Revisiting Witness-Succinct UC-lifting Compilers

Davide Li Calsi, Luigi Russo, Dominique Schröder, and Federico Valbusa

TU Wien, Vienna, Austria
{davide.li.calsi, luigi.russo, dominique.schroeder, federico.valbusa}@tuwien.ac.at

Abstract. Zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) are increasingly used within larger cryptographic protocols, making security under composition essential. Generic compilation to Universal Composability (UC) requires straight-line, black-box extraction while preserving succinctness and controlling prover cost.

We introduce verifiable reconstructible straightline codes (VRSCs), a primitive that separates public hiding, transcript-based reconstruction, binding, and proof non-rerandomizability. This abstraction exposes a missing security requirement in the witness-succinct UC compiler of Ganesh et al. (EUROCRYPT 2023). Permuting previously revealed evaluation openings produces a fresh accepting proof with non-negligible probability without violating non-extrapolation, invalidating the claimed simulation argument. Proof non-rerandomizability captures this attack at the certificate level. Our construction prevents transfers between repetitions through verifier-enforced disjoint evaluation domains.

We formulate a generic UC lifting theorem in terms of VRSCs and give a pairing-based instantiation using a Fischlin-style search over authenticated codeword symbols. Structured Reed–Solomon encoding and jointly computed KZG openings reduce certificate-generation work from quadratic to $O_\lambda(n \log n)$, where n is the witness size. Only a linear-time encoding check enters the underlying SNARK circuit. To our knowledge, this is the first generic UC lifting compiler that simultaneously achieves quasi-linear prover overhead and constant additional proof size and verification work with respect to the witness size.

1 Introduction

Zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) [Mic00] certify the validity of a statement while hiding the associated witness beyond what the public statement reveals. This combination of succinctness and privacy makes them a natural building block for larger protocols, such as ZK rollups, zkLogin [BCJ+24], and privacy-preserving cryptocurrencies such as Zcash [BCG+14]. In these applications, the security of the surrounding protocol depends on how the proof system interacts with other protocol components and concurrent executions. Security in isolation does not, by itself, establish these guarantees.

Composable security frameworks such as Universal Composability [Can01] provide tools for reasoning about such interactions. Realizing the zkSNARK functionality in UC requires *straight-line extraction*: the simulator must extract a witness from a prover's execution without rewinding it, because it interacts with an external environment whose execution it cannot rewind.

Obtaining this guarantee generically remains costly. Widely used SNARKs with constant-size proofs, such as Groth16 [Gro16] and the Plonk family [GWC19], do not automatically provide the straight-line extraction that realizing the SNARK functionality in UC requires. Existing composability results must also be understood relative to their models and ideal functionalities. For example, the UC treatment of Groth16 by [BFK+24] uses a generic-group model within UC and a modified functionality.

Generic lifting compilers address this gap, but their prover costs can asymptotically exceed those of the underlying SNARK. Let n denote the number of field elements in the witness supplied to the lifting compiler. We write O_λ to suppress dependence on the security parameter. The lifting steps of [GKO+23] and [BCC+25] require $O_\lambda(n^2)$ prover work. The preprocessing-based compiler of [CCC+25] reduces this to $O_\lambda(n^{1+\varepsilon})$, but does not retain proof size independent of n. These results leave open the following question:

Can generic UC lifting achieve quasi-linear prover overhead while preserving witness succinctness?

We answer this question affirmatively by making the extraction mechanism a standalone cryptographic primitive. The compiler of [GKO+23] already separates polynomial encoding and commitment schemes from their use in UC lifting. We abstract their combined extraction role into a *Verifiable Reconstructible Straightline Code* (VRSC), whose security properties suffice for a generic UC lifting theorem. This abstraction lets us revisit both the security and the efficiency of UC lifting. It exposes a missing security requirement in the original compiler and guides our quasi-linear construction.

The abstraction exposes a security failure. The VRSC abstraction reveals a missing requirement in the compiler of [GKO+23]: proof non-rerandomizability. Their non-extrapolation property prevents producing an opening at a new evaluation point, but does not prevent rearranging existing openings into a fresh accepting proof. We give such an attack with non-negligible success probability, invalidating the claimed simulation argument.[1] Our construction prevents these transfers by assigning disjoint evaluation domains to repetitions and enforcing membership during verification.

Following our disclosure, the authors of [GKO+23] have incorporated this fix into the current version of their full version [GKO+22], which acknowledges the issue and restores the security proof.

From the primitive to UC lifting. We combine a VRSC with a simulation-extractable SNARK. The SNARK certifies that the encoded word contains a valid witness for the statement, while the VRSC enables its reconstruction from the prover's oracle queries without rewinding. The resulting lifting theorem separates the compiler's security argument from the construction used to provide the VRSC.

A coding-based construction. The abstraction also leads to an efficient realization. We interpret the authenticated polynomial evaluations as symbols of a Reed–Solomon code. Fischlin's search [Fis05] exposes enough symbols through the oracle queries for reconstruction, while random padding hides the witness in the smaller public view. Choosing a structured domain of roots of unity enables fast codeword computation and joint computation of KZG opening proofs [KZG10; FK23]. Together, these reduce certificate-generation work from quadratic to $O_\lambda(n \log n)$. The disjoint repetition domains partition this single structured domain, preserving the efficiency of the joint computation.

Keeping the added circuit small. The VRSC interface separates the encoding algorithm Encode, which computes the commitment, from certificate generation CLeak. Only the linear-time encoding check enters the SNARK circuit; codeword computation and opening-proof generation run outside it. Consequently, the compiler adds $O_\lambda(n)$ gates and requires $O_\lambda(n \log n)$ work for certificate generation, with additional proof size and verification work independent of n. If the original circuit has size M and $n = O(M)$, a quasi-linear underlying SNARK prover retains $O_\lambda(M \log M)$ complexity after compilation.

Beyond UC lifting. The VRSC abstraction also enables a black-box construction of *publicly verifiable secret sharing*. Combined with a SNARK and public-key encryption, it provides non-interactive dealing, publicly verifiable encrypted shares, and straight-line extractability under static corruption. Our instantiation gives constant-size public records per party. Qualified parties reconstruct the secret as a field element, and the extractor recovers the shares directly from the dealer's oracle-query transcript. This makes the shared values available for subsequent field arithmetic, for example as inputs to MPC. The construction uses VRSC only through its interface, demonstrating its value as a cryptographic primitive beyond the lifting compiler.

1.1 Our Contributions

Our contributions are organized as follows.

[1] We focus on the attack against the original compiler, noting that similar attacks also apply to subsequent adaptations, including [RZM+26, Sec. 4.1] and the "RO-only" variant of [BCC+25, Sec. 4.1].

A cryptographic primitive for straight-line extraction. We define VRSCs through public hiding, binding, transcript-based reconstructability, and proof non-rerandomizability. The definition separates the ability to supply an opening from the ability to reconstruct one without rewinding. This makes the extraction component a separate object of cryptographic study. As such, VRSCs are useful beyond the compiler we build with them here; we illustrate this with a black-box construction of publicly verifiable secret sharing (Section 4.3). With our code-based instantiation, this gives us the first publicly verifiable t out of n secret-sharing scheme with straight-line extractability and shares of size independent of t.

A concrete attack and an explicit security requirement. We identify a permutation attack on the compiler of [GKO+23] that succeeds with non-negligible probability. VRSC proof non-rerandomizability captures the missing requirement, and our concrete construction uses verifier-enforced disjoint evaluation domains to prevent transfers of openings between repetitions.

A witness-succinct UC compiler from VRSCs. We formulate a lifting theorem combining a VRSC with a simulation-extractable SNARK (see Section 5). The reduction uses the SNARK to certify a valid opening and the VRSC to make that opening recoverable from the oracle transcript. The theorem separates the cost of checking the encoding inside the SNARK from that of generating and verifying the VRSC certificate outside it.

A quasi-linear instantiation. We give a Reed–Solomon and KZG construction with $O_\lambda(n)$ encoding time, $O_\lambda(n \log n)$ certificate-generation time, and $O_\lambda(1)$ certificate size and verification work. To our knowledge, this gives the first generic UC lifting compiler that simultaneously achieves quasi-linear prover overhead and constant additional proof size and verification work with respect to the witness size. We can instantiate our compiler with Plonk, which is simulation extractable in the ROM [Lip25]. Besides Plonk, other possible candidates would be the scheme of [GM17] or those derived from applying the transform of [ARS20], which however rely on knowledge assumptions.

As a further result, we rule out sparse generator matrices as a route to linear-time MDS encoding for the parameters used by the Fischlin construction. This identifies a restriction on one possible optimization strategy, rather than a general lower bound for MDS encoding or a proof of optimality of quasi-linear prover time.

	Prover work	**Verifier work**	**Proof size**
[GKO+23]	$O_\lambda(n^2)$	$O_\lambda(1)$	$O_\lambda(1)$
[BCC+25]	$O_\lambda(n^2)$	$O_\lambda(n)$	$O_\lambda(\log n)$
[CCC+25]	$O_\lambda(n^{1+\varepsilon})$	$O_\lambda(\text{polylog}\, n)$	$O_\lambda(\text{polylog}\, n)$
This work	$O_\lambda(n \log n)$	$O_\lambda(1)$	$O_\lambda(1)$

Table 1. Asymptotic costs of the lifting component, excluding the underlying SNARK. The parameter n is the witness size in field elements, and O_λ suppresses dependence on the security parameter. The construction of [CCC+25] uses preprocessing.

1.2 Related work

The first witness-succinct UC-lifting compiler was proposed by [GKO+23]. Their compiler achieves witness-succinctness while preserving the verifier complexity of the underlying SNARK, but incurs $O_\lambda(n^2)$ prover time. Their work additionally requires a trusted structured CRS, i.e., a local setup assumption.

The work of [BCC+25] removes the need for such a structured local CRS by working in the global, non-programmable random oracle model. This improvement comes at the cost of a relaxation of the UC framework and an increase in proof size to $O_\lambda(\log n)$.

More recently, [CCC+25] improved the prover complexity of [GKO+23] using a form of polynomial processing and a different encoding strategy. Their compiler achieves $O_\lambda(n^{1+\varepsilon})$ prover time for any constant

$\varepsilon > 0$. This improvement, however, comes with both polylog n proof size and a preprocessing requirement. Our compiler further improves these trade-offs. In the pairing-based setting, we achieve $O_\lambda(n \log n)$ prover time while retaining the witness-succinctness and constant-size proofs of the most efficient instantiation of [GKO+23]. In particular, our construction improves both the asymptotic and practical prover complexity without requiring the polynomial preprocessing used by [CCC+25].

Other generic lifting compilers have been proposed, but all incurring an overhead that prevents simultaneously achieving witness succinctness and black-box extraction. At high level, they augment a zkSNARK by encrypting the witness and proving the correctness of the encryption. The underlying idea was introduced in [Sah99; DDO+01; Gro06], was further optimized in the C∅C∅ framework [KZM+15], and then specialized to Groth16 in [AB19; Bag19].

The result of [CF24] leverages the straight-line extractability of Kilian-like constructions. This approach is applicable only to arguments arising from Interactive Oracle Proofs [BCS16], an interactive form of PCPs, and therefore does not provide a generic compiler for arbitrary SNARKs.

Finally, a line of research [GM17; GKK+22; DG23; KPT23; FFK+23; FFR24; CFR25; Lip25; FR26] has shown that notable zkSNARKs are simulation-extractable, but they either require non-blackbox extraction, rewinding, or idealized models like GGM/AGM in addition to ROM.

Non-interactive PVSS [Sta96; Sch99; HV09; CD17; GJM+21; CD20; Bag25; Gro21] has a dealer commit to a degree-t polynomial ϕ with $\phi(0) = s$ and distribute encrypted evaluations with proofs of correct encryption; all of these reconstruct only $S := h^s \in \mathbb{G}$, a one-way image of s, via extraction that rewinds the dealer — the limitation we discuss in Section B. Bacho and Loss [BL23] gave the first adaptively secure PVSS, but only under the algebraic group model and the non-standard one-more discrete-logarithm assumption; Bacho, Chen, and Loss [BCL26] remove this reliance via a straight-line-extractable ElGamal encryption of S, still only the one-way image, and via bespoke NIZKs. Our black-box PVSS construction (Section B) instead reuses VRSC off the shelf to straight-line extract the raw scalar s itself, employing constant-size shares. Katz [Kat24] similarly extracts raw shares straight-line via a NIZK knowledge extractor, as part of a distributed key-generation protocol, but only for an honest majority ($t < n/2$), since full security there requires guaranteed output delivery; our PVSS instead supports an arbitrary threshold $t < n$.

2 Technical Overview

The extraction mechanism in a UC lifting compiler must make the witness recoverable without rewinding while keeping the public proof hiding and succinct. In the mechanism of [GKO+23], the public proof and the prover's random-oracle queries reveal different amounts of information about the witness. Our starting point is to treat this combined extraction mechanism as a cryptographic primitive in its own right. We isolate the guarantees needed by the compiler in a *Verifiable Reconstructible Straightline Code* (VRSC), separating them from the polynomial encoding and commitment algorithms used to provide them. This abstraction yields two insights. First, it exposes a missing security requirement: proof non-rerandomizability. We demonstrate the gap by rearranging existing evaluation openings into a fresh accepting proof. Second, viewing the authenticated evaluations as codeword symbols reveals the freedom to choose a structured evaluation domain and compute opening proofs jointly. Our realization combines disjoint domains for the repetitions with this structure, preventing the transfers used by the attack while reducing certificate-generation work from quadratic to quasi-linear in the witness size. Figure 1 summarizes how the primitive connects these security and efficiency insights to UC lifting.

2.1 From an Extraction Mechanism to a Primitive

In the compiler of [GKO+23], the verifier and the extractor learn different amounts of information about the same polynomial. For concreteness, consider their coefficient encoding: the prover encodes a witness $\mathrm{w} \in \mathbb{F}^n$ as a polynomial f_{w} of degree below d, with fresh, uniformly random field elements as additional coefficients. These random coefficients mask the witness's contribution to the small number of evaluations revealed in the public proof. Revealing only these few evaluations preserves hiding, while exposing d evaluations at distinct

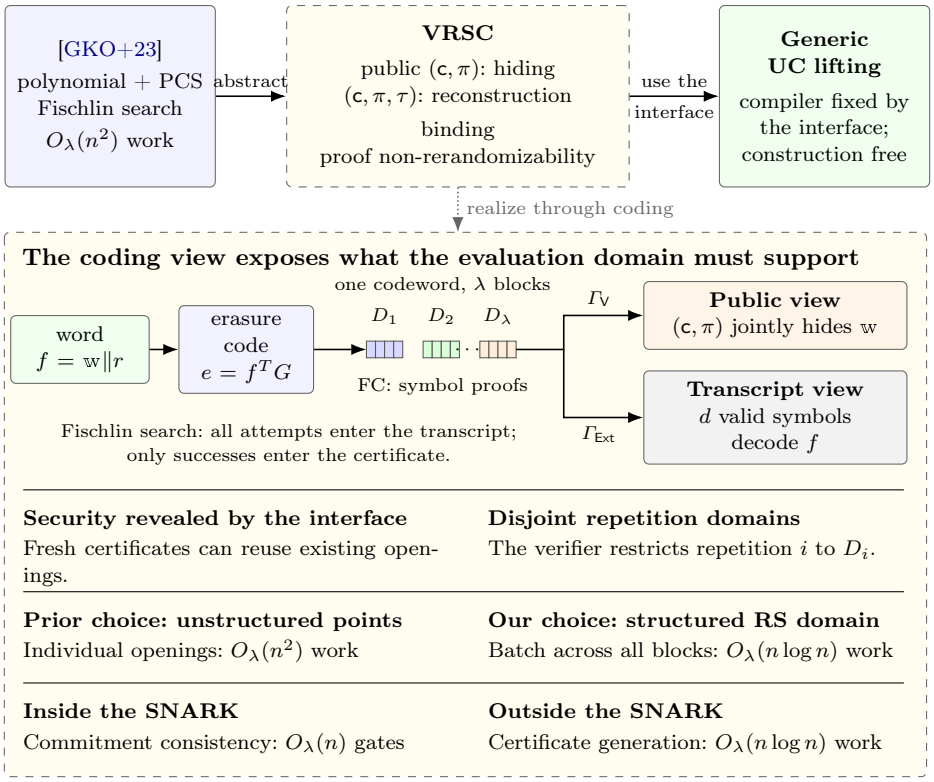

Fig. 1. The conceptual path from [GKO+23] to our framework. We isolate the extraction mechanism's cryptographic requirements in a VRSC, which supports generic UC lifting and exposes both a missing security obligation and freedom in the choice of construction. After padding, the commitment c and certificate π jointly hide the witness. Here τ denotes the oracle transcript. Given the commitment, certificate, and oracle transcript, reconstruction recovers a consistent word without receiving an opening or rewinding; the guarantee applies when the adversary can also supply a valid opening. Proof non-rerandomizability addresses fresh certificates formed from existing openings, a possibility not excluded by non-extrapolation. In our realization, the codeword is partitioned into blocks indexed by pairwise disjoint evaluation domains D_i. The verifier restricts repetition i to D_i, preventing transfers of openings between repetitions. These blocks partition one roots-of-unity domain, retaining fast Reed–Solomon evaluation and joint computation of opening proofs across the entire codeword. Only commitment consistency enters the SNARK circuit; certificate generation remains outside. Additional proof size and verification work remain independent of witness size.

points allows the extractor to recover the polynomial and hence the witness. The public view of an honest execution must remain hiding, while fresh accepting proofs must expose enough information for extraction after previously simulated proofs have been accounted for.

To ensure that the evaluations refer to the witness being proved, the prover commits to $f_{\mathbb{w}}$ with a polynomial commitment c. A SNARK for an extended relation certifies both the original statement and consistency of c with the witness encoding. In the UC security proof, the adversary may see simulated proofs before producing a proof of its own. We therefore require *simulation extractability*, which allows the security reduction to extract a valid witness for the extended relation from a fresh SNARK proof even after such exposure. This witness includes a valid opening of c, establishing the premise needed for VRSC reconstructability. The UC simulator itself recovers the witness from the oracle-query transcript without rewinding. Evaluation proofs then authenticate the revealed values against this same commitment. Hiding concerns the joint view of the commitment, evaluations, and opening proofs. Bounded independence of the evaluations alone is not sufficient. Their KZG instantiation uses one more random coefficient than the

number of revealed evaluations to mask the commitment as well, as the commitment c may be viewed as an additional polynomial evaluation.

Fischlin's search [Fis05] turns this difference between the public proof and the oracle transcript into an extraction guarantee. For their simple parameter choice of λ repetitions, the prover samples evaluation points without replacement within each repetition, proves the evaluations against c, and hashes each attempt until the first b output bits are zero. The hash input binds the attempt to the session, extended circuit, commitment, evaluation point and value, opening proof, and repetition index. Each fresh hash query succeeds with probability 2^{-b}. Only the first successful attempt from each repetition enters the final proof, whereas the oracle transcript also contains the failed attempts. Thus the verifier sees only λ evaluations, while the extractor can read the evaluations and opening proofs used throughout the search.

Why must even a malicious prover expose enough distinct evaluations? Evaluation binding and unique opening proofs ensure that, for a fixed commitment, a point provides at most one valid hash candidate per repetition, except with negligible probability. Repeating a query gives the same answer, and changing the value or opening proof cannot provide another valid candidate. Set $b = 1 + \log d$, assuming for simplicity that d is a power of two. With fewer than d distinct points, the probability of finding a successful valid candidate in any one repetition is at most $d/2^b = 1/2$. The repetition index separates the hash inputs, so passing all λ repetitions while exposing fewer than d distinct points in total has probability at most $2^{-\lambda}$ for a fixed statement and commitment. The formal analysis accounts for adaptive queries and the polynomially many statement-commitment pairs considered by the adversary [GKO+23]. After handling previously simulated proofs, a fresh accepting proof exposes at least d valid evaluations at distinct points in the observed transcript, except with negligible probability. The extractor filters valid openings, deduplicates evaluation points across repetitions, interpolates f_{w}, and decodes the witness without rewinding.

For the honest prover, each repetition in [GKO+23] is capped at $T = 2d(\lambda + \log \lambda)$ attempts and aborts if none succeeds. This choice makes honest search failure negligible. The cap controls honest running time and the distinct-candidate argument above explains extraction even when the prover does not follow the prescribed search.

Their generic theorem requires evaluation binding, unique opening proofs, joint evaluation hiding, and *non-extrapolation*. The last property tries to prevent extending the evaluations and proofs of a randomized encoding of a "dummy" witness to a *new* point. This is necessary because their simulator encodes and commits to a dummy witness, and invokes the NIZK simulator to produce a proof; without non-extrapolation, an adversary could recycle the same simulated NIZK proof while supplying new evaluations for the same commitment, forging a fresh compiled proof. As we elaborate next, this property is however not sufficient to prevent re-randomization attacks.

For the ordinary KZG instance, obtained by setting the maximum degree bound to d, an individually computed opening takes $O(d)$ group operations, and the search makes $O(\lambda d)$ attempts in expectation. With $d = n + O(\lambda)$, the additional prover work is $O_\lambda(n^2)$. This is the cost of that implementation, rather than a lower bound for their generic compiler.

Isolating the Requirements. Ganesh et al. [GKO+23] already separate their generic compiler from the polynomial encoding and commitment schemes used to instantiate it. We take the *combined extraction mechanism* as a *primitive in its own right* and ask which guarantees its interface must provide. This change of perspective is the main insight to bypass the quadratic prover overhead.

The compiler needs a public encoding and certificate that hide the witness, together with a way to reconstruct an opening from the prover's oracle queries. This requirement can be stated independently of polynomials, evaluation points, or a particular commitment scheme. We capture it in a cryptographic primitive, called a *Verifiable Reconstructible Straightline Code* (VRSC). After setup generates $(\mathsf{cpk}, \mathsf{cvk})$, the interface consists of

$$\mathsf{c} \leftarrow \mathsf{VRSC.Encode}^{\mathcal{O}}(\mathsf{cpk}, f), \quad \pi \leftarrow \mathsf{VRSC.CLeak}^{\mathcal{O}}(\mathsf{cpk}, f), \quad \mathsf{VRSC.Vf}^{\mathcal{O}}(\mathsf{cvk}, \mathsf{c}, \pi) \in \{0, 1\}.$$

Here f is a word, c its encoding, and π a publicly verifiable certificate (which we view as a controlled form of leakage on the witness, hence the name CLeak), with certificate generation allowed to return $\perp$. In our

application $f = \mathbb{w}\|r$ contains a short random padding. The certificate π attests that the prover has made enough information available through its oracle queries to enable reconstruction under the conditions below. It can be checked using cvk without revealing f. We separate Encode from CLeak because the compiler checks consistency with the encoding inside the SNARK, while certificate generation runs outside it. The primitive has four security requirements:

- *Hiding:* after fresh padding, an efficient distinguisher cannot distinguish witness choices even with access to the same (random) oracle before and after receiving (c, π). Our construction additionally gives statistical hiding of the public pair alone.
- *Binding:* it is infeasible to produce two distinct words that encode to the same c.
- *Reconstructability:* whenever an adversary produces an accepting certificate and can also supply a valid opening of c, an extractor given (c, π, τ) recovers a word consistent with c, except with negligible probability. Here τ is the oracle-query transcript. The extractor receives no opening and does not rewind the adversary.
- *Proof non-rerandomizability:* a certificate for an encoding of a random word cannot efficiently be transformed into a different accepting certificate for the same encoding, except with negligible probability.

The distinction between supplying an opening and recovering it from a transcript is essential: the former is a premise of the security guarantee, while the latter is the task performed by the extractor. Correctness requires honest certificate generation and verification to succeed except with negligible probability. Succinctness requires certificate size and verification time to be $O_\lambda(\mathrm{polylog}\,|f|)$. Our instantiation will make both *independent* of the witness size.

A Missing Compositional Security Requirement. The VRSC abstraction separates reconstruction from proof non-rerandomizability. The former concerns recovering an encoded word from the oracle transcript. The latter prevents an adversary from converting an exposed certificate into a different accepting certificate for the same encoding. This distinction reveals a concrete failure in the simulation argument of [GKO+23]. Their reduction to non-extrapolation relies on the implication that a modified vector of evaluation points $\widetilde{\mathbf{z}}$ contains a previously unopened point $z_i \in \mathbf{z}$. However,

$$\widetilde{\mathbf{z}} \neq \mathbf{z} \quad \not\Rightarrow \quad \exists i : \ \widetilde{z}_i \notin \{z_1, \ldots, z_\lambda\}.$$

An adversary can change the certificate while preserving its entire set of authenticated evaluations.

Specifically, retain the commitment and the simulated underlying SNARK proof, and exchange two complete evaluation-opening triples at distinct points. All algebraic verification checks remain satisfied. Although the hash inputs include the repetition index, only the two moved triples require additional successful hash checks. The remaining checks retain their previously successful inputs. For fresh cross-position queries, the transposition therefore succeeds with probability

$$2^{-2b} = \frac{1}{4d^2} \qquad \text{when } b = 1 + \log_2 d,$$

assuming for simplicity that d is a power of two. This probability is non-negligible for polynomially bounded d. In particular, increasing the number of repetitions does not amplify security against this modification: only the changed positions incur a new search cost.

The resulting certificate contains no new evaluation point, so it cannot yield a winning instance of non-extrapolation. Nevertheless, it is a fresh compiled proof retaining the previously simulated SNARK component, and hence triggers precisely the simulator rejection event that the reduction claims is negligible. Thus, the attack invalidates the claimed hybrid indistinguishability, rather than merely weakening its quantitative bound.

The VRSC interface makes the missing obligation explicit through proof non-rerandomizability, which concerns the entire certificate. Our construction addresses this case by assigning pairwise disjoint admissible evaluation domains to repetitions and enforcing domain membership during verification. An opening valid for one repetition cannot then be transferred to another. This restriction acts on the authenticated evaluation points themselves.

Beyond UC-lifting compiler. Treating this extraction mechanism as a primitive in its own right pays off beyond the compiler we build with it here. Since $\mathsf{VRSC.Encode}$ and $\mathsf{VRSC.CLeak}$ already give a non-interactive, publicly verifiable, straight-line-reconstructible commitment to a witness, pairing them with a small outer SNARK and a public-key encryption scheme yields, essentially for free, a black-box construction of publicly verifiable secret sharing (Section 4.3, with the full treatment in Section B). Unlike the classical Stadler–Schoenmakers lineage, which reconstructs only a one-way image of the secret via rewinding, this construction recovers the dealt secret itself, straight-line, with constant-size shares.

2.2 Using the VRSC Interface

Binding Reconstruction to the Statement. Recovering a word from c is useful for UC lifting only if that word contains a witness for the statement being proved. Akin to [GKO+23], a SNARK establishes this connection. For an original relation $\mathcal{R}$, the extended relation $\mathcal{R}_{\mathsf{cpk}}$ checks, with oracle access suppressed,

$$(\mathbb{x}, \mathbb{w}) \in \mathcal{R} \quad \wedge \quad \mathsf{c} = \mathsf{VRSC.Encode}(\mathsf{cpk}, \mathbb{w} \| r),$$

on public input $(\mathbb{x}, \mathsf{c})$ and witness $(\mathbb{w}, r)$. The prover produces one VRSC certificate π_1 for the padded word and one SNARK proof π_2 for $\mathcal{R}_{\mathsf{cpk}}$. The compiled proof is $(\mathsf{c}, \pi_1, \pi_2)$, and verification checks both proofs.

The two components have complementary roles in the security argument. For a fresh SNARK proof, simulation extractability supplies a valid opening of c inside the reduction. This establishes the premise needed to invoke VRSC reconstructability. The straight-line simulator itself runs the VRSC extractor on $(\tau, \mathsf{c}, \pi_1)$; it does not run the SNARK's non-black-box extractor. Binding then connects the reconstructed word to the witness certified by the SNARK. For proofs derived from previously simulated executions, proof non-rerandomizability controls the creation of a fresh certificate for an existing encoding. One further case remains: the adversary copies a simulated pair (c, π_1) and attaches a fresh SNARK proof. By binding, the word certified by that proof is the padded zero word chosen by the simulator, and the simulator recovers it from the oracle queries it made itself when generating π_1, which it keeps alongside the observed transcript.

The resulting lifting theorem depends on the VRSC properties, so changing the VRSC construction does not require changing the generic compiler. Theorem 10 formalizes this argument in the presence of a global random oracle with query observation and a local setup functionality. The SNARK setup uses fresh randomness independent of the VRSC setup, allowing simulation of the SNARK while preserving the VRSC key used in the security reduction.

2.3 Realizing the Interface Through Coding

To realize this interface efficiently, we return to the evaluations used in the original compiler. A polynomial evaluation is a linear function of its coefficients. Viewing these evaluations as codeword symbols exposes which properties the evaluation domain must provide, and hence which choices are available for improving efficiency.

One Codeword, Two Views. Write the padded witness as a coefficient vector $f = \mathbb{w} \| r \in \mathbb{F}^d$. Evaluating its polynomial at z is the inner product of f with $(1, z, \ldots, z^{d-1})^T$. Placing these vectors side by side gives a generator matrix $G \in \mathbb{F}^{d \times \overline{N}}$ and a codeword

$$e = f^T G \in \mathbb{F}^{\overline{N}}, \qquad e_{i,j} = \langle f, g_{i,j} \rangle.$$

Here G is partitioned into λ blocks of N columns, $\overline{N} = \lambda N$, and $g_{i,j}$ denotes column j of block i. Repetition i opens only columns in its own block. We use a terminology pertaining to coding theory as G can be viewed as the generator matrix of a linear code. An opening authenticates the selected symbol against c. For reconstruction we use an MDS code: any d distinct symbols of the full codeword determine f. The search induces two erasure channels on this codeword, that we denote Γ_{V} and Γ_{Ext} (cf. Figure 1). The verifier receives through Γ_{V} only the successful response from each of λ repetitions, so its view contains at most λ

distinct symbols. The extractor observes through Γ_{Ext} the oracle queries, in which valid symbols absent from that transcript are erasures from its perspective. The parameters force a successful prover to expose enough distinct valid symbols for decoding, except with negligible probability. Padding masks the first view, while an erasure code supports reconstruction from the second.

Authenticating Distinct Symbols. An inner-product functional commitment [CFK+22] binds c to f and permits proofs $\sigma_{i,j}$ that $e_{i,j} = \langle f, g_{i,j} \rangle$. In repetition i, the prover searches only its block for a position satisfying

$$\mathsf{H}(\mathsf{c}, i, j, e_{i,j}, \sigma_{i,j}) = 0^b.$$

Counting hash queries alone is insufficient: repeated queries about the same symbol would provide no additional information for decoding. Evaluation binding prevents inconsistent accepted values at a position. Unique opening proofs prevent the prover from generating additional hash candidates for a fixed position and value merely by changing $\sigma_{i,j}$. These properties connect successful search to the number of distinct authenticated codeword symbols in the transcript. In our KZG instantiation they give the quasi-unique responses required by Fischlin's original analysis: for a fixed commitment and challenge, producing two distinct accepting responses is infeasible. This allows us to use the original transform. Without quasi-uniqueness, extraction can fail; the randomized variant of [Ks22] addresses that setting.

For the MDS instantiation, set $b = 1 + \log d$ as before. The distinct-candidate argument from the polynomial setting now applies to codeword positions: fewer than d distinct valid symbols give too few opportunities to pass all λ repetitions, except with negligible probability. Theorem 5 gives the formal argument. The code tolerates $\overline{N} - d$ erasures. The extractor verifies the openings in the transcript and solves the linear system given by any d distinct valid symbols. Evaluation binding makes these symbols consistent with the valid opening supplied in the reconstructability experiment, so decoding recovers a word that encodes to c.

Masking the Joint Public View. Reconstruction alone does not ensure public hiding. We pad with $r \leftarrow\!\$\ \mathbb{F}^\ell$ for $\ell = \lambda + 1$, so $d = n + \ell$, matching the ℓ-hiding definition, and write

$$e = \mathbb{w}^T G_{\mathrm{hi}} + r^T G_{\mathrm{lo}},$$

where the two row blocks correspond to the witness and padding. For a set S of at most λ revealed positions, full column rank of G_{lo}^S makes the random term mask the revealed symbols. The commitment must be hidden jointly with them. In our KZG instantiation, the commitment is one further evaluation in the exponent. The padding rows of the augmented evaluation matrix have full column rank, except for a negligible setup event. The extra random coordinate masks this evaluation as well. The opening proofs are determined by these evaluations and the setup, yielding statistical hiding of the complete public pair (c, π).

2.4 Turning the Coding View into an Efficient Instantiation

Choosing a Structured Domain. Reconstruction, masking, and verifiable responses determine the security requirements on the domain. They leave room for a structured choice, as shown in the lower panel of Figure 1. To determine its size, observe that each fresh hash check in honest certificate generation succeeds with probability $2^{-b} = 1/(2d)$. With $N = 2\lambda d$ candidate positions per repetition, the probability that one repetition fails is at most

$$(1 - 1/(2d))^N \leq e^{-\lambda}.$$

A union bound over the λ repetitions gives negligible abort probability. Each repetition uses its own block of N positions, so the complete codeword has length $\overline{N} = \lambda N$ with $\overline{N} = 2\lambda^2 d$. Membership in the assigned block is enforced by the verifier, preventing transfers of openings between repetitions.

We choose a Reed–Solomon code whose locators are the $\overline{N}$-th roots of unity in $\mathbb{F}_p$, with $\overline{N} \mid p - 1$. For power-of-two $\overline{N}$, the NTT computes the full codeword in $O(\overline{N} \log \overline{N})$ field operations. The technique of [FK23] jointly computes the $\overline{N}$ KZG opening proofs within the same asymptotic operation count. Each

repetition then searches its own precomputed block. This yields $O_\lambda(n \log n)$ certificate-generation time, while retaining the explicit λ^2 factor in the full-domain batch size.

Both parts of the computation matter: fast codeword generation alone would leave the cost of individual opening proofs unchanged. The structured domain accelerates both, while the VRSC interface keeps the compiler independent of this choice. The resulting commitment and certificate have size $O_\lambda(1)$, and verification takes $O_\lambda(1)$ work, with respect to witness size.

Keeping the Added Circuit Small. To retain this benefit in the compiled SNARK, we must also account for the work added to its circuit. The VRSC algorithm Encode computes the commitment c to the padded word in $O_\lambda(n)$ time. The larger codeword $e = f^T G$ and its opening proofs are computed by CLeak, outside the SNARK circuit. Only consistency with Encode is checked inside the extended relation. Consequently, the additional circuit has $O_\lambda(n)$ gates even though certificate generation takes $O_\lambda(n \log n)$ work.

If the original circuit has size M, a quasi-linear SNARK prover applied to the extended relation therefore costs $O_\lambda((M+n)\log(M+n))$, in addition to the certificate-generation work. When $n = O(M)$, this is $O_\lambda(M \log M)$ overall. Thus the distinction between Encode and CLeak, shown in the last row of Figure 1, preserves the prover's quasi-linear asymptotic cost while keeping the additional proof size and verification work independent of witness size.

A Constraint on Further Improvements. The same view helps assess possible optimizations. For example, one might try to compute $f^T G$ in linear time by making G sparse. For an $[\overline{N}, d]$ MDS code, every row of G has at most $d-1$ zeros: otherwise, d zero positions give a singular $d \times d$ submatrix. Hence G has at least $d(\overline{N} - d + 1)$ nonzero entries, ruling out linear-time encoding by sparse matrix multiplication for our parameters. This is a restriction on that optimization strategy; it does not establish a general lower bound for MDS encoding or optimality of the quasi-linear construction. For the power-of-two root domains used here, the standard NTT bound is $O(\overline{N} \log \overline{N})$. The sparsity argument above excludes one route to linear-time multiplication. It does not prove that the NTT or the full VRSC prover is asymptotically optimal. However, while not known to be impossible, encoding $[n, k]$ MDS codes in $o(n \log n)$ time is a long-standing open problem in coding theory, which suggests that $O_\lambda(n \log n)$ is the best complexity achievable by constructions of this type with current techniques.

3 Preliminaries

Notation. For an integer $n \geq 1$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. A function f is negligible in λ (we write $f \in \mathsf{negl}(\lambda)$) if it approaches zero faster than the reciprocal of any polynomial. We write $f = O_\lambda(g(n))$ if there exists a fixed polynomial λ^c such that $f = O(\lambda^c \cdot g(n))$. We write $y \leftarrow \mathcal{A}(x)$ to denote the output of algorithm $\mathcal{A}$ on input x. If $\mathcal{A}$ is randomized, we write $\mathcal{A}(x; r)$ to make its random coins r explicit, and omit r when the coins are sampled independently and uniformly. We write PPT for probabilistic polynomial-time algorithms, whose running time is bounded by a polynomial in the input length for every choice of random coins.

3.1 The Universal Composability Framework

The Universal Composability (UC) framework [Can01] follows a simulation-based paradigm. Rather than reasoning directly about the security of a protocol, one first specifies the desired behavior by means of an *ideal functionality* $\mathcal{F}_X$, which models a trusted party implementing the target task. Security is then defined by comparing two worlds. In the *ideal world*, honest parties interact only with $\mathcal{F}_X$, while corrupted parties are controlled by an ideal-world adversary Sim, called the *simulator*. In the *real world*, parties execute the actual protocol in the presence of a real-world adversary.

A protocol Π UC-realizes $\mathcal{F}_X$ if, for every real-world adversary $\mathcal{A}$, there exists a simulator Sim such that no interactive environment $\mathcal{Z}$ can distinguish whether it is interacting with the real protocol execution or with the ideal execution involving $\mathcal{F}_X$. Intuitively, any attack against the real protocol can therefore be

reproduced in the ideal world without violating the security guarantees captured by the ideal functionality. A fundamental advantage of the UC framework is its universal composition theorem, which guarantees that protocols proven UC-secure remain secure when composed with arbitrary UC-secure protocols.

Generalized UC and global functionalities. Our constructions are analyzed in the Generalized Universal Composability (GUC) framework [CDP+07]. Unlike the standard UC model, GUC allows protocols to access *global functionalities*, namely ideal functionalities that are shared across multiple protocol instances and are therefore not local to a single execution. Such functionalities model global resources that are consistently accessible throughout the system. In this work, the only global functionality we use is a global random oracle.

Specifically, we work in the $\mathcal{G}_{\mathsf{RO}}$-hybrid model, where parties have oracle access to the non-programmable global random oracle introduced in [CJS14]. We adopt the simplified formulation of [CDG+18] and we formally define $\mathcal{G}_{\mathsf{RO}}$ in Figure 2.

The functionality can be accessed by every party as well as by the simulator through the commands QUERY and OBSERVE. Upon receiving a fresh query (QUERY, x), $\mathcal{G}_{\mathsf{RO}}$ lazily samples a uniformly random output, stores the resulting input–output pair in an internal table $\mathcal{Q}$, and returns the sampled value. Repeated queries are answered consistently by looking up the previously stored value in $\mathcal{Q}$.

A key feature of $\mathcal{G}_{\mathsf{RO}}$ is the notion of *illegitimate queries*. Every party is associated with a party identifier pid and a session identifier sid. Whenever a party issues a query (QUERY, x), the functionality interprets the input as (s, x'), where s denotes the session identifier encoded in the query. The query is said to be *illegitimate* whenever s differs from the session identifier associated with the querying party. Intuitively, these are oracle queries that refer to executions outside the current protocol session. Illegitimate queries can be retrieved through the OBSERVE command, and this is essential for simulation. During the security proof, the simulator reconstructs the view of corrupted parties by observing their oracle queries. However, the environment may issue additional oracle queries by spawning dummy parties that are not part of the protocol execution being simulated. Since the simulator is unaware of these auxiliary parties, it would otherwise fail to reproduce the environment's view. By classifying such queries as illegitimate and making them available through OBSERVE, the functionality ensures that the simulator receives exactly the additional information required to carry out the simulation.

Finally, observe that oracle queries issued by parties participating in the current session, as well as those issued by the simulator itself, are never classified as illegitimate. Consequently, the simulator's own oracle queries remain hidden from both the environment and corrupted parties. This property is crucial for establishing indistinguishability between the real and ideal executions in our security proofs.

The $\mathcal{G}_{RO}$ functionality

Upon receiving (QUERY, x) from a party $\mathcal{P} = (\mathsf{pid}, \mathsf{sid})$ or the simulator Sim:

- Check if there is a pair (x, v) for some $v \in \{0,1\}^{\ell(\lambda)}$ in the (initially empty) list $\mathcal{Q}$ of past queries. Else, choose uniformly $v \in \{0,1\}^{\ell(\lambda)}$ and store the pair (x, v) in $\mathcal{Q}$.
- Parse x as (s, x'). If $\mathsf{sid} \neq s$ then add (s, x', v) to the (initially empty) list of illegitimate queries for SID s, that is denoted by $\mathcal{Q}_{|s}$.
- Return v to $\mathcal{P}$.

Upon receiving $(\text{OBSERVE}, \mathsf{sid})$ from the simulator Sim: return the list $\mathcal{Q}_{|\mathsf{sid}}$ of illegitimate queries for SID sid to Sim.

Fig. 2. The Global RO functionality [CDG+18], parametrized by output length $\ell(\lambda)$.

3.2 Non-interactive Arguments

A *non-interactive argument system* (NARG) for relation $\mathcal{R}$ in the random oracle model, denoted by Π, consists of a tuple of algorithms $(\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ having black-box access to a random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\ell(\lambda)}$, with the following syntax:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: takes as input the security parameter 1^λ and outputs public parameters $\mathsf{pp} := (\mathsf{pk}, \mathsf{vk})$.
- $\pi \leftarrow \mathsf{P}^{\mathsf{H}}(\mathsf{pk}, \mathbb{x}, \mathbb{w})$: takes as input a proving key pk, a statement $\mathbb{x}$ and witness $\mathbb{w}$, and outputs a proof π if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$.
- $b \leftarrow \mathsf{V}^{\mathsf{H}}(\mathsf{vk}, \mathbb{x}, \pi)$: takes as input a verifying key vk, a statement $\mathbb{x}$ and proof π, and it accepts ($b = 1$) or rejects ($b = 0$).

A NARG is said *succinct*, and simply denoted as SNARG, if the running time of V is $O_\lambda(|\mathbb{x}| + \mathrm{polylog}\,|\mathbb{w}|)$ and the proof size $|\pi|$ is $O_\lambda(\mathrm{polylog}\,|\mathbb{w}|)$. Explicitly distinguishing a prover key pk from a verifier key vk is essential for succinct arguments: while pk (and thus pp) may be linear in the witness size, vk may be much shorter. Giving the verifier the entire pp would therefore preclude verifier running time sublinear in $|\mathbb{w}|$.

Definition 1 (Completeness). *A NARG Π satisfies* completeness *if for every $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, it holds that*

$$\Pr\left[b = 1 : (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda); \pi \leftarrow \mathsf{P}^{\mathsf{H}}(\mathsf{pk}, \mathbb{x}, \mathbb{w}); b \leftarrow \mathsf{V}^{\mathsf{H}}(\mathsf{vk}, \mathbb{x}, \pi)\right] = 1.$$

In the following, we define soundness and if this property holds with respect to a computationally unbounded adversary, we say that the argument has *statistical* (or *information-theoretic*) soundness, and we call the argument a *proof* system.

Definition 2 (Soundness). *A NARG Π satisfies* soundness *if for every PPT adversary $\mathcal{A}$, it holds that*

$$\Pr\left[b = 1 \wedge \mathbb{x} \notin \mathcal{L}_{\mathcal{R}} : (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda); (\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{pk}, \mathsf{vk}); b \leftarrow \mathsf{V}^{\mathsf{H}}(\mathsf{vk}, \mathbb{x}, \pi)\right] \in \mathsf{negl}(\lambda),$$

where $\mathcal{L}_{\mathcal{R}} := \{\mathbb{x} \mid \exists \mathbb{w} : (\mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$ is the set of true-statements.

Zero-knowledge. Informally, an argument is zero-knowledge if a proof reveals no information about the witness [GMR85]. We formalize hereafter a notion of zero-knowledge in the non-programmable random oracle model (NPROM) [Wee09], where the simulator $\mathcal{S}$ is not allowed to reprogram the random oracle. We therefore define zero-knowledge with respect to a trapdoor generated by $\mathsf{SimSetup}$, and all parties, including the simulator, query the same honest oracle H. The simulator neither observes nor programs the queries of the adversary and its power comes from the trapdoor alone. Looking ahead, this allows to run the code of $\mathcal{S}$ without interfering with a gloabl random oracle.

Zero-knowledge is a security property that is only guaranteed for valid statements in the language, hence $\mathcal{S}$ should only be queried on true statements. For this reason, we introduce a wrapper oracle $\mathcal{S}'$ that, on input a trapdoor td and a query $(\mathbb{x}, \mathbb{w})$ returns the output of $\mathcal{S}^{\mathsf{H}}(\mathsf{td}, \mathbb{x})$ if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, otherwise returns $\perp$.

Definition 3 (Unbounded Zero-knowledge). *A NARG Π is zero-knowledge if there exist PPT algorithms $\mathsf{SimSetup}, \mathcal{S}$ (with wrapper oracle $\mathcal{S}'$) such that for all PPT adversaries $\mathcal{A}$:*

$$\Pr\left[\begin{matrix}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathcal{A}^{\mathsf{P},\mathsf{H}}(\mathsf{pp}) = 1\end{matrix}\right] \approx \Pr\left[\begin{matrix}(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{SimSetup}(1^\lambda) \\ \mathcal{A}^{\mathcal{S}'(\mathsf{td},\cdot),\mathsf{H}}(\mathsf{pp}) = 1\end{matrix}\right]$$

A zero-knowledge NARG with statistical soundness is simply referred to as a non-interactive zero-knowledge proof (NIZK) [BFM88].

Simulation extractability. A strong security property of NARGs is *simulation extractability* [GM17]. Informally, from a prover producing a valid proof after seeing polynomially many proofs provided by a simulator (even for *false* statements), it is possible to extract a valid witness. This property can be formalized in many different ways (see [DDO+01; PR05; JP14]), depending on the model and the constraints of the adversary and the extractor. Hereafter, we present a non-blackbox definition in the explicitly programmable random oracle model (EPROM) [Wee09] adapted from [GM17]. In particular, following the syntax of [FKM+12], we define a stateful zero-knolwedge simulator $\mathcal{S}$, whose initial state $\mathsf{st} = \mathsf{pp}$, as an algorithm that operates in two modes. The first mode, $(y, \mathsf{st}') \leftarrow \mathcal{S}(1, \mathsf{st}, a)$, takes care of handling calls to the oracle H on input a query string a. The second mode, $(\pi, \mathsf{st}') \leftarrow \mathcal{S}(2, \mathsf{st}, \mathbb{x})$ simulates a proof for the statement $\mathbb{x}$. We also define two wrapper oracles that share their internal state st.

- $\mathcal{S}_1(a)$ returns the first output of $\mathcal{S}(1, \mathsf{st}, a)$
- $\mathcal{S}_2(\mathbb{x})$ returns the first output of $\mathcal{S}(2, \mathsf{st}, \mathbb{x})$

Definition 4 (Simulation extractability). *A NARG Π is (non-black-box)* simulation-extractable *with respect to a zero-knowledge stateful simulator $\mathcal{S}$, with wrapper oracles $\mathcal{S}_1, \mathcal{S}_2$, if for all PPT adversaries $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{V}^{\mathcal{S}_1}(\mathbb{x}, \pi) = 1 & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \wedge\ (\mathbb{x}, \pi) \notin \mathcal{Q} & (\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\mathcal{S}_1, \mathcal{S}_2}(\mathsf{pp}) \\ \wedge\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} & \mathbb{w} \leftarrow \mathcal{E}(\mathsf{st}, \mathbb{x}, \pi) \end{array}\right] \in \mathsf{negl}(\lambda)$$

where $\mathcal{Q}$ is the set of statement-proof pairs obtained by $\mathcal{A}$ when interacting with $\mathcal{S}_2$.

The previous definition implies the weaker notion of *knowledge soundness*, which differs only in that the adversary is not given access to $\mathcal{S}_2$. A NARG (resp. SNARG, if succinct) that is knowledge-sound is called NARK (resp. SNARK) and we may use the prefix "zk" to indicate that the system additionally satisfies zero knowledge. In Figure 3 we give a formal definition of a UC functionality for a non-interactive argument of knowledge.

The $\mathcal{F}_{\mathsf{NIZK}}$ functionality

Upon receiving $(\textsc{prove}, \mathsf{sid}, \mathbb{x}, \mathbb{w})$ from a party $\mathcal{P}_i$:

- Ignore if $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$, otherwise send $(\textsc{prove}, \mathsf{sid}, \mathbb{x})$ to Sim.
- Upon receiving $(\textsc{proof}, \mathsf{sid}, \mathbb{x}, \pi)$ from the simulator Sim, store $(\mathbb{x}, \pi)$ in the (initially empty) list $\mathcal{Q}$ and output $(\textsc{proof}, \mathsf{sid}, \mathbb{x}, \pi)$ to $\mathcal{P}_i$.

Upon receiving $(\textsc{verify}, \mathsf{sid}, \mathbb{x}, \pi)$ from a party $\mathcal{P}_i$:

- If $(\mathbb{x}, \pi)$ is not stored in $\mathcal{Q}$, then send $(\textsc{verify}, \mathsf{sid}, \mathbb{x}, \pi)$ to Sim.
- Upon receiving $(\textsc{witness}, \mathbb{w})$ from Sim, if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, store $(\mathbb{x}, \pi)$ in $\mathcal{Q}$.
- Return $(\textsc{verification}, \mathsf{sid}, (\mathbb{x}, \pi) \in_? \mathcal{Q})$ to $\mathcal{P}_i$

Fig. 3. The NIZK functionality adapted from [GOS06], parametrized by polynomial-time-decidable relation $\mathcal{R}$, run with N parties $\mathcal{P}_1, \ldots, \mathcal{P}_N$.

3.3 Finite Fields, Primes, Polynomials

In the following p denotes a large prime number. It is a well-known fact that all the finite fields of order p are isomorphic to $(\mathbb{Z}_p, +, \cdot)$, with $+$ and $\cdot$ denoting the sum and product modulo p, respectively. We will use the

notation $\mathbb{F}_p$ to refer to any representation of said field. The multiplicative group $\mathbb{F}_p^* = (\mathbb{F}_p \setminus \{0\}, \cdot)$ contains $p-1$ elements, and is cyclic. An element $\theta \in \mathbb{F}_p$ has order s iff s is the smallest positive integer such that $\theta^s = 1$; if $s = p-1$ then $\mathbb{F}_p^* = \langle\theta\rangle$. Furthermore, the following is a well-known fact.

Proposition 1. *The number of distinct elements of $\mathbb{F}_p^*$ with order s is*

$$\begin{cases} 0 & p \not\equiv 1 \mod s \\ \varphi(s) & p \equiv 1 \mod s \end{cases},$$

where φ denotes Euler's totient function.

Furthermore, one can efficiently generate all said elements from a generator g of $\mathbb{F}_p^*$ by computing $g^{k \cdot \frac{p-1}{s}}$ for any integer $1 \leq k \leq s$ coprime with s.

Fix a positive integer m dividing $p-1$. The elements of $\mathbb{F}_p^*$ whose order divides m form the group of m-th roots of unity. This group contains exactly m elements . We call the elements that have order *exactly* m the *primitive* m-th roots of unity. We will use the following well-known result.

Theorem 1. *Let m be a power of two and p a prime with $m \mid p-1$. Then $\mathbb{F}_p$ has $\varphi(m)$ primitive m-th roots of unity and exactly m m-th roots of unity. Choose a primitive m-th root ζ and order the evaluation points as $1, \zeta, \ldots, \zeta^{m-1}$. For $f \in \mathbb{F}_p[X]$ of degree less than m, the NTT computes these m evaluations in $O(m \log m)$ field operations, and the inverse NTT recovers its coefficients in the same operation count.*

3.4 Error Correcting Codes

Let $\mathbb{F}_q$ denote the finite field with q elements.

Definition 5 (Linear code). *A* linear code *C of length n and dimension k over $\mathbb{F}_q$ is a k-dimensional linear subspace of $\mathbb{F}_q^n$. We refer to C as an $[n,k]_q$ code.*

It is a well-known fact that a linear q-ary code can be viewed as the span over $\mathbb{F}_q$ of a *generator* matrix $G \in \mathbb{F}_q^{k \times n}$. As a consequence, any codeword $c \in C$ equals $x^T G$ for some $x \in \mathbb{F}_q^k$.

Definition 6 (Hamming distance and weight). *For $x, y \in \mathbb{F}_q^n$, the* Hamming distance *$d(x,y)$ is the number of coordinates in which x and y differ. The* Hamming weight *of x is $\mathrm{wt}(x) = d(x,0)$. The* minimum distance *of a code C is*

$$d(C) = \min_{\substack{x,y \in C \\ x \neq y}} d(x,y) = \min_{\substack{x \in C \\ x \neq 0}} \mathrm{wt}(x),$$

where the second equality holds because C is linear.

An $[n,k]_q$ code with minimum distance d is called an $[n,k,d]_q$ code.

Proposition 2 (Singleton bound). *Every $[n,k,d]_q$ code satisfies $d \leq n-k+1$.*

Definition 7 (MDS code). *An $[n,k,d]_q$ code is called* maximum distance separable (MDS) *if it meets the Singleton bound with equality, i.e., $d = n-k+1$.*

With a slight abuse of terminology we say that a $k \times n$ matrix G is MDS if it generates a linear MDS code. Vandermonde matrices and the codes they generate, known as Reed-Solomon codes, constitute a prime example of MDS matrices and codes, respectively.

In this work we make use of the following standard result.

Theorem 2 (Reed Solomon codes are MDS). *Let $q \geq n$. Fix any n pairwise distinct $a_1, \ldots, a_n \in \mathbb{F}_q$. The Reed-Solomon code with locators $a_1, \ldots, a_n$ is the $[n, k]$ code generated by*

$$G = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ a_1 & a_2 & \ldots & a_n \\ a_1^2 & a_2^2 & \ldots & a_n^2 \\ \vdots & \vdots & \ldots & \vdots \\ a_1^{k-1} & a_2^{k-1} & \ldots & a_n^{k-1} \end{pmatrix}.$$

Such a code is MDS.

Recall that a code with minimum distance d can correct up to $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ errors. Since, by the Singleton bound, an MDS code achieves the largest possible minimum distance $d = n - k + 1$ for a given length n and dimension k, it follows that, among all $[n, k]_q$ codes, MDS codes achieve the best possible error-correcting capability. Likewise, MDS codes are remarkably effective against *erasures*. Namely, given any k symbols of a codeword $c = x^T G$ one can reconstruct x unambiguously, thereby tolerating an optimal $n - k$ number of erasures.

3.5 Functional Commitments and Encodings

A functional commitment scheme [LRY16] allows committing to a function f and later open the function at a desired input x. We follow the formalization of [CP23], but we specialize it to support succinct arguments.

Definition 8 (Functional Commitment). *A functional commitment scheme* FCom *for a function family $\mathcal{F} = \{f : \mathbb{X} \to \mathbb{Y}\}$, where $\mathbb{X}$ is a set of opening inputs and $\mathbb{Y}$ is the space of outputs, is a tuple of algorithms with the following interfaces:*

- $(\mathsf{fpk}, \mathsf{fvk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *: on input the security parameter 1^λ, outputs a commitment key* fpk *and a verifying key* fvk*;*
- $\mathsf{c} \leftarrow \mathsf{Com}(\mathsf{fpk}, f \in \mathcal{F})$*: on input a commitment key* fpk *and a function f, outputs a commitment* c *to f;*
- $\pi \leftarrow \mathsf{Open}(\mathsf{fpk}, f, x \in \mathbb{X})$*: on input a commitment key* fpk*, function f, and opening input x, outputs a proof π attesting to the value of $f(x)$:*
- $b \leftarrow \mathsf{Vf}(\mathsf{fvk}, \mathsf{c}, x \in \mathbb{X}, y \in \mathbb{Y}, \pi)$*: on input a verifying key* fvk*, some* c *that purportedly commits to some function f, an opening input x, a claimed value y, and a purported proof π that $f(x) = y$, it outputs a bit b.*

Throughout this work we mostly fix $\mathcal{F}$ to the family of inner products with a fixed d-dimensional vector over a finite field, i.e.

$$\mathcal{F} = \{f_v : \mathbb{F}_q^d \to \mathbb{F}_q \mid v \in \mathbb{F}_q^d\}$$

where $f_v(x) := \langle v, x \rangle$ for all $x \in \mathbb{F}_q^d$. This implicitly sets $\mathbb{X} = \mathbb{F}_q^d$ and $\mathbb{Y} = \mathbb{F}_q$. We refer to this specialization as *inner product functional commitments*. Henceforth we occasionally abuse notation and write v in place of f_v, and vice versa, when the intended meaning is clear from context. We also write $\mathbb{F}$ for $\mathbb{F}_q$, since all computations are over $\mathbb{F}_q$ unless stated otherwise.

A related but distinct primitive is that of polynomial commitments. One can view them as a special class of inner product functional commitments where the openings are restricted to vectors of the form $v = \left(1\ a\ a^2\ \ldots\ a^{d-1}\right)^T$ for some $a \in \mathbb{F}$. For the sake of completeness, we provide a formal definition in Section A.

A functional commitment scheme satisfies the following notions of completeness and evaluation-binding.

Definition 9 (FCS Completeness). FCom *has perfect completeness if for every $x \in \mathbb{X}$, and $f \in \mathcal{F}$*

$$\Pr\left[b = 1 \;\middle|\; \begin{array}{l} (\mathsf{fpk}, \mathsf{fvk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{c} \leftarrow \mathsf{Com}(\mathsf{fpk}, f) \\ b \leftarrow \mathsf{Vf}(\mathsf{fvk}, \mathsf{c}, x, f(x), \mathsf{Open}(\mathsf{fpk}, f, x)) \end{array} \right] = 1$$

Definition 10 (FCS Binding). FCom *is binding if for every PPT* $\mathcal{A}$

$$\Pr\left[f \neq f' \wedge \mathsf{c} = \mathsf{c}' \;\middle|\; \begin{array}{l}(\mathsf{fpk}, \mathsf{fvk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (f, f') \leftarrow \mathcal{A}(\mathsf{fpk}, \mathsf{fvk}) \\ \mathsf{c} \leftarrow \mathsf{Com}(\mathsf{fpk}, f) \\ \mathsf{c}' \leftarrow \mathsf{Com}(\mathsf{fpk}, f')\end{array}\right] \in \mathsf{negl}(\lambda)$$

Definition 11 (FCS Evaluation-Binding). FCom *is evaluation-binding if for every PPT* $\mathcal{A}$

$$\Pr\left[\begin{array}{l}y \neq y' \\ \wedge b = 1 \\ \wedge b' = 1\end{array} \;\middle|\; \begin{array}{l}(\mathsf{fpk}, \mathsf{fvk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{c}, x, y, y', \pi, \pi') \leftarrow \mathcal{A}(\mathsf{fpk}, \mathsf{fvk}) \\ b \leftarrow \mathsf{Vf}(\mathsf{fvk}, \mathsf{c}, x, y, \pi) \\ b' \leftarrow \mathsf{Vf}(\mathsf{fvk}, \mathsf{c}, x, y', \pi')\end{array}\right] \in \mathsf{negl}(\lambda)$$

[GKO+23] introduced the notion of *unique proofs* for polynomial commitments (see Section A). Informally, it requires that finding two distinct accepting proofs for a fixed commitment, input and output is computationally infeasible.

Definition 12 (FCS Unique Proof). FCom *has unique proofs if for all PPT adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{l}\pi \neq \pi' \\ \wedge b = 1 \\ \wedge b' = 1\end{array} \;\middle|\; \begin{array}{l}(\mathsf{fpk}, \mathsf{fvk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{c}, x, y, \pi, \pi') \leftarrow \mathcal{A}(\mathsf{fpk}, \mathsf{fvk}) \\ b \leftarrow \mathsf{Vf}(\mathsf{fvk}, \mathsf{c}, x, y, \pi) \\ b' \leftarrow \mathsf{Vf}(\mathsf{fvk}, \mathsf{c}, x, y, \pi')\end{array}\right] \in \mathsf{negl}(\lambda)$$

The definition above admits a natural strengthening that gets rid of the computational bound on $\mathcal{A}$. More precisely, we say that FCom has perfect unique response if there exists a setup-time bad event E with negligible probability and, conditioned on $\neg E$, any commitment-input-output triple (c, x, y) admits at most a unique accepting proof with respect to fvk.

Vector encoding schemes allow to encode a witness into a vector, and to possibly decode it at a later point in time.

Definition 13 (Vector Encoding [GKO+23]). *A vector encoding scheme* FEnc *is a pair of PPT algorithms* $(\mathsf{Encode}, \mathsf{Decode})$ *with the following interfaces:*

- $f_w \leftarrow \mathsf{Encode}(\mathbb{w}, \ell; r)$*: given a witness* $\mathbb{w} \in \mathbb{F}^n$ *and evaluation bound* $\ell \in \mathbb{N}$ *it outputs a vector* $f_w \in \mathbb{F}^{n+\ell}$*.*
- $\mathbb{w} \leftarrow \mathsf{Decode}(f_w, n, \ell)$*: given a vector* $f_w \in \mathbb{F}^{n+\ell}$ *and integers* $n, \ell \in \mathbb{N}$*, it outputs a witness* $\mathbb{w}$*.*

The encoding scheme is *correct* if for all $n, \ell > 0$ and $\mathbb{w} \in \mathbb{F}^n$, $\mathsf{Decode}(\mathsf{Encode}(\mathbb{w}, \ell; r), n, \ell) = \mathbb{w}$ with probability 1 over the sampling of r.

4 Verifiable Reconstructible Straightline Code

In this section, we formalize our new primitive *Verifiable Reconstructible Straightline Code*.

Definition 14 (VRSC). *Let* $\mathcal{O}$ *denote a fixed oracle. A Verifiable Reconstructible Straightline Code* VRSC *consists of oracle-aided PPT algorithms* $(\mathsf{Setup}, \mathsf{Encode}^{\mathcal{O}}, \mathsf{CLeak}^{\mathcal{O}}, \mathsf{Vf}^{\mathcal{O}})$ *with the following interfaces:*

- $(\mathsf{cpk}, \mathsf{cvk}) \leftarrow \mathsf{Setup}(1^\lambda)$*: on input the security parameter* 1^λ*, it outputs an encoding key* cpk *and a verifying key* cvk*.*
- $\mathsf{c} \leftarrow \mathsf{Encode}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w})$*: on input the encoding key* cpk *and a word* $\mathbb{w}$*, it outputs an encoding* c*.*
- $\pi \leftarrow \mathsf{CLeak}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w})$*: on input the encoding key* cpk *and a word* $\mathbb{w}$*, it outputs a proof* π *or the abort symbol* $\perp$*.*
- $b \leftarrow \mathsf{Vf}^{\mathcal{O}}(\mathsf{cvk}, \mathsf{c}, \pi)$*: on input the verifying key* cvk*, an encoding* c *and a proof* π*, it outputs a bit* b*.*

We say that VRSC *is* succinct *if the running time of verifying a proof is* $O_\lambda(\text{polylog}\,|\mathbb{w}|)$ *and the proof size* $|\pi|$ *is* $O_\lambda(\text{polylog}\,|\mathbb{w}|)$.

Additionally, we use the expression *oracle-free encoding* to denote VRSCs such that the Encode procedure makes no call to $\mathcal{O}$.

We say that VRSC is *correct* if for any $\mathbb{w}$ the procedure $\mathsf{c} \leftarrow \mathsf{Encode}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w})$, $\pi \leftarrow \mathsf{CLeak}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w})$ and $b \leftarrow \mathsf{Vf}^{\mathcal{O}}(\mathsf{cvk}, \mathsf{c}, \mathsf{CLeak}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w}))$ satisfies

$$\Pr[\pi \neq \bot \wedge b = 1] \geq 1 - \mathsf{negl}(\lambda)$$

for some negligible function $\mathsf{negl}(\lambda)$. We further formalize the properties of hiding, binding, reconstructability, and proof non-rerandomizability. At high level, the first requires that an accepting proof and the encoding leak no information on $\mathbb{w}$. The second dictates it is infeasible to find two distinct words mapping to the same encoding c. The third (which is a stronger version of binding) requires that it is possible to extract a word $\mathbb{w}$ compatible with an encoded word c given an accepting proof π and the transcript τ of queries made to $\mathcal{O}$. Finally, the last one requires that no adversary is able to re-randomize a proof π of an encoding c to a random word.

Definition 15 (Hiding). *For* $\beta \in \{0, 1\}$, *define* $\mathsf{Hide}^{\mathcal{A}}_{\beta}$ *by:*

1. *Generate* $(\mathsf{cpk}, \mathsf{cvk}) \leftarrow \mathsf{Setup}(1^\lambda)$.
2. *Run* $(\mathbb{w}_0, \mathbb{w}_1, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}_1(\mathsf{cpk}, \mathsf{cvk})$, *where* $\mathbb{w}_0, \mathbb{w}_1 \in \mathbb{F}^n$ *have the same length.*
3. *Independently sample* $r \leftarrow\!\!\$\ \mathbb{F}^\ell$ *and compute* $\mathsf{c} \leftarrow \mathsf{Encode}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w}_\beta \| r)$ *and* $\pi \leftarrow \mathsf{CLeak}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w}_\beta \| r)$.
4. *Return* $\mathcal{A}^{\mathcal{O}}_2(\mathsf{st}, \mathsf{c}, \pi)$.

A VRSC is ℓ-hiding *if, for every PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, *the distinguishing gap*

$$\left|\Pr[\mathsf{Hide}^{\mathcal{A}}_0 = 1] - \Pr[\mathsf{Hide}^{\mathcal{A}}_1 = 1]\right| \leq \mathsf{negl}(\lambda).$$

Notice that the adversary may retain auxiliary state generated with the public parameters, and choose its challenge words after polynomially many oracle queries. Adaptive polynomially many challenges, with fresh padding each time, are covered by a hybrid over challenge calls.

Definition 16 (Binding). *A* VRSC *is binding if for any PPT adversary* $\mathcal{A}$ *the following experiment succeeds with probability at most negligible in* λ*:*

- *Generate* $(\mathsf{cpk}, \mathsf{cvk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{c}, \mathbb{w}, \mathbb{w}') \leftarrow \mathcal{A}(\mathsf{cpk}, \mathsf{cvk})$.
- *Return 1 iff* $\mathsf{c} = \mathsf{Encode}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w}) = \mathsf{Encode}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w}')$ *and* $\mathbb{w} \neq \mathbb{w}'$.

Definition 17 (Reconstructability). *A* VRSC *is reconstructible if there exists a PPT algorithm* Ext *such that for any PPT* $\mathcal{A}$ *the following experiment succeeds with probability at most negligible in* λ*:*

1. *Compute* $(\mathsf{cpk}, \mathsf{cvk}) \leftarrow \mathsf{VRSC.Setup}(1^\lambda)$, *and* $(\mathbb{w}, \mathsf{c}, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{cpk}, \mathsf{cvk})$.
2. *Let* τ *be the transcript of queries to* $\mathcal{O}$ *made by* $\mathcal{A}$. *Compute* $\overline{\mathbb{w}} \leftarrow \mathsf{Ext}(\mathsf{cvk}, \mathsf{cpk}, \tau, \mathsf{c}, \pi)$.
3. *Return 1 if and only if* $\mathbb{w} \neq \overline{\mathbb{w}}$, $\mathsf{Encode}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w}) = \mathsf{c}$, $\mathsf{Vf}^{\mathcal{O}}(\mathsf{cvk}, \mathsf{c}, \pi) = 1$ *and* $\mathsf{Encode}^{\mathcal{O}}(\mathsf{cpk}, \overline{\mathbb{w}}) \neq \mathsf{c}$.

Definition 18 (Proof Non-Rerandomizability). *A* VRSC *is proof-non-rerandomizable if for any PPT* $\mathcal{A}$ *the following experiment succeeds with probability at most negligible in* λ*:*

- *Generate* $(\mathsf{cpk}, \mathsf{cvk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{c} \leftarrow \mathsf{Encode}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w})$, *for a word* $\mathbb{w}$ *sampled uniformly at random from* $\mathbb{F}^d$, *and* $\pi \leftarrow \mathsf{CLeak}^{\mathcal{O}}(\mathsf{cpk}, \mathbb{w})$.
- *Let* $\pi' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{cpk}, \mathsf{cvk}, \mathsf{c}, \pi)$.
- *Return 1 iff* $\mathsf{Vf}^{\mathcal{O}}(\mathsf{cvk}, \mathsf{c}, \pi') = 1$ *and* $\pi \neq \pi'$.

4.1 A Code-Theoretic Construction

In the following we assume $\mathbb{w} \in \mathbb{F}^n$, $d = n + \lambda + 1$, and $N = 2\lambda d$. Furthermore, we replace the abstract oracle $\mathcal{O}$ with a random oracle H.

Construction 1 (Fischlin VRSC) *Let* FCom *denote an inner product functional commitment scheme. Furthermore, let* $G \in \mathbb{F}^{d \times N\lambda}$ *be the generator matrix of a linear code consisting of* λ *blocks*

$$G = \left(G_1 \ldots G_\lambda\right).$$

The Fischlin VRSC with parameters (d, N, b, H) *is a VRSC as defined in Figure 4.*

Encode generates a commitment c to the input $\mathbb{w}$ via functional commitment FCom.

Write $\overline{N} = \lambda N$ and let $g_{i,j}$ be column j of $G_i \in \mathbb{F}^{d \times N}$, equivalently column $(i-1)N + j$ of G. For an encoded word $\mathbb{w} \in \mathbb{F}^d$, write $e_{i,j} = e_{(i-1)N+j} = \langle \mathbb{w}, g_{i,j} \rangle$.

For each $i = 1, \ldots, \lambda$, the CLeak procedure fetches the i-th block G_i and loops through its N columns. When testing column $g_{i,j}$ it evaluates the inner product $e_{i,j}$, computes a proof $\pi_{i,j}$ certifying the correct evaluation w.r.t. c. If the resulting transcript $(\mathsf{c}, i, j, e_{j+(i-1)N}, \pi_{i,j})$ maps to 0^b under H, CLeak records the success and moves to the next i. Verification attests that each proof of evaluation verifies, as well as that the aforementioned hash-based check is passed. Furthermore, it enforces domain separation, ensuring that the challenges used in the i-th iteration cannot be recycled for any index in $[\lambda] \setminus \{i\}$.

$\mathsf{Setup}(1^\lambda)$

1 : $(\mathsf{cpk}, \mathsf{cvk}) \leftarrow \mathsf{FCom.Setup}(1^\lambda)$
2 : **return** $(\mathsf{cpk}, \mathsf{cvk})$

$\mathsf{Encode}(\mathsf{cpk}, \mathbb{w})$

1 : $\mathsf{c} \leftarrow \mathsf{FCom.Com}(\mathsf{cpk}, \mathbb{w})$
2 : **return** c

$\mathsf{Vf}^{\mathsf{H}}(\mathsf{cvk}, \mathsf{c}, \pi)$

1 : **if** $\pi == \perp$: **return** 0
2 : $\{i, j_i, e_{i,j_i}, \pi_{i,j_i}\}_{i \in [\lambda]} \leftarrow \pi$
3 : **if** $\exists i$ s.t. $j_i \notin [N]$: **return** 0
4 : **for** $i \in [\lambda]$:
5 : $\quad \overline{G} \leftarrow G_i$
6 : $\quad$ **if** $\mathsf{FCom.Vf}(\mathsf{cvk}, \mathsf{c}, \overline{G}_{j_i}, e_{i,j_i}, \pi_{i,j_i}) = 0 \vee$
7 : $\quad \mathsf{H}(\mathsf{c}, i, j_i, e_{i,j_i}, \pi_{i,j_i}) \neq 0^b$:
8 : $\quad\quad$ **return** 0
9 : **return** 1

$\mathsf{CLeak}^{\mathsf{H}}(\mathsf{cpk}, \mathbb{w})$

1 : $e \leftarrow (\mathbb{w})^T G$
2 : $\pi \leftarrow \emptyset$
3 : $\mathsf{c} \leftarrow \mathsf{Encode}(\mathsf{cpk}, \mathbb{w})$
4 : **for** $i \in [\lambda]$:
5 : $\quad succ_i \leftarrow 0$
6 : $\quad$ **for** $j \in [N]$:
7 : $\quad\quad$ // Fetch i-th block of G
8 : $\quad\quad \overline{G} \leftarrow G_i$
9 : $\quad\quad$ // Open the j-th column of i-th block
10 : $\quad\quad \pi_{i,j} \leftarrow \mathsf{FCom.Open}(\mathsf{cpk}, \mathbb{w}, \overline{G}_j)$
11 : $\quad\quad$ **if** $\mathsf{H}(\mathsf{c}, i, j, e_{j+(i-1)N}, \pi_{i,j}) = 0^b$
12 : $\quad\quad\quad succ_i \leftarrow 1$
13 : $\quad\quad\quad \pi \leftarrow \pi \,||\, (i, j, e_{j+(i-1)N}, \pi_{i,j})$
14 : $\quad\quad\quad$ break
15 : **if** $\exists\, i \in [\lambda] \mid succ_i == 0$
16 : $\quad$ **return** $\perp$
17 : **else return** π

Fig. 4. Fischlin VRSC. The CLeak procedure enforces domain separation by only using challenges in the i-th block of G for the i-th parallel repetition. Vf enforces it by fetching the corresponding block at verification time. The output π is canonically sorted by increasing i.

Theorem 3. *If* $b = 1 + \log d$, $N = 2\lambda d$ *and* H *is modeled as a random oracle, Fischlin VRSC is correct.*

Proof. Fix an index $i \in [\lambda]$. The corresponding **for** loop evaluates H at most N times, and stops if one output equals 0^b. Each check succeeds with probability 2^{-b}; the probability that all of them fail is

$$p_i \coloneqq (1 - 2^{-b})^N \leq \exp\left((-N)/2^b\right).$$

By the union bound, the probability that this holds for one value of i is at most:

$$\sum_{i \in [\lambda]} p_i \leq \lambda e^{-\lambda}$$

which is negligible in λ.

Therefore CLeak returns $\perp$ with probability no more than that. Conditioned on CLeak not returning $\perp$, it is straightforwardly verifiable that the verifier accepts with probability 1. The claim follows.

□

For the purpose of proving that the VRSC is hiding, it is useful to define a specific encoding scheme FEnc for linear functions and a specific hiding property related to FEnc and FCom. The encoding scheme FEnc does:

- Encode: on input $\mathbb{w}$ sample $r \leftarrow\!\$\, \mathbb{F}^{1+\lambda}$. Output $\mathbb{w}||r$.
- Decode: recover $\mathbb{w}$ from the first $|\mathbb{w}|$ coordinates of $\mathbb{w}'$.

FEnc is trivially correct. Furthermore, we view the computation of $(\mathbb{w}')^T G$ as $\overline{N}$ evaluations of a linear function encoded in $\mathbb{w}'$.

Related to this specific encoding procedure, we define a notion of hiding.

Definition 19 (ℓ-Evaluation Hiding). FCom *is ℓ-evaluation hiding w.r.t* FEnc *if for all (not necessarily efficient) adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*

$$\Pr\left[b = b' \wedge |z| \leq \ell \wedge \mathsf{Valid}_H(z) \;\middle|\; \begin{array}{l} (\mathsf{fpk}, \mathsf{fvk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathbb{w}, z) \leftarrow \mathcal{A}_1(\mathsf{fpk}, \mathsf{fvk}) \\ b \leftarrow\!\$\, \{0,1\}, r \leftarrow\!\$\, \mathbb{F}^{\ell+1} \\ f_w \leftarrow \mathsf{FEnc.Encode}(b \cdot \mathbb{w}, \ell+1; r) \\ \mathsf{c} \leftarrow \mathsf{FCom.Com}(\mathsf{fpk}, f_w) \\ \pi_i \leftarrow \mathsf{Open}(\mathsf{fpk}, f_w, z_i) \text{ for } i \in [|z|] \\ b' \leftarrow \mathcal{A}_2((\pi_i, f_w(z_i))_{i \in [|z|]}, \mathsf{c}) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $z = (z_1, \ldots, z_{|z|})$ with $z_i \in \mathbb{X} \subseteq \mathbb{F}^{n+\ell+1}$, where $\mathbb{X}$ is the prescribed public opening domain, and $\mathcal{A}_1$ and $\mathcal{A}_2$ share state. The predicate $\mathsf{Valid}_H(z)$ holds exactly when the last $\ell+1$ columns of z, viewed as a $|z| \times (n+\ell+1)$ matrix over $\mathbb{F}$, have rank $|z|$. We adopt the convention that the empty list is valid, with rank zero.

For the Fischlin construction, it suffices to use $\mathbb{X} = \{g_{i,j} : i \in [\lambda], j \in [N]\}$. Furthermore, for a matrix $\hat{G}$ whose columns are evaluation points, we use $\mathsf{Valid}_H\left(\hat{G}\right)$ for the same predicate applied to its column list, equivalently to the row matrix $\hat{G}^T$.

Theorem 4. *Suppose* FCom *is λ-evaluation hiding with respect to* FEnc*, is evaluation-binding, and has deterministic, oracle-free commitment and opening algorithms and perfect completeness. Assume every λ-column submatrix of G satisfies $\mathsf{Valid}_H(\cdot)$. Except with setup probability $\eta = \mathsf{negl}(\lambda)$, assume perfect uniqueness of accepting openings and the following commitment min-entropy condition: for every fixed key, every $\mathbb{w} \in \mathbb{F}^n$, and every commitment c,*

$$\Pr_{r \leftarrow\!\$ \mathbb{F}^{\lambda+1}}[\mathsf{FCom.Com}(\mathsf{cpk}, \mathbb{w}||r) = c] \leq \mu, \qquad \mu = \mathsf{negl}(\lambda).$$

With the correctness parameters above, the Fischlin VRSC is $(\lambda+1)$-hiding in the sense of Definition 15.

Informally, high min-entropy makes a fresh commitment unlikely to occur in any prior oracle query. The stopping indices then have a word-independent geometric law. Evaluation hiding masks the public commitment and the one opening revealed in each block. To handle subsequent oracle queries, a reduction generates the conditioned oracle answers from this public view alone. Perfect uniqueness identifies the honest opening at each position, and evaluation binding bounds queries containing an accepted opening with an incorrect value. Thus public-view hiding extends to computational hiding with oracle access.

Proof (of Theorem 4). Fix $\mathbb{w}' \in \mathbb{F}^n$, let $\ell = \lambda + 1$, and sample $r \leftarrow\!\!\$\ \mathbb{F}^\ell$, where $d = n + \ell$.

We first recall that repetition i queries $\mathsf{H}(\mathsf{c}, i, j, e_{i,j}, \pi_{i,j})$ in increasing order of $j \in [N]$. Every query within this execution is fresh because its input includes (i, j). Its output is therefore independent and uniform, irrespective of the encoded word. Each trial succeeds with probability 2^{-b}, where b is the public hash length. Thus the stopping indices $J = (j_1, \ldots, j_\lambda)$ follow a distribution independent of the word, padding, and commitment — namely λ independent geometric distributions, where an outcome larger than N is converted to $\perp$. If one of the stopping indices exceeds N, the global output of CLeak is set to $\perp$, thus revealing no information on $\mathbb{w}$.

We then turn to the case where all the stopping indices are N or less. Label $z_i := g_{i,j_i}$ for $i \in [\lambda]$ and $z = (z_1, \ldots, z_\lambda)$. Note that z is a submatrix of G consisting of λ columns; by assumption, it satisfies $\mathsf{Valid}_H(z) = 1$. This enables a reduction to λ-evaluation hiding (c.f. Definition 19) to show that, for any stopping indices J sampled as above, the joint distribution of

$$(\mathsf{c}, \pi_{1,j_1}, \ldots, \pi_{\lambda,j_\lambda})$$

is within negligible statistical distance from an analogous distribution of where the commitment c and λ evaluations e_{i,j_i} and proofs π_{i,j_i} were obtained by committing and evaluating $0||r$.

It remains to rule out any information leakage due to queries to H made by $\mathcal{A}$ before or after running CLeak:

- *Prior oracle queries*: by the min-entropy condition on c, if $\mathcal{A}$ makes q_0 queries distinct queries before the invocation of CLeak, the probability that some query contains c is at most $q_0\mu$ by a simple union bound. As a consequence, none of the prior oracle queries overlaps with some query made by CLeak, and their outputs follow a uniform distribution carrying no information on $\mathbb{w}$.
- *Subsequent oracle queries*: at this stage $\mathcal{A}$ knows c, and can make queries containing it. All queries that do not contain c are fresh, and as such entail a uniformly random response independent of $\mathbb{w}$. If H receives a query of the form $(\mathsf{c}, i, j, e_{i,j}, \pi_{i,j})$ then the output is 1) 0^b if $j = j_i$, with the latter denoting the sampled stopping index; 2) a random independent element of $\{0,1\}^b \setminus \{0^b\}$ if $0 < j < j_i$; 3) a random independent element of $\{0,1\}^b$ if $j > j_i$. In virtue of the unique-openings property, the aforementioned description faithfully represents the distribution of the random oracle response. Note that this stochastic process is simulatable without knowledge of $\mathbb{w}$, and as such subsequent oracle queries provide no information on it.

□

The binding of the Fischlin VRSC immediately follows from that of the functional commitment scheme. We then prove reconstructability.

Theorem 5. *If* FCom *is evaluation-binding with perfectly unique proofs, G is MDS, and $b = 1 + \log d$, then the Fischlin VRSC is reconstructible.*

The extractor keeps valid transcript openings and interpolates from d distinct global positions. Evaluation binding makes these values agree with the committed word, and the MDS property then gives that word uniquely. Perfect uniqueness and the Fischlin counting argument bound the probability of acceptance with fewer than d transcript positions, including certificates containing unqueried oracle inputs.

Proof (of Theorem 5). The extractor scans τ for tuples $(\mathsf{c}, i, j, y, \pi)$ with $i \in [\lambda]$, $j \in [N]$, and $\mathsf{FCom.Vf}(\mathsf{cvk}, \mathsf{c}, g_{i,j}, y, \pi) = 1$. It associates each such tuple with its global position $t = (i-1)N + j \in [\overline{N}]$. If fewer than d distinct positions occur, it returns $\perp$. Otherwise, it chooses d of them, with index set S, and solves $x^T G_S = y_S^T$ for x. The MDS property makes G_S invertible. It returns x.

Evaluation binding rules out two accepted values at the same position, except with negligible probability. Perfect uniqueness then gives at most one accepted opening for each fixed (c, i, j, y). Hence, outside the evaluation-binding bad event, distinct accepted oracle inputs for a fixed commitment correspond to distinct global positions $(i-1)N + j$.

We bound the probability of an accepting certificate when fewer than d such positions occur in τ. For each commitment and each repetition, expose the oracle outputs at distinct valid opening tuples in their order of first query. These outputs are independent uniform strings. A final certificate can also contain a valid tuple that the adversary never queried. Include its fresh oracle output when the verifier checks it. If the transcript has fewer than d positions in total, there are at most d candidate outputs per repetition after this final check is included. Acceptance requires a zero output among these candidates in every repetition. The candidate sequences for distinct repetitions are independent because the hash input includes i. Padding each sequence with unused independent outputs to length d gives the upper bound

$$\left(1 - (1 - 2^{-b})^d\right)^\lambda \leq (d2^{-b})^\lambda = 2^{-\lambda}.$$

This argument allows adaptive query allocation. If q_{H} is the number of adversarial oracle queries, at most $q_{\mathsf{H}}+1$ commitments are relevant, including an unqueried final commitment. A union bound gives $(q_{\mathsf{H}}+1)2^{-\lambda}$.

Finally, consider a winning execution of Definition 17, so $\mathsf{c} = \mathsf{Encode}(\mathsf{cpk}, \mathbb{w})$ for the word output by the adversary. Every accepted transcript value must equal $\langle \mathbb{w}, g_{i,j} \rangle$, except with negligible probability. Otherwise, after learning $\mathbb{w}$, a reduction computes its honest opening at that same $g_{i,j}$ and outputs the two distinct values and their valid proofs, contradicting evaluation binding. Thus, whenever d positions are available, $x^T G_S = \mathbb{w}^T G_S$ and invertibility gives $x = \mathbb{w}$. The reconstruction failure probability is at most the sum of the evaluation-binding bad-event probabilities and $(q_{\mathsf{H}} + 1)2^{-\lambda}$, which is negligible.

□

Theorem 6. *Let $d = n + \lambda + 1$ with $n \geq 1$. Assume* FCom *has deterministic, oracle-free commitment and opening algorithms, perfect completeness, evaluation binding, perfectly unique proofs, and λ-evaluation hiding with respect to* FEnc*. Suppose G has row rank d and the last $\lambda + 1$ coordinates of any $\lambda + 1$ columns form an invertible matrix. Then the domain-separated Fischlin VRSC with canonical certificate ordering is proof-non-rerandomizable.*

Domain separation ensures that any changed accepted certificate contains a fresh opening, unless evaluation binding or perfect uniqueness fails. Together with the original λ openings, this determines the padding for any proposed witness prefix. Statistical evaluation hiding lets a reduction resample that prefix while preserving, up to negligible distance, the test that the reconstructed word has commitment c. With overwhelming probability the new prefix differs, giving two distinct words with one commitment and hence an evaluation-binding violation.

Proof (of Theorem 6). Assume, towards contradiction, that a PPT $\mathcal{A}$ causes the experiment of Definition 18 to return 1 with non-negligible probability. We sequentially rule out certain events concerning the proof π' returned by $\mathcal{A}$. We parse π' as a sequence of tuples

$$(i, j', e_{i,j'}, \pi_{i,j'}) \quad \text{for } i \in [\lambda]$$

sorted by increasing i. Likewise, parse π as an analogously sorted sequence of tuples $(i, j, e_{i,j}, \pi_{i,j})$.

1. *Rerandomization of an opening* (badEv_1): there exists one i with $(j', e_{i,j'}) = (j, e_{i,j})$ and $\pi_{i,j'} \neq \pi_{i,j}$. Then $\pi_{i,j}$ and $\pi_{i,j'}$ are two distinct accepting proofs for the same commitment-input-output triple $(\mathsf{c}, g_{i,j}, e_{i,j})$; by perfect uniqueness of the proofs of FCom, $\Pr[\mathsf{badEv}_1] = 0$.
2. *Incorrect Evaluation* (badEv_2): there exists one i with corresponding tuple $(i, j', e_{i,j'}, \pi_{i,j'})$ such that $\pi_{i,j'}$ is accepted but $e_{i,j'} \neq \langle \mathbb{w}, g_{i,j'} \rangle$. We rule out this event by reducing to the evaluation-binding of FCom. The reduction $\mathcal{B}$ receives $(\mathsf{cpk}, \mathsf{cvk})$, samples $\mathbb{w}$, computes c and π honestly, runs $\mathcal{A}$ and obtains π' from it, and searches for such an i. It computes the honest opening $\hat{\pi} \leftarrow \mathsf{FCom.Open}(\mathsf{cpk}, \mathbb{w}, g_{i,j'})$ and outputs $(\mathsf{c}, g_{i,j}, e_{i,j'}, \langle \mathbb{w}, g_{i,j'} \rangle, \pi_{i,j'}, \hat{\pi})$, so that $\Pr[\mathsf{badEv}_2] \leq \mathsf{Adv}^{\mathsf{evbind}}_{\mathcal{B}}(\lambda)$.

3. *Evaluation guessing at a fresh column* (badEv_3): there exists an i such that in the tuples $(i, j', e_{i,j'}, \pi_{i,j'})$ and $(i, j, e_{i,j}, \pi_{i,j})$ it holds $j \neq j'$ and $e_{i,j'} = \langle \mathbb{w}, g_{i,j'} \rangle$. For reasons that will be clarified later, we upper bound $\Pr[\mathsf{badEv}_3 \wedge \neg\mathsf{badEv}_2]$. Conditioning on $\neg\mathsf{badEv}_2$ requires $\mathcal{A}$ to genuinely guess an evaluation of the inner product with $\mathbb{w}$ at a fresh challenge. Consider the following hybrid argument: H_0 is the non-rerandomizability experiment; in H_1 the challenger still samples $\mathbb{w} = \mathbb{w}_L \| r$ with $\mathbb{w}_L \in \mathbb{F}^n$ and $r \in \mathbb{F}^{\lambda+1}$, but computes c and π from $\overline{\mathbb{w}} := 0 \| r$. By the $(\lambda+1)$-hiding of the Fischlin VRSC (Theorem 4, whose hypothesis on λ-column submatrices is implied by the $\mathsf{Valid}_H(\cdot)$ condition on $(1+\lambda)$-column submatrices assumed here), the view of $\mathcal{A}$ in H_0 and H_1 is within statistical distance $\mathsf{negl}(\lambda)$. Write $x = L' \| R'$ with $L' \in \mathbb{F}^n$ and $R' \in \mathbb{F}^{\lambda+1}$, so that $\langle \mathbb{w}, x \rangle = \langle \mathbb{w}_L, L' \rangle + \langle r, R' \rangle$. In H_1 the view of $\mathcal{A}$ is independent of $\mathbb{w}_L$, and $L' \neq 0$ because, by hypothesis, the top n rows of G contain no null column; hence $\langle \mathbb{w}_L, L' \rangle$ is uniform over $\mathbb{F}$ and independent of everything $\mathcal{A}$ sees, and $\mathcal{A}$ outputs $e'_i = \langle \mathbb{w}, x \rangle$ with probability exactly $1/|\mathbb{F}|$. A union bound over the at most λ indices i with $j'_i \neq j_i$ gives

$$\Pr[\mathsf{badEv}_3 \wedge \neg\mathsf{badEv}_2] \leq \frac{\lambda}{|\mathbb{F}|} + \mathsf{negl}(\lambda).$$

The events $\mathsf{badEv}_1, \mathsf{badEv}_2$, and $\mathsf{badEv}_3 \mid \neg\mathsf{badEv}_2$ are exhaustive, in the sense that ruling them out prevents any viable adversarial strategy to produce a fresh accepting π'. Taken together, they require that $\mathcal{A}$ reuses the same inputs, evaluations, and proofs. In principle $\mathcal{A}$ could supply a permutation of the λ input-output-proof tuples $(j, e_{i,j}, \pi_{i,j})$ of π, but this is ruled out by domain separation: a column $g_{i,j}$ of G constituting a valid challenge for i is invalid for all indices in $[\lambda] \setminus \{i\}$. Altogether, $\Pr[\mathcal{A} \text{ wins}] \leq \mathsf{Adv}^{\mathsf{evbind}}_{\mathcal{B}}(\lambda) + \lambda/|\mathbb{F}| + \mathsf{negl}(\lambda)$, which is negligible.

□

4.2 KZG-based VRSC

In this subsection, we show how to instantiate the VRSC of Figure 4 with a pairing-based approach, achieving linear encoding time, quasi-linear prover time, and constant-size proof.

Complexity Assumptions on Bilinear Groups. We work with a bilinear group generator $\mathcal{G}(1^\lambda)$ that outputs $\mathbf{p} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient, non-degenerate bilinear pairing (Type-3, i.e. no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ is known), and $[1]_\iota$ is a fixed generator of $\mathbb{G}_\iota$. We use the standard bracket notation: for $a \in \mathbb{Z}_p$ we write $[a]_\iota := a \cdot [1]_\iota$, and $[a]_1 \bullet [b]_2 := \hat{e}([a]_1, [b]_2) = [ab]_T$. Let $\mathbb{F} = \mathbb{Z}_p$. For a set $S \subset \mathbb{F}$ we write $Z_S(X) := \prod_{t \in S}(X - t)$ for its vanishing polynomial.

Definition 20 (n-SDH [BB04]). *The n-Strong Diffie-Hellman (n-SDH) assumption holds for $\mathcal{G}$ if, for every PPT adversary $\mathcal{A}$:*

$$\Pr\left[y = [1/(\tau + c)]_1 \wedge c \neq 0 \;\middle|\; \begin{array}{l} \mathbf{p} \leftarrow \mathcal{G}(1^\lambda) \\ \tau \xleftarrow{\$} \mathbb{F} \\ \mathrm{srs} \leftarrow \left(\mathbf{p}, [\tau, \ldots, \tau^n]_1, [\tau]_2\right) \\ (y, c) \leftarrow \mathcal{A}(\mathrm{srs}) \end{array} \right] \in \mathsf{negl}(\lambda)$$

In what follows we state two falsifiable assumptions that underpin the knowledge soundness of Plonk.

Definition 21 (n-ARSDH [LPS24]). *The n-*Adaptive Rational Strong Diffie-Hellman *(n-ARSDH) assumption holds for $\mathcal{G}$ in $\mathbb{G}_1$ if, for every PPT adversary $\mathcal{A}$:*

$$\Pr\left[\begin{array}{c} S \subset \mathbb{F}, \;\; |S| = n+1, \\ [g]_1 \neq [0]_1, \;\; [g]_1 \bullet [1]_2 = [h]_1 \bullet [Z_S(\tau)]_2 \end{array} \;\middle|\; \begin{array}{l} \mathbf{p} \leftarrow \mathcal{G}(1^\lambda) \\ \tau \xleftarrow{\$} \mathbb{F} \\ \mathrm{srs} \leftarrow \left(\mathbf{p}, [\tau, \ldots, \tau^n]_1, [\tau]_2\right) \\ (S, [g, h]_1) \leftarrow \mathcal{A}(\mathrm{srs}) \end{array} \right] \in \mathsf{negl}(\lambda)$$

Definition 22 (par-SplitRSDH [LPS25]). *Let $m, n_\psi, n_1, n_S \in \mathrm{poly}(\lambda)$ be positive integers with $n_S > n_1 + n_\psi + 1$, and let $\psi_k(X) \in \mathbb{F}_{\le n_\psi}[X]$ for $k \in [m]$. Set $\mathsf{par} = (m, n_\psi, n_1, n_S, (\psi_k)_{k=1}^m)$. The* par-SplitRSDH *assumption holds for $\mathcal{G}$ if, for every PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c|l} S \subset \mathbb{F},\ |S| = n_S,\ L(X) \in \mathbb{F}[X], & \mathbf{p} \leftarrow \mathcal{G}(1^\lambda) \\ \deg L(X) \in [n_1 + n_\psi + 1,\ n_S - 1], & \tau \xleftarrow{\$} \mathbb{F} \\ \sum_{k=1}^{m} [x_k\, \psi_k(\tau)]_1 \ =\ [L(\tau)]_1 + [\phi\, Z_S(\tau)]_1 & \mathrm{srs} \leftarrow \left(\mathbf{p},\ [\tau, \ldots, \tau^{n_1}]_1,\ [\tau]_2\right) \\ & \left(S, L(X), [(x_k)_{k=1}^m, \phi]_1\right) \leftarrow \mathcal{A}(\mathrm{srs}) \end{array}\right] \in \mathsf{negl}(\lambda)$$

In the context of Plonk, the relevant instantiation uses parameters $\mathsf{par}_n = (m, n_\psi, n_1, n_S, (\psi_k)_{k=1}^m)$ with $m = 3$, $n_\psi = 2n$, $n_1 = n + 5$, $n_S = 4n_1 + 1$, and $\psi_k(X) = X^{(k-1)n}$ [LPS25].

While n-ARSDH implies n-SDH [LPS24], no reduction between ARSDH and SplitRSDH is currently known [LPS25].

Generator Matrix. In the following we assume for simplicity that λ and d are both powers of 2. Let $N = 2\lambda d$, $\overline{N} = N\lambda$, $s = \varphi(\overline{N}) = \lambda^2 d$, and p prime such that $\overline{N} \mid p - 1$. By Theorem 1 there exist s distinct *primitive* $\overline{N}$-roots of unity in $\mathbb{F}_p$, and as a consequence $\overline{N}$ distinct $\overline{N}$-th roots of unity $\mathbf{r}_1, \ldots, \mathbf{r}_{\overline{N}}$. We set G to equal the generator matrix of a $[\overline{N}, d]$ Reed-Solomon code with code locators $\mathbf{r}_1, \ldots, \mathbf{r}_{\overline{N}}$. This ensures that G is MDS; furthermore, given $f \in \mathbb{F}_p[x]$ of degree less than d, $\mathsf{NTT}(f)$ returns $f(\mathbf{r}_1), \ldots, f(\mathbf{r}_{\overline{N}})$ in time $O(\overline{N} \log \overline{N}) = O_\lambda(d \log d)$. This immediately implies an encoding procedure with time $O_\lambda(d \log d)$. G crucially satisfies two further properties:

1. For any $1 \le m \le d$ the bottom m rows of G form a Vandermonde MDS submatrix; for $m = \lambda + 1$, this gives the invertible padding submatrices required by Theorem 6.
2. Each submatrix of λ columns satisfies the $\mathsf{Valid}_H(\cdot)$ condition. We elaborate further on this in a few paragraphs, when dwelling on the hiding of the whole construction.

Furthermore G admits a natural block division, with λ blocks, each pertaining to roots of unity $\mathbf{r}_{(i-1)\cdot N+1}, \ldots, \mathbf{r}_{iN}$.

Functional Commitment. Adopting Reed-Solomon codes allows us to employ polynomial commitments, which we can view as specialized inner product functional commitments where the committed vectors are of the form $\left(1\ a\ a^2\ \ldots\ a^{n-1}\right)^T$. Concretely, we jointly view $\mathbb{w}$ and r as a polynomial $f \in \mathbb{F}[x]$ of degree less than d, and commit to it via KZG. For efficient proof computation we leverage the approach of [FK23], which we briefly recap. Assume $\tau \leftarrow\!\$\ \mathbb{F}_p$ is secret, and elements

$$[1], [\tau], \ldots, [\tau^t] := g, g^\tau, \ldots, g^{\tau^t} \in \mathbb{G}$$

were published for some integer $t > d$ and a generator g of $\mathbb{G}$. Fix $i \in [\overline{N}]$. The i-th KZG quotient is by definition

$$q_i(Z) := \frac{f(Z) - f(\mathbf{r}_i)}{Z - \mathbf{r}_i}.$$

In the worst case the prover needs to compute $\overline{N}$ such quotients, evaluate them on τ, and raise g to the $q_i(\tau)$-th power, thus obtaining the terms $[q_1(\tau)], \ldots, [q_{\overline{N}}(\tau)]$. The crucial observation of [FK23] is that there exists a polynomial h of degree d such that $h(\mathbf{r}_i) = q_i(\tau)$ for all $i \in [\overline{N}]$. Through a clever combination of circulant matrix multiplication and DFTs it is possible to compute the coefficients of h from the first d powers of τ and the coefficients of f in time $O_\lambda(d \log d)$. While τ is kept secret, $[1], [\tau], \ldots, [\tau^{d-1}]$ is public, and the aforementioned computation is amenable to a computation in the exponent. That is, it is possible to compute $[h_1], \ldots, [h_d]$ in $O(d \log d)$ group operations. Finally, by running NTT in the exponent one can evaluate $\{[h(\mathbf{r}_i)]\}_{i \in [\overline{N}]}$ in $O(\overline{N} \log \overline{N}) = O_\lambda(d \log d)$ group operations.

This batch computation covers all $\overline{N} = \lambda N$ openings jointly. Repetition i uses only those with global indices $(i-1)N + 1, \ldots, iN$, so domain separation preserves the $O_\lambda(d \log d)$ proving time.

Ganesh et al. [GKO+23, Lem 4.2] showed that KZG has unique proofs, and is evaluation-binding under the Strong Diffie-Hellman assumption (see Definition 20). For polynomial commitments the binding notion of Definition 10 is exactly polynomial binding of [KZG10], and it is satisfied under the n-SDH assumption.

Hiding. KZG is λ-evaluation hiding for the polynomial encoding $\mathbb{w}\|r$. Put $\ell = \lambda$ and sample $r \leftarrow\!\$\ \mathbb{F}_p^{\ell+1}$, so the coefficient vector has length $d = n + \ell + 1$ and the polynomial has degree less than d. Let $\alpha_1, \ldots, \alpha_k$ be any $k \le \ell$ distinct locators selected from the $\overline{N}$-th roots of unity, and set $\alpha_0 = \tau$. Condition on τ being nonzero and outside that root set. This excludes an event of probability at most $(\overline{N}+1)/p$ when τ is uniform in $\mathbb{F}_p$. For every fixed good τ and every fixed choice of locators,

$$(f_w(\alpha_j))_{j=0}^{k} = \mathbb{w}A + rB, \qquad B_{a,j} = \alpha_j^{n+a} \quad (0 \le a \le \ell,\ 0 \le j \le k),$$

where A contains the first n powers of the locators. The first $k+1$ rows of B form a Vandermonde matrix multiplied by $\mathsf{diag}(\alpha_0^n, \ldots, \alpha_k^n)$. They are invertible because the locators are distinct and nonzero. Thus B has column rank $k+1$, and the displayed evaluation vector is uniform over $\mathbb{F}_p^{k+1}$, independently of $\mathbb{w}$. This holds even if the evaluation locators are chosen as a function of the public key, since we have fixed τ before applying the rank argument.

The commitment is $g^{f_w(\tau)}$. Each opening at α_j is

$$g^{q_j(\tau)} = g^{(f_w(\tau) - f_w(\alpha_j))/(\tau - \alpha_j)},$$

so commitments and opening proofs are deterministic postprocessing of τ and the evaluation vector. Their joint distribution is therefore identical for $\mathbb{w}\|r$ and $0\|r$ on the good-setup event. Unconditionally, the statistical distance is at most $(\overline{N}+1)/p$, which is negligible provided the field size is chosen accordingly. Finally, the last $\lambda + 1$ rows of G have full column rank on any at most $\lambda + 1$ distinct locators, by the same scaled Vandermonde argument.

For the hiding guarantee of Theorem 4, fix a good setup as above. Since $\tau \neq 0$, the coefficient of the first padding symbol in $f_w(\tau)$ is $\tau^n \neq 0$. Thus, for every fixed witness and key, fresh padding makes $f_w(\tau)$ uniform in $\mathbb{F}_p$ and the commitment uniform in $\mathbb{G}$, implying high commitment entropy holds with $\mu = 1/p$. Commitments and openings are deterministic and oracle-free. For each allowed locator $\alpha \neq \tau$, the KZG verification equation has exactly one accepting proof for fixed commitment and value, and as already mentioned, evaluation binding holds under SDH. Together with the exceptional setup probability $\eta \le (\overline{N}+1)/p$, these facts establish computational hiding under SDH.

Non-Rerandomizability. The same construction is also proof-non-rerandomizable by Theorem 6. Commitments and openings are deterministic and oracle-free, and Reed–Solomon gives both full row rank and the required padding submatrices. For $n \ge 1$ and cryptographic field size, p^{-n} is negligible. On the good-setup event above, τ is outside the opening domain, so the KZG verification equation uniquely determines an opening proof. Charging the remaining setup probability $(\overline{N}+1)/p$ to the error yields non-rerandomizability under SDH, with no additional hardness assumption.

Theorem 7 (KZG-based VRSC). *Under the SDH assumption (see Definition 20), there exists a verifiable reconstructible straightline code* VRSC *in the ROM, with* $O_\lambda(|\mathbb{w}|)$ Encode *time and* $O_\lambda(|\mathbb{w}| \log |\mathbb{w}|)$ CLeak *time.*

Can the full-domain computation be linear? Our $[\overline{N}, d]$ Reed–Solomon code is evaluated in $O(\overline{N} \log \overline{N}) = O_\lambda(d \log d)$ operations. A sparse generator matrix would permit multiplication in time proportional to its number of nonzero entries, but the MDS property rules out linear sparsity for these parameters. This restricts that particular approach. It is neither a lower bound for all MDS algorithms nor a necessary condition for every possible VRSC design.

Theorem 8. *Pick an arbitrary prime q. If $G \in \mathbb{F}_q^{k \times n}$ and contains w non-zero entries with $w \le k(n-k)$ then G is not MDS.*

Proof. By assumption G contains exactly $nk - w$ zero entries. By the generalized pigeonhole principle, there exists a row index $j \in [k]$ such that the j-th row contains at least $\lceil n - w/k \rceil \ge n - w/k$ entries. By our assumption on w this is greater or equal than k. Let us pick any k of these zero entries, and let $i_1, \ldots, i_k \in [n]$ denotes their distinct column indices. The submatrix consisting of the corresponding columns contains a row of zeros, and is therefore singular.

□

In our setting the row argument gives $w \geq d(\overline{N} - d + 1) = \Theta_\lambda(d^2)$.

4.3 Further Applications

Beyond the black-box UC compiler developed in the rest of this paper, VRSC's combination of straight-line reconstructability, hiding, and binding is a useful ingredient wherever a primitive needs non-interactive, publicly verifiable guarantees without rewinding an adversary a concurrent environment might also be observing. We illustrate this with a black-box construction of *publicly verifiable secret sharing* (PVSS): a dealer non-interactively commits to a secret s and distributes to each of n parties a public, individually verifiable record, from which any $t+1$ parties can later reconstruct s. Unlike the classical PVSS lineage [Sta96; Sch99; CD17; GJM+21; Gro21], which extracts only a one-way image h^s via rewinding, and unlike the fastest known VSS-style schemes [MDR23; PPP25], which are interactive, our construction is non-interactive, publicly verifiable, extracts the raw secret s itself straight-line, and gives constant-size shares.

Informally: *correctness* guarantees honestly dealt shares reconstruct s; *commitment consistency* guarantees that any two qualified sets of accepting shares reconstruct to the same value, even for a malicious dealer; *secrecy* guarantees that up to t corrupted parties learn nothing about s; and *straight-line Extractability* guarantees an extractor recovers every party's share directly from the dealer's oracle-query transcript, without rewinding it.

Theorem 9 (Black-box PVSS from VRSC). *Let* VRSC *be hiding, binding, and reconstructible, with oracle-free encoding and a deterministic transcript extractor. Let* Π *satisfy trapdoor zero knowledge and honest-setup offline knowledge soundness as specified in Definitions 3 and 4, for* $\mathcal{R}_{\mathsf{share}}$*. Let* PKE *be IND-CPA secure and perfectly correct. For* $0 \leq t < n < |\mathbb{F}|$ *over a prime field, the construction in Section B satisfies correctness, commitment consistency, secrecy, and straight-line extractability under static corruption. A per-party public record has size* $|c_i| + |\pi_{\mathsf{VRSC}}| + |\pi_\Pi|$*, in addition to the common commitment. It is* $O_\lambda(1)$ *when ciphertexts, VRSC certificates, and SNARK proofs all have that size.*

Informally, the dealer encodes a random degree-t polynomial ϕ with $\phi(0) = s$ as a single VRSC word $\mathsf{c} \leftarrow \mathsf{VRSC.Encode}(\mathsf{cpk}, \phi \| r)$, together with one certificate $\pi \leftarrow \mathsf{VRSC.CLeak}(\mathsf{cpk}, \phi \| r)$ shared across all n parties; each party i then only receives an encryption c_i of its evaluation $\phi(i)$ under its own public key, plus a proof π_i' that c_i and c commit to the same ϕ. This is captured by $\mathcal{R}_{\mathsf{share}}$ whose full definition is in Section B.1. Party i verifies (π, π_i') against c and c_i, and, once qualified, recovers $\phi(i)$ by simply decrypting c_i.

The proof reduces both commitment consistency and straight-line extractability to a single fact: a malicious dealer cannot make c admit accepting records for two different underlying polynomials. This follows by combining Π's knowledge soundness (which pins down a candidate witness per accepting index) with VRSC's reconstructability (one straight-line extraction, seeded from a single accepting index) and binding (a union bound ruling out disagreement for every accepting record). The public extractor uses only the VRSC transcript algorithm. Any private SNARK extraction used to prove its correctness occurs solely in an offline security reduction.

Note that the extractor is able to reconstruct the full secret from a *single* share. Secrecy follows a standard hybrid: VRSC's hiding masks (c, π), Π's zero-knowledge lets the honest parties' proofs be simulated, and PKE's IND-CPA security then hides each honest ciphertext.

We defer the full syntax, construction, and proofs to Section B.

5 A Blackbox UC Compiler for SNARKs

We now detail how to leverage any VRSC to lift an arbitrary zkSNARK to UC-secure zkSNARK.

Our goal is to construct a zkSNARK for an NP relation $\mathcal{R}$. We define a new relation

$$\mathcal{R}_{\mathsf{cpk}} = \{((\mathbb{x}, \mathsf{c}), (\mathbb{w}, r)) \mid (\mathbb{x}, \mathbb{w}) \in \mathcal{R} \wedge \mathsf{c} = \mathsf{VRSC.Encode}(\mathsf{cpk}, \mathbb{w} \| r)\}$$

and assume the existence of a zkSNARK $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for $\mathcal{R}_{\mathsf{cpk}}$. We then combine Π and VRSC into a single protocol $\Pi_{\mathsf{UC\text{-}SNARK}}$ according to the description of Figure 6.

Local Setup. We consider a UC functionality $\mathcal{F}_{\mathsf{Setup}}$ (Figure 5) that models the setup required by the original (simulation-extractable) zkSNARK that our compiler lifts to UC. At high level, $\mathcal{F}_{\mathsf{Setup}}$ takes care of generating a trusted CRS. Previous work [GKO+23; BCC+25] does not distinguish between the proving and verifying keys. As a result, their ideal setup functionalities do not capture the succinct setting considered here since their verifier is given the entire pp. Specifically, in the concrete KZG-based instantiation of [GKO+23], pp includes the linear-size pk, which is incompatible with a verifier running in time sublinear in $|\mathbb{w}|$. Moreover, although [GKO+23] proves security in the Global ROM, their local setup functionality is in fact compatible with a local ROM, and their UC simulator may need to intercept the prover's random-oracle queries directly in order to run the zero-knowledge simulator. We sidestep this odd mixture of random oracles by adopting the Global ROM outright, which forces our ZK simulator to rely on a trapdoor rather than RO programmability.

The $\mathcal{F}_{\mathsf{Setup}}$ functionality

Upon receiving $(\textsc{genprovingkey}, \mathsf{sid})$ from party $\mathcal{P}$:

- If no parameters pp has been stored for sid, run $(\mathsf{cpk}, \mathsf{cvk}) \leftarrow \mathsf{VRSC.Setup}(1^\lambda)$, and derive $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ for the relation $\mathcal{R}_{\mathsf{cpk}}$. Store $(\mathsf{pp}, \mathsf{sid})$, where $\mathsf{pp} := (\mathsf{pk}, \mathsf{vk}, \mathsf{cpk}, \mathsf{cvk})$.
- Send $(\textsc{provingkey}, \mathsf{sid}, \mathsf{pk}, \mathsf{cpk})$ to $\mathcal{P}$.

Upon receiving $(\textsc{genverifyingkey}, \mathsf{sid})$ from party $\mathcal{P}$:

- If no parameters pp has been stored for sid, run $(\mathsf{cpk}, \mathsf{cvk}) \leftarrow \mathsf{VRSC.Setup}(1^\lambda)$, and derive $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ for the relation $\mathcal{R}_{\mathsf{cpk}}$. Store $(\mathsf{pp}, \mathsf{sid})$, where $\mathsf{pp} := (\mathsf{pk}, \mathsf{vk}, \mathsf{cpk}, \mathsf{cvk})$.
- Send $(\textsc{verifyingkey}, \mathsf{sid}, \mathsf{vk}, \mathsf{cvk})$ to $\mathcal{P}$.

Fig. 5. The setup functionality, parametrized by the security parameter λ.

Theorem 10 (UC lifting compiler). *Let Π be a zero-knowledge argument according to Definition 3, and simulation-extractable as in Definition 4 with respect to a possibly different zero-knowledge simulator. Let VRSC be ℓ-hiding, reconstructible, and proof-non-rerandomizable. Then $\Pi_{\mathsf{UC\text{-}SNARK}}$ UC-realizes $\mathcal{F}_{\mathsf{NIZK}}$ in the $(\mathcal{G}_{\mathsf{RO}}, \mathcal{F}_{\mathsf{Setup}})$-hybrid model, with static corruption.*

Let M be the original circuit size, n the witness length, and $|\mathbb{x}|$ the public-input length. Suppose the encoding has a circuit of size $O_\lambda(T_{\mathsf{Encode}}(n))$, and let $C = M + O_\lambda(T_{\mathsf{Encode}}(n))$. The costs are

- $O_\lambda(T_{\mathsf{P}}(C) + T_{\mathsf{Encode}}(n) + T_{\mathsf{CLeak}}(n))$ *prover time,*
- $O_\lambda(T_{\mathsf{V}}(C, |\mathbb{x}| + |\mathsf{c}|) + T_{\mathsf{Vf}}(n))$ *verifier time,*
- $O_\lambda(|\mathsf{c}| + \pi_\Pi(C) + \pi_{\mathsf{VRSC}}(n))$ *proof size.*

Proof intuition. We move from the real protocol to the ideal one in three hybrids:

1. We first replace every honest SNARK proof π_2 with a simulated proof. Since all statements are still true, this follows directly from zero-knowledge of Π. The CRS is sampled with "fresh" independent randomness, so this does not affect VRSC.
2. We then replace each encoding c of the real witness with an encoding of a dummy value $0^n \| r$. The proof π_1 is generated *honestly* with respect to this dummy witness. This step follows from hiding of VRSC. Notice that some simulated proofs may now correspond to false statements, but any resulting change can be charged to the hiding advantage.
3. Finally, we replace verification by extraction: the verifier extracts a witness from c and checks $\mathcal{R}$. A failure can occur either because the same (c, π_2) is paired with a different π_1, violating non-rerandomizability of VRSC, or because a fresh π_2 yields a witness inconsistent with the one extracted from c, violating simulation-extractability of Π together with reconstructability of VRSC.

The $\Pi_{\mathsf{UC\text{-}SNARK}}$ protocol

Upon receiving $(\textsc{prove}, \mathsf{sid}, \mathbb{x}, \mathbb{w})$ ignore if $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$. Otherwise:

- Send $(\textsc{genprovingkey}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{Setup}}$ and wait for answer $(\textsc{provingkey}, \mathsf{sid}, \mathsf{pk}, \mathsf{cpk})$.
- Parse $\mathbb{w}$ as $\mathbb{w} \in \mathbb{F}^n$.
- Sample $r \leftarrow\!\$\ \mathbb{F}^\ell$ and compute $\mathsf{c} \leftarrow \mathsf{VRSC.Encode}(\mathsf{cpk}, \mathbb{w}\|r)$.
- Obtain $\pi_1 \leftarrow \mathsf{VRSC.CLeak}(\mathsf{cpk}, \mathbb{w}\|r)$. Whenever CLeak makes a call to H with input a, send $(\textsc{query}, \mathsf{sid}, a)$ to $\mathcal{G}_{\mathsf{RO}}$ and forward the response to CLeak.
- If $\pi_1 = \bot$, abort by outputting $(\textsc{proof}, \mathsf{sid}, \mathsf{runout_eval})$
- Obtain $\pi_2 \leftarrow \Pi.\mathsf{P}(\mathsf{pk}, \mathbb{x}, \mathsf{c}, \mathbb{w}, r)$. Whenever P makes a call to H with input a, send $(\textsc{query}, \mathsf{sid}, a)$ to $\mathcal{G}_{\mathsf{RO}}$ and forward the response to P.
- Output $(\textsc{proof}, \mathsf{sid}, \overline{\pi})$ with $\overline{\pi} = (\mathsf{c}, \pi_1, \pi_2)$.

Upon receiving $(\textsc{verify}, \mathsf{sid}, \mathbb{x}, \overline{\pi})$:

- Send $(\textsc{genverifyingkey}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{Setup}}$ and wait for answer $(\textsc{verifyingkey}, \mathsf{sid}, \mathsf{vk}, \mathsf{cvk})$.
- Parse $\overline{\pi}$ as $(\mathsf{c}, \pi_1, \pi_2)$.
- $\mathsf{VRSC.Vf}(\mathsf{cvk}, \mathsf{c}, \pi_1) = 1$ and $\Pi.\mathsf{V}(\mathsf{vk}, \mathbb{x}, \mathsf{c}, \pi_2) = 1$ output $(\textsc{verification}, \mathsf{sid}, 1)$. Otherwise, output $(\textsc{verification}, \mathsf{sid}, 0)$. Whenever V (resp. Vf) makes a call to H with input a, send $(\textsc{query}, \mathsf{sid}, a)$ to $\mathcal{G}_{\mathsf{RO}}$ and forward the response to V (resp. Vf).

Fig. 6. The $\Pi_{\mathsf{UC\text{-}SNARK}}$ protocol, parametrized by the security parameter λ, a finite field $\mathbb{F}$, a relation $\mathcal{R}$.

The resulting hybrid is exactly the ideal world.

Beyond the use of an abstract extractability primitive (our VRSC) rather than a polynomial commitment, our proof strategy crucially differs from [GKO+23] in the order of the hybrids. [GKO+23] directly simulates their proof π_1 component, requiring zero-knowledge of the Fischlin transform. We instead first simulate the SNARK proofs while their statements are honest, and only then switch the VRSC encoding.

This also means that we do not need full simulation-completeness for $\mathcal{R}_{\mathsf{cpk}}$, but standard zero-knowledge (Definition 3) already covers, due to the hiding of VRSC, the class of potentially false statements arising in the reduction.

Corollary 1 (KZG-based UC-SNARK). *Under ARSDH (Definition 21) and par-splitRSDH (Definition 22), there exists a UC-secure zkSNARK with proof size $O_\lambda(1)$, prover time $O_\lambda((M+n)\log(M+n))$, and verifier time $O_\lambda(|\mathbb{x}|+1)$, for relations with circuit size M and witness length n. Its additional verification work is $O_\lambda(1)$ with respect to witness size.*

Proof. The KZG-based VRSC in Theorem 7 has linear encoding cost, quasi-linear certificate-generation cost, and constant commitment and certificate size with respect to n. The encoding check adds $O_\lambda(n)$ gates. Substituting Plonk's costs at circuit size $C = M + O_\lambda(n)$ into Theorem 10 gives the claim. Plonk is both zero-knowledge with respect to a trapdoor simulator [Sef24], and simulation extractable (with respect to another trapdoorless zero-knowledge simulator) [Lip25].

Finally, the simulation extractability of Plonk follows under the stated assumptions from [Lip25]. Since ARSDH implies the SDH assumption used by the KZG VRSC, Theorem 10 applies.

□

Proof (of Theorem 10). It is immediate to see that the completeness of the protocol $\Pi_{\mathsf{UC\text{-}SNARK}}$ follows directly from the completeness of Π and VRSC. Let ϵ_1 (resp. ϵ_2) be the completeness error of Π (resp. VRSC). For a valid pair $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, the probability that a proof $\overline{\pi}$ produced by an honest prover party does not verify is at most $\epsilon_1 + \epsilon_2$, which is negligible in λ by assumption.

Succinctness. We consider three important metrics: proof size, prover and verifier time. The circuit associated with the relation $\mathcal{R}_{\mathsf{cpk}}$ is obtained by composing the one for the original relation $\mathcal{R}$, and the one that describes the encoding procedure of VRSC, which requires $O_\lambda(T_{\mathsf{Encode}}(n))$ additional gates. Let M be the size of the original circuit, and let $C = M + O_\lambda(T_{\mathsf{Encode}}(n))$ be the size of the modified circuit. Notice that the prover has to additionally encode the witness $\mathbb{w}$ into c and produce a proof for the code VRSC. We conclude that the total prover time is $O_\lambda(T_{\mathsf{P}}(C) + T_{\mathsf{Encode}}(n) + T_{\mathsf{CLeak}}(n))$.

Similarly, we derive the the verifier time as $O_\lambda(T_{\mathsf{V}}(C, |\mathbb{x}| + |\mathsf{c}|) + T_{\mathsf{Vf}}(n))$, accounting for the time to verify the enhanced relation $\mathcal{R}_{\mathsf{cpk}}$ (reading the additional public inputs $(\mathbb{x}, \mathsf{c})$), and verify the proof π_1 for VRSC.

We can easily derive $O_\lambda(\pi_\Pi(C) + \pi_{\mathsf{VRSC}}(n) + |\mathsf{c}|)$ as the total proof size, due to the three elements c, π_1, π_2 in $\overline{\pi}$.

The ideal adversary Sim. As standard in the UC-setting, we recall that it suffices to find a simulator Sim for the "dummy adversary", i.e., and adversary $\mathcal{A}$ that simply acts as a proxy between the environment $\mathcal{Z}$ and parties.

The ideal adversary Sim, parametrized by witness size n, does the following.

- Simulates $\mathcal{F}_{\mathsf{Setup}}$ by running $\mathsf{SimSetup}(1^\lambda)$ rather than $\Pi.\mathsf{Setup}(1^\lambda)$ for the relation $\mathcal{R}_{\mathsf{cpk}}$, and let td be the simulation trapdoor for Π. The keys $(\mathsf{cpk}, \mathsf{cvk})$ are generated honestly.
- Upon receiving $(\textsc{prove}, \mathsf{sid}, \mathbb{x})$ from $\mathcal{F}_{\mathsf{NIZK}}$, sample randomness $r \leftarrow\!\$\ \mathbb{F}^\ell$ and compute $\mathsf{c} \leftarrow \mathsf{VRSC.Encode}(\mathsf{cpk}, 0^n \| r)$. Obtain a proof $\pi_1 \leftarrow \mathsf{VRSC.CLeak}(\mathsf{cpk}, 0^n \| r)$ and abort if $\pi_1 = \bot$. Run the zero-knowledge simulator for $\mathcal{R}_{\mathsf{cpk}}$ and obtain $\pi_2 \leftarrow \Pi.\mathcal{S}(\mathsf{td}, (\mathbb{x}, \mathsf{c}))$. Finally, send $(\textsc{proof}, \mathsf{sid}, (\mathsf{c}, \pi_1, \pi_2))$ to $\mathcal{F}_{\mathsf{NIZK}}$.
- Upon receiving $(\textsc{verify}, \mathsf{sid}, \mathbb{x}, \overline{\pi})$ from $\mathcal{F}_{\mathsf{NIZK}}$, retrieve the parameters associated with sid, and send $(\textsc{verification}, \mathsf{sid}, 0)$ if either $\mathsf{VRSC.Vf}(\mathsf{cvk}, \mathsf{c}, \pi_1) = 0$ or $\Pi.\mathsf{V}(\mathsf{vk}, \mathbb{x}, \pi_2) = 0$, where $\overline{\pi} := (\mathsf{c}, \pi_1, \pi_2)$. Otherwise, query $\mathcal{G}_{\mathsf{RO}}$ on $(\textsc{observe}, \mathsf{sid})$ and receive the set $Q := \{(\textsc{query}, \mathsf{sid}, s_i)_i\}$ of illegitimate queries associated with sid, and run $\mathbb{w} \leftarrow \mathsf{VRSC.Ext}(\mathsf{c}, \pi, \{s_i\}_i)$, and parse $\mathbb{w}$ as $(\mathbb{w}', r) \in \mathbb{F}^n \times \mathbb{F}^\ell$. Send $(\textsc{witness}, \mathbb{w}')$ back to $\mathcal{F}_{\mathsf{NIZK}}$.

UC-security. Let $\mathrm{Exec}(\mathcal{F}_{\mathsf{NIZK}}, \mathsf{Sim}, \mathcal{Z})$ be the random variable that denotes the output of the experiment where the PPT environment $\mathcal{Z}$ invokes (potentially) many sessions of $\mathcal{F}_{\mathsf{NIZK}}$ and interacts with the simulator Sim. Let $\mathrm{Exec}(\mathit{\Pi}_{\mathsf{UC\text{-}SNARK}}, \mathcal{A}, \mathcal{Z})$ be the random variable that denotes the output of the experiment where the executions of $\mathcal{F}_{\mathsf{NIZK}}$ are replaced with invocations of $\mathit{\Pi}_{\mathsf{UC\text{-}SNARK}}$ in which the dummy adversary $\mathcal{A}$ is playing. We will proceed to show that:

$$\mathrm{Exec}(\mathcal{F}_{\mathsf{NIZK}}, \mathsf{Sim}, \mathcal{Z}) \approx \mathrm{Exec}(\mathit{\Pi}_{\mathsf{UC\text{-}SNARK}}, \mathcal{A}, \mathcal{Z})$$

If both parties are corrupted then the functionalities can be treated as part of the environment $\mathcal{Z}$. Therefore, we only consider the case where $\mathcal{Z}$ participates in sessions *either* with a corrupted prover (prover session) or a corrupted verifier (verifier session). Moreover, let q_{P} and q_{V} be an upper-bound to the number of prover and verifier queries, respectively, to the $\mathcal{F}_{\mathsf{NIZK}}$ functionality.

Game_0. This game corresponds to the real world execution, namely $\mathsf{Game}_0 \equiv \mathrm{Exec}(\mathit{\Pi}_{\mathsf{UC\text{-}SNARK}}, \mathcal{A}, \mathcal{Z})$.

Game_1. This experiment is the same as Game_0 except that the proof π_2 for a statement $(\mathbb{x}, \mathsf{c}) \in \mathcal{R}_{\mathsf{cpk}}$ computed by honest prover parties is replaced with a simulated proof using $\Pi.\mathcal{S}(\mathbb{x}, \mathsf{c})$.

Lemma 1. $\mathsf{Game}_1 \approx \mathsf{Game}_0$.

Proof. Let H_0 be equivalent to Game_0, and for all $i \in [q_{\mathsf{P}}]$ let H_i be equivalent to H_{i-1}, except that in H_i the i-th proof generated by the honest parties is replaced by simulating the proof π_2 using the zero-knowledge simulator for Π. By definition, it follows that $H_{q_{\mathsf{P}}} \equiv \mathsf{Game}_1$, thus it suffices to show that for all $i \in [q_{\mathsf{P}}]$, we have that $H_{i-1} \approx H_i$. We show that this holds by reducing to the zero-knowledge property of Π.

Formally, the reduction $\mathcal{B}$ would take care of simulating $\mathcal{G}_{\mathsf{RO}}$ in the eyes of $\mathcal{Z}$, and would handle calls to $\mathcal{F}_{\mathsf{Setup}}$ and $\mathcal{F}_{\mathsf{NIZK}}$ consistently. The only difference between the two hybrids lies in the fact that the i-th proof

computed by an honest party is either honest (H_{i-1}) or simulated (H_i), for a statement $(\mathbb{x}, \mathsf{c})$ for which the reduction $\mathcal{B}$ knows the corresponding witness $\mathbb{w}$, as it is provided by the honest prover party to $\mathcal{F}_{\mathsf{NIZK}}$. Thus, when the i-th proof computed by an honest party has to be computed, $\mathcal{B}$ would simply invoke the challenger of the zero-knowledge game. All the first $i-1$ proofs of the honest prover parties are computed by "locally" running the code of the simulator $\mathcal{S}$ for Π (that by assumption is possible by using the trapdoor td generated by $\mathsf{SimSetup}$), while the remaining $q_{\mathsf{P}} - i$ proofs are computed honestly running the code of the prover algorithm.

By a simple union bound, we derive that probability to distinguish H_0 from H_1 is at most $q_{\mathsf{P}} \cdot \epsilon_{\mathrm{zk}}$, where ϵ_{zk} is the zero-knowledge error of Π.

□

Game_2. This experiment is the same as Game_1 except that the proof π_1 and the encoding c for a statement $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ computed by honest prover parties are replaced with a proof $\tilde{\pi}_1 \leftarrow \mathsf{VRSC.CLeak}(\mathsf{cpk}, 0\|r)$ and an encoding $\tilde{\mathsf{c}} \leftarrow \mathsf{VRSC.Encode}(\mathsf{cpk}, 0\|r)$, for a uniformly random $r \leftarrow\!\!\$\ \mathbb{F}^{\ell}$.

Lemma 2. $\mathsf{Game}_2 \approx \mathsf{Game}_1$.

Proof. We can proceed with a hybrid argument over the number of honest prover queries to $\mathcal{F}_{\mathsf{NIZK}}$. Let H_0 be equivalent to Game_2, and for all $i \in [q_{\mathsf{P}}]$ let H_i be equivalent to H_{i-1} except on how the i-th proof generated by some honest party is computed, specifically the encoding $\tilde{\mathsf{c}}$ (and the corresponding proof $\tilde{\pi}_1$) is used in H_i rather than the honest one. The advantage of any PPT distinguisher between two consecutive hybrids can be upper-bounded by the probability of breaking the ℓ-hiding property of VRSC. By a simple union bound, we derive that no PPT algorithm can distinguish Game_2 from Game_1 with probability more than $q_{\mathsf{P}} \cdot \epsilon_{\mathrm{hid}}$, where ϵ_{hid} is the hiding error of VRSC.

□

Game_3. This experiment is the same as Game_2 except that the proofs of the form $\overline{\pi} := (\mathsf{c}, \pi_1, \pi_2)$ for a statement $\mathbb{x}$ (and some sid) verified by honest verifier parties are verified using the following procedure. The hybrid retrieves the list of queries to the random oracle associated with the same sid, it derives a transcript τ, as described in the code of Sim, computes $\mathbb{w} \leftarrow \mathsf{VRSC.Ext}(\mathsf{c}, \pi_2, \tau)$, parses it as $(\mathbb{w}', r) \in \mathbb{F}^n \times \mathbb{F}^{\ell}$, and returns 1 if and only if $(\mathbb{w}', r) \in \mathcal{R}$.

Lemma 3. $\mathsf{Game}_3 \approx \mathsf{Game}_2$.

Proof. We can proceed with a hybrid argument over the number of honest verifier queries to $\mathcal{F}_{\mathsf{NIZK}}$. Let H_0 be equivalent to Game_3, and for all $i \in [q_{\mathsf{V}}]$ let H_i be equivalent to H_{i-1} except for how the i-th proof queried by some honest party is actually verified, specifically the additional extractor procedure described above.

There are two cases:

- **($(\mathbb{x}, \mathsf{c}, \pi_2)$ is part of a simulated proof)** If (π_2, c) was part of a proof $\overline{\pi}' := (\mathsf{c}', \pi_1', \pi_2')$ produced by an honest party (in a different session sid) for the same statement $\mathbb{x}$, it must be that $\pi_1 \neq \pi_1'$. In this subcase, the π_1 component must be different as otherwise $\mathcal{F}_{\mathsf{NIZK}}$ would not notify Sim to verify the proof. Moreover, due to the change in Game_1, π_2 has been computed using the zero-knowledge simulator of Π, while c has been computed as an honest encoding of the $0^n\|r'$ word, for some $r' \in \mathbb{F}^{\ell}$, due to the change introduced in Game_2. We can reduce to the proof-non-rerandomizability of VRSC. The reduction $\mathcal{B}$ intercepts the random oracle queries and forwards them to H, and simulates the the calls to $\mathcal{F}_{\mathsf{NIZK}}$ in the eyes of $\mathcal{Z}$, and then outputs as forgery π_1, as a valid re-randomization of $\overline{\pi}_1$.
- **($(\mathbb{x}, \mathsf{c}, \pi_2)$ is <u>not</u> part of a simulated proof)** Since Π is simulation extractable with respect to a zero-knowledge simulator $\mathcal{S}'$, we can reduce to reconstructability of VRSC as follows. The reduction $\mathcal{B}$ locally runs the code of the zero-knowledge simulator $\mathcal{S}'$, and forwards all the RO queries to H. It then can run the extractor $\Pi.\mathcal{E}$ to extract a witness $\overline{\mathbb{w}} = (\overline{\mathbb{w}}', \overline{r})$ that with overwhelming probability is a valid word encoded into c. Notice that this (possibly rewinding-based) extractor is fed with a fresh statement-proof

pair, as $\mathbb{x}, \mathsf{c}, \pi_2$ was not part of a simulated proof for $\Pi_{\mathsf{UC\text{-}SNARK}}$, namely π_2 was not generated using the zero-knowledge simulator $\Pi.\mathcal{S}'$ on input $(\mathbb{x}, \mathsf{c})$. If $\mathbb{w}' \neq \overline{\mathbb{w}}'$, however, it follows that $\overline{\mathbb{w}} \neq \mathbb{w}$, thus $\mathcal{B}$ can successfully forge in the reconstructability game and output $(\overline{\mathbb{w}}, \mathsf{c}, \pi_1)$. Finally, we observe that the reduction may need to use a possibly different zero-knowledge simulator $\mathcal{S}'$ than the simulator $\Pi.\mathcal{S}$ used in Game_3; however, since both of them are valid zero-knowledge simulators, this change modifies the view of a PPT distinguisher by at most a negligible amount.

Remark 1. Notice that in the above reductions $\mathcal{B}$ can run the zero-knowledge simulator of Π because the CRS of $\mathcal{R}_{\mathsf{cpk}}$ is sampled freshly, with independent randomness. A subtle but crucial aspect of our reduction is that $\mathcal{B}$ must be able to invoke the zero-knowledge simulator of Π, which may require the ability to simulate a trapdoor CRS for $\mathcal{R}_{\mathsf{cpk}}$. This is possible precisely because the CRS of $\mathcal{R}\mathsf{cpk}$ is sampled freshly with independent randomness, leaving the existing commitment key of VRSC untouched. In principle, Π being a zkSNARK for a commit-and-prove relation, could reuse the commitment key of VRSC (e.g., when instantiated with a KZG key and a KZG-based SNARK); were that the case, simulating the CRS would interfere with the commitment key, and the reduction would break down. The ideal functionality of [GKO+23] does not make this independence explicit,[2] and we rely on it critically here.

By union bound, it follows that the probability to distinguish Game_3 from Game_2 is at most $q_{\mathsf{V}} \cdot (\epsilon_{\mathsf{Ext}} + \epsilon_{\text{sim-ext}})$, where ϵ_{Ext} is the reconstructability error of VRSC and $\epsilon_{\text{sim-ext}}$ is the simulation-extractability error of Π.

□

Finally, it is easy to see that Game_3 is equivalent to $\mathrm{Exec}(\mathcal{F}_{\mathsf{NIZK}}, \mathsf{Sim}, \mathcal{Z})$, because of all the changes in previous hybrids.

□

Acknowledgments

We thank Mahak Pancholi and Matteo Campanelli for general discussions about this work.

References

[AB19] Shahla Atapoor and Karim Baghery. *Simulation Extractability in Groth's zk-SNARK*. Cryptology ePrint Archive, Report 2019/641. 2019. URL: https://eprint.iacr.org/2019/641.

[ARS20] Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. "Lift-and-Shift: Obtaining Simulation Extractable Subversion and Updatable SNARKs Generically". In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1987–2005. DOI: 10.1145/3372297.3417228.

[Bag19] Karim Baghery. "Subversion-Resistant Simulation (Knowledge) Sound NIZKs". In: *17th IMA International Conference on Cryptography and Coding*. Ed. by Martin Albrecht. Vol. 11929. LNCS. Springer, Cham, Dec. 2019, pp. 42–63. DOI: 10.1007/978-3-030-35199-1_3.

[Bag25] Karim Baghery. "Π: A Unified Framework for Computational Verifiable Secret Sharing". In: *PKC 2025, Part IV*. Ed. by Tibor Jager and Jiaxin Pan. Vol. 15677. LNCS. Springer, Cham, May 2025, pp. 110–142. DOI: 10.1007/978-3-031-91829-2_4.

[BB04] Dan Boneh and Xavier Boyen. "Short Signatures Without Random Oracles". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Berlin, Heidelberg, May 2004, pp. 56–73. DOI: 10.1007/978-3-540-24676-3_4.

[2] We thank the authors of [GKO+23] for confirming this point.

[BCC+25] Christian Badertscher, Matteo Campanelli, Michele Ciampi, Luigi Russo, and Luisa Siniscalchi. "Universally Composable SNARKs with Transparent Setup without Programmable Random Oracle". In: *CRYPTO 2025, Part VII*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16006. LNCS. Springer, Cham, Aug. 2025, pp. 225–258. DOI: 10.1007/978-3-032-01907-3_8.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. "Zerocash: Decentralized Anonymous Payments from Bitcoin". In: *2014 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2014, pp. 459–474. DOI: 10.1109/SP.2014.36.

[BCJ+24] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Yan Ji, Jonas Lindstrøm, Deepak Maram, Ben Riva, Arnab Roy, Mahdi Sedaghat, and Joy Wang. "zkLogin: Privacy-Preserving Blockchain Authentication with Existing Credentials". In: *ACM CCS 2024*. Ed. by Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie. ACM Press, Oct. 2024, pp. 3182–3196. DOI: 10.1145/3658644.3690356.

[BCL26] Renas Bacho, Yanbo Chen, and Julian Loss. *Adaptively Secure (Aggregatable) PVSS from Standard Assumptions*. Cryptology ePrint Archive, Paper 2026/1100. 2026. URL: https://eprint.iacr.org/2026/1100.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *TCC 2016-B, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. LNCS. Springer, Berlin, Heidelberg, 2016, pp. 31–60. DOI: 10.1007/978-3-662-53644-5_2.

[Bea92] Donald Beaver. "Efficient Multiparty Protocols Using Circuit Randomization". In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Berlin, Heidelberg, Aug. 1992, pp. 420–432. DOI: 10.1007/3-540-46766-1_34.

[BFK+24] Jan Bobolz, Pooya Farshim, Markulf Kohlweiss, and Akira Takahashi. "The Brave New World of Global Generic Groups and UC-Secure Zero-Overhead SNARKs". In: *TCC 2024, Part I*. Ed. by Elette Boyle and Mohammad Mahmoody. Vol. 15364. LNCS. Springer, Cham, Dec. 2024, pp. 90–124. DOI: 10.1007/978-3-031-78011-0_4.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. "Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)". In: *20th ACM STOC*. ACM Press, May 1988, pp. 103–112. DOI: 10.1145/62212.62222.

[BL23] Renas Bacho and Julian Loss. "Adaptively Secure (Aggregatable) PVSS and Application to Distributed Randomness Beacons". In: *ACM CCS 2023*. Ed. by Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda. ACM Press, Nov. 2023, pp. 1791–1804. DOI: 10.1145/3576915.3623106.

[Can01] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *42nd FOCS*. IEEE Computer Society Press, Oct. 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888.

[CCC+25] Matteo Campanelli, Mario Carrillo, Ignacio Cascudo, Dario Fiore, Danilo Francati, and Rosario Gennaro. *On the Power of Polynomial Preprocessing: Proving Computations in Sublinear Time, and More*. Cryptology ePrint Archive, Paper 2025/238. 2025. URL: https://eprint.iacr.org/2025/238.

[CD17] Ignacio Cascudo and Bernardo David. "SCRAPE: Scalable Randomness Attested by Public Entities". In: *ACNS 2017*. Ed. by Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi. Vol. 10355. LNCS. Springer, Cham, July 2017, pp. 537–556. DOI: 10.1007/978-3-319-61204-1_27.

[CD20] Ignacio Cascudo and Bernardo David. "ALBATROSS: Publicly AttestabLe BATched Randomness Based On Secret Sharing". In: *ASIACRYPT 2020, Part III*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12493. LNCS. Springer, Cham, Dec. 2020, pp. 311–341. DOI: 10.1007/978-3-030-64840-4_11.

[CDG+18] Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. "The Wonderful World of Global Random Oracles". In: *EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. LNCS. Springer, Cham, 2018, pp. 280–312. DOI: 10.1007/978-3-319-78381-9_11.

[CDP+07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. "Universally Composable Security with Global Setup". In: *TCC 2007*. Ed. by Salil P. Vadhan. Vol. 4392. LNCS. Springer, Berlin, Heidelberg, Feb. 2007, pp. 61–85. DOI: 10.1007/978-3-540-70936-7_4.

[CF24] Alessandro Chiesa and Giacomo Fenzi. "zkSNARKs in the ROM with Unconditional UC-Security". In: *TCC 2024, Part I*. Ed. by Elette Boyle and Mohammad Mahmoody. Vol. 15364. LNCS. Springer, Cham, Dec. 2024, pp. 67–89. DOI: 10.1007/978-3-031-78011-0_3.

[CFK+22] Hien Chu, Dario Fiore, Dimitris Kolonelos, and Dominique Schröder. "Inner Product Functional Commitments with Constant-Size Public Parameters and Openings". In: *SCN 22*. Ed. by Clemente Galdi and Stanislaw Jarecki. Vol. 13409. LNCS. Springer, Cham, Sept. 2022, pp. 639–662. DOI: 10.1007/978-3-031-14791-3_28.

[CFR25] Matteo Campanelli, Antonio Faonio, and Luigi Russo. "SNARKs for Virtual Machines Are Non-malleable". In: *EUROCRYPT 2025, Part IV*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15604. LNCS. Springer, Cham, May 2025, pp. 153–183. DOI: 10.1007/978-3-031-91134-7_6.

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. "Practical UC security with a Global Random Oracle". In: *ACM CCS 2014*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM Press, Nov. 2014, pp. 597–608. DOI: 10.1145/2660267.2660374.

[CP23] Leo de Castro and Chris Peikert. "Functional Commitments for All Functions, with Transparent Setup and from SIS". In: *EUROCRYPT 2023, Part III*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14006. LNCS. Springer, Cham, Apr. 2023, pp. 287–320. DOI: 10.1007/978-3-031-30620-4_10.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. "Robust Non-interactive Zero Knowledge". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Berlin, Heidelberg, Aug. 2001, pp. 566–598. DOI: 10.1007/3-540-44647-8_33.

[DG23] Quang Dao and Paul Grubbs. "Spartan and Bulletproofs are Simulation-Extractable (for Free!)" In: *EUROCRYPT 2023, Part II*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14005. LNCS. Springer, Cham, Apr. 2023, pp. 531–562. DOI: 10.1007/978-3-031-30617-4_18.

[FFK+23] Antonio Faonio, Dario Fiore, Markulf Kohlweiss, Luigi Russo, and Michal Zajac. "From Polynomial IOP and Commitments to Non-malleable zkSNARKs". In: *TCC 2023, Part III*. Ed. by Guy N. Rothblum and Hoeteck Wee. Vol. 14371. LNCS. Springer, Cham, 2023, pp. 455–485. DOI: 10.1007/978-3-031-48621-0_16.

[FFR24] Antonio Faonio, Dario Fiore, and Luigi Russo. "Real-World Universal zkSNARKs are Non-Malleable". In: *ACM CCS 2024*. Ed. by Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie. ACM Press, Oct. 2024, pp. 3138–3151. DOI: 10.1145/3658644.3690351.

[Fis05] Marc Fischlin. "Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors". In: *CRYPTO 2005*. Ed. by Victor Shoup. Vol. 3621. LNCS. Springer, Berlin, Heidelberg, Aug. 2005, pp. 152–168. DOI: 10.1007/11535218_10.

[FK23] Dankrad Feist and Dmitry Khovratovich. *Fast amortized KZG proofs*. Cryptology ePrint Archive, Report 2023/033. 2023. URL: https://eprint.iacr.org/2023/033.

[FKM+12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. "On the Non-malleability of the Fiat-Shamir Transform". In: *INDOCRYPT 2012*. Ed. by Steven D. Galbraith and Mridul Nandi. Vol. 7668. LNCS. Springer, Berlin, Heidelberg, Dec. 2012, pp. 60–79. DOI: 10.1007/978-3-642-34931-7_5.

[FR26] Antonio Faonio and Luigi Russo. *Sumcheck-based zkSNARKs are Non-Malleable*. Cryptology ePrint Archive, Paper 2026/335. 2026. URL: https://eprint.iacr.org/2026/335.

[GJM+21] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. "Aggregatable Distributed Key Generation". In: *EUROCRYPT 2021, Part I*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12696. LNCS. Springer, Cham, Oct. 2021, pp. 147–176. DOI: 10.1007/978-3-030-77870-5_6.

[GKK+22] Chaya Ganesh, Hamidreza Khoshakhlagh, Markulf Kohlweiss, Anca Nitulescu, and Michal Zajac. "What Makes Fiat-Shamir zkSNARKs (Updatable SRS) Simulation Extractable?" In: *SCN*

22. Ed. by Clemente Galdi and Stanislaw Jarecki. Vol. 13409. LNCS. Springer, Cham, Sept. 2022, pp. 735–760. DOI: 10.1007/978-3-031-14791-3_32.

[GKO+22] Chaya Ganesh, Yashvanth Kondi, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. *Witness-Succinct Universally-Composable SNARKs*. Cryptology ePrint Archive, Report 2022/1618. 2022. URL: https://eprint.iacr.org/2022/1618.

[GKO+23] Chaya Ganesh, Yashvanth Kondi, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. "Witness-Succinct Universally-Composable SNARKs". In: *EUROCRYPT 2023, Part II*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14005. LNCS. Springer, Cham, Apr. 2023, pp. 315–346. DOI: 10.1007/978-3-031-30617-4_11.

[GM17] Jens Groth and Mary Maller. "Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs". In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Cham, Aug. 2017, pp. 581–612. DOI: 10.1007/978-3-319-63715-0_20.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)". In: *17th ACM STOC*. ACM Press, May 1985, pp. 291–304. DOI: 10.1145/22145.22178.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Perfect Non-interactive Zero Knowledge for NP". In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Berlin, Heidelberg, 2006, pp. 339–358. DOI: 10.1007/11761679_21.

[Gro06] Jens Groth. "Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures". In: *ASIACRYPT 2006*. Ed. by Xuejia Lai and Kefei Chen. Vol. 4284. LNCS. Springer, Berlin, Heidelberg, Dec. 2006, pp. 444–459. DOI: 10.1007/11935230_29.

[Gro16] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments". In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Berlin, Heidelberg, May 2016, pp. 305–326. DOI: 10.1007/978-3-662-49896-5_11.

[Gro21] Jens Groth. *Non-interactive distributed key generation and key resharing*. Cryptology ePrint Archive, Report 2021/339. 2021. URL: https://eprint.iacr.org/2021/339.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Report 2019/953. 2019. URL: https://eprint.iacr.org/2019/953.

[HV09] Somayeh Heidarvand and Jorge L. Villar. "Public Verifiability from Pairings in Secret Sharing Schemes". In: *SAC 2008*. Ed. by Roberto Maria Avanzi, Liam Keliher, and Francesco Sica. Vol. 5381. LNCS. Springer, Berlin, Heidelberg, Aug. 2009, pp. 294–308. DOI: 10.1007/978-3-642-04159-4_19.

[JP14] Abhishek Jain and Omkant Pandey. "Non-Malleable Zero Knowledge: Black-Box Constructions and Definitional Relationships". In: *SCN 14*. Ed. by Michel Abdalla and Roberto De Prisco. Vol. 8642. LNCS. Springer, Cham, Sept. 2014, pp. 435–454. DOI: 10.1007/978-3-319-10879-7_25.

[Kat24] Jonathan Katz. "Round-Optimal, Fully Secure Distributed Key Generation". In: *CRYPTO 2024, Part VII*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14926. LNCS. Springer, Cham, Aug. 2024, pp. 285–316. DOI: 10.1007/978-3-031-68394-7_10.

[KPT23] Markulf Kohlweiss, Mahak Pancholi, and Akira Takahashi. "How to Compile Polynomial IOP into Simulation-Extractable SNARKs: A Modular Approach". In: *TCC 2023, Part III*. Ed. by Guy N. Rothblum and Hoeteck Wee. Vol. 14371. LNCS. Springer, Cham, 2023, pp. 486–512. DOI: 10.1007/978-3-031-48621-0_17.

[Ks22] Yashvanth Kondi and abhi shelat. "Improved Straight-Line Extraction in the Random Oracle Model with Applications to Signature Aggregation". In: *ASIACRYPT 2022, Part II*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13792. LNCS. Springer, Cham, Dec. 2022, pp. 279–309. DOI: 10.1007/978-3-031-22966-4_10.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. "Constant-Size Commitments to Polynomials and Their Applications". In: *ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477.

LNCS. Springer, Berlin, Heidelberg, Dec. 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11.

[KZM+15] Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. *C∅C∅: A Framework for Building Composable Zero-Knowledge Proofs*. Cryptology ePrint Archive, Report 2015/1093. 2015. URL: https://eprint.iacr.org/2015/1093.

[Lip25] Helger Lipmaa. "Plonk is Simulation Extractable in ROM Under Falsifiable Assumptions". In: *TCC 2025, Part IV*. Ed. by Benny Applebaum and Huijia (Rachel) Lin. Vol. 16271. LNCS. Springer, Cham, Dec. 2025, pp. 3–36. DOI: 10.1007/978-3-032-12290-2_1.

[LPS24] Helger Lipmaa, Roberto Parisella, and Janno Siim. "Constant-Size zk-SNARKs in ROM from Falsifiable Assumptions". In: *EUROCRYPT 2024, Part VI*. Ed. by Marc Joye and Gregor Leander. Vol. 14656. LNCS. Springer, Cham, May 2024, pp. 34–64. DOI: 10.1007/978-3-031-58751-1_2.

[LPS25] Helger Lipmaa, Roberto Parisella, and Janno Siim. "On Knowledge-Soundness of Plonk in ROM from Falsifiable Assumptions". In: *CRYPTO 2025, Part VII*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16006. LNCS. Springer, Cham, Aug. 2025, pp. 362–395. DOI: 10.1007/978-3-032-01907-3_12.

[LRY16] Benoît Libert, Somindu C. Ramanna, and Moti Yung. "Functional Commitment Schemes: From Polynomial Commitments to Pairing-Based Accumulators from Simple Assumptions". In: *ICALP 2016*. Ed. by Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi. Vol. 55. LIPIcs. Schloss Dagstuhl, July 2016, 30:1–30:14. DOI: 10.4230/LIPIcs.ICALP.2016.30.

[MDR23] Atsuki Momose, Sourav Das, and Ling Ren. "On the Security of KZG Commitment for VSS". In: *ACM CCS 2023*. Ed. by Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda. ACM Press, Nov. 2023, pp. 2561–2575. DOI: 10.1145/3576915.3623127.

[Mic00] Silvio Micali. "Computationally Sound Proofs". In: *SIAM Journal on Computing* 30.4 (2000), pp. 1253–1298. DOI: 10.1137/S0097539795284959. URL: https://doi.org/10.1137/S0097539795284959.

[PPP25] Christodoulos Pappas, Dimitrios Papadopoulos, and Charalampos Papamanthou. "HydraProofs: Optimally Computing All Proofs in a Vector Commitment (With Applications to Efficient zk-SNARKs Over Data from Multiple Users)". In: *2025 IEEE Symposium on Security and Privacy*. Ed. by Marina Blanton, William Enck, and Cristina Nita-Rotaru. IEEE Computer Society Press, May 2025, pp. 3421–3439. DOI: 10.1109/SP61157.2025.00204.

[PR05] Rafael Pass and Alon Rosen. "New and improved constructions of non-malleable cryptographic protocols". In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 533–542. DOI: 10.1145/1060590.1060670.

[RZM+26] Nahid Roustaeifar, Maryam Zarezadeh, Neel Mehendale, Behzad Abdolmaleki, Mahshid Delavar, and Stefan Köpsell. *Witness-Succinct UC-Secure SNARKs with Weaker Trust Assumptions*. Cryptology ePrint Archive, Paper 2026/2434. 2026. URL: https://eprint.iacr.org/2026/2434.

[Sah99] Amit Sahai. "Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security". In: *40th FOCS*. IEEE Computer Society Press, Oct. 1999, pp. 543–553. DOI: 10.1109/SFFCS.1999.814628.

[Sch99] Berry Schoenmakers. "A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic". In: *CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Berlin, Heidelberg, Aug. 1999, pp. 148–164. DOI: 10.1007/3-540-48405-1_10.

[Sef24] Marek Sefranek. "How (Not) to Simulate PLONK". In: *SCN 24, Part I*. Ed. by Clemente Galdi and Duong Hieu Phan. Vol. 14973. LNCS. Springer, Cham, Sept. 2024, pp. 96–117. DOI: 10.1007/978-3-031-71070-4_5.

[Sta96] Markus Stadler. "Publicly Verifiable Secret Sharing". In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 190–199. DOI: 10.1007/3-540-68339-9_17.

[Wee09] Hoeteck Wee. "Zero Knowledge in the Random Oracle Model, Revisited". In: *ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, Berlin, Heidelberg, Dec. 2009, pp. 417–434. DOI: 10.1007/978-3-642-10366-7_25.

[ZXH+22] Jiaheng Zhang, Tiancheng Xie, Thang Hoang, Elaine Shi, and Yupeng Zhang. "Polynomial Commitment with a One-to-Many Prover and Applications". In: *USENIX Security 2022*. Ed. by Kevin R. B. Butler and Kurt Thomas. USENIX Association, Aug. 2022, pp. 2965–2982. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/zhang-jiaheng.

A Polynomial Commitments

A polynomial commitment enables a prover to commit to a target polynomial f over a finite field $\mathbb{F}$, and subsequently reveal the evaluations of f at some point without leaking any additional information. We adopt the formalization of [KZG10].

Definition 23 (Polynomial Commitment [KZG10]). *A polynomial commitment scheme* PCom *for univariate polynomials over a prime field* $\mathbb{F}$ *with* $|\mathbb{F}| = p$ *is defined by the following algorithms:*

- $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, d)$*: on input the security parameter* 1^λ *and a maximum degree* d *outputs a commitment key* ck*.*
- $\mathsf{c} \leftarrow \mathsf{Com}(\mathsf{ck}, f; r)$*: on input the commitment key* ck*, the polynomial* $f \in \mathbb{F}[x]$ *with degree* $\deg(f) \leq d$*, and possibly a randomness* r*, it outputs a commitment* c*.*
- $\pi \leftarrow \mathsf{Open}(\mathsf{ck}, f, x)$*: on input the commitment key* ck*, polynomial* $f \in \mathbb{F}[x]$ *and input* $x \in \mathbb{F}$ *it outputs a proof* π*.*
- $b \leftarrow \mathsf{Vf}(\mathsf{ck}, \mathsf{c}, x, y, \pi)$*: on input the commitment key* ck*, commitment* c*, a pair* $(x, y) \in \mathbb{F}^2$*, and a proof* π*, it outputs a bit* b*.*

We recall the definition of *unique proofs* given by Ganesh *et al.*; informally, it requires that finding two distinct accepting proofs for a fixed commitment, input and output is computationally infeasible.

Definition 24 (Unique Proof [GKO+23]). PCom *has unique proofs if for all PPT adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{l|l} \pi \neq \pi' & \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ \wedge b = 1 & (\mathsf{c}, x, y, \pi, \pi') \leftarrow \mathcal{A}(\mathsf{ck}) \\ \wedge b' = 1 & b \leftarrow \mathsf{Vf}(\mathsf{ck}, \mathsf{c}, x, y, \pi) \\ & b' \leftarrow \mathsf{Vf}(\mathsf{ck}, \mathsf{c}, x, y, \pi') \end{array}\right] \in \mathsf{negl}(\lambda)$$

Furthermore, we require the standard notions of completeness and evaluation-binding for polynomial commitments.

Definition 25 (PCS Completeness). PCom *has perfect completeness if for every* $d \in \mathbb{N}$*,* $x \in \mathbb{F}$*, and* $f \in \mathbb{F}[x]$ *with* $\deg(f) \leq d$

$$\Pr\left[b = 1 \;\middle|\; \begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ \mathsf{c} \leftarrow \mathsf{Com}(\mathsf{ck}, f; r) \\ \pi \leftarrow \mathsf{Open}(\mathsf{ck}, f, x) \\ b \leftarrow \mathsf{Vf}(\mathsf{ck}, \mathsf{c}, x, f(x), \pi) \end{array}\right] = 1$$

Definition 26 (PCS Evaluation-Binding). PCom *is evaluation-binding if for every* $d \in \mathbb{N}$ *and PPT* $\mathcal{A}$

$$\Pr\left[\begin{array}{l|l} y \neq y' & \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ \wedge b = 1 & (\mathsf{c}, x, y, y', \pi, \pi') \leftarrow \mathcal{A}(\mathsf{ck}) \\ \wedge b' = 1 & b \leftarrow \mathsf{Vf}(\mathsf{ck}, \mathsf{c}, x, y, \pi) \\ & b' \leftarrow \mathsf{Vf}(\mathsf{ck}, \mathsf{c}, x, y', \pi') \end{array}\right] \in \mathsf{negl}(\lambda)$$

A related notion, interplaying with polynomial commitments in prior work, is that of polynomial encoding. Informally, a polynomial encoding scheme enables a honest prover to encode a witness into a polynomial with certain properties.

Definition 27 (Polynomial Encoding [GKO+23]). *A polynomial encoding scheme* PEnc *is a pair of PPT algorithms* $(\mathsf{Encode}, \mathsf{Decode})$ *with the following interfaces:*

- $f_w \leftarrow \mathsf{Encode}(\mathbb{w}, \ell; r)$*: given a witness* $\mathbb{w} \in \mathbb{F}^n$ *and evaluation bound* $\ell \in \mathbb{N}$ *it outputs a polynomial* $f_w \in \mathbb{F}[x]$*.*
- $\mathbb{w} \leftarrow \mathsf{Decode}(f_w, n, \ell)$*: given a polynomial* $f_w \in \mathbb{F}[x]$ *and integers* $n, \ell \in \mathbb{N}$*, it outputs a witness* $\mathbb{w}$*.*

The encoding scheme is *correct* if for all $n, \ell > 0$ and $\mathbb{w} \in \mathbb{F}^n$, $\mathsf{Decode}(\mathsf{Encode}(\mathbb{w}, \ell; r), n, \ell) = \mathbb{w}$ with probability 1 over the sampling of r.

We further recall two properties pertaining to the interaction of a polynomial commitment scheme with a polynomial encoding scheme. The first one, known as evaluation-hiding, requires that the joint distribution of the encoding f_w and a bounded number of evaluations of f_w hides $\mathbb{w}$. The second one, known as non-extrapolation, requires that is infeasible to extrapolate the evaluation of f_w at a new point given ℓ' evaluations, even when f_w is an encoding of the 0 witness.

Definition 28 (Evaluation-Hiding). PCom *is evaluation hiding w.r.t* PEnc *if for all* $\ell \in \mathbb{N}$ *and PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$

$$\Pr\left[b = b' \wedge |z| \leq \ell \;\middle|\; \begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ (\mathbb{w}, z) \leftarrow \mathcal{A}_1(\mathsf{ck}) \\ b \leftarrow\!\$\, \{0,1\}, r \leftarrow\!\$\, \mathbb{F}^{1+\ell}, d = |\mathbb{w}| + \ell \\ f_w \leftarrow \mathsf{PEnc.Encode}(b \cdot \mathbb{w}, \ell; r) \\ \mathsf{c} \leftarrow \mathsf{PCom.Com}(\mathsf{ck}, f_w, d) \\ y_i \leftarrow f_w(z_i) \text{ for } i \in [|z|] \\ \pi_i \leftarrow \mathsf{Open}(\mathsf{ck}, f_w, z_i) \text{ for } i \in [|z|] \\ b' \leftarrow \mathcal{A}_2(\mathsf{c}, z, \{y_i, \pi_i\}_{i \in [|z|]}) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *share the same state.*

B Verifiable Secret Sharing

We give a standalone, game-based definition of (publicly) verifiable secret sharing. A dealer shares a secret among n parties so that any $t+1$ of them can reconstruct it, while any t or fewer learn nothing; *verifiability* means that anyone (not only the intended party) can check, from public information alone, that a share is well-formed with respect to the dealer's commitment. Crucially, this public check must not itself leak the share: the object anyone can verify (denoted c_i below) is decoupled from the actual share value sh_i, which only party i can recover, using a secret key sk_i established once at setup.

Intended applications and scope. We do not claim this construction dominates the state of the art on efficiency or adaptive security. Two comparisons apply. First, the fastest VSS-style schemes [MDR23; PPP25] are interactive: a share is valid only once a live, synchronized quorum runs a BFT-consensus sub-protocol, ruling them out whenever recipients are not simultaneously online (e.g., shares posted to a bulletin board and picked up asynchronously). This is a comparison of different primitives, not of efficiency.

Second, the closest non-interactive competitor [BCL26], and the classical lineage it belongs to [Sta96; Sch99; CD17; GJM+21; CD20; Bag25; Gro21], reconstruct only $S := h^s \in \mathbb{G}$, the image of s under the one-way map $s \mapsto h^s$ in a hard-DL group — not a domain mismatch (any two order-p sets are abstractly in bijection), but genuine one-wayness: no party, honest or not, can recover s from S, and this is intrinsic to how those schemes interpolate "in the exponent." Our own $\mathsf{VRSC.Encode}$ is likewise one-way, but only to gate public verifiability of c; Reveal decrypts via PKE, an invertible map, and returns s itself.

This matters wherever the shared value feeds further field arithmetic rather than only exponentiation — an MPC input or a Beaver triple [Bea92] — where the scalar must be recoverable without rewinding, for the same reason as the paper's main construction: a UC simulator cannot rewind an adversary a concurrent environment is also observing. Such preprocessing is conventionally analyzed under static corruption, matching our own construction's scope (below). No known construction combines non-interactive dealing, constant-size shares, public verifiability, and straight-line extraction of the raw scalar: [MDR23; PPP25] give the latter two only interactively, and [BCL26] and its lineage give the first two but extract only a one-way image of the scalar.

Definition 29 (Verifiable Secret Sharing). *A* publicly verifiable secret sharing scheme *for secrets in a domain* $\mathcal{D}_s$ *and* n *parties with threshold* $t < n$ *is a tuple of algorithms* $\mathsf{PVSS} = (\mathsf{Setup}, \mathsf{Deal}, \mathsf{ShareVerify}, \mathsf{Reveal}, \mathsf{Reconstruct})$ *that work as follows.*

- $\mathsf{Setup}(1^\lambda, n, t) \to (\mathsf{pp}, (\mathsf{sk}_i)_{i\in[n]})$*: on input the security parameter, the number of parties, and the threshold, outputs public parameters* pp *(which include a public key* pk_i *for each party* i*) and, for each* $i \in [n]$*, a secret key* sk_i *given to party* i *alone.*
- $\mathsf{Deal}(\mathsf{pp}, s) \to (\mathsf{c}, (c_i, \pi_i)_{i\in[n]})$*: on input the public parameters and a secret* $s \in \mathcal{D}_s$*, the (possibly malicious) dealer outputs a commitment* c *and, for each party* $i \in [n]$*, a public share record* c_i *together with a proof* π_i *of its well-formedness. Crucially,* c_i *need not reveal the share* sh_i *it encodes.*
- $\mathsf{ShareVerify}(\mathsf{pp}, \mathsf{c}, i, c_i, \pi_i) \to b \in \{0,1\}$*: a public algorithm, requiring no secret key, that accepts or rejects the record* c_i *of party* i *with respect to* c*.*
- $\mathsf{Reveal}(\mathsf{sk}_i, c_i) \to \mathsf{sh}_i$*: run by party* i *alone, recovers the share encoded in* c_i*.*
- $\mathsf{Reconstruct}(\mathsf{pp}, \{(i, \mathsf{sh}_i)\}_{i\in T}) \to s$*: on input any set* $T \subseteq [n]$ *with* $|T| \geq t+1$ *of (revealed) shares, outputs the reconstructed secret. For definiteness, when* $|T| > t+1$ *we take* $\mathsf{Reconstruct}$ *to act on the* $t+1$ *smallest-indexed elements of* T *and ignore the rest, so the algorithm is a well-defined, deterministic function of* T*.*

Definition 30 (Correctness). PVSS *is* correct *if for all* λ, n, t *and all* $s \in \mathcal{D}_s$*, the following probability is* $\geq 1 - \mathsf{negl}(\lambda)$*:*

$$\Pr\left[\begin{array}{l}\forall i \in [n],\ \mathsf{ShareVerify}(\mathsf{pp}, \mathsf{c}, i, c_i, \pi_i) = 1 \ \wedge \\ \forall T \subseteq [n],\ |T| \geq t+1, \\ \quad \mathsf{Reconstruct}\big(\mathsf{pp}, \{(i, \mathsf{Reveal}(\mathsf{sk}_i, c_i))\}_{i\in T}\big) = s\end{array} \middle| \begin{array}{l}(\mathsf{pp}, (\mathsf{sk}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, n, t) \\ (\mathsf{c}, (c_i, \pi_i)_{i\in[n]}) \leftarrow \mathsf{Deal}(\mathsf{pp}, s)\end{array}\right].$$

We require that a PVSS scheme satisfies the following security properties:

Definition 31 (Commitment Consistency). PVSS *satisfies* commitment consistency *if for every (non-uniform) efficient adversary* $\mathcal{A}$*, the following probability is* $\mathsf{negl}(\lambda)$*:*

$$\Pr\left[\begin{array}{c}\mathsf{Reconstruct}\big(\mathsf{pp}, \{(i, \mathsf{Reveal}(\mathsf{sk}_i, c_i))\}_{i\in T}\big) \\ \neq \\ \mathsf{Reconstruct}\big(\mathsf{pp}, \{(i, \mathsf{Reveal}(\mathsf{sk}_i, c'_i))\}_{i\in T'}\big)\end{array} \middle| \begin{array}{l}(\mathsf{pp}, (\mathsf{sk}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, n, t) \\ (\mathsf{c}, T, T', (c_i, \pi_i)_{i\in T}, (c'_i, \pi'_i)_{i\in T'}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathsf{ShareVerify}(\mathsf{pp}, \mathsf{c}, i, c_i, \pi_i) = 1\ \forall i \in T \\ \mathsf{ShareVerify}(\mathsf{pp}, \mathsf{c}, i, c'_i, \pi'_i) = 1\ \forall i \in T'\end{array}\right]$$

where $|T|, |T'| \geq t+1$*.*

Definition 32 (Secrecy). PVSS *satisfies* secrecy *if for every (non-uniform) efficient adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, the following probability is at most* $\frac{1}{2} + \mathsf{negl}(\lambda)$*:*

$$\Pr\left[|I| \leq t \ \wedge\ s_0, s_1 \in \mathcal{D}_s \ \wedge\ b' = b \middle| \begin{array}{l}(\mathsf{pp}, (\mathsf{sk}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, n, t) \\ (I, s_0, s_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ b \leftarrow\!\$\ \{0,1\} \\ (\mathsf{c}, (c_i, \pi_i)_{i\in[n]}) \leftarrow \mathsf{Deal}(\mathsf{pp}, s_b) \\ b' \leftarrow \mathcal{A}_2\big(\mathsf{st}, \mathsf{c}, (c_i, \pi_i)_{i\in[n]}, (\mathsf{sk}_i)_{i\in I}\big)\end{array}\right].$$

This VSS construction is not the paper's central contribution: our main result is the black-box compiler of Theorem 10, and this section presents VSS as a direct, illustrative application of VRSC to a smaller, self-contained relation. Correspondingly, we only target Secrecy against non-adaptive corruption; extending it to adaptive corruption would require additional machinery (e.g., non-committing encryption) orthogonal to VRSC itself, and we leave it for future work.

Unlike prior PVSS, which requires only *some* extractor — typically realized via rewinding [ZXH+22; PPP25] or by re-executing an interactive consensus subprotocol [MDR23] — the property below requires a *straight-line* extractor: one that recovers a full opening of c from the dealer's oracle-query transcript, without rewinding or a second interaction.

Definition 33 (Straight-Line Extractability). *Let* H *be a random oracle.* PVSS *satisfies* straight-line extractability *if there exists a (non-uniform) efficient extractor* $\mathcal{E}$ *such that for every (non-uniform) efficient adversary* $\mathcal{A}$ *(the possibly malicious dealer),*

$$\Pr\left[\begin{array}{c}\exists i \in [n] : \mathsf{ShareVerify}(\mathsf{pp}, \mathsf{c}, i, c_i, \pi_i) = 1 \\ \wedge \\ \mathsf{Reveal}(\mathsf{sk}_i, c_i) \neq f(i)\end{array}\middle|\begin{array}{l}(\mathsf{pp}, (\mathsf{sk}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, n, t) \\ (\mathsf{c}, (c_i, \pi_i)_{i\in[n]}) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{pp}) \\ f \leftarrow \mathcal{E}\big(\mathsf{pp}, \mathsf{c}, (c_i, \pi_i)_{i\in[n]}, Q\big)\end{array}\right] = \mathsf{negl}(\lambda),$$

where $f : [n] \to \mathcal{D}_s$ *is the function of accepted share values* $\mathcal{E}$ *claims to recover, i.e.,* $f(i) = \mathsf{sh}_i$*, and* Q *denotes the transcript of* $\mathcal{A}$*'s queries to* H *(and their answers) during its execution.*

B.1 A Black-Box Construction from VRSC

We now give a PVSS construction that treats VRSC as a black box, using only its four algorithms $(\mathsf{Setup}, \mathsf{Encode}, \mathsf{CLeak}, \mathsf{Vf})$, and instead pairs it with an outer simulation-extractable NIZK Π (Section 3.2) — exactly the two ingredients combined by $\Pi_{\mathsf{UC\text{-}SNARK}}$ in Figure 6 — together with a public-key encryption scheme that keeps each c_i from revealing its share. The construction is a direct specialization of that compiler to a different, much smaller outer relation, and its proof reuses the same reduction techniques.

Public-key encryption. We recall the standard syntax and security notion, since it is not used elsewhere in the paper.

Definition 34 (Public-key encryption). *A public-key encryption scheme with message space* $\mathcal{M}$ *and randomness space* $\mathcal{R}_{\mathsf{Enc}}$ *is a triple* $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *where* $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*,* $\mathsf{Enc}(\mathsf{pk}, m; \rho) \to c$ *for* $m \in \mathcal{M}$*,* $\rho \in \mathcal{R}_{\mathsf{Enc}}$*, and* $\mathsf{Dec}(\mathsf{sk}, c) \to m$*. It is* perfectly correct *if for all* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{KeyGen}(1^\lambda)$*, all* $m \in \mathcal{M}$*, and all* $\rho \in \mathcal{R}_{\mathsf{Enc}}$*,* $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m; \rho)) = m$*.*

Definition 35 (IND-CPA security). PKE *is* IND-CPA secure *if for every (non-uniform) efficient adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,*

$$\left|\Pr\left[\mathcal{A}_2(\mathsf{st}, c) = \beta \middle| \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}) \\ \beta \leftarrow\!\!\$\ \{0,1\};\ \rho \leftarrow\!\!\$\ \mathcal{R}_{\mathsf{Enc}} \\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_\beta; \rho)\end{array}\right] - \tfrac{1}{2}\right| = \mathsf{negl}(\lambda).$$

Our Construction. We detail the full construction in Figure 7, and elaborate on its design in the following. For a threshold t, define the *evaluation-and-encryption relation*

$$\mathcal{R}_{\mathsf{share}} \coloneqq \left\{ \big(((i, \mathsf{pk}_i, c_i),\ \mathsf{c}),\ (\mathbb{w}, r, \rho)\big) \;\middle|\; \begin{array}{l}\mathsf{c} = \mathsf{VRSC.Encode}(\mathsf{cpk}, \mathbb{w} \| r) \ \wedge \\ c_i = \mathsf{PKE.Enc}(\mathsf{pk}_i, \phi_{\mathbb{w}}(i); \rho)\end{array} \right\},$$

$$\phi_{\mathbb{w}}(X) \coloneqq \sum_{k=0}^{t} \mathbb{w}_k X^k,$$

parameterized by cpk. Here $\mathbb{w} = (\mathbb{w}_0, \ldots, \mathbb{w}_t) \in \mathbb{F}^{t+1}$ throughout, so that $\phi_{\mathbb{w}}(0) = \mathbb{w}_0$; the witness space for $\mathbb{w}$ is fixed to $\mathbb{F}^{t+1}$ — i.e., the degree bound t is compiled directly into the relation, not enforced by an auxiliary check. Note that $\mathcal{R}_{\mathsf{share}}$ no longer mentions the plaintext share $\mathsf{sh}_i = \phi_{\mathbb{w}}(i)$ at all in its public statement: only the ciphertext c_i is public, which is exactly what closes the gap of having Deal output shares in the clear. Assume a simulation-extractable NIZK $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for $\mathcal{R}_{\mathsf{share}}$, with a non-programming trapdoor simulator $(\mathsf{SimSetup}, \mathcal{S})$ as in Definition 3.

We assume $\mathbb{F} = \mathbb{F}_p$ for some prime p and $0 \le t < n < p$. All security games and algorithms use the same random oracle, with distinct tags for VRSC and SNARK queries. The VRSC encoding is oracle-free and its extractor is a deterministic algorithm on the recorded transcript. The two proof-system setups use independent randomness.

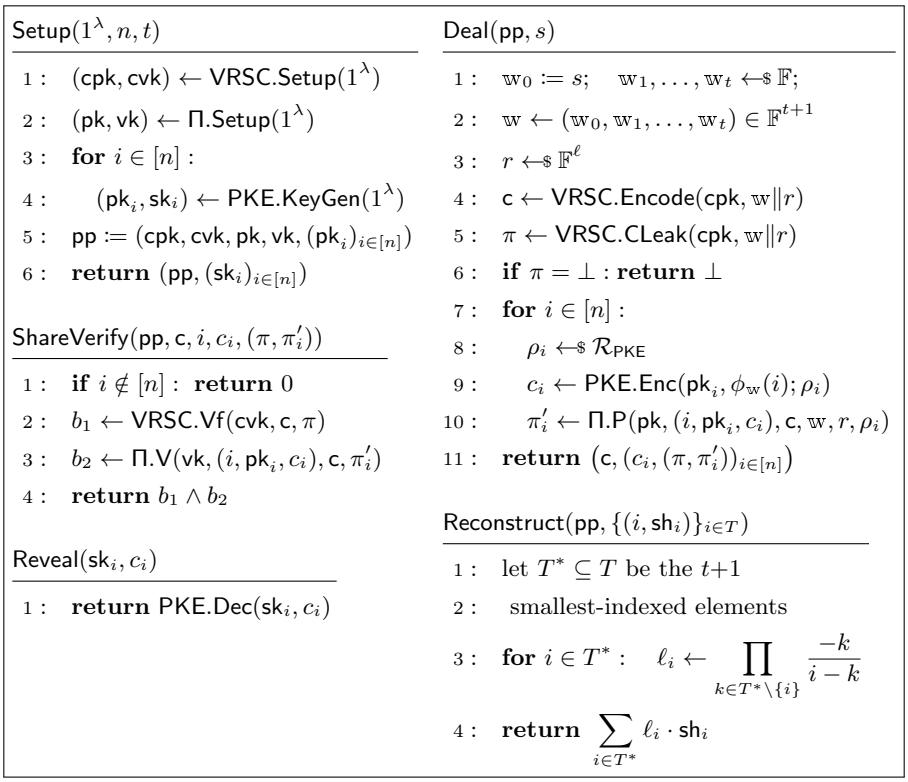

Fig. 7. A black-box PVSS using VRSC, a NIZK Π for $\mathcal{R}_{\mathsf{share}}$, and PKE under the hypotheses of Theorem 11. Reconstruct evaluates the Lagrange interpolant of T^* at 0, which equals $\phi_{\mathbb{w}}(0) = \mathbb{w}_0$ whenever the shares $\{\mathsf{sh}_i\}_{i\in T^*}$ all lie on a common degree-$\leq t$ polynomial $\phi_{\mathbb{w}}$.

Note that, in Figure 7, (c, π) does not depend on i: it is generated once per dealing and reused, unmodified, across all n per-party proofs $\pi'_1, \ldots, \pi'_n$. This is also why Setup does not depend on n beyond generating the n key pairs: since i is only ever a public input to $\mathcal{R}_{\mathsf{share}}$, the same $(\mathsf{cpk}, \mathsf{cvk}, \mathsf{pk}, \mathsf{vk})$ supports a later-decided number of parties below p, for the fixed threshold and supported input sizes. New recipients still need encryption keys and distinct nonzero field indices.

The abstract π_i of Definition 29 is instantiated here as the pair (π, π'_i): π (the same string for every i) certifies that c opens to *some* $\mathbb{w} \in \mathbb{F}^{t+1}\|\mathbb{F}^\ell$, and π'_i certifies that c_i encrypts $\phi_{\mathbb{w}}(i)$ for that same $\mathbb{w}$.

Theorem 11 (Black-box PVSS from VRSC). *Let* VRSC *be ℓ-hiding, binding, and reconstructible, with oracle-free encoding and a deterministic transcript-only extractor. Let* Π *have the trapdoor zero-knowledge and honest-setup offline knowledge-soundness properties specified above. Let* PKE *be IND-CPA secure and perfectly correct for every valid message and encryption random tape, with message space $\mathbb{F}_p$. For $0 \leq t < n < p$, Figure 7 satisfies correctness, commitment consistency, secrecy, and straight-line extractability under the static corruption experiment of Definition 32.*

We isolate the technical core of Commitment Consistency and Straight-Line Extractability into a single lemma, since both properties are consequences of the same fact: a malicious dealer cannot make c admit accepting records for two different underlying witnesses:

Lemma 4 (Unique extraction). *Under the hypotheses of Theorem 11, there is a uniform efficient public algorithm $\mathcal{E}_{\mathsf{PVSS}}$ with the following property. Let a dealer output* c *and a polynomial list L of records $(i, c_i, (\pi_i, \pi'_i))$, allowing repeated indices and different records at the same index. Given $(\mathsf{pp}, \mathsf{c}, L, Q)$, where*

Q is the dealer's original VRSC query transcript, the algorithm outputs a polynomial ϕ of degree at most t. Except with negligible probability, every accepting record in L decrypts to $\phi(i)$.

Proof. In list order, for each record compute $x \leftarrow \mathsf{VRSC.Ext}(\mathsf{cvk}, \mathsf{cpk}, Q, \mathsf{c}, \pi_i)$, where Ext is the extractor guaranteed to exist by VRSC's Reconstructability (Definition 17). If x parses as $\mathbb{w}\|r \in \mathbb{F}^{t+1} \times \mathbb{F}^{\ell}$ and $\mathsf{VRSC.Encode}(\mathsf{cpk}, x) = \mathsf{c}$, return $\phi_{\mathbb{w}}$ for the first such candidate. If none exists, return the zero polynomial.

For each of the at most $m = |L|$ records, honest-setup offline knowledge soundness supplies a witness $(\mathbb{w}_i, r_i, \rho_i)$ for its original SNARK statement whenever that proof accepts, except with error ϵ_{ks}. These witnesses are produced only inside the reduction, from private copies of the original execution. The offline computation makes no further live VRSC queries. Thus, for any record whose VRSC proof also verifies, $\mathsf{VRSC.Vf}(\mathsf{cvk}, \mathsf{c}, \pi_i) = 1$, a reconstructability reduction can output $(\mathbb{w}_i\|r_i, \mathsf{c}, \pi_i)$ with exactly transcript Q: except with reconstructability error ϵ_{rec}, the candidate $\mathsf{VRSC.Ext}(\mathsf{cvk}, \mathsf{cpk}, Q, \mathsf{c}, \pi_i)$ is then a well-formed opening of c. In particular, if any record is fully accepting (both checks of $\mathsf{ShareVerify}$ pass), at least one candidate therefore passes the public extractor's encoding check.

Let $x^* = \mathbb{w}^*\|r^*$ be the first candidate that passes. Compare it to the knowledge witness for every accepting record, including records with a repeated index or a repeated VRSC certificate. Both encode to c. If they differ, the reduction outputs two distinct openings and breaks binding. A union bound over records gives error at most $m(\epsilon_{\mathrm{ks}} + \epsilon_{\mathrm{rec}} + \epsilon_{\mathrm{bind}})$. Outside these events, all accepting records have $\mathbb{w}_i\|r_i = x^*$. Perfect correctness for the extracted encryption randomness then yields $\mathsf{PKE.Dec}(\mathsf{sk}_i, c_i) = \phi_{\mathbb{w}^*}(i)$. If no record accepts, the claim is vacuous. This proves the lemma.

□

We remark that, similar to the proof of reconstructability of our main construction, only the security reductions use SNARK knowledge extraction. The public extractor is the explicit transcript algorithm above, which is rewinding-free. We now use Lemma 4 to prove the main claim of this section.

Proof (of Theorem 11).

Correctness. An honest dealing samples a degree-t polynomial $\phi_{\mathbb{w}}$ with $\phi_{\mathbb{w}}(0) = s$. Except with the VRSC completeness error, its certificate is generated and verifies. Each per-index SNARK statement has the explicitly supplied witness $(\mathbb{w}, r, \rho_i)$, so perfect SNARK completeness makes its proof verify. Perfect encryption correctness gives

$$\mathsf{Reveal}(\mathsf{sk}_i, c_i) = \phi_{\mathbb{w}}(i) \qquad \text{for every } i \in [n].$$

Any qualified set contains $t+1$ distinct field indices. Interpolation recovers $\phi_{\mathbb{w}}$ and hence its value s at zero. Thus correctness fails only with the VRSC completeness error, which is negligible.

Commitment consistency. Apply Lemma 4 to the concatenation of the two output collections indexed by T and T'. Keep both records at every index in their intersection, even if they differ. This list has at most $2n$ entries, so the lemma applies with a polynomial loss. Except with negligible probability, every accepting decrypted share in both collections equals $\phi(i)$ for the same degree-$\leq t$ polynomial. Each qualified set contains at least $t+1$ distinct nonzero indices, so both reconstructions return $\phi(0)$.

Straight-line extractability. On the dealer's indexed output, form the corresponding list of records and run the public algorithm in Lemma 4. Its conclusion is exactly the per-accepting-record guarantee in Definition 33. A single accepting record, together with the dealer's transcript, suffices to obtain a candidate for the entire polynomial. The actual extractor performs no rewinding and runs no SNARK knowledge extractor.

Secrecy. Fix the adversary, let I be its selected corrupted set with $|I| \leq t$, and sample the challenge bit b. An honest certificate-generation abort occurs with negligible probability (matching VRSC's completeness error) and is absorbed into the final bound. Use the following hybrids.

Game_0. The real secrecy experiment.

Game_1. Generate the SNARK parameters using $\mathsf{SimSetup}$ and simulate *all* dealer proofs, including those for $i \in I$, with $\mathcal{S}(\mathsf{td}, ((i, \mathsf{pk}_i, c_i), \mathsf{c}))$. Commitments and ciphertexts are still honest, so all requests have known valid witnesses. Independent VRSC and encryption keys are generated locally by the reduction. Multi-query zero knowledge accounts jointly for this setup switch and the n proof requests. That is, provided that Π is zero-knowledge, Game_0 and Game_1 are indistinguishable, albeit with (negligible) probability $n \cdot \epsilon_{\mathrm{zk}}$.

Game_2. Replace the common VRSC pair (c, π) by a pair generated by calling $\mathsf{VRSC.Encode}$ and $\mathsf{VRSC.CLeak}$ on input $0^{t+1}\|r$, with fresh padding, and simulate every SNARK proof for its updated public statement. A reduction to hiding proves that this step only introduces a negligible gap between Game_2 and Game_1. The reduction knows the independent SNARK trapdoor and can simulate for the returned commitment without knowing its opening. These calls remain well-defined and accepting even once the updated statement has no known witness: by Game_1, $\mathcal{S}$ already accepts with overwhelming probability on the honest (true) statement, by zero-knowledge and completeness of Π; since the only difference between the honest and updated statement is exactly the VRSC-hiding challenge on c, with everything else fixed, any drop in $\mathcal{S}$'s acceptance probability after the swap would itself be an efficient distinguisher against VRSC's hiding. No old proof is retained for a new commitment.

Game_3. For $i \notin I$, replace the encryption of $\phi_{\mathrm{w}_b}(i)$ by an encryption of zero, one index at a time, and simulate the proof for the resulting statement. Each change is justified by IND-CPA. To embed a challenge at an index i, use the challenger's public key there and generate the other keys locally. If the adversary selects $i \in I$, abort this reduction with a fixed output: the two adjacent hybrids coincide on that event because corrupted ciphertexts are never changed. Otherwise no secret key for i is needed. Summing over the n fixed index positions gives a distinguishing advantage between Game_3 and Game_2 of at most $n\epsilon_{\mathrm{cpa}}$.

In Game_3, the VRSC pair and all uncorrupted ciphertexts are independent of b. The corrupted ciphertexts depend on b only through $(\phi_{\mathrm{w}_b}(i))_{i \in I}$. Because these are evaluations at at most t distinct nonzero points of a random degree-t polynomial with constant coefficient s_b, their joint distribution is independent of s_b. All per-index proofs are now computed from public statements by the simulator, so they introduce no further witness dependence. The corrupted secret keys were sampled independently of b. The entire final view is therefore independent of b.

The real distinguishing gap is bounded by $n\epsilon_{\mathrm{zk}} + \epsilon_{\mathrm{hid}} + n\epsilon_{\mathrm{cpa}}$, and is negligible by the hypotheses of Theorem 11.

□

Remark 2 (Genericity versus prior non-interactive PVSS). Non-interactive PVSS has, to our knowledge, always been instantiated monolithically: construction and proof are derived together for one fixed algebraic setting, and porting to a different assumption means starting over. This holds across the whole lineage we compare against in Section B — Stadler [Sta96], Schoenmakers [Sch99], SCRAPE [CD17], Aggregatable DKG [GJM+21], Groth's NI-DKG [Gro21], and Bacho–Loss [BL23; BCL26] — where extraction, when it exists at all, is a rewinding argument tied to one specific Σ-protocol or NIZK, not a component that could be swapped out.

Theorem 11, by contrast, is generic in all three ingredients: any VRSC satisfying Hiding, Binding, and Reconstructability, any simulation-extractable NIZK for $\mathcal{R}_{\mathsf{share}}$, and any IND-CPA secure PKE. Its proof depends only on these black-box properties, so a future VRSC instantiation — a different code, commitment, or post-quantum assumption — yields a new PVSS instantiation without touching Definitions 31 to 33 at all. We currently give only one concrete VRSC instantiation (Theorem 7), so this genericity is a property of the theorem statement, not yet exercised by a second backend.