. First, consider simulation for an uncorrupted prover. We simulate the behaviors of $\mathcal{F}_{\mathsf{Setup}}$ and π' component of real-world proofs produced by honest provers using the underlying NIZK simulator $\mathit{\Pi}_{\mathcal{R}}.\mathcal{S}$. After the first r queries to $\mathcal{G}_{\mathsf{RO}}$ are programmed to be 0^b, commitments to witness-encoding polynomials are replaced with simulated commitments to randomized polynomials encoding a dummy witness (i.e., 0-vector). This transition is justified by the evaluation hiding property (Definition 2.9). Then we stop programming the $\mathcal{G}_{\mathsf{RO}}$ responses in the next hybrid. At this stage, simulation of uncorrupted provers is essentially done.
Next, we describe simulation for an uncorrupted verifier. The requirement here is to extract a witness from whatever $(\tilde{\mathcal{C}}, \tilde{\varpi} = (\tilde{\pi}', \tilde{c}, \tilde{\mathbf{z}}, \tilde{\mathbf{y}}, \tilde{\boldsymbol{\pi}}_{\mathsf{PCS}}))$ submitted by uncorrupted verifiers unless they have been created during the simulation of uncorrupted provers. Here, we first rule out the case where at least one of $(\tilde{\mathbf{z}}, \tilde{\mathbf{y}}, \tilde{\boldsymbol{\pi}}_{\mathsf{PCS}})$ differs from previously simulated $(\mathbf{z}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCS}})$ for the same statement $\tilde{\mathcal{C}}$ and $\tilde{c}$. This can be shown by constructing a reduction to evaluation binding, evaluation hiding, or unique proof. Finally, the extraction algorithm interpolates the witness-encoding polynomial f for $\tilde{c}$ by observing $\mathcal{G}_{\mathsf{RO}}$ queries and decodes f to a candidate witness $w = \mathbf{w} \in \mathbb{F}^n$. The analysis concludes by bounding the probability that extracted w is invalid as follows. We run a non-black-box $\mathsf{SIM\text{-}EXT}$ extractor $\mathcal{E}_{\mathcal{Z}}$ of the underlying proof system against successful $\mathcal{Z}$ on statement the extended circuit $\tilde{\mathcal{C}}'$, and proof $\tilde{\pi}'$ to obtain another candidate witness $w' = (\mathbf{w}^*, \boldsymbol{\rho}_w, \rho_c)$. This fails in the case that $\tilde{\pi}'$ is a previously simulated proof. However, we rule this case out by relying on non-extrapolation property. Given this, the event $\tilde{\mathcal{C}}'(w') = 0$ happens only with negligible probability thanks to the simulation-extractability property. Hence, assuming $\tilde{\mathcal{C}}'(w') = 1$, it also holds that $\tilde{\mathcal{C}}(\mathbf{w}^*) = 1$ by the definition of $\tilde{\mathcal{C}}'$. Then we show that $\mathbf{w} = \mathbf{w}^*$. Otherwise, one can construct another witness-encoding polynomial $f^* \neq f$ that "explains" the same commitment $\tilde{c}$, breaking evaluation binding. With this we conclude that the extracted witness $\mathbf{w}$ is a valid witness as $\mathbf{w} = \mathbf{w}^*$ and $\tilde{\mathcal{C}}(\mathbf{w}^*) = 1$.
The above proof sketch describes simulation strategy for a single prover and verifier. In the formal proof, this is extended to incorporate multiple uncorrupted provers and verifiers in a session.

[12] We thank an anonymous reviewer for bringing this observation to our attension.

Let us turn to formal proof. Complete simulation algorithm is given in Fig. 5. The environment $\mathcal{Z}$ starts a session by initializing a certain number of parties and adversary $\mathcal{A}$. In a particular session sid, the environment $\mathcal{Z}$ instructs the parties with two commands: PROVE and VERIFY. The real world behavior is as follows. An honest party P_i on input $(\text{PROVE}, \mathsf{sid}, \mathsf{ssid}, \mathcal{C}, w)$ from $\mathcal{Z}$ executes the honest prover's algorithm in $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ to generate the proof. And on receiving $(\text{VERIFY}, \mathsf{sid}, \mathsf{ssid}, \mathcal{C}, \varpi)$, it verifies by running the honest verifier's algorithm. In the ideal world, the honest parties forward their inputs to the functionality $\mathcal{F}_{\mathsf{NIZK}}$. The corrupt parties' behavior is controlled by $\mathcal{A}$ in both the worlds. Within a session sid, we assume that $\mathcal{Z}$ issues s_1 queries of the type $(\text{PROVE}, \mathsf{sid}, \mathsf{ssid}, \mathcal{C}, w)$ meant for an honest party, and s_2 of the type $(\text{VERIFY}, \mathsf{sid}, \mathsf{ssid}, \mathcal{C}, \varpi)$ for either honest or corrupt party. Let $\mathsf{s} = \mathsf{s}_1 + \mathsf{s}_2$. Proofs for indistinguishability of hybrids are deferred to Appendix A.

- Hyb_0 : This is the real world.
- Hyb_1: Replace all the honest parties with a single party $\mathcal{B}$. This party is responsible for simulating the view of the adversary and the environment for the rest of the protocol. In particular, $\mathcal{B}$ acts on behalf of the honest parties and does exactly what an honest party would do in the real world. In addition, it intercepts the $\mathcal{G}_{\mathsf{RO}}$ queries made by any corrupt party P_i within the session, forwards it the $\mathcal{G}_{\mathsf{RO}}$ and relays the response back to P_i. Similarly, it intercepts all $\mathcal{F}_{\mathsf{Setup}}$ queries made by P_i in the session and relays it back and forth between $\mathcal{F}_{\mathsf{Setup}}$ and P_i.
- Hyb_2: Instead of forwarding P_i's calls to $\mathcal{F}_{\mathsf{Setup}}$ functionality, $\mathcal{B}$ answers them by executing steps in **Simulation of $\mathcal{F}_{\mathsf{Setup}}$** in Sim. The rest of the execution remains the same as before, i.e., the $\mathcal{B}$ executes on behalf of the honest parties by executing the honest algorithm.
- For $j \in [\mathsf{s}_1]$, Hyb_{2+j}: For the j-th PROVE command with input $(\mathcal{C}, w)$ for an honest party P_i from $\mathcal{Z}$, replace honest prover's algorithm with Step 1-7 in **Simulation of uncorrupted prover** (in Sim) for input $\mathcal{C}$.
- For $j \in [\mathsf{s}_2]$, $\mathsf{Hyb}_{2+\mathsf{s}_1+j}$: For the j-th VERIFY command with input $(\mathcal{C}, \varpi)$ for an honest party P_i from $\mathcal{Z}$, replace honest verifier's algorithm with Step 1-12 in **Simulation of uncorrupted verifier** (in Sim). We assume that all the VERIFY commands are made only by the honest parties. This is without loss of generality as any query that a corrupt party wants to make can instead be routed through an honest party via the environment.
- $\mathsf{Hyb}_{3+\mathsf{s}}$: This is the ideal world execution. Replace $\mathcal{B}$ with Sim, where the steps in Sim are executed for each $(\text{PROVE}, \mathsf{sid}, \mathsf{ssid}, \mathcal{C}, w)$, and $(\text{VERIFY}, \mathsf{sid}, \mathsf{ssid}, \mathcal{C}, \varpi)$ command (as explained in the above hybrids), and sends corresponding $(\text{PROOF}, \mathsf{sid}, \mathsf{ssid}, \mathsf{P}_i, \varpi)$ and $(\text{WITNESS}, \mathsf{sid}, \mathsf{ssid}, \mathsf{P}_i, w)$ to $\mathcal{F}_{\mathsf{NIZK}}$.

Succinctness. From completeness and simulation analysis we obtain the following constraints for parameters r, b, T: $T = (\lambda + \log(r)) \cdot 2^b$ and $\lambda = r(b - \log(d))$. Consider the simple parameter choice $r = \lambda$. This gives, $b = \log(d) + 1$ and $T = 2d(\lambda + \log(\lambda))$. More generally, the parameter choices, $r = \mathcal{O}(\lambda/\log(\lambda)) = \mathcal{O}_\lambda(1), b = \mathcal{O}(\log(d) + \log(\lambda)) = \mathcal{O}_\lambda(\log(d))$, and $T = \mathcal{O}((\lambda + \log(\lambda/\log(\lambda)))\lambda d) = \mathcal{O}_\lambda(d)$ satisfies the conditions.
Assume that PCS produces constant size $(\mathcal{O}_\lambda(1))$ commitments and evaluation proofs, and $\Pi_{\mathcal{R}}$ produces $\mathcal{O}_\lambda(1)$ size proofs. Later in Section 4 we discuss candidate schemes satisfying these constraints. The total size of the proof ϖ is one commitment c of size $\mathcal{O}_\lambda(1)$, vectors $\mathbf{z}, \mathbf{y}$ consisting of r field elements, r evaluation proofs π_i of size $\mathcal{O}_\lambda(1)$, and one NIZK proof π' for statement $\mathcal{C}'$. Recall that $\mathcal{C}'$ is composed of $\mathcal{C}$ and the circuit that describes the Com and Enc operations. Thus, $\mathcal{C}'$ is only $\mathcal{O}(\mathsf{poly}(\lambda, n))$ bigger than $\mathcal{C}$, where n is the witness size. Since $\Pi_{\mathcal{R}}$ produces constant size proofs, proof for $\mathcal{C}'$ is also of size $\mathcal{O}_\lambda(1)$. Finally, since, $r = \mathcal{O}_\lambda(1)$, the size of ϖ remains $\mathcal{O}_\lambda(1)$.

Remark 3.3. Here, r is independent of the degree of the polynomial. The proof size only grows with the number of repetitions and thus remains independent of the degree, assuming constant size PCS and NIZK $\Pi_{\mathcal{R}}$ proofs. However, the prover's computational effort increases with the increase in degree d.

4 Instantiating our Compiler

In this section, we discuss a few candidates for PCS, PES and NIZK schemes for instantiating our compiler.

Simulator: Sim

Sim is parameterized by $\lambda, \mathbb{F}, \mathcal{D}_{\mathsf{Enc}}, \phi, T, r, b, D$ and has access to the global functionality $\mathcal{G}_{\mathsf{RO}}$ as $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ does. It simulates real prover's proof for arbitrary $(\mathcal{C}, w) \in \mathcal{R}$, extracts a witness from a valid proof $(\mathcal{C}, \varpi)$ chosen by the environment (as long as it hasn't been recorded by $\mathcal{F}_{\mathsf{NIZK}}$), and simulates the local setup functionality $\mathcal{F}_{\mathsf{Setup}}$. It internally keeps track of the state information st of the underlying NIZK simulator $\Pi_{\mathcal{R}}.\mathcal{S}$, which is initially set to ε.

- **Initialization** follows [Gro06]: We use the notation $\tilde{\mathsf{P}}_i$ for a dummy party in the ideal process, which simply forwards inputs and outputs between the environment $\mathcal{Z}$ and the ideal functionality $\mathcal{F}_{\mathsf{NIZK}}$, and P_i for a simulated party. Sim starts by invoking a copy of a PPT adversary $\mathcal{A}$. It will run a simulated interaction of $\mathcal{A}$, the parties, and the environment. In particular, whenever $\mathcal{A}$ communicates with $\mathcal{Z}$, Sim just passes this information along. And whenever $\mathcal{A}$ corrupts a party P_i, Sim corrupts the corresponding dummy party $\tilde{\mathsf{P}}_i$.
- **Simulation of $\mathcal{F}_{\mathsf{Setup}}$**
  - **Parameters** Upon receiving input $(\text{GENPARAMS}, \mathsf{sid})$ from a party P_i, if no pp has been stored, run $\mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda)$, let $\mathsf{st} := \mathsf{pp}$, and store pp. Send $(\text{PARAMS}, \mathsf{sid}, \mathsf{pp})$ to P_i.
  - **Commitment Key** This is identical to $\mathcal{F}_{\mathsf{Setup}}$.
  - **Setup** Upon receiving input $(\text{SETUP}, \mathsf{sid}, \mathsf{in})$ from a party P_i, ignore if st has never been initialized with pp. Otherwise run $(\mathsf{out}, \mathsf{st}) \leftarrow \Pi_{\mathcal{R}}.\mathcal{S}(1, \mathsf{st}, \mathsf{in})$ using the current state and send $(\text{SETUP}, \mathsf{sid}, \mathsf{out})$ to P_i.
- **Handling $\mathcal{G}_{\mathsf{RO}}$ queries**
  1. Initialize empty set $\mathcal{Q}_{\mathsf{ro}}$.
  2. Upon receiving input (QUERY, x) from a party P_i, forward it to the $\mathcal{G}_{\mathsf{RO}}$ and forward the response v back to P_i.
  3. Record x in $\mathcal{Q}_{\mathsf{ro}}$.
- **Simulation of uncorrupted prover** Upon receiving input $(\text{PROVE}, \mathsf{sid}, \mathcal{C})$ from $\mathcal{F}_{\mathsf{NIZK}}$:
  1. Derive the witness size n from the description of $\mathcal{C}$. Compute ℓ and d as **Proof** of $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ would and if $d > D$ abort by outputting $(\text{PROOF}, \bot)$.
  2. Generate a polynomial encoding of dummy witness: $f \leftarrow \mathsf{Enc}(0^n, n, \ell; \boldsymbol{\rho}_w)$, where $\boldsymbol{\rho}_w \leftarrow\$ \mathbb{F}^\ell$.
  3. Generate a commitment to the polynomial encoding as **Proof** of $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ would: $c \leftarrow \mathsf{Com}(\mathsf{ck}, f, d; \rho_c)$.
  4. Define the circuit $\mathcal{C}'$ as **Proof** of $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ would.
  5. Run $\Pi_{\mathcal{R}}.\mathcal{S}(2, \mathsf{st}, \mathcal{C}')$ to obtain a proof-state pair (π', st).
  6. Create $\mathbf{z}$, $\mathbf{y}$, and $\boldsymbol{\pi}_{\mathsf{PCS}}$ as **Proof** of $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ would.
  7. Send (PROOF, ϖ) to $\mathcal{F}_{\mathsf{NIZK}}$, where $\varpi := (\pi', c, \mathbf{z}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCS}})$
- **Simulation of uncorrupted verifier** Upon receiving input $(\text{VERIFY}, \mathsf{sid}, \mathcal{C}, \varpi)$ from $\mathcal{F}_{\mathsf{NIZK}}$:
  1 Perform verification checks similar to **Verification** of $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$, but use pp and ck generated during the simulation of $\mathcal{F}_{\mathsf{Setup}}$. Calls to $\mathcal{O}_{\mathsf{Setup}}$ made by $\Pi_{\mathcal{R}}.\mathcal{V}$ are handled by running $(\mathsf{out}, \mathsf{st}) \leftarrow \Pi_{\mathcal{R}}.\mathcal{S}(1, \mathsf{st}, \mathsf{in})$ and forwarding out to $\mathcal{V}$. If invalid, send $(\text{WITNESS}, \bot_1)$ to $\mathcal{F}_{\mathsf{NIZK}}$. This will eventually cause $\mathcal{F}_{\mathsf{NIZK}}$ to output $(\text{VERIFICATION}, \mathsf{sid}, \mathsf{ssid}, 0)$ to a dummy party $\tilde{\mathsf{P}}_i$.

  2 Parse proof ϖ as $(\pi', c, \mathbf{z}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCS}})$.

  3 Query $\mathcal{G}_{\mathsf{RO}}$ on $(\text{OBSERVE}, \mathsf{sid})$ and receive the set of illegitimate queries $\mathcal{Q}_{|\mathsf{sid}}$.

  4 Update $\mathcal{Q}_{\mathsf{ro}} = \mathcal{Q}_{\mathsf{ro}} \cup \mathcal{Q}_{|\mathsf{sid}}$.

  5 Define circuit $\mathcal{C}'$ as **Verification** of $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ would.

  6 Define $\mathcal{Q}_c$ as the set of queries in $\mathcal{Q}_{\mathsf{ro}}$ of the form $(\text{QUERY}, (\mathsf{sid}, (\mathcal{C}', c, \cdot, \cdot, \cdot, \cdot)))$ such that evaluation proof is valid. If there are more than one queries with the same evaluation point z then, irrespective of the iteration i, include only the very first such query in $\mathcal{Q}_c$.

  7 In the set $\mathcal{Q}_c$, if for the same (c, z), there exists (y, π) and (y', π') such that $y \neq y'$ or $\pi \neq \pi'$, then set $w := \bot_2$ and go to 12.

  8 If $(\mathcal{C}', \pi')$ was previously generated by $\Pi_{\mathcal{R}}.\mathcal{S}$ then set $w := \bot_3$ and go to 12.

  9 If $|\mathcal{Q}_c| < d$ then set $w := \bot_4$ and go to 12.

  10 Otherwise, parse $\mathcal{Q}_c$ as tuples $\{(\mathcal{C}', c, z_j, y_j, \pi_j, i_j)\}$, where each z_j is distinct. Collect polynomial evaluations (z_j, y_j) and interpolate the polynomial f of degree $d-1$ such that for $j \in [d]$, $y_j = f(z_j)$.

  11 If $(\mathcal{C}, \mathsf{Dec}(f)) \notin \mathcal{R}$ set $w := \bot_5$; Else, set $w := \mathsf{Dec}(f)$.

  12 Send $(\text{WITNESS}, w)$ to $\mathcal{F}_{\mathsf{NIZK}}$.

Fig. 5: Simulator for $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$.

4.1 A Candidate PCS and PES Scheme

We show that using KZG commitments [KZG10] as the PCS scheme along with Coefficient (PES_1) or Lagrange (PES_2) encoding scheme (§ 2.3) satisfies all necessary conditions required to instantiate our compiler, i.e., it is succinct, has evaluation binding, has unique proofs, is evaluation hiding, and has non-extrapolation.

We describe the polynomial commitment scheme $\mathsf{PCS}_{\mathsf{KZG}} = (\mathsf{KGen}, \mathsf{Com}, \mathsf{Eval}, \mathsf{Check})$. The formalization below follows the deterministic scheme of [CHM⁺20, §C.2] supporting multiple degree bounds up to the maximum degree D. Note that if $d = D$, one can skip computing/checking $\hat{c}, \hat{\pi}$, and $\hat{y}$.

– $\mathsf{KGen}(1^\lambda, D)$: Generate the parameters of a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, g, h, e)$ where $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = q$ is prime, $\langle g\rangle = \mathbb{G}_1$, $\langle h\rangle = \mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable, non-degenerate bilinear map. The group order q also determines $\mathbb{F} := \mathbb{F}_q$ and a set of supported polynomials $\mathbb{F}_{<D}[X]$. Sample $\alpha \in \mathbb{F}$ uniformly, and compute $\sigma = (g, g^\alpha, \ldots, g^{\alpha^{D-1}}, h, h^\alpha)$. Output $\mathsf{ck} = (\mathcal{G}, \sigma)$.

– $\mathsf{Com}(\mathsf{ck}, f, d)$: On input ck, a polynomial $f \in \mathbb{F}_{<d}[X]$, and a degree bound $d \le D$, compute a shifted polynomial $\hat{f} = X^{D-d} \cdot f$, and generate a commitment as $\mathbf{c} = (g^{f(\alpha)}, g^{\hat{f}(\alpha)})$ and output $\mathbf{c}$.

– $\mathsf{Eval}(\mathsf{ck}, c, d, z, f(z), f)$: Compute $\omega(X) = (f(X)-f(z))/(X-z)$ and $\hat{\omega}(X) = (\hat{f}(X)-\hat{f}(z))/(X-z)$ where $\hat{f}$ is computed as above. Output $\boldsymbol{\pi} = (g^{\omega(\alpha)}, g^{\hat{\omega}(\alpha)}, \hat{f}(z))$.

– $\mathsf{Check}(\mathsf{ck}, \mathbf{c}, d, z, y, \boldsymbol{\pi})$: Parse $\mathbf{c} = (c, \hat{c})$ and $\boldsymbol{\pi} = (\pi, \hat{\pi}, \hat{y})$. Accept if and only if $e(c/g^y, h) = e(\pi, h^\alpha/h^z)$, $e(\hat{c}/g^{\hat{y}}, h) = e(\hat{\pi}, h^\alpha/h^z)$, and $\hat{y} = z^{D-d} \cdot y$.

The security of $\mathsf{PCS}_{\mathsf{KZG}}$ relies on the SDH assumption [BB04].

Definition 4.1 (SDH Assumption). *The strong Diffie-Hellman assumption (SDH) holds with respect to a bilinear group generator* BGen *if for all* PPT *adversaries $\mathcal{A}$ and degree bound $D > 0$,*

$$\Pr\left[t = g^{\frac{1}{\alpha+c}} \ : \ \mathcal{G} \leftarrow \mathsf{BGen}(1^\lambda); \alpha \leftarrow\!\!\$\ \mathbb{F}; \sigma := (\{g^{\alpha^i}\}_{i=0}^{D-1}, h^\alpha); (t, c) \leftarrow \mathcal{A}(\mathcal{G}, \sigma)\right] \le \mathsf{negl}(\lambda)$$

Lemma 4.2. $\mathsf{PCS}_{\mathsf{KZG}}$ *is perfectly unique (Definition 2.7), computationally evaluation binding under the SDH assumption, perfectly ϕ-evaluation hiding (Definition 2.9), and computationally ϕ-non-extrapolation (Definition 2.10) with respect to any polynomial encoding scheme* PES *with bounded independence (Definition 2.8), where $\phi(r) := r + 1$.*

Proof. **Unique Proof.** We prove there exists unique $\boldsymbol{\pi} = (\pi, \hat{\pi}, \hat{y})$ for a fixed $\mathbf{c} = (c, \hat{c})$, d, z, and y. Due to the pairing equation, a valid π is uniquely determined by $(c/g^y)^{\frac{1}{\alpha-z}}$. The same holds for $\hat{\pi}$. Finally, a valid $\hat{y}$ is uniquely determined by $z^{D-d}y$.

Evaluation Binding. Suppose the adversary outputs $\mathbf{c} = (c, \hat{c}), d, z, y, y' \ne y, \boldsymbol{\pi} = (\pi, \hat{\pi}, \hat{y}), \boldsymbol{\pi} = (\pi', \hat{\pi}', \hat{y}')$ such that both proofs verify. If $g^z = g^\alpha$, then SDH is broken with solution $(g^{1/z}, 0)$. Otherwise, we have $(\pi/\pi')^{\frac{1}{y'-y}} = g^{\frac{1}{\alpha-z}}$ thanks to the pairing equation and thus SDH is broken with solution $((\pi/\pi')^{\frac{1}{y'-y}}, -z)$.[13]

Evaluation Hiding. Let $r = |\mathbf{z}|$ be the number of evaluations requested by the adversary. Due to the bounded independence of PES, any set of $\phi(r) = r + 1$ evaluations of encoded polynomial f in $\mathcal{D}_{\mathsf{Enc}}$ are independently and uniformly distributed in $\mathbb{F}$. The commitment $\mathbf{c} = (c, \hat{c})$ leaks at most a single evaluation $f(\alpha)$. For $i \in [r]$, each proof $(\pi_i, \hat{\pi}_i, \hat{y}_i)$ leaks at most $f(\alpha)$ and $f(z_i)$. Overall, the adversary observes at most $r + 1$ evaluations of f, whose distribution is independent and uniform in $\mathbb{F}$.

Non-Extrapolation. For KZG polynomial commitment scheme used with PES, $\phi(r) := r + 1$. We show the following hybrids to prove non-extrapolation.

1. Hyb_0: The same as the game defined in Definition 2.10, i.e., an all-zero vector of length n is encoded as a polynomial and the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is provided with up to r distinct evaluations plus proofs.

2. Hyb_1: The challenger's code is changed as: Instead of encoding an all-0 vector, sample d random evaluations $y_i \leftarrow\!\!\$\ \mathbb{F}$. Recall, degree of the encoded polynomial is denoted by $d - 1$. Let $|\mathbf{z}|_{\mathsf{u}}$ denote

[13] Since the reduction only relies on the first component of the proof the scheme even satisfies a slightly stronger variant of evaluation binding where the adversary gets to choose distinct degree bounds for different evaluation proofs.

the number of distinct values in $\mathbf{z}$. Let $r' := |\mathbf{z}|_{\mathsf{u}}$ and $n' := d - r'$. Note that, when there are no repeat elements in $\mathbf{z}$, $r = r'$. Sample n' evaluation points from $\mathcal{D}_{\mathsf{Enc}}^{n'}$ and interpolate the polynomial f defined by d points (z_i, y_i), where the first r' z_i's are from $\mathcal{A}_1$, and the rest are sampled by the challenger. Computing commitments and evaluation proofs is same as before.

This hybrid remains indistinguishable from the previous one because of ϕ-evaluation hiding of the PCS scheme. In particular, up to $r+1$ distinct evaluations and proofs do not reveal anything about the underlying committed polynomial. For $i \in [r]$, each proof leaks at most one evaluation $f(z_i)$ and $f(\alpha)$. Thus, overall the adversary learns at most $r+1$ distinct evaluations only.

Now, after the execution of Hyb_1, $\mathcal{A}_2$ outputs a valid evaluation proof for a new point (z^*, y^*, π^*). Let $\tilde{y} = f(z^*)$. Since, the committed polynomial f is random and has degree $d-1 = n+r$, and $\mathcal{A}$ learns at most $r+1$, there is at least one degree of freedom corresponding to which the evaluation of f is uniformly distributed in $\mathbb{F}$. This implies that the probability of $y^* = \tilde{y}$ is $1/|\mathbb{F}|$, which is negligible. In case, $y^* \neq \tilde{y}$, the challenger obtains two different evaluations and valid proof for the same point which contradicts evaluation binding for the PCS scheme. Thus, $\mathcal{A}$ wins only with negligible probability.

4.2 Candidate NIZK Schemes

Our compiler lifts any simulation extractable SNARK (SE-SNARK) to a UC NIZK. Plugging in any SE-SNARK therefore yields a UC NIZK under the same assumptions. However, the security analyses of many SNARKs in the literature are in idealized models like the Generic Group Model (GGM) or Algebraic Group Model (AGM) [FKL18], and such analyses do not provide any guarantees outside of those models. As we wish to prove composition with respect to any environment (not just algebraic ones, for instance), the most meaningful candidates to plug into our compiler are those that provide guarantees about *any* adversary, even by making use of (non-black-box) knowledge assumptions.

Immediately Compatible SE-SNARKs. Groth and Maller [GM17] construct an SE-SNARK from a knowledge assumption that they formulate, called the eXtended Power Knowledge of Exponent (XPKE) assumption. Lipmaa [Lip19] presents SE-SNARKs under 'hash-algebraic' knowledge assumptions. Abdolmaleki et al. [ARS20] show how to lift any zk-SNARK to an SE-SNARK (with non-black-box extraction), and present a concrete instantiation based on the zk-SNARK of Groth et al. [GKM+18], which in turn relies on knowledge assumptions that they introduce. One could of course apply Abdolmaleki et al.'s approach to any $O_\lambda(1)$-sized zk-SNARK to obtain a $O_\lambda(1)$-sized SE-SNARK under the same knowledge assumptions. All of these SE-SNARKs are $O_\lambda(1)$-sized and can be plugged into our compiler to obtain $O_\lambda(1)$-sized UC NIZKs with provably secure composition with respect to any environment, under the same knowledge assumptions.

Remark 4.3. Note that, a typical UC-security definition does not pose any constraint on the auxiliary string z provided to the environment. However, when the above mentioned SNARKs are plugged into our compiler, the compiled SNARK is UC-secure assuming that the auxiliary string provided to the environment come from a *benign* distribution, i.e., it is benign with respect to the knowledge assumption under which the underlying SNARK is proven to be secure. This is a necessary assumption as otherwise extraction is known to be impossible [BCPR14, BP15].

Future Work: Alternative Instantiations. While we have been focused on obtaining $O_\lambda(1)$-sized UC NIZKs in this paper, our compiler can be more widely applicable. In general, given a NIZK that produces proofs of size $\mathcal{O}_\lambda(f(|C|+|w|))$ and a polynomial commitment scheme that produces evaluation proofs of size $\mathcal{O}_\lambda(g(|w|))$ for some functions f, g, our compiler produces a UC NIZK (in the ROM) where the proofs are of size $O_\lambda(f(|C|+|w|)+g(|w|))$, under the same setup and knowledge assumptions as the NIZK and polynomial commitment. With the right input SNARKs, we can obtain witness-succinct UC NIZKs that have benefits orthogonal to $O_\lambda(1)$-sized proofs. Consider the following:

– One interesting question here is regarding *'transparent' input SNARK*— one that does not require a structured common reference string—would result in a transparent UC NIZK with the same succinctness upon applying our compiler. For instance, the recent work of Arun et al. [AGL+22] gives such a constant sized transparent SNARK using class groups, however their analysis is in the generic group model, and simulation extractability of their construction has yet to be analyzed.

– If one were to plug in a SNARK that does not require non-black-box knowledge assumptions, we would obtain a UC NIZK that does not either. For instance, plugging in Bulletproofs [BBB+18] into our compiler with a transparent polynomial commitment scheme (in the ROM) would result in a $O_\lambda(\log(|C| + |w|))$ sized transparent NIZK in the ROM alone, that does not rely on any knowledge assumptions, and only assumes the hardness of computing discrete logarithms. Although Bulletproofs was recently shown to satisfy simulation-extractability only in the random oracle model [GOP+23, DG23], a caveat is that zero-knowledge simulator of Fiat-Shamir Bulletproofs requires a programmable random oracle, which either requires non-global random oracle [Nie02] or the programmable variant of $\mathcal{G}_{\mathsf{RO}}$ [CDG+18].

The scope of this paper is limited to the design and analysis of our general compiler, and so we leave such custom instantiations to future work.

Acknowledgment

The authors would like to thank abhi shelat for helpful discussions about an early version of this work. We thank Markulf Kohlweiss for his suggestions on presentation of our result. We thank anonymous reviewers of Eurocrypt 2023 for valuable comments and suggestions.

The work described in this paper has received funding from: the Concordium Blockhain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC); Core Research Grant CRG/2020/004488, SERB, Department of Science and Technology; Infosys Young Investigator Award, Infosys Foundation, Bangalore; the Protocol Labs Research Grant Program PL-RGP1-2021-064.

References

AB19. S. Atapoor and K. Baghery. Simulation extractability in groth's zk-SNARK. Cryptology ePrint Archive, Report 2019/641, 2019. https://eprint.iacr.org/2019/641.

ABK+21. M. Abdalla, M. Barbosa, J. Katz, J. Loss, and J. Xu. Algebraic adversaries in the universal composability framework. In *ASIACRYPT 2021, Part III*, vol. 13092 of *LNCS*, pp. 311–341. Springer, Heidelberg, 2021.

AGL+22. A. Arun, C. Ganesh, S. Lokam, T. Mopuri, and S. Sridhar. Dew: Transparent constant-sized zkSNARKs. Cryptology ePrint Archive, Report 2022/419, 2022. https://eprint.iacr.org/2022/419.

AHIV17. S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *ACM CCS 2017*, pp. 2087–2104. ACM Press, 2017.

ARS20. B. Abdolmaleki, S. Ramacher, and D. Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In *ACM CCS 2020*, pp. 1987–2005. ACM Press, 2020.

Bag19. K. Baghery. Subversion-resistant simulation (knowledge) sound NIZKs. In *17th IMA International Conference on Cryptography and Coding*, vol. 11929 of *LNCS*, pp. 42–63. Springer, Heidelberg, 2019.

BB04. D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *EUROCRYPT 2004*, vol. 3027 of *LNCS*, pp. 223–238. Springer, Heidelberg, 2004.

BBB+18. 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*, pp. 315–334. IEEE Computer Society Press, 2018.

BBHR18. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. https://eprint.iacr.org/2018/046.

BCC+16. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT 2016, Part II*, vol. 9666 of *LNCS*, pp. 327–357. Springer, Heidelberg, 2016.

BCGV16. E. Ben-Sasson, A. Chiesa, A. Gabizon, and M. Virza. Quasi-linear size zero knowledge from linear-algebraic PCPs. In *TCC 2016-A, Part II*, vol. 9563 of *LNCS*, pp. 33–64. Springer, Heidelberg, 2016.

BCMS20. B. Bünz, A. Chiesa, P. Mishra, and N. Spooner. Recursive proof composition from accumulation schemes. In *TCC 2020, Part II*, vol. 12551 of *LNCS*, pp. 1–18. Springer, Heidelberg, 2020.

BCPR14. N. Bitansky, R. Canetti, O. Paneth, and A. Rosen. On the existence of extractable one-way functions. In *46th ACM STOC*, pp. 505–514. ACM Press, 2014.

BCR$^+$19. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT 2019, Part I*, vol. 11476 of *LNCS*, pp. 103–128. Springer, Heidelberg, 2019.

BCS16. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *TCC 2016-B, Part II*, vol. 9986 of *LNCS*, pp. 31–60. Springer, Heidelberg, 2016.

BFM90. M. Blum, P. Feldman, and S. Micali. Proving security against chosen cyphertext attacks. In *CRYPTO'88*, vol. 403 of *LNCS*, pp. 256–268. Springer, Heidelberg, 1990.

BG18. S. Bowe and A. Gabizon. Making groth's zk-snark simulation extractable in the random oracle model. Cryptology ePrint Archive, Paper 2018/187, 2018. https://eprint.iacr.org/2018/187.

BGG19. S. Bowe, A. Gabizon, and M. D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. In *FC 2018 Workshops*, vol. 10958 of *LNCS*, pp. 64–77. Springer, Heidelberg, 2019.

BKSV21. K. Baghery, M. Kohlweiss, J. Siim, and M. Volkhov. Another look at extraction and randomization of groth's zk-snark. In *FC 2021*, vol. 12674 of *Lecture Notes in Computer Science*, pp. 457–475. Springer, 2021.

BP15. E. Boyle and R. Pass. Limits of extractability assumptions with distributional auxiliary input. In *ASIACRYPT 2015, Part II*, vol. 9453 of *LNCS*, pp. 236–261. Springer, Heidelberg, 2015.

BPR20. K. Baghery, Z. Pindado, and C. Ràfols. Simulation extractable versions of groth's zk-SNARK revisited. In *CANS 20*, vol. 12579 of *LNCS*, pp. 453–461. Springer, Heidelberg, 2020.

BR93. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pp. 62–73. ACM Press, 1993.

BS21. K. Baghery and M. Sedaghat. Tiramisu: Black-box simulation extractable nizks in the updatable CRS model. In *CANS 2021*, vol. 13099 of *Lecture Notes in Computer Science*, pp. 531–551. Springer, 2021.

BSBHR18. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018. https://eprint.iacr.org/2018/046.

BSMP91. M. Blum, A. D. Santis, S. Micali, and G. Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

Can01. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pp. 136–145. IEEE Computer Society Press, 2001.

CD00. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *ASIACRYPT 2000*, vol. 1976 of *LNCS*, pp. 331–345. Springer, Heidelberg, 2000.

CDG$^+$18. J. Camenisch, M. Drijvers, T. Gagliardoni, A. Lehmann, and G. Neven. The wonderful world of global random oracles. In *EUROCRYPT 2018, Part I*, vol. 10820 of *LNCS*, pp. 280–312. Springer, Heidelberg, 2018.

CDPW07. R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In *TCC 2007*, vol. 4392 of *LNCS*, pp. 61–85. Springer, Heidelberg, 2007.

CFF$^+$21. 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 *ASIACRYPT 2021, Part III*, vol. 13092 of *LNCS*, pp. 3–33. Springer, Heidelberg, 2021.

CGKS22. M. Campanelli, C. Ganesh, H. Khoshakhlagh, and J. Siim. Impossibilities in succinct arguments: Black-box extraction and more. Cryptology ePrint Archive, Report 2022/638, 2022. https://eprint.iacr.org/2022/638.

CHM$^+$20. A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zk-SNARKs with universal and updatable SRS. In *EUROCRYPT 2020, Part I*, vol. 12105 of *LNCS*, pp. 738–768. Springer, Heidelberg, 2020.

CJS14. R. Canetti, A. Jain, and A. Scafuro. Practical UC security with a global random oracle. In *ACM CCS 2014*, pp. 597–608. ACM Press, 2014.

CLOS02. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pp. 494–503. ACM Press, 2002.

COS20. A. Chiesa, D. Ojha, and N. Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In *EUROCRYPT 2020, Part I*, vol. 12105 of *LNCS*, pp. 769–793. Springer, Heidelberg, 2020.

CSW20. R. Canetti, P. Sarkar, and X. Wang. Triply adaptive UC NIZK. Cryptology ePrint Archive, Report 2020/1212, 2020. https://eprint.iacr.org/2020/1212.

DDN91. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pp. 542–552. ACM Press, 1991.

DDO$^+$01. A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *CRYPTO 2001*, vol. 2139 of *LNCS*, pp. 566–598. Springer, Heidelberg, 2001.

DG23. Q. Dao and P. Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). Cryptology ePrint Archive, Paper 2023/494, 2023. https://eprint.iacr.org/2023/494.

DSW08. Y. Dodis, V. Shoup, and S. Walfish. Efficient constructions of composable commitments and zero-knowledge proofs. In *CRYPTO 2008*, vol. 5157 of *LNCS*, pp. 515–535. Springer, Heidelberg, 2008.

Fis05. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO 2005*, vol. 3621 of *LNCS*, pp. 152–168. Springer, Heidelberg, 2005.

FKL18. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *CRYPTO 2018, Part II*, vol. 10992 of *LNCS*, pp. 33–62. Springer, Heidelberg, 2018.

FKMV12. S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In *INDOCRYPT 2012*, vol. 7668 of *LNCS*, pp. 60–79. Springer, Heidelberg, 2012.

GGPR13. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT 2013*, vol. 7881 of *LNCS*, pp. 626–645. Springer, Heidelberg, 2013.

GH98. O. Goldreich and J. Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, 1998.

GKK$^+$22. C. Ganesh, H. Khoshakhlagh, M. Kohlweiss, A. Nitulescu, and M. Zajac. What makes fiat-shamir zksnarks (updatable SRS) simulation extractable? In *SCN 2022*, vol. 13409 of *Lecture Notes in Computer Science*, pp. 735–760. Springer, 2022.

GKM$^+$18. J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In *CRYPTO 2018, Part III*, vol. 10993 of *LNCS*, pp. 698–728. Springer, Heidelberg, 2018.

GM17. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In *CRYPTO 2017, Part II*, vol. 10402 of *LNCS*, pp. 581–612. Springer, Heidelberg, 2017.

GMY06. J. A. Garay, P. D. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, 2006.

GOP$^+$22. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In *EUROCRYPT 2022, Part II*, vol. 13276 of *LNCS*, pp. 397–426. Springer, Heidelberg, 2022.

GOP$^+$23. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-shamir bulletproofs are non-malleable (in the random oracle model). Cryptology ePrint Archive, Paper 2023/147, 2023. https://eprint.iacr.org/2023/147.

GOS06. J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT 2006*, vol. 4004 of *LNCS*, pp. 339–358. Springer, Heidelberg, 2006.

GOS12. J. Groth, R. Ostrovsky, and A. Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012.

Gro06. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT 2006*, vol. 4284 of *LNCS*, pp. 444–459. Springer, Heidelberg, 2006.

Gro16. J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016, Part II*, vol. 9666 of *LNCS*, pp. 305–326. Springer, Heidelberg, 2016.

GVW02. O. Goldreich, S. Vadhan, and A. Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1):1–53, 2002.

GWC19. 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.

JP14. A. Jain and O. Pandey. Non-malleable zero knowledge: Black-box constructions and definitional relationships. In *SCN 14*, vol. 8642 of *LNCS*, pp. 435–454. Springer, Heidelberg, 2014.

Kat21. S. Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. In *CRYPTO 2021, Part II*, vol. 12826 of *LNCS*, pp. 580–610, Virtual Event, 2021. Springer, Heidelberg.

Kil92. J. Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pp. 723–732. ACM Press, 1992.

KKK21. T. Kerber, A. Kiayias, and M. Kohlweiss. Composition with knowledge assumptions. In *CRYPTO 2021, Part IV*, vol. 12828 of *LNCS*, pp. 364–393, Virtual Event, 2021. Springer, Heidelberg.

KPV19. A. Kattis, K. Panarin, and A. Vlasov. RedShift: Transparent SNARKs from list polynomial commitment IOPs. Cryptology ePrint Archive, Report 2019/1400, 2019. https://eprint.iacr.org/2019/1400.

Ks22. Y. Kondi and a. shelat. Improved straight-line extraction in the random oracle model with applications to signature aggregation. Cryptology ePrint Archive, Report 2022/393, 2022. https://eprint.iacr.org/2022/393.

KZG10. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT 2010*, vol. 6477 of *LNCS*, pp. 177–194. Springer, Heidelberg, 2010.

KZM+15. A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. Papamanthou, R. Pass, a. shelat, and E. Shi. C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093, 2015. https://eprint.iacr.org/2015/1093.

Lip19. H. Lipmaa. Simulation-extractable SNARKs revisited. Cryptology ePrint Archive, Report 2019/612, 2019. https://eprint.iacr.org/2019/612.

LR22a. A. Lysyanskaya and L. N. Rosenbloom. Efficient and universally composable non-interactive zero-knowledge proofs of knowledge with security against adaptive corruptions. Cryptology ePrint Archive, Paper 2022/1484, 2022. https://eprint.iacr.org/2022/1484.

LR22b. A. Lysyanskaya and L. N. Rosenbloom. Universally composable sigma-protocols in the global random-oracle model. Cryptology ePrint Archive, Report 2022/290, 2022. https://eprint.iacr.org/2022/290.

Mau11. U. Maurer. Constructive cryptography - A new paradigm for security definitions and proofs. In *Theory of Security and Applications - Joint Workshop, TOSCA 2011,*, vol. 6993 of *LNCS*, pp. 33–56. Springer, 2011.

MBKM19. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In *ACM CCS 2019*, pp. 2111–2128. ACM Press, 2019.

Mic00. S. Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

Nie02. J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO 2002*, vol. 2442 of *LNCS*, pp. 111–126. Springer, Heidelberg, 2002.

Pas03. R. Pass. On deniability in the common reference string and random oracle model. In *CRYPTO 2003*, vol. 2729 of *LNCS*, pp. 316–337. Springer, Heidelberg, 2003.

PHGR13. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pp. 238–252. IEEE Computer Society Press, 2013.

PR05. R. Pass and A. Rosen. New and improved constructions of non-malleable cryptographic protocols. In *37th ACM STOC*, pp. 533–542. ACM Press, 2005.

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

Unr15. D. Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *EUROCRYPT 2015, Part II*, vol. 9057 of *LNCS*, pp. 755–784. Springer, Heidelberg, 2015.

A Omitted Proofs

In this part we prove sub-claims for proving Theorem 3.1.

Lemma A.1. Hyb_0 *and* Hyb_1 *are indistinguishable.*

Proof. In Hyb_1, $\mathcal{B}$ simulates the behavior of the honest parties in the real world identically. Additionally it only intercepts all $\mathcal{G}_{\mathsf{RO}}$ and $\mathcal{F}_{\mathsf{Setup}}$ queries and does not tamper them in any way. Thus, the view of $\mathcal{Z}$ is identical in the two hybrids.

Lemma A.2. Hyb_1 *and* Hyb_2 *are indistinguishable.*

Proof. Indistinguishability with the previous hybrids is guaranteed by zero-knowledge property of $\Pi_{\mathcal{R}}.\mathcal{S}$. In particular, since $\Pi_{\mathcal{R}}$ satisfies zero-knowledge (Definition 2.4), output of $\Pi_{\mathcal{R}}.\mathcal{S}(1, \mathsf{st}, \mathsf{in})$ is distributed indistinguishably from that of $\mathcal{O}_{\mathsf{Setup}}$.

Lemma A.3. *For* $j = 0, \ldots, \mathsf{s}_1 - 1$, Hyb_{2+j} *and* Hyb_{2+j+1} *are indistinguishable.*

Proof. Recall that $\Pi_{\mathcal{R}}$ is a NIZK with straight-line simulation in the $\mathcal{F}_{\mathsf{Setup}}$- hybrid model. The difference between two consecutive hybrids is that in Hyb_{2+j} the j-th PROVE command for an honest party is performed by executing the honest prover's algorithm in $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$, while in Hyb_{2+j+1}, this is done by executing the simulator Sim's steps in **Simulation of uncorrupted prover** (Step 1 to 7). We argue indistinguishability via a series of sub-hybrids. In these sub-hybrids, we only describe execution with respect to the j-th PROVE command for an honest party.

1. $\mathsf{Hyb}_{j,0}$: This is the same as Hyb_{2+j}.
2. $\mathsf{Hyb}_{j,1}$: This is the same as $\mathsf{Hyb}_{j,0}$ except the following differences. For the j-th PROVE command for an honest party P_i, generate a simulated proof π' instead of an honest proof. In particular, execute steps 1 through 6 in $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$ for input $\mathcal{C}, w$ and compute $\mathcal{C}', w'$ as the honest prover would.

Then in place of step 7 (i.e., $\Pi_{\mathcal{R}}.\mathcal{P}(\mathsf{pp}, \mathcal{C}', w')$), implement step 5 from Sim (i.e., $\Pi_{\mathcal{R}}.\mathcal{S}(2, \mathsf{st}, \mathcal{C}')$) to generate proof π'.

The only difference between $\mathsf{Hyb}_{j,0}$ and $\mathsf{Hyb}_{j,1}$ is in the way π' is computed. We argue that this difference is indistinguishable because of zero-knowledge property (Definition 2.4) of $\Pi_{\mathcal{R}}$. In particular, a distinguisher for the two hybrids can be used to build a distinguisher who breaks zero-knowledge property of $\Pi_{\mathcal{R}}$.

Let $\mathcal{A}_{\mathsf{H}}$ be the adversary who distinguishes between $\mathsf{Hyb}_{j,0}$ and $\mathsf{Hyb}_{j,1}$ with advantage ε. Note that since $\Pi_{\mathcal{R}}$ is a NIZK it implies that the following indistinguishability holds:

$$\left| \Pr\left[b = 1 \ : \ \begin{matrix} \mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda); \\ b \leftarrow \mathcal{A}^{\mathcal{S}_1, \mathcal{P}}(\mathsf{pp}) \end{matrix} \right] - \Pr\left[b = 1 \ : \ \begin{matrix} \mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda); \\ b \leftarrow \mathcal{A}^{\mathcal{S}_1, \mathcal{S}_2}(\mathsf{pp}) \end{matrix} \right] \right| < \mathsf{negl}(\lambda). \tag{1}$$

The only difference with respect to the original NIZK definition is that, here in the "real-world" experiment, $\mathcal{A}$ runs with access to $\mathcal{S}_1$ and $\mathcal{P}$ instead of $\mathcal{O}_{\mathsf{Setup}}$ and $\mathcal{P}$. Given $\mathcal{A}_{\mathsf{H}}$, we construct an efficient distinguishing adversary $\mathcal{A}_{\mathsf{NIZK}}$ that succeeds in distinguishing the two events in Equation. 1 with advantage ε.

$\mathcal{A}_{\mathsf{NIZK}}$ receives as input the set of commands issued by $\mathcal{Z}$, and black-box access to the adversary $\mathcal{A}_{\mathsf{H}}$. It starts an experiment with the NIZK challenger. It simulates the view of $\mathcal{A}_{\mathsf{H}}$ with respect to the commands issued by $\mathcal{Z}$ by running the simulated code for the first $j-1$ PROVE commands, and running the honest prover's algorithm for the $j+1, \ldots, \mathsf{s}_1$ commands. All the $\mathcal{F}_{\mathsf{Setup}}$ commands are forwarded to the NIZK challenger who answers them by running $\mathcal{S}_1$. $\mathcal{A}_{\mathsf{NIZK}}$ simulates the view for $\mathcal{G}_{\mathsf{RO}}$ queries by creating an instance of the functionality locally and behaving just as the functionality. For the j-th command PROVE, $\mathcal{A}_{\mathsf{NIZK}}$ does the following.

– Run steps 1 to 6 honestly as described in $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$(**Proof**). During this execution, $\mathcal{A}_{\mathsf{NIZK}}$ queries $\mathcal{G}_{\mathsf{RO}}$ directly whenever necessary.

– Defines an input statement $\mathcal{C}'$ from $\mathcal{C}$ and c.

– Query NIZK challenger on input $\mathcal{C}'$ and $w' = (\mathbf{w}, \boldsymbol{\rho}_w, \rho_c)$, and receive a proof π' which is either real or simulated.

– Embed the challenge distribution as the step 7 message π'.

– When $\mathcal{A}_{\mathsf{H}}$ outputs bit b', $\mathcal{A}_{\mathsf{NIZK}}$ outputs the same bit.

When $\mathcal{A}_{\mathsf{NIZK}}$ receives the real world distribution, π' is generated according to $\Pi_{\mathcal{R}}$. This correctly emulates $\mathsf{Hyb}_{j,0}$. Similarly, when $\mathcal{A}_{\mathsf{NIZK}}$ receives the ideal world distribution, π' is generated with the help of a simulator $\Pi_{\mathcal{R}}.\mathcal{S}$. Thus, the distinguishing probability of $\mathcal{A}_{\mathsf{NIZK}}$ is the same as that of $\mathcal{A}_{\mathsf{H}}$.

3. $\mathsf{Hyb}_{j,2}$: This is the same as $\mathsf{Hyb}_{j,1}$ except that, in the j-th command to an honest party P_i, the first r $\mathcal{G}_{\mathsf{RO}}$ queries made by P_i are programmed to be 0^b. Formally, this hybrid executes the following steps for the j-th command. The simulation for the rest of the commands are handled as before.

   – Implement steps 1 to 6 from $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$(**Proof**).

   – Implement step 7 as in $\mathsf{Hyb}_{j,1}$, i.e., by computing $\Pi_{\mathcal{R}}.\mathcal{S}(2, \cdot, \cdot)$.

   – For each $i \in [r]$, pick an evaluation point z_i at random from $\mathcal{D}_{\mathsf{Enc}}$.

   – Evaluate evaluation proofs for each of these points and program $\mathcal{G}_{\mathsf{RO}}$ output at points $(\mathsf{sid}, (\mathcal{C}', c, z_i, y_i, \pi_i, i))$ to be 0^b.

   There are two differences here compared to the last hybrid. First, here, an honest will always break out of the loop successfully (right after r runs), while in the previous hybrid an honest prover might fail to find a satisfying prefix. The latter event is bounded by the completeness error, which as shown earlier is negligible. The second difference is that the $\mathcal{G}_{\mathsf{RO}}$ outputs are programmed at r points. We will argue that, assuming that the prover does not fail in the previous hybrid, the view generated by programming the RO in this experiment is statistically indistinguishable from that of the last hybrid. The difference in the view in the two hybrids is with respect to $(\mathcal{C}', c, z_i, y_i, \pi_i, i)$ and queries to $\mathcal{G}_{\mathsf{RO}}$ (both QUERY and OBSERVE). Recall that $\mathcal{G}_{\mathsf{RO}}$ queries made in the session are not recorded by the functionality and cannot be observed. Thus queries to $\mathcal{G}_{\mathsf{RO}}$ on OBSERVE command remain the same in the two hybrids. Now, consider the distribution of tuple $(\mathcal{C}', c, z_i, y_i, \pi_i, i)$. $\mathcal{C}', c$ are generated honestly in both hybrids and given z_i, the values y_i, π_i are unique. Hence, the only difference is in the way the honest prover obtains the evaluation points z_i. In $\mathsf{Hyb}_{j,1}$ it finds z_i by sampling the challenge space randomly and querying the $\mathcal{G}_{\mathsf{RO}}$ till it finds one of the preimage of 0^b. Thus, each z_i is distributed uniformly in $\mathcal{D}_{\mathsf{Enc}}$. $\mathbf{z}$ is clearly uniformly random in $\mathsf{Hyb}_{j,2}$ as the prover picks them at random. This implies that the tuple $(\mathcal{C}', c, z_i, y_i, \pi_i, i)$ is distributed identically in both hybrids. Finally, we argue indistinguishability for $\mathcal{G}_{\mathsf{RO}}$ responses on command QUERY. The difference between the two hybrids lies in the programming of $\mathcal{G}_{\mathsf{RO}}$ in $\mathsf{Hyb}_{j,2}$. First we note that a distinguisher is successful if it is able to query $\mathcal{G}_{\mathsf{RO}}$ on a point that is later programmed to output 0. As c is generated honestly and has high min-entropy (see Remark A.4 below), the probability of this event occurring is negligible. Next, we argue that (conditioned on no points being queried prior to programming) the distribution of the responses of $\mathcal{G}_{\mathsf{RO}}$ is not noticeably different between the hybrids. Let zc_1 and zc_2 be random variables that count the number of queries of the form (QUERY, $(\mathsf{sid}, (\mathcal{C}', c, z_i, y_i, \pi_i, i))$) $(\forall z_i \in \mathcal{D}_{\mathsf{Enc}})$ that result in 0 as a response, in $\mathsf{Hyb}_{j,1}$ and $\mathsf{Hyb}_{j,2}$ respectively. Observe that zc_2 is effectively zc_1 shifted by (at most) r, to account for the r programmed points. Since zc_1 follows the Binomial distribution with parameters $|\mathcal{D}_{\mathsf{Enc}}| = O(2^\lambda)$ tries and probability $2^{-b} \in 1/\mathsf{poly}(\lambda)$ of success—meaning that $\mathsf{zc}_1 \in O(2^\lambda)$—and $r \in \mathsf{poly}(\lambda)$, the statistical distance between zc_1 and zc_2 is negligible. This suffices to show the indistinguishability of the responses of $\mathcal{G}_{\mathsf{RO}}$, as queries that are not of this form and the remaining non-zero responses are unaffected. Thus, $\mathcal{G}_{\mathsf{RO}}$ responses on QUERY command are indistinguishable as well.

Remark A.4. We note that evaluation hiding for a polynomial commitment scheme already implies high min-entropy. To see why this is true, suppose that the commitment scheme does not have high min-entropy. Then, there exists a value $\mathbf{w}^*$, a commitment value c^*, and a polynomial $\mathsf{poly}(\lambda)$ such that $\Pr[\mathsf{Com}(\mathsf{ck}, \mathsf{Enc}(\mathbf{w}^*, n, \ell; \boldsymbol{\rho}_w), d; \rho_c) = c^*] \geq 1/\mathsf{poly}(\lambda)$, where the probability is taken over choice of coins $\boldsymbol{\rho}_w, \rho_c$, and ck, n, ℓ, d are defined according to Definition 2.9. Let $\mathcal{A}_{\mathsf{ev}} = (\mathcal{A}_1, \mathcal{A}_2)$ be the adversary against evaluation hiding game. $\mathcal{A}_{\mathsf{ev}}$ on receiving ck outputs $\mathbf{w}^*$ and in response receives a commitment corresponding to either $\mathbf{w}^*$ or to an all-0 vector. If $\mathcal{A}_2$ receives the commitment value c^*, it outputs its decision bit as 1, and otherwise, 0. If the commitment corresponding to the all-0 vector takes value c^* only with negligible probability (which we argue below), $\mathcal{A}_{\mathsf{ev}}$ succeeds in distinguishing with non-negligible probability.

Observe that a commitment for the encoding of all-0 vector will take the value c^* only with negligible probability as otherwise there exists an adversary $\mathcal{A}_{\mathsf{eb}}$ against the evaluation binding property for the commitment scheme. $\mathcal{A}_{\mathsf{eb}}$ fixes commitment c^*, and messages $\mathbf{w}^*, 0^n$. It then finds opening information $(\mathbf{w}^*, \boldsymbol{\rho})$ by sampling random coins, encoding $\mathbf{w}^*$, committing to the encoded polynomial, and checking if this is equal to c^*. $\mathcal{A}_{\mathsf{eb}}$ does the same for 0^n to obtain opening information $(0^n, \boldsymbol{\rho}')$. Since the commitment takes value c^* with non-negligible probability, it is efficient to find this opening information. Thus, for the same commitment c^*, $\mathcal{A}_{\mathsf{eb}}$ can now produce two sets of valid evaluations and proofs (one each for the polynomial encoding $\mathbf{w}^*$ and 0^n).

4. $\mathsf{Hyb}_{j,3}$: This is the same as $\mathsf{Hyb}_{j,2}$ except that instead of encoding the actual witness w, encode a dummy witness 0^n during execution of the j-th PROVE command to honest part P_i. Suppose there is an adversary who can distinguish between two hybrids then we can build a distinguisher against the ϕ-Evaluation Hiding property (Definition 2.9).

   Let $\mathcal{A}_{\mathsf{H}}$ be the adversary who can distinguish between $\mathsf{Hyb}_{j,2}$ and $\mathsf{Hyb}_{j,3}$ with advantage ε. We build an adversary $\mathcal{A}_{\mathsf{ev}}$ for the evaluation hiding property with the same advantage. Let $\mathsf{PCS}, \mathsf{PES}$ be the schemes used in the protocol (and by the simulator), and proceed as follows for the j-th PROVE command.

   – $\mathcal{A}_{\mathsf{ev}}$ receives as input witness w, chooses evaluation points $\mathbf{z}$ at random and sends $(w, \mathbf{z})$ to the challenger.
   – $\mathcal{A}_{\mathsf{ev}}$ receives $(c, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCS}})$ in return and parses $\boldsymbol{\pi}_{\mathsf{PCS}}$ as a list of evaluation proofs $(\pi_1, \ldots, \pi_r)$.
   – Define $\mathcal{C}'$ given $\mathcal{C}$ as an honest prover would.
   – For each $i \in [r]$, $\mathcal{A}_{\mathsf{ev}}$ program $\mathcal{G}_{\mathsf{RO}}$ output at points $(\mathcal{C}', c, z_i, y_i, \pi_i, i)$ to be 0^b.
   – Run the underlying simulator $\Pi_{\mathcal{R}}.\mathcal{S}$ on input $\mathcal{C}'$ to obtain proof π'.
   – Define $\varpi = (\pi', c, \mathbf{z}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCS}})$, invokes $\mathcal{A}_{\mathsf{H}}$ internally and sends ϖ to $\mathcal{A}$.
   – $\mathcal{A}_{\mathsf{H}}$ ouptuts a bit b', $\mathcal{A}_{\mathsf{ev}}$ outputs the same bit.

   View of $\mathcal{A}_{\mathsf{H}}$ consists of ϖ, queries to $\mathcal{F}_{\mathsf{Setup}}$ and to $\mathcal{G}_{\mathsf{RO}}$. Queries to $\mathcal{F}_{\mathsf{Setup}}$ and $\mathcal{G}_{\mathsf{RO}}$ are handled exactly the same in the two hybrids. Otherwise, there are two difference between the views: (i) the proof ϖ is with respect to a witness w or 0^n, and (ii) proof π' is for a different statement (and potentially for a wrong statement in $\mathsf{Hyb}_{j,3}$). If the challenger's bit $b = 1$, the view ϖ correctly emulates $\mathsf{Hyb}_{j,2}$, where w is encoded and π' is simulated proof for an honest statement. Otherwise it emulates $\mathsf{Hyb}_{j,3}$, where 0^n is encoded and π' is simulated proof for a potentially false statement. Thus, the distinguishing advantage of $\mathcal{A}_{\mathsf{ev}}$ is the same as that of $\mathcal{A}_{\mathsf{H}}$.

Remark A.5. Note that, the ability of $\Pi_{\mathcal{R}}.\mathcal{S}$ to generate proofs for a false statements, which is required in the correctness of indistinguishability argument above, is guaranteed by simulation-extractability for underlying $\Pi_{\mathcal{R}}$.

5. $\mathsf{Hyb}_{j,4}$: This is the same as Hyb_{3+j}. The difference from $\mathsf{Hyb}_{j,3}$ is that here the preimage of 0^b is obtained by making repeated queries to the $\mathcal{G}_{\mathsf{RO}}$, while in $\mathsf{Hyb}_{j,3}$, they are programmed according to the challenge received in the evaluation hiding game. By a reasoning similar to the indistinguishability argument between $\mathsf{Hyb}_{j,1}$ and $\mathsf{Hyb}_{j,2}$ we can conclude that $\mathsf{Hyb}_{j,3}$ and $\mathsf{Hyb}_{j,4}$ are indistinguishable. Specifically, assuming that the algorithm in $\mathsf{Hyb}_{j,4}$ does not fail in finding appropriate preimages, the view of the distinguisher consists of ϖ, queries to $\mathcal{G}_{\mathsf{RO}}$, and to $\mathcal{F}_{\mathsf{Setup}}$. The simulator fails in finding preimages with negligible probability corresponding to the correctness error. Queries to $\mathcal{F}_{\mathsf{Setup}}$ are handled in the same way. Once again, recall that any $\mathcal{G}_{\mathsf{RO}}$ for a session sid by a party participating in the same session is not recorded by the functionality. Thus, $\mathcal{G}_{\mathsf{RO}}$ queries made in the execution of $\mathsf{Hyb}_{j,4}$ are not recorded and so responses to OBSERVE command behave indistinguishably. Now, we argue the same for responses to QUERY command. $c, \mathcal{C}'$ is computed in exactly the same way, and given c, z_i, the values y_i and π_i are unique. Thus the only difference is in the way z_i's are found. In $\mathsf{Hyb}_{j,3}$, z_i's are distributed uniformly random as they are picked at random. This is also the case for $\mathsf{Hyb}_{j,4}$ since z_i's are found by querying the $\mathcal{G}_{\mathsf{RO}}$. Finally, a distinguisher can distinguish if he queries $\mathcal{G}_{\mathsf{RO}}$ on a point that is later programmed to output 0^b. However, since c is generated honestly and is assumed to have high min-entropy, the probability of this event occurring is negligible.

Lemma A.6. *For $j = 0, \dots, \mathsf{s}_2 - 1$, $\mathsf{Hyb}_{2+\mathsf{s}_1+j}$ and $\mathsf{Hyb}_{2+\mathsf{s}_1+j+1}$ are indistinguishable.*

Proof. In these hybrids we replace the j-th VERIFY command to an honest party P_i with the code in **Simulation of uncorrupted verifier** (in Sim). In particular, all the $\mathcal{F}_{\mathsf{Setup}}$ queries are implemented by executing **Simulation of $\mathcal{F}_{\mathsf{Setup}}$**. All the s_1 PROVE commands are executing by steps **Simulation of uncorrupted prover**. The first $j - 1$ VERIFY commands are implemented by running steps in **Simulation of uncorrupted verifier**, and the last $\{j + 1, \dots, \mathsf{s}_2\}$ VERIFY commands are done by running honest verifier's algorithm in $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$. For the j-th VERIFY command to an honest party P_i, in $\mathsf{Hyb}_{2+\mathsf{s}_1+j}$ the honest verifier's code is executed, while in $\mathsf{Hyb}_{2+\mathsf{s}_1+j+1}$ code in **Simulation of uncorrupted verifier** is implemented. We prove indistinguishability between consecutive hybrids using the following sub-hybrids.

1. $\mathsf{Hyb}'_{\mathsf{j},0}$: The same as $\mathsf{Hyb}_{2+\mathsf{s}_1+j}$
2. $\mathsf{Hyb}'_{\mathsf{j},1}$: Same the previous hybrid except that, for the j-th query the honest verifier's algorithm is replaced by steps 1 to 7 in **Simulation for uncorrupted verifier** are additionally executed. The difference in the two hybrids is the abort probability introduced by $\perp_2$. This happens with negligible probability as otherwise it violates either evaluation binding (when $y \neq y'$ for the same c, z) or unique proofs (when $\pi \neq \pi'$ for the same c, z, y).
3. $\mathsf{Hyb}'_{\mathsf{j},2}$: Same as the previous hybrid except that, for the j-th query, execute step 8 in **Simulation for uncorrupted verifier** in addition to the previous steps. The difference with the previous hybrid is that, here, there is an additional abort probability introduced by the event that $(\mathcal{C}', \pi')$ in ϖ was previously generated by $\Pi_{\mathcal{R}}.\mathcal{S}$ execution. We argue that this happens with negligible probability, otherwise, we break non-extrapolation for the PCS scheme.
   Suppose, there exists an environment $\mathcal{Z}$ and adversary $\mathcal{A}$ such that $\mathsf{Hyb}'_{\mathsf{j},2}$ results in $\perp_3$ with probability ε. $\mathcal{Z}$ sees s_1 simulated proofs ϖ for its choice of statements, and later comes up with a proof $\tilde{\varpi} = (\tilde{\pi}', \tilde{c}, \tilde{\mathbf{z}}, \tilde{\mathbf{y}}, \tilde{\boldsymbol{\pi}}_{\mathsf{PCS}})$ such that $(\tilde{\mathcal{C}}', \tilde{\pi}')$ appears in one of the simulated proofs, where $\tilde{\mathcal{C}}'$ is computed as usual given $\mathcal{C}$ and c. We construct an adversary $\mathcal{A}_{\mathsf{ne}}$ who breaks non-extrapolation with probability $\varepsilon / \mathsf{s}_1$.
   - $\mathcal{A}_{\mathsf{ne}}$ starts the ϕ-non-extrapolation game and receives ck from the challenger. It starts simulating the view for $\mathcal{Z}$ for $\mathsf{Hyb}'_{\mathsf{j},2}$. It creates an instance for $\mathcal{G}_{\mathsf{RO}}$ functionality to simulate queries to

$\mathcal{G}_{\mathsf{RO}}$. All $\mathcal{F}_{\mathsf{Setup}}$ queries are answered by running the steps in **Simulating $\mathcal{F}_{\mathsf{Setup}}$** except that when queried for **Commitment Key** send ck.

– Recall, $\mathcal{Z}$ issues s_1 PROVE commands. These are answered as: $\mathcal{A}_{\mathsf{ne}}$ picks an index $k \in [\mathsf{s}_1]$ at random and answers all but the k-th PROVE command by generating simulated proofs as in previous hybrids. For the k-th command PROVE, $\mathcal{A}_{\mathsf{ne}}$ invokes the non-extrapolation challenger by sending $(n, \mathbf{z})$, where n is the size of witness and each z_i is sampled randomly from $\mathcal{D}_i$.

– The challenger returns proofs $(c, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCS}})$, where c corresponds to a commitment to a polynomial encoding 0^n. $\mathcal{A}_{\mathsf{ne}}$ programs the $\mathcal{G}_{\mathsf{RO}}$ at points corresponding to this proof to 0^b. Note that, since $\mathcal{A}_{\mathsf{ne}}$ invokes $\mathcal{Z}$ internally and simulates the entire view for $\mathcal{Z}$, it is allowed to program the $\mathcal{G}_{\mathsf{RO}}$. $\mathcal{A}_{\mathsf{ne}}$ computes $\mathcal{C}'$, simulates its proof π' by running $\Pi_{\mathcal{R}}.\mathcal{S}$, and forwards $\varpi = (\pi', c, \mathbf{z}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCS}})$ to $\mathcal{Z}$.
The commitment c and evaluation proofs correspond to an encoding of 0^n, the view obtained by $\mathcal{Z}$ is exactly as in $\mathsf{Hyb}'_{\mathsf{j},2}$.

– Recall, $\mathcal{Z}$ issues s_2 VERIFY commands. For the first $j-1$ VERIFY commands are implemented by running steps in **Simulation of uncorrupted verifier**, and the last $\{j+1, \ldots, \mathsf{s}_2\}$ VERIFY commands are done by running honest verifier's algorithm in $\Pi_{\mathsf{UC}\text{-}\mathcal{R}}$. During this execution, $\mathcal{Z}$ issues a command VERIFY such that $(\tilde{\mathcal{C}'}, \tilde{\pi'})$ was previously simulated. With probability $1/\mathsf{s}_1$, this corresponds to the k-th PROVE command, i.e., $\tilde{\mathcal{C}'} = \mathcal{C}'$ and $\tilde{\pi'}$ was simulated by $\mathcal{A}_{\mathsf{ne}}$ for input $\mathcal{C}'$. Then, since $\tilde{\varpi}$ differs from all previous proofs and $(\pi', c, \mathcal{C}) = (\tilde{\pi'}, \tilde{c}, \tilde{\mathcal{C}})$, it must be that $(\mathbf{z}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCS}}) \neq (\tilde{\mathbf{z}}, \tilde{\mathbf{y}}, \tilde{\boldsymbol{\pi}}_{\mathsf{PCS}})$. And since (because of the previous hybrid), if $\mathbf{z} = \tilde{\mathbf{z}}$, then $\mathbf{y} = \tilde{\mathbf{y}}$ and $\boldsymbol{\pi}_{\mathsf{PCS}} = \tilde{\boldsymbol{\pi}}_{\mathsf{PCS}}$, it must be that $\mathbf{z} \neq \tilde{\mathbf{z}}$. Thus, $\mathcal{A}_{\mathsf{ne}}$ obtains at least one more evaluation $(\tilde{z}_i, \tilde{y}_i)$ with proof $\tilde{\pi}_i$ from $\tilde{\varpi}$ in addition to the r evaluations from ϖ. $\mathcal{A}_{\mathsf{ne}}$ forwards $(\tilde{z}_i, \tilde{y}_i, \tilde{\pi}_i)$ to the non-extrapolation challenger. If $\mathcal{Z}$ reuses a simulated proof with probability ε, $\mathcal{A}_{\mathsf{ne}}$ breaks non-extrapolation with probability ε/s_1.

4. $\mathsf{Hyb}'_{\mathsf{j},3}$: Same as the previous hybrid except that, for the j-th VERIFY command with some input $(\mathcal{C}, \varpi)$, step 9 in **Simulation for uncorrupted verifier** is additionally executed. The difference from the previous hybrid is that an additional abort condition is introduced by $\perp_4$. Note that if $\perp_4$ is not the output then the simulator succeeds in interpolating the polynomial and decoding a witness.

Let the event where $\perp_4$ is output be denoted by X_1. We define the event X_2 as: there exists a pair $(\mathcal{C}^*, c^*)$ for which $\mathcal{G}_{\mathsf{RO}}$ queries of the form $(\mathsf{sid}, (\mathcal{C}^*, c^*, \cdot, \cdot, \cdot, i))$, for each $i \in \{1, \ldots, r\}$, were made and $|\mathcal{Q}_{c^*}| < d$. Observe that X_1 happens only if X_2 happens. Note that, in X_2, we can ignore the pairs $(\mathcal{C}^*, c^*)$ for which no H query was made for some iteration. This is because such queries can never be the part of final proof. Now we bound the probability of X_2 which will bound also the probability of X_1 happening.

Let the total number of $\mathcal{G}_{\mathsf{RO}}$ queries made be Q. Fix some tuple $(\mathsf{sid}, (\mathcal{C}^*, c^*, \cdot, \cdot, \cdot, \cdot))$, and now we bound the probability that $|\mathcal{Q}_{c^*}| < d$ for this specific tuple (again assuming that there is at least one query per iteration). As argued in the previous hybrid, for a given (c^*, z^*), the evaluation and the evaluation proof are unique. Thus, there is only one query in the set $\mathcal{Q}_{\mathsf{ro}}$ corresponding to any commitment, evaluation point pair (c^*, z^*). This means that the condition $|\mathcal{Q}_{c^*}| < d$ happens if there are less than d distinct z for the same c^* in the query set $\mathcal{Q}_{\mathsf{ro}}$.
Each iteration i is independent as the evaluation points come from a separate sub-domain. Consider the event that there is no iteration in which the adversary takes more that d queries to find a preimage for 0^b. Let good_k denote the event that the k-th RO query is a good query, i.e. hashes to 0^b. Then the probability that, in iteration i, the adversary $\mathcal{A}$ wins, i.e., finds a preimage for 0^b in less than d queries to the $\mathcal{G}_{\mathsf{RO}}$ can be bounded as follows.

$$\Pr[\mathcal{A} \text{ wins in iteration } i] \leq \sum_{k=1}^{d-1} \Pr[\mathsf{good}_k] < \frac{d}{2^b}$$

For the pair $(\mathcal{C}^*, c^*)$, $|\mathcal{Q}_{c^*}| < d$, if $\mathcal{A}$ wins in each iteration. Thus,

$$\Pr[|\mathcal{Q}_{c^*}| < d] \leq \left(\frac{d}{2^b}\right)^r = \left(\frac{2^{r\log(d)}}{2^{rb}}\right)$$

For $r(b - \log(d)) = \lambda$, we get negligible probability for $|\mathcal{Q}_{c^*}| < d$ for a fixed pair $(\mathcal{C}^*, c^*)$.

$$\Pr[|\mathcal{Q}_{c^*}| < d] \leq \frac{1}{2^\lambda}$$

Taking a union bound over all tuples $(\mathcal{C}^*, c^*)$ among Q $\mathcal{G}_{\mathsf{RO}}$ queries, we get,

$$\Pr[\mathsf{X}_1] \leq \Pr[\mathsf{X}_2] \leq \frac{Q}{2^\lambda}$$

Since the number of $\mathcal{G}_{\mathsf{RO}}$ queries can only be polynomially many, the probability of event X_1 happening is negligible. Thus, the simulator outputs $\bot_4$ only with negligible probability and otherwise succeeds in extracting a witness.

5. $\mathsf{Hyb}'_{j,4}$: Implement all the steps in **Simulation for uncorrupted verifier** (steps 1 to 12). This hybrid differs from $\mathsf{Hyb}'_{j,3}$ in the additional abort condition introduced by $\bot_5$. We argue that this happens only with negligible probability.
For a given configuration of $\mathcal{A}$ and $\mathcal{Z}$, we construct an adversary $\mathcal{A}_{\mathsf{eb}}$ that breaks evaluation binding for the PCS scheme if $\bot_5$ happens. $\mathcal{A}_{\mathsf{eb}}$ internally runs the following cheating prover $\mathcal{B}$ against SIM-EXT of $\Pi_{\mathcal{R}}$ which further invokes $\mathcal{A}$ and $\mathcal{Z}$ inside.
   (a) $\mathcal{B}$ simulates the view of $\mathcal{A}$ and $\mathcal{Z}$ in $\mathsf{Hyb}'_{j,4}$, except that it forwards setup queries in to $\mathcal{S}_1$ to receive out, and queries $\mathcal{C}'$ (derived from queried $\mathcal{C}$ and self-produced c) to $\mathcal{S}'_2$ to receive π', respectively.
   (b) When $\mathcal{Z}$ outputs accepting $(\mathcal{C}, (\pi', c, \mathbf{z}, \mathbf{y}, \boldsymbol{\pi}_{\mathsf{PCS}}))$, $\mathcal{B}$ constructs the corresponding extended circuit $\mathcal{C}'$ and outputs $(\mathcal{C}', \pi')$

   $\mathcal{A}_{\mathsf{eb}}$ then answers queries made by $\mathcal{B}$ by running $\mathcal{S}_1$ and $\mathcal{S}'_2$. Thanks to the abort conditions defined in previous hybrids, the output $(\mathcal{C}', \pi')$ of $\mathcal{B}$ differs from any of previously generated statement-proof pairs recorded during queries to $\mathcal{S}'_2$. Thus, the SIM-EXT property suggests there exists a (non-black-box) extractor $\mathcal{E}_{\mathcal{B}}$ such that it successfully outputs a valid witness w' satisfying $\mathcal{C}'(w') = 1$ except with negligible probability.
   Let us describe how $\mathcal{A}_{\mathsf{eb}}$ proceeds to break evaluation binding assuming $\mathcal{C}'(w') = 1$ (otherwise it aborts). Parse w' as $(\mathbf{w}^*, \boldsymbol{\rho}^*_w, \rho^*_c)$. It holds that $c = \mathsf{Com}(\mathsf{ck}, \mathsf{Enc}(\mathbf{w}^*, n, \ell; \boldsymbol{\rho}^*_w), d; \rho^*_c)$ and $\mathcal{C}(\mathbf{w}^*) = 1$. Recall that w (interpreted as vector $\mathbf{w}$) is the witness extracted by $\mathsf{Hyb}'_{j,4}$ by observing $\mathcal{G}_{\mathsf{RO}}$ queries. To cause the event $\bot_5$ while $(\mathcal{C}', w') \in \mathcal{R}$, it must be that $\mathbf{w}^* \neq \mathbf{w}$. $\mathcal{A}_{\mathsf{eb}}$ recomputes the encoding with respect to $(\mathbf{w}^*, \boldsymbol{\rho}^*_w)$ as $f^* \leftarrow \mathsf{Enc}(\mathbf{w}^*, n, \ell; \boldsymbol{\rho}^*_w)$ of degree less than d. Since $\mathbf{w} \neq \mathbf{w}^*$, we also have $f \neq f^*$ as otherwise it contradicts correctness of the encoding function. Then $\mathcal{A}_{\mathsf{eb}}$ evaluates f^* over all z_i found in $\mathcal{Q}_c$ to obtain y^*_i, and honestly computes new evaluation proofs $\pi^*_i \leftarrow \mathsf{Eval}(\mathsf{ck}, c, d, z_i, y^*_i, f^*; \rho^*_c)$. Obviously, completeness of PCS ensures all these proofs pass the verification check. Note that the size of $\mathcal{Q}_c$ is at least d, so we are guaranteed to get d valid evaluation proofs both from the RO queries and f^* as constructed above. Because f and f^* are distinct polynomials of degree less than d, it must be that $y_j \neq y^*_j$ for some j. Therefore, for such j, the tuple $(c, z_j, y_j, y^*_j, d, \pi_j, \pi^*_j)$ is indeed an instance breaking the evaluation binding.
   Overall, we get

$$\Pr[\bot_5] \leq \Pr[(\mathcal{C}', w') \notin \mathcal{R}] + \Pr\left[\mathbf{w} \neq \mathbf{w}^* \;\middle|\; (\mathcal{C}', w') \in \mathcal{R}\right] \leq \varepsilon_{\mathsf{se}} + \varepsilon_{\mathsf{eb}} \leq \mathsf{negl}(\lambda)$$

   where $\varepsilon_{\mathsf{se}}$ is the SIM-EXT knowledge error of $\Pi_{\mathcal{R}}$ and $\varepsilon_{\mathsf{eb}}$ is the advantage of breaking evaluation binding property of PCS.

Lemma A.7. $\mathsf{Hyb}_{2+\mathsf{s}}$ *and* $\mathsf{Hyb}_{3+\mathsf{s}}$ *are indistinguishable.*

Proof. The change here is merely syntactic, i.e., replacing $\mathcal{B}$ in hybrid $\mathsf{Hyb}_{2+\mathsf{s}}$ with Sim. $\mathsf{Hyb}_{3+\mathsf{s}}$ runs in exactly the same way as $\mathsf{Hyb}_{2+\mathsf{s}}$, except that the outputs (simulated proofs and extracted witness in response to PROVE and VERIFY commands, resp.) produced in the hybrid experiment are forwarded to the $\mathcal{F}_{\mathsf{NIZK}}$ functionality.