





Witness-Succinct Universally-Composable SNARKs^{*}

Chaya Ganesh¹, Yashvanth Kondi², Claudio Orlandi², Mahak Pancholi²,
Akira Takahashi³, and Daniel Tschudi⁴

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`

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.

^{*} The authors would like to thank abhi shelat for helpful discussions about an early version of this work. We thank anonymous reviewers of Eurocrypt 2023 for valuable comments and suggestions.

The work described in this paper has received funding from: the Concordium Blockchain 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.

1 Introduction

The UC framework and UC Secure NIZKs. The Universal Composability (UC) framework [28] 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) [16, 17] 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) [37], which is often referred to as *simulation-extractability* in context of UC (NI)ZK [35, 38, 46, 56, 72, 73]. 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 [35] was shown to be UC secure in [31]. Subsequently, [50, 54] constructed UC secure NIZKs in the presence of adaptive adversaries, and [50] 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 arguments⁵ of knowledge) where the communication is sublinear (ideally polylogarithmic or constant⁶) 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 size $O(|w|)$ and π is the SNARG itself. This approach was described by De Santis et al. [35] in the context of lifting ordinary NIZK to simulation-sound NIZK, and implemented as part of the C0C0 framework for circuit-succinct UC NIZK by Kosba et al. [63], with optimizations for concrete efficiency using the state-of-the-art SNARKs at the time. C0C0 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, C0C0 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” [63, 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

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 [48, 49].

⁶ Polynomial only in the security parameter.

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. [21]). 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 [10].

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 [1] 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 [60] 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 [68] 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 [47, 70]) are not black-box extractable⁷. 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. [27] 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 [61], Micali’s CS proofs [69], and the recent works on Interactive Oracle Proofs [15]. 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 [3, 11, 69] are black-box straight-line extractable, simulation-extractability

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 [12]	ROM	ROM	✓	✓	✓	unknown
Aurora [14]	ROM	ROM	✓	✓	✓	unknown
RedShift [59]	ROM	ROM	✓	✓	✓	unknown
Bulletproofs [22]	DLOG	ROM	✓	✓	✗	✓ [45]
SONIC [67]	AGM & q -DLOG	CRS & ROM	✗	✗	✓	✓ [42]
PLONK [41]	AGM	CRS & ROM	✗	✗	✓	✓ [42]
Marlin [33]	AGM	CRS & ROM	✗	✗	✓	✓ [42]
Groth16 [51]	GGM	CRS(& ROM for NM)	✗	✗	✓	✓ [20]
Groth-Maller [53]	XPKE & Poly	CRS	✗	✗	✓	✓
LAMASSU [2]	q -MC & q -MK & BDH & DL	CRS	✗	✗	✓	✓
Ours + [53] + [57]	XPKE & Poly & SDH	CRS & GROM	✗	✓	✓	✓
Ours + [2] + [57]	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 [2, Theorem 4] where they adapt [52] as an underlying SNARK.

of these has not been shown. SNARKs in the ROM that are logarithmic in the statement and witness size are known from conservative computational assumptions such as the hardness of computing discrete logarithms [19, 22] in the standalone setting. Bulletproofs [22] are known to be simulation-extractable, but currently only in the AGM+ROM [44] or in the ROM with rewinding [45]. 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 [42]. 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**?*

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 $\Pi_{\text{UC-}\mathcal{R}}$ in the (global random oracle $(\mathcal{G}_{\text{RO}})$, local setup $(\mathcal{F}_{\text{Setup}})$)-hybrid model, where \mathcal{G}_{RO} is observable but non-programmable as in [30] and $\mathcal{F}_{\text{Setup}}$ 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 [2, 53] into our compiler gives us as a corollary the first constant sized UC NIZKs in the $(\mathcal{G}_{\text{RO}}, \mathcal{F}_{\text{crs}})$ -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 [38, 44]. 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 [57].

1.1 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 [25, 36]. 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 $|\text{ct}| \in \Omega(|w|)$. 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 [25, 71]. 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 [23, 34]) 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 $(\text{cm}, \pi_{\text{cm}})$, 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 $|\text{cm}| \in O_\lambda(1)$, 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 $q \in \omega(\text{poly}(\lambda))$ 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 degree⁸ $d - 1$ 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.

$O_\lambda(1)$ 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

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.

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 $n \in \text{poly}(d, \lambda)$.

Authenticating Evaluation Openings at $O_\lambda(1)$ Cost via Fischlin’s Technique [39]: 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 [15, 69], 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 [39] 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 $\pi_{\text{cm}}^{(i)} = (z_i, f(z_i))$ with uniform z_i and the corresponding *evaluation proof* $\pi_{\text{ev}}^{(i)}$ that ensures the polynomial f committed to in cm has been correctly evaluated at z_i , and (2)

store $(\pi_{\text{cm}}^{(i)}, \pi_{\text{ev}}^{(i)})$ and go to the next iteration if $H(\text{cm}, i, \pi_{\text{cm}}^{(i)}, \pi_{\text{ev}}^{(i)}) = 0$ for a random oracle H with b -bit outputs, and go to step (1) otherwise. Thanks to the evaluation proof, $\pi_{\text{cm}}^{(i)}$ is tied to a given commitment cm . In practice, succinct evaluation proof can be easily implemented by naively invoking the underlying SNARK prover⁹ or by instantiating cm with a dedicated *polynomial commitment scheme* such as [57], 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 $H(\text{cm}, i, \cdot) = 0$ within $d - 1$ queries. For any given i , the probability that P^* finds $H(\text{cm}, i, \cdot) = 0$ within d queries is at most $d/2^b$; therefore the probability that P^* finds $H(\text{cm}, i, \cdot) = 0$ 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 [57] 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 $\text{poly}(\lambda)$ as well as $O_\lambda(d)$, i.e. it scales 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 $\pi_{\text{cm}} = (\pi_{\text{cm}}^{(i)})_{i \in [r]}$ —this proof is at the heart of forcing the environment to use the witness within the context of the protocol. The proof π_{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 π_{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

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.

outside the protocol—is extracted in a black-box, straight-line fashion via π_{cm} within the context of the protocol.

1.2 Related Work

Straight-line Extraction. Our UC-lifting technique is inspired by Fischlin’s transform [39] based on Proof-of-Work. Kondi and shelat [62] 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 [65, 66] 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 [32] realized *triply adaptive* UC-secure NIZK using a straight-line extractable commitment in the CRS model. Pass [71] 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 [24] 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 [74] extended [71] to retain security against an adversary making superposition queries to the RO (the so-called *quantum random oracle model*). Recently, Katsumata [58] 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 [35, 50, 73], optimized in the C0C0 framework [63], and tailored to Groth16 in [5, 6]. 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 [2] framework extends the C0C0 lifting technique to work with updatable SNARKs giving a generic compiler from updatable CRS SNARKS to SE SNARKs. TIRAMISU [9] 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 succinctness or blackbox extraction is lost, unlike our compiler. There are works on lifting specific SNARKs into SE; the work of Groth and Maller [53] presents an SE SNARK, but the simulation extractability is non-black-box. There is a line of work on analysing the simulation extractability [7, 8, 20] 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 $\{a, a + 1, \dots, b\}$. 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 $\text{poly}(\lambda)$ it holds that $f(\lambda) < 1/\text{poly}(\lambda)$ for sufficiently large $\lambda > 0$. We write “ $f(\lambda) < \text{negl}(\lambda)$ ” 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 [28] 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 [29] 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}_{RO} as introduced in [30]. We follow the simplified description from [25]. The \mathcal{G}_{RO} functionality can be queried by any party and the ideal adversary with two commands: QUERY and OBSERVE. The environment can query \mathcal{G}_{RO} by spawning additional dummy parties outside the context of the current protocol execution. \mathcal{G}_{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}_{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}_{RO} in Fig. 1.

Remark 2.1. In [30] 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 [25] 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.

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

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

- **Query** Upon receiving a query (QUERY, x), from some party $\mathcal{P} = (\text{pid}, \text{sid})$ 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 $\text{sid} \neq s$ 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 (OBSERVE, sid) from the adversary **Sim**, return the list $\mathcal{Q}_{|\text{sid}}$ of illegitimate queries for SID sid to the adversary.

Fig. 1: Functionality for Global Random Oracle \mathcal{G}_{RO} [25]

Intuitively, these illegitimate queries are required for proving security of our protocols. The ideal adversary (or the simulator) works by observing \mathcal{G}_{RO} queries made by the corrupt party during the protocol execution. However, the environment can bypass this by querying \mathcal{G}_{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 [25] by marking such queries as illegitimate and disclosing them to the simulator via OBSERVE command. Note that any \mathcal{G}_{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.

Definition 2.2 (UC Security in the Global ROM [29, 30]). *Let $\mathcal{F}, \mathcal{F}'$ be m -party functionalities and Π be a protocol. We say that Π UC-realizes \mathcal{F} in the $\mathcal{G}_{\text{RO}}, \mathcal{F}'$ -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:*

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

where **REAL** is the outputs of the honest parties and the adversary \mathcal{A} after a real execution of protocol Π with input $\mathbf{x} = (x_1, \dots, x_m)$ for parties P_1, \dots, 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 $(\text{PGen}, \mathcal{O}_{\text{Setup}}, \mathcal{P}, \mathcal{V})$.

- $\text{pp} \leftarrow \text{PGen}(1^\lambda)$: 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.
- $\text{out} \leftarrow \mathcal{O}_{\text{Setup}}(\text{in})$: A stateful setup oracle that takes an input string in and outputs out .
- $\pi \leftarrow \mathcal{P}^{\mathcal{O}_{\text{Setup}}}(x, w)$: Takes as input a statement x and witness w , and outputs a proof π if $(x, w) \in \mathcal{R}$.
- $b \leftarrow \mathcal{V}^{\mathcal{O}_{\text{Setup}}}(x, \pi)$: Takes as input a statement x and proof π , and outputs a bit b , indicating “accept” or “reject”.

We introduce the setup oracle $\mathcal{O}_{\text{Setup}}$ 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 $\mathcal{O}_{\text{Setup}}$ 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, $\mathcal{O}_{\text{Setup}}$ is initialized with an empty query-response table and proceeds as follows. On receiving $\text{in} \in \{0, 1\}^*$, if in has never been queried, sample uniform $\text{out} \in \{0, 1\}^{\ell(\lambda)}$, store (in, 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

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

We define zero-knowledge by following the syntax of [38, 44]. A zero-knowledge simulator \mathcal{S} is defined as a stateful algorithm with initial state $\text{st} = \text{pp}$ that operates in two modes. The first mode, $(\text{out}, \text{st}') \leftarrow \mathcal{S}(1, \text{st}, \text{in})$ takes care of handling calls to the oracle $\mathcal{O}_{\text{Setup}}$ on input in . The second mode, $(\pi, \text{st}') \leftarrow \mathcal{S}(2, \text{st}, x)$ simulates a proof for the input statement x . For convenience we define three “wrapper” oracles. These oracles are stateful and share the internal state st , which initially contains an empty string.

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

Definition 2.4 ((Unbounded) Zero-Knowledge). Let $\Pi_{\mathcal{R}} = (\text{PGen}, \mathcal{O}_{\text{Setup}}, \mathcal{P}, \mathcal{V})$ 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 wrap-

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

$\mathcal{F}_{\text{Setup}}$ is parametrized by a security parameter λ and a degree bound $D > 0$ and runs with parties P_1, \dots, P_N and an ideal process adversary Sim .

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

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

per oracles \mathcal{S}_1 and \mathcal{S}_2 such that for all PPT adversaries \mathcal{A} it holds that

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

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 [53] 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.

Definition 2.5 ((Non-black-box) Simulation Extractability). Consider a non-interactive proof system $\Pi_{\mathcal{R}} = (\text{PGen}, \mathcal{O}_{\text{Setup}}, \mathcal{P}, \mathcal{V})$ 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 (*SIM-EXT*) with respect to \mathcal{S} , if for any PPT adversary \mathcal{A} , there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}$ such that

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

where st is the final state of the simulator \mathcal{S} , $\text{state}_{\mathcal{A}}$ 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.

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

$\mathcal{F}_{\text{NIZK}}$ is parametrized by polynomial-time-decidable relation $\mathcal{R} \in \{0, 1\}^* \times \{0, 1\}^*$, and runs with parties P_1, \dots, P_N and an ideal process adversary Sim . It stores proof table \mathcal{Q} which is initially empty.

- **Proof** Upon receiving input $(\text{PROVE}, \text{sid}, \text{ssid}, x, w)$ from a party P_i , ignore if $(x, w) \notin \mathcal{R}$. Otherwise, send $(\text{PROVE}, \text{sid}, x)$ to Sim . Upon receiving (PROOF, π) from Sim , store (x, π) in \mathcal{Q} and send $(\text{PROOF}, \text{sid}, \text{ssid}, \pi)$ to P_i .
- **Verification** Upon receiving input $(\text{VERIFY}, \text{sid}, \text{ssid}, x, \pi)$ from a party P_i , if (x, π) is not stored in \mathcal{Q} , then send $(\text{VERIFY}, \text{sid}, x, \pi)$ to Sim . Upon receiving $(\text{WITNESS}, w)$ from Sim , if $(x, w) \in \mathcal{R}$, store (x, π) in \mathcal{Q} . Finally, return $(\text{VERIFICATION}, \text{sid}, \text{ssid}, (x, \pi) \in? \mathcal{Q})$ to P_i .

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

The ideal functionality $\mathcal{F}_{\text{Setup}}$ that provides the setup and oracle for non-interactive proof system $\Pi_{\mathcal{R}} = (\text{PGen}, \mathcal{O}_{\text{Setup}}, \mathcal{P}, \mathcal{V})$ 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 $\Pi_{\text{UC-}\mathcal{R}}$. The ideal functionality for Non-interactive Zero-Knowledge $\mathcal{F}_{\text{NIZK}}$ is defined in Fig. 3. The functionality is taken from [55] with a minor difference being that $\mathcal{F}_{\text{NIZK}}$ explicitly informs Sim of the associated session ID.

2.3 Succinct Polynomial Commitment Scheme

The following definition is adapted from the full version of [33]. 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 $(\text{KGen}, \text{Com}, \text{Eval}, \text{Check})$:*

1. $\text{ck} \leftarrow \text{KGen}(1^\lambda, D)$: Takes as input the security parameter λ and the maximum degree bound D and generates commitment key ck as output.
2. $c \leftarrow \text{Com}(\text{ck}, f, d; \rho_c)$: 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 $\rho_c = \perp$.
3. $\pi \leftarrow \text{Eval}(\text{ck}, c, d, z, y, f; \rho_c)$: 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. $b \leftarrow \text{Check}(\text{ck}, c, d, z, y, \pi)$: Takes as input statement (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

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

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

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

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 [57] indeed satisfy these.

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

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

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 [33, §2.3]¹⁰, 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 (Enc, Dec) defined over an evaluation domain \mathcal{D}_{Enc} (which also determines the forbidden domain $\mathcal{S}_{\text{Enc}} = \mathbb{F} \setminus \mathcal{D}_{\text{Enc}}$).

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

¹⁰ This property is also known as k -knowledge bound in [13].

We say PES is correct if $\mathbf{w} = \text{Dec}(\text{Enc}(\mathbf{w}, n, \ell; \boldsymbol{\rho}), n, \ell)$ 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}^ℓ , any set of ℓ evaluations of $f \leftarrow \text{Enc}(\mathbf{w}, n, \ell; \boldsymbol{\rho})$ in \mathcal{D}_{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. $|\mathcal{D}_{\text{Enc}}| \in O(2^\lambda)$. Below we recall some candidate encoding schemes that are implicitly employed in many SNARK constructions.

- **Coefficient Encoding** $\text{PES}_1 = (\text{Enc}_1, \text{Dec}_1)$: PES can be instantiated using simple coefficient encoding as in [67]. Here $\mathcal{D}_{\text{Enc}} = \mathbb{F} \setminus \{0\}$ 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** $\text{PES}_2 = (\text{Enc}_2, \text{Dec}_2)$: This encoding method has been used in e.g. [26, 33, 41]. Suppose a subset $H \subset \mathbb{F}$ of cardinality n and an evaluation domain $\mathcal{D}_{\text{Enc}} = \mathbb{F} \setminus (H \cup \{0\})$. 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, $\text{Enc}_2(\mathbf{w}, n, \ell; \boldsymbol{\rho})$ 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 $\mathcal{S}_{\text{Enc}} = H$. On the other hand, up to ℓ evaluations of f in \mathcal{D}_{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 [33, 57]): 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 [26, 33, 41, 67] 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 $\phi : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ calculating the expansion factor for encoding randomness: given the number 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 $\text{PCS} = (\text{KGen}, \text{Com}, \text{Eval}, \text{Check})$ be a polynomial commitment scheme and $\text{PES} = (\text{Enc}, \text{Dec})$ be a polynomial encoding scheme. We say PCS is ϕ -evaluation hiding with respect to PES if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,*

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

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 $\pi \leftarrow \text{Eval}(\text{ck}, c, d, \mathbf{z}, \mathbf{y}, f; \rho_c)$ denotes setting $\pi_i \leftarrow \text{Eval}(\text{ck}, c, d, z_i, y_i, f; \rho_c)$ 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 $\text{PCS} = (\text{KGen}, \text{Com}, \text{Eval}, \text{Check})$ be a polynomial commitment scheme and $\text{PES} = (\text{Enc}, \text{Dec})$ 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

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

where \mathcal{A}_1 and \mathcal{A}_2 share the internal states, $\mathbf{y} := f(\mathbf{z})$, π 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 $\Pi_{\mathcal{R}} = (\text{PGen}, \mathcal{O}_{\text{Setup}}, \mathcal{P}, \mathcal{V})$ for the *arithmetic circuit satisfiability relation* $\mathcal{R} = \{(\mathcal{C}, w) : \mathcal{C}(w) = 1\}$, and (2) a **PCS** $= (\text{KGen}, \text{Com}, \text{Eval}, \text{Check})$ with suitable properties. The resulting protocol, denoted by $\Pi_{\text{UC-}\mathcal{R}}$, UC-realizes $\mathcal{F}_{\text{NIZK}}$ in the $(\mathcal{G}_{\text{RO}}, \mathcal{F}_{\text{Setup}})$ -hybrid model, where $\mathcal{F}_{\text{Setup}}$ is described in Fig. 2.

Theorem 3.1. *Let $\Pi_{\mathcal{R}}$ be a **SIM-EXT NIZK** proof system, 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, ϕ -evaluation hiding, and ϕ -non-extrapolation with respect to the encoding scheme $\text{PES} = (\text{Enc}, \text{Dec})$. Then, $\Pi_{\text{UC-}\mathcal{R}}$ described in Fig. 4 UC-realizes $\mathcal{F}_{\text{NIZK}}$ in the $(\mathcal{G}_{\text{RO}}, \mathcal{F}_{\text{Setup}})$ -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 $\in \mathcal{D}_{\text{Enc}}$, 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 $\mathcal{G}_{\text{RO}}(\text{QUERY}, (\text{sid}, (\mathcal{C}', c, z_i, y_i, \pi_i, i))) \neq 0^b$. 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.

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

The protocol $\Pi_{\text{UC-}\mathcal{R}}$ is parameterized by: security parameter λ , finite field \mathbb{F} , evaluation domain \mathcal{D}_{Enc} for **PES**, evaluation hiding expansion factor $\phi : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$, 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 $(\text{PROVE}, \text{sid}, \text{ssid}, \mathcal{C}, w)$, ignore if $\mathcal{C}(w) \neq 1$. Otherwise, P_i does:

1. Send $(\text{GENPARAMS}, \text{sid})$ to $\mathcal{F}_{\text{Setup}}$ and wait for answer $(\text{PARAMS}, \text{sid}, \text{pp})$.
2. Send $(\text{GENKEY}, \text{sid})$ to $\mathcal{F}_{\text{Setup}}$ and wait for answer $(\text{COMKEY}, \text{sid}, \text{ck})$.
3. Parse $w = \mathbf{w} \in \mathbb{F}^n$. Let $\ell := \phi(r)$ and $d := n + \ell$. If $d > D$, abort by outputting $(\text{PROOF}, \text{sid}, \text{ssid}, \perp)$.
4. Generate a polynomial encoding of the witness vector: $f \leftarrow \text{Enc}(\mathbf{w}, n, \ell; \rho_w)$, where $\rho_w \leftarrow \$ \mathbb{F}^\ell$.
5. Generate a commitment to the polynomial encoding: $c \leftarrow \text{Com}(\text{ck}, f, d; \rho_c)$, 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 $w' = (\mathbf{w}, \rho_w, \rho_c)$ if and only if the following conditions are met:

$$\mathcal{C}(\mathbf{w}) = 1 \quad \wedge \quad c = \text{Com}(\text{ck}, \text{Enc}(\mathbf{w}, n, \ell; \rho_w), d; \rho_c)$$

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 $\mathcal{O}_{\text{Setup}}$ with input in , send $(\text{SETUP}, \text{sid}, \text{in})$ to receive a response $(\text{SETUP}, \text{sid}, \text{out})$ and forward out to \mathcal{P} .
8. Initialize empty sets \mathbf{z} , \mathbf{y} , and π_{PCS} .
9. For each iteration $i \in [r]$ do:
 - (a) Initialize counter $\text{ctr} := 0$ and an empty set of used evaluation points \mathcal{D}_i .
 - (b) If $\text{ctr} = T$, abort by outputting $(\text{PROOF}, \text{sid}, \text{ssid}, \text{runout_eval})$.
 - (c) Sample an evaluation point: $z_i \leftarrow \$ \mathcal{D}_{\text{Enc}} \setminus \mathcal{D}_i$. Update $\text{ctr} := \text{ctr} + 1$. Update $\mathcal{D}_i := \mathcal{D}_i \cup \{z_i\}$.
 - (d) Compute $y_i = f(z_i)$ and evaluation proof $\pi_i \leftarrow \text{Eval}(\text{ck}, c, d, z_i, y_i, f; \rho_c)$.
 - (e) Send $(\text{QUERY}, (\text{sid}, (\mathcal{C}', c, z_i, y_i, \pi_i, i)))$ to \mathcal{G}_{RO} . Upon receiving v from \mathcal{G}_{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 π_{PCS} , respectively.
10. Output $(\text{PROOF}, \text{sid}, \text{ssid}, \varpi)$, where $\varpi := (\pi', c, \mathbf{z}, \mathbf{y}, \pi_{\text{PCS}})$.

– **Verification** Upon receiving input $(\text{VERIFY}, \text{sid}, \text{ssid}, \mathcal{C}, \varpi)$, P_i does:

1. Send $(\text{GENPARAMS}, \text{sid})$ to $\mathcal{F}_{\text{Setup}}$ and wait for answer $(\text{PARAMS}, \text{sid}, \text{pp})$.
2. Send $(\text{GENKEY}, \text{sid})$ to $\mathcal{F}_{\text{Setup}}$ and wait for answer $(\text{COMKEY}, \text{sid}, \text{ck})$.
3. Parse $\varpi = (\pi', c, \mathbf{z}, \mathbf{y}, \pi_{\text{PCS}})$. 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 $(\text{VERIFICATION}, \text{sid}, \text{ssid}, 0)$.
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 $\pi_{\text{PCS}} = (\pi_i)_{i \in [r]}$.
6. Output $(\text{VERIFICATION}, \text{sid}, \text{ssid}, 1)$ if all of the following checks pass, otherwise output $(\text{VERIFICATION}, \text{sid}, \text{ssid}, 0)$:
 - (a) $\Pi_{\mathcal{R}}.\mathcal{V}$ on input pp , \mathcal{C}' , and π' outputs 1. Calls to $\mathcal{O}_{\text{Setup}}$ by \mathcal{V} are handled similar to the above.
 - (b) For all $i \in [r]$: $1 = \text{Check}(\text{ck}, c, d, z_i, y_i, \pi_i)$.
 - (c) For all $i \in [r]$: send $(\text{QUERY}, (\text{sid}, (\mathcal{C}', c, z_i, y_i, \pi_i, i)))$ to \mathcal{G}_{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 $(\mathcal{G}_{\text{RO}}, \mathcal{F}_{\text{Setup}})$ -hybrid model. $\Pi_{\text{UC-}\mathcal{R}}$ internally runs $\Pi_{\mathcal{R}}$, **PCS**, and **PES**.

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

Thus, an honest prover manages to find a preimage of 0^b in polynomial time except with probability $2^{-\lambda}$. We also remark that the completeness error increases only additively even if the underlying proof system $\Pi_{\mathcal{R}}$ is statistically complete.¹¹ We defer the analysis in this case to the full version.

Simulation. We begin by sketching the overall simulation strategy. First, consider simulation for an uncorrupted prover. We simulate the behaviors of $\mathcal{F}_{\text{Setup}}$ and π' component of real-world proofs produced by honest provers using the underlying NIZK simulator $\Pi_{\mathcal{R}}.\mathcal{S}$. After the first r queries to \mathcal{G}_{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}_{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 $(\tilde{\mathcal{C}}, \tilde{\omega} = (\tilde{\pi}', \tilde{c}, \tilde{\mathbf{z}}, \tilde{\mathbf{y}}, \tilde{\pi}_{\text{PCS}}))$ 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 $(\tilde{\mathbf{z}}, \tilde{\mathbf{y}}, \tilde{\pi}_{\text{PCS}})$ differs from previously simulated $(\mathbf{z}, \mathbf{y}, \pi_{\text{PCS}})$ 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}_{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 SIM-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 $w' = (\mathbf{w}^*, \rho_w, \rho_c)$. 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.

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.

¹¹ We thank an anonymous reviewer for bringing this observation to our attention.

An honest party P_i on input $(\text{PROVE}, \text{sid}, \text{ssid}, \mathcal{C}, w)$ from \mathcal{Z} executes the honest prover's algorithm in $\Pi_{\text{UC-}\mathcal{R}}$ to generate the proof. And on receiving $(\text{VERIFY}, \text{sid}, \text{ssid}, \mathcal{C}, \varpi)$, it verifies by running the honest verifier's algorithm. In the ideal world, the honest parties forward their inputs to the functionality $\mathcal{F}_{\text{NIZK}}$. The corrupt parties' behavior is controlled by \mathcal{A} in both the worlds. Within a session sid , we assume that \mathcal{Z} issues \mathbf{s}_1 queries of the type $(\text{PROVE}, \text{sid}, \text{ssid}, \mathcal{C}, w)$ meant for an honest party, and \mathbf{s}_2 of the type $(\text{VERIFY}, \text{sid}, \text{ssid}, \mathcal{C}, \varpi)$ for either honest or corrupt party. Let $\mathbf{s} = \mathbf{s}_1 + \mathbf{s}_2$. Proofs for indistinguishability of hybrids are deferred to the full version [43].

- 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}_{RO} queries made by any corrupt party P_i within the session, forwards it the \mathcal{G}_{RO} and relays the response back to P_i . Similarly, it intercepts all $\mathcal{F}_{\text{Setup}}$ queries made by P_i in the session and relays it back and forth between $\mathcal{F}_{\text{Setup}}$ and P_i .
- Hyb_2 : Instead of forwarding P_i 's calls to $\mathcal{F}_{\text{Setup}}$ functionality, \mathcal{B} answers them by executing steps in **Simulation of $\mathcal{F}_{\text{Setup}}$** 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 $j \in [\mathbf{s}_1]$, 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 $j \in [\mathbf{s}_2]$, $\text{Hyb}_{2+\mathbf{s}_1+j}$: For the j -th **VERIFY** command with input (\mathcal{C}, ϖ) 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.
- $\text{Hyb}_{3+\mathbf{s}}$: This is the ideal world execution. Replace \mathcal{B} with **Sim**, where the steps in **Sim** are executed for each $(\text{PROVE}, \text{sid}, \text{ssid}, \mathcal{C}, w)$, and $(\text{VERIFY}, \text{sid}, \text{ssid}, \mathcal{C}, \varpi)$ command (as explained in the above hybrids), and sends corresponding $(\text{PROOF}, \text{sid}, \text{ssid}, P_i, \varpi)$ and $(\text{WITNESS}, \text{sid}, \text{ssid}, P_i, w)$ to $\mathcal{F}_{\text{NIZK}}$.

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 can-

didate 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 $\mathcal{O}(\text{poly}(\lambda, n))$ 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 3.2. 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.

4.1 A Candidate **PCS** and **PES** Scheme

We show that using KZG commitments [57] 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 $\text{PCS}_{\text{KZG}} = (\text{KGen}, \text{Com}, \text{Eval}, \text{Check})$. The formalization below follows the deterministic scheme of [33, §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} .

- **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 $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ 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 $\sigma = (g, g^\alpha, \dots, g^{\alpha^{D-1}}, h, h^\alpha)$. Output $\text{ck} = (\mathcal{G}, \sigma)$.
- **Com**(ck, f, d): On input ck , a polynomial $f \in \mathbb{F}_{<d}[X]$, and a degree bound $d \leq D$, 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} .
- **Eval**($\text{ck}, c, d, z, f(z), f$): 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 $\boldsymbol{\pi} = (g^{\omega(\alpha)}, g^{\hat{\omega}(\alpha)}, \hat{f}(z))$.
- **Check**($\text{ck}, \mathbf{c}, d, z, y, \boldsymbol{\pi}$): 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 $\hat{y} = z^{D-d} \cdot y$.

Simulator: Sim

Sim is parameterized by $\lambda, \mathbb{F}, \mathcal{D}_{\text{Enc}}, \phi, T, r, b, D$ and has access to the global functionality \mathcal{G}_{RO} as $\Pi_{\text{UC-}\mathcal{R}}$ does. It simulates real prover's proof for arbitrary $(\mathcal{C}, w) \in \mathcal{R}$, extracts a witness from a valid proof (\mathcal{C}, ϖ) chosen by the environment (as long as it hasn't been recorded by $\mathcal{F}_{\text{NIZK}}$), and simulates the local setup functionality $\mathcal{F}_{\text{Setup}}$. 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 [50]: We use the notation \tilde{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 $\mathcal{F}_{\text{NIZK}}$, 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{P}_i .
- **Simulation of $\mathcal{F}_{\text{Setup}}$**
 - **Parameters** Upon receiving input $(\text{GENPARAMS}, \text{sid})$ from a party P_i , if no pp has been stored, run $\text{pp} \leftarrow \text{PGen}(1^\lambda)$, let $\text{st} := \text{pp}$, and store pp . Send $(\text{PARAMS}, \text{sid}, \text{pp})$ to P_i .
 - **Commitment Key** This is identical to $\mathcal{F}_{\text{Setup}}$.
 - **Setup** Upon receiving input $(\text{SETUP}, \text{sid}, \text{in})$ from a party P_i , ignore if st has never been initialized with pp . Otherwise run $(\text{out}, \text{st}) \leftarrow \Pi_{\mathcal{R}, \mathcal{S}}(1, \text{st}, \text{in})$ using the current state and send $(\text{SETUP}, \text{sid}, \text{out})$ to P_i .
- **Handling \mathcal{G}_{RO} queries**
 1. Initialize empty set \mathcal{Q}_{ro} .
 2. Upon receiving input (QUERY, x) from a party P_i , forward it to the \mathcal{G}_{RO} and forward the response v back to P_i .
 3. Record x in \mathcal{Q}_{ro} .
- **Simulation of uncorrupted prover** Upon receiving input $(\text{PROVE}, \text{sid}, \mathcal{C})$ from $\mathcal{F}_{\text{NIZK}}$:
 1. Derive the witness size n from the description of \mathcal{C} . Compute ℓ and d as **Proof** of $\Pi_{\text{UC-}\mathcal{R}}$ would and if $d > D$ abort by outputting (PROOF, \perp) .
 2. Generate a polynomial encoding of dummy witness: $f \leftarrow \text{Enc}(0^n, n, \ell; \rho_w)$, where $\rho_w \leftarrow \mathbb{F}^\ell$.
 3. Generate a commitment to the polynomial encoding as **Proof** of $\Pi_{\text{UC-}\mathcal{R}}$ would: $c \leftarrow \text{Com}(\text{ck}, f, d; \rho_c)$.
 4. Define the circuit \mathcal{C}' as **Proof** of $\Pi_{\text{UC-}\mathcal{R}}$ would.
 5. Run $\Pi_{\mathcal{R}, \mathcal{S}}(2, \text{st}, \mathcal{C}')$ to obtain a proof-state pair (π', st) .
 6. Create \mathbf{z}, \mathbf{y} , and π_{PCS} as **Proof** of $\Pi_{\text{UC-}\mathcal{R}}$ would.
 7. Send (PROOF, ϖ) to $\mathcal{F}_{\text{NIZK}}$, where $\varpi := (\pi', c, \mathbf{z}, \mathbf{y}, \pi_{\text{PCS}})$.
- **Simulation of uncorrupted verifier** Upon receiving input $(\text{VERIFY}, \text{sid}, \mathcal{C}, \varpi)$ from $\mathcal{F}_{\text{NIZK}}$:
 1. Perform verification checks similar to **Verification** of $\Pi_{\text{UC-}\mathcal{R}}$, but use pp and ck generated during the simulation of $\mathcal{F}_{\text{Setup}}$. Calls to $\mathcal{O}_{\text{Setup}}$ made by $\Pi_{\mathcal{R}, \mathcal{V}}$ are handled by running $(\text{out}, \text{st}) \leftarrow \Pi_{\mathcal{R}, \mathcal{S}}(1, \text{st}, \text{in})$ and forwarding out to \mathcal{V} . If invalid, send $(\text{WITNESS}, \perp_1)$ to $\mathcal{F}_{\text{NIZK}}$. This will eventually cause $\mathcal{F}_{\text{NIZK}}$ to output $(\text{VERIFICATION}, \text{sid}, \text{ssid}, 0)$ to a dummy party \tilde{P}_i .
 2. Parse proof ϖ as $(\pi', c, \mathbf{z}, \mathbf{y}, \pi_{\text{PCS}})$.
 3. Query \mathcal{G}_{RO} on $(\text{OBSERVE}, \text{sid})$ and receive the set of illegitimate queries $\mathcal{Q}_{|\text{sid}}|$.
 4. Update $\mathcal{Q}_{\text{ro}} = \mathcal{Q}_{\text{ro}} \cup \mathcal{Q}_{|\text{sid}}|$.
 5. Define circuit \mathcal{C}' as **Verification** of $\Pi_{\text{UC-}\mathcal{R}}$ would.
 6. Define \mathcal{Q}_c as the set of queries in \mathcal{Q}_{ro} of the form $(\text{QUERY}, (\text{sid}, (\mathcal{C}', c, \cdot, \cdot, \cdot, \cdot)))$ 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 $w := \perp_2$ and go to 12.
 8. If (\mathcal{C}', π') was previously generated by $\Pi_{\mathcal{R}, \mathcal{S}}$ then set $w := \perp_3$ and go to 12.
 9. If $|\mathcal{Q}_c| < d$ then set $w := \perp_4$ 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 $(\mathcal{C}, \text{Dec}(f)) \notin \mathcal{R}$ set $w := \perp_5$; Else, set $w := \text{Dec}(f)$.
 12. Send $(\text{WITNESS}, w)$ to $\mathcal{F}_{\text{NIZK}}$.

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

The security of PCS_{KZG} relies on the SDH assumption [18].

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 $D > 0$,*

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

Lemma 4.2. *PCS_{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 , $y' \neq y$, $\boldsymbol{\pi} = (\pi, \hat{\pi}, \hat{y})$, $\boldsymbol{\pi}' = (\pi', \hat{\pi}', \hat{y}')$ 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 $((\pi/\pi')^{\frac{1}{y'-y}}, -z)$.¹²

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}_{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 $|\mathbf{z}|_u$ denote the number of distinct values in \mathbf{z} . Let $r' := |\mathbf{z}|_u$ and $n' := d - r'$. Note that, when there are no repeat elements in \mathbf{z} , $r = r'$. Sample n' evaluation points from $\mathcal{D}_{\text{Enc}}'$ and interpolate the polynomial f defined by d points (z_i, y_i) , where the first r' z_i 's are from

¹² 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.

\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) [40], 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 [53] construct an SE-SNARK from a knowledge assumption that they formulate, called the eXtended Power Knowledge of Exponent (XPKE) assumption. Lipmaa [64] presents SE-SNARKs under ‘hash-algebraic’ knowledge assumptions. Abdolmaleki et al. [2] 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. [52], 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.

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 $O_\lambda(f(|C| + |w|))$ and a polynomial commitment scheme that produces evaluation proofs of size $O_\lambda(g(|w|))$ 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:

- A ‘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. [4] 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 [22] into our compiler with a transparent polynomial commitment scheme (in the ROM) would result in a $O_\lambda(\log(|C| + |w|))$ sized transparent UC NIZK in the ROM alone, that does not rely on any knowledge assumptions, and only assumes the hardness of computing discrete logarithms. One hurdle to overcome for such an instantiation is that while Bulletproofs are known to be simulation extractable in the AGM [44], there is at present no such analysis in the random oracle model alone (to our knowledge).

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.

References

1. Abdalla, M., Barbosa, M., Katz, J., Loss, J., Xu, J.: Algebraic adversaries in the universal composability framework. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part III. LNCS, vol. 13092, pp. 311–341. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92078-4_11
2. Abdolmaleki, B., Ramacher, S., Slamanig, D.: Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1987–2005. ACM Press (Nov 2020). <https://doi.org/10.1145/3372297.3417228>
3. Ames, S., Hazay, C., Ishai, Y., Venkatasubramanian, M.: Liger: Lightweight sub-linear arguments without a trusted setup. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 2087–2104. ACM Press (Oct / Nov 2017). <https://doi.org/10.1145/3133956.3134104>
4. Arun, A., Ganesh, C., Lokam, S., Mopuri, T., Sridhar, S.: Dew: Transparent constant-sized zkSNARKs. Cryptology ePrint Archive, Report 2022/419 (2022), <https://eprint.iacr.org/2022/419>
5. Atapoor, S., Bagheri, K.: Simulation extractability in groth’s zk-SNARK. Cryptology ePrint Archive, Report 2019/641 (2019), <https://eprint.iacr.org/2019/641>
6. Bagheri, K.: Subversion-resistant simulation (knowledge) sound NIZKs. In: Albrecht, M. (ed.) 17th IMA International Conference on Cryptography and Coding. LNCS, vol. 11929, pp. 42–63. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-35199-1_3

7. Bagheri, K., Kohlweiss, M., Siim, J., Volkhov, M.: Another look at extraction and randomization of groth's zk-snark. In: Borisov, N., Díaz, C. (eds.) FC 2021. Lecture Notes in Computer Science, vol. 12674, pp. 457–475. Springer (2021). https://doi.org/10.1007/978-3-662-64322-8_22, https://doi.org/10.1007/978-3-662-64322-8_22
8. Bagheri, K., Pindado, Z., Ràfols, C.: Simulation extractable versions of groth's zk-SNARK revisited. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) CANS 20. LNCS, vol. 12579, pp. 453–461. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-65411-5_22
9. Bagheri, K., Sedaghat, M.: Tiramisu: Black-box simulation extractable nizks in the updatable CRS model. In: Conti, M., Stevens, M., Krenn, S. (eds.) CANS 2021. Lecture Notes in Computer Science, vol. 13099, pp. 531–551. Springer (2021). https://doi.org/10.1007/978-3-030-92548-2_28, https://doi.org/10.1007/978-3-030-92548-2_28
10. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993). <https://doi.org/10.1145/168588.168596>
11. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046 (2018), <https://eprint.iacr.org/2018/046>, <https://eprint.iacr.org/2018/046>
12. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046 (2018), <https://eprint.iacr.org/2018/046>
13. Ben-Sasson, E., Chiesa, A., Gabizon, A., Virza, M.: Quasi-linear size zero knowledge from linear-algebraic PCPs. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part II. LNCS, vol. 9563, pp. 33–64. Springer, Heidelberg (Jan 2016). https://doi.org/10.1007/978-3-662-49099-0_2
14. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 103–128. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_4
15. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_2
16. Blum, M., Feldman, P., Micali, S.: Proving security against chosen cyphertext attacks. In: Goldwasser, S. (ed.) CRYPTO'88. LNCS, vol. 403, pp. 256–268. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34799-2_20
17. Blum, M., Santis, A.D., Micali, S., Persiano, G.: Noninteractive zero-knowledge. SIAM J. Comput. **20**(6), 1084–1118 (1991). <https://doi.org/10.1137/0220068>, <https://doi.org/10.1137/0220068>
18. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_14
19. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_12

20. Bowe, S., Gabizon, A.: 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>, <https://eprint.iacr.org/2018/187>
21. Bowe, S., Gabizon, A., Green, M.D.: A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. In: Zohar, A., Eyal, I., Teague, V., Clark, J., Bracciali, A., Pintore, F., Sala, M. (eds.) FC 2018 Workshops. LNCS, vol. 10958, pp. 64–77. Springer, Heidelberg (Mar 2019). https://doi.org/10.1007/978-3-662-58820-8_5
22. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press (May 2018). <https://doi.org/10.1109/SP.2018.00020>
23. Bünz, B., Chiesa, A., Mishra, P., Spooner, N.: Recursive proof composition from accumulation schemes. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 1–18. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64378-2_1
24. Camenisch, J., Damgård, I.: Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 331–345. Springer, Heidelberg (Dec 2000). https://doi.org/10.1007/3-540-44448-3_25
25. Camenisch, J., Drijvers, M., Gagliardoni, T., Lehmann, A., Neven, G.: The wonderful world of global random oracles. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 280–312. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78381-9_11
26. Campanelli, M., Faonio, A., Fiore, D., Querol, A., Rodríguez, H.: Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part III. LNCS, vol. 13092, pp. 3–33. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92078-4_1
27. Campanelli, M., Ganesh, C., Khoshakhlagh, H., Siim, J.: Impossibilities in succinct arguments: Black-box extraction and more. Cryptology ePrint Archive, Report 2022/638 (2022), <https://eprint.iacr.org/2022/638>
28. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001). <https://doi.org/10.1109/SFCS.2001.959888>
29. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (Feb 2007). https://doi.org/10.1007/978-3-540-70936-7_4
30. Canetti, R., Jain, A., Scafuro, A.: Practical UC security with a global random oracle. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014. pp. 597–608. ACM Press (Nov 2014). <https://doi.org/10.1145/2660267.2660374>
31. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC. pp. 494–503. ACM Press (May 2002). <https://doi.org/10.1145/509907.509980>
32. Canetti, R., Sarkar, P., Wang, X.: Triply adaptive UC NIZK. Cryptology ePrint Archive, Report 2020/1212 (2020), <https://eprint.iacr.org/2020/1212>
33. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.P.: Marlin: Pre-processing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45721-1_26

34. Chiesa, A., Ojha, D., Spooner, N.: Fractal: Post-quantum and transparent recursive proofs from holography. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 769–793. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45721-1_27
35. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_33
36. Dodis, Y., Shoup, V., Walfish, S.: Efficient constructions of composable commitments and zero-knowledge proofs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 515–535. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_29
37. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: 23rd ACM STOC. pp. 542–552. ACM Press (May 1991). <https://doi.org/10.1145/103418.103474>
38. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 60–79. Springer, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34931-7_5
39. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_10
40. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_2
41. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019), <https://eprint.iacr.org/2019/953>
42. Ganesh, C., Khoshakhlagh, H., Kohlweiss, M., Nitulescu, A., Zajac, M.: What makes fiat-shamir zkSNARKs (updatable SRS) simulation extractable? In: Galdi, C., Jarecki, S. (eds.) SCN 2022. Lecture Notes in Computer Science, vol. 13409, pp. 735–760. Springer (2022). https://doi.org/10.1007/978-3-031-14791-3_32, https://doi.org/10.1007/978-3-031-14791-3_32
43. Ganesh, C., Kondi, Y., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Witness-succinct universally-composable snarks. Cryptology ePrint Archive, Paper 2022/1618 (2022), <https://eprint.iacr.org/2022/1618>, <https://eprint.iacr.org/2022/1618>
44. Ganesh, C., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 397–426. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_14
45. Ganesh, C., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Fiat-shamir bulletproofs are non-malleable (in the random oracle model). Cryptology ePrint Archive, Paper 2023/147 (2023), <https://eprint.iacr.org/2023/147>, <https://eprint.iacr.org/2023/147>
46. Garay, J.A., MacKenzie, P.D., Yang, K.: Strengthening zero-knowledge protocols using signatures. Journal of Cryptology **19**(2), 169–209 (Apr 2006). <https://doi.org/10.1007/s00145-005-0307-3>

47. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_37
48. Goldreich, O., Håstad, J.: On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.* **67**(4), 205–214 (1998)
49. Goldreich, O., Vadhan, S., Wigderson, A.: On interactive proofs with a laconic prover. *Computational Complexity* **11**(1), 1–53 (2002)
50. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (Dec 2006). https://doi.org/10.1007/11935230_29
51. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_11
52. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 698–728. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96878-0_24
53. Groth, J., Maller, M.: Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 581–612. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63715-0_20
54. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_21
55. Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. *J. ACM* **59**(3), 11:1–11:35 (2012). <https://doi.org/10.1145/2220357.2220358>, <https://doi.org/10.1145/2220357.2220358>
56. Jain, A., Pandey, O.: Non-malleable zero knowledge: Black-box constructions and definitional relationships. In: Abdalla, M., Prisco, R.D. (eds.) SCN 14. LNCS, vol. 8642, pp. 435–454. Springer, Heidelberg (Sep 2014). https://doi.org/10.1007/978-3-319-10879-7_25
57. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8_11
58. Katsumata, S.: A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 580–610. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84245-1_20
59. Kattis, A., Panarin, K., Vlasov, A.: RedShift: Transparent SNARKs from list polynomial commitment IOPs. *Cryptology ePrint Archive*, Report 2019/1400 (2019), <https://eprint.iacr.org/2019/1400>
60. Kerber, T., Kiayias, A., Kohlweiss, M.: Composition with knowledge assumptions. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 364–393. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_13

61. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: 24th ACM STOC. pp. 723–732. ACM Press (May 1992). <https://doi.org/10.1145/129712.129782>
62. Kondi, Y., shelat, a.: 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>
63. Kosba, A., Zhao, Z., Miller, A., Qian, Y., Chan, H., Papamanthou, C., Pass, R., shelat, a., Shi, E.: $C\emptyset c\emptyset$: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093 (2015), <https://eprint.iacr.org/2015/1093>
64. Lipmaa, H.: Simulation-extractable SNARKs revisited. Cryptology ePrint Archive, Report 2019/612 (2019), <https://eprint.iacr.org/2019/612>
65. Lysyanskaya, A., Rosenbloom, L.N.: 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>, <https://eprint.iacr.org/2022/1484>
66. Lysyanskaya, A., Rosenbloom, L.N.: Universally composable sigma-protocols in the global random-oracle model. Cryptology ePrint Archive, Report 2022/290 (2022), <https://eprint.iacr.org/2022/290>
67. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2111–2128. ACM Press (Nov 2019). <https://doi.org/10.1145/3319535.3339817>
68. Maurer, U.: Constructive cryptography - A new paradigm for security definitions and proofs. In: Mödersheim, S., Palamidessi, C. (eds.) Theory of Security and Applications - Joint Workshop, TOSCA 2011, LNCS, vol. 6993, pp. 33–56. Springer (2011). https://doi.org/10.1007/978-3-642-27375-9_3, https://doi.org/10.1007/978-3-642-27375-9_3
69. Micali, S.: Computationally sound proofs. SIAM J. Comput. **30**(4), 1253–1298 (2000). <https://doi.org/10.1137/S0097539795284959>, <https://doi.org/10.1137/S0097539795284959>
70. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252. IEEE Computer Society Press (May 2013). <https://doi.org/10.1109/SP.2013.47>
71. Pass, R.: On deniability in the common reference string and random oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_19
72. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 533–542. ACM Press (May 2005). <https://doi.org/10.1145/1060590.1060670>
73. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS. pp. 543–553. IEEE Computer Society Press (Oct 1999). <https://doi.org/10.1109/SFFCS.1999.814628>
74. Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 755–784. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_25