Nenya: Efficient zkSNARKs for Integer Rings $\mathbb{Z}/2^k\mathbb{Z}$ via Reverse Multiplication-Friendly Embeddings

Tianyu Zhang Nikita Borisov Yupeng Zhang
University of Illinois Urbana-Champaign
{tianyuz,nikita,zhangyp}@illinois.edu

Abstract

Zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) are powerful cryptographic tools for certifying the correctness of computations with short proofs, fast verification, and zero-knowledge privacy guarantees. However, most existing zkSNARK schemes support arithmetic only over finite fields defined by large primes or their extensions. This creates substantial overhead when proving computations written in high-level programming languages, which typically operate on fixed-width integer types such as 32-bit and 64-bit integers.

In this paper, we introduce Nenya, the first practical SNARK scheme that supports native arithmetic over the integer ring $\mathbb{Z}/2^k\mathbb{Z}$. At the core of Nenya are new reductions from rank-1 constraint systems (R1CS) over $\mathbb{Z}/2^k\mathbb{Z}$ to constraints over Galois rings via reverse multiplication-friendly embeddings (RMFEs). We also design a polynomial commitment scheme based on suitable classes of error-correcting codes, which supports well-formedness checks and packing rotations on committed witnesses essentially for free, resulting in sublinear verifier time.

Experimental results demonstrate that Nenya achieves strong concrete performance. For a rank-1 constraint system of size 2^{25} for data-parallel computations over 32-bit integers, Nenya achieves a prover time of only 4.8 seconds. This is four orders of magnitude faster than the state-of-the-art SNARK scheme over $\mathbb{Z}/2^k\mathbb{Z}$, and only 1.06× slower than a SNARK over a field with the same number of constraints. For general computation, the prover time of Nenya is 34.6 seconds. Moreover, for proving pixel-level image transformations that arise in media-provenance workflows, such as filtering that operates natively modulo 2^8, Nenya proves the same computation 12× faster than a field-based SNARK.

Contents

1 Introduction **4**
- 1.1 Related Work . . . 5

2 Technical Overview **6**

3 Preliminaries **7**
- 3.1 Galois Rings . . . 8
  - 3.1.1 Unit-Separated Sets . . . 8
  - 3.1.2 Polynomials over Galois Rings . . . 8
  - 3.1.3 Modules and Linear Codes over Rings . . . 9
- 3.2 zkSNARKs . . . 9
  - 3.2.1 R1CS over Rings . . . 10
  - 3.2.2 Polynomial Interactive Oracle Proof over Rings . . . 10
  - 3.2.3 Polynomial Commitment Scheme over Rings . . . 11
- 3.3 Multiplicative Packing . . . 11
  - 3.3.1 Reduction of the Multiplication Relation . . . 12

4 Reducing R1CS Relations via Multiplicative Packing and PIOPs **12**
- 4.1 SIMD Circuits . . . 12
  - 4.1.1 Reduction of SIMD Circuits . . . 13
  - 4.1.2 PIOP for SIMD Circuits . . . 13
- 4.2 MIMD Circuits . . . 14
  - 4.2.1 Reduction of MIMD Circuits . . . 14
  - 4.2.2 PIOP for MIMD Circuits . . . 15
- 4.3 General Circuits . . . 15
  - 4.3.1 Reduction of General Circuits . . . 16
  - 4.3.2 PIOP for General Circuits . . . 17
- 4.4 Making the Verifier Sublinear . . . 17

5 Polynomial Commitment Scheme over Rings **18**
- 5.1 Lifting Binary Linear Codes to Galois Rings . . . 19
- 5.2 Submodule Checks Are Free . . . 20
  - 5.2.1 Quotient Code . . . 21
  - 5.2.2 Free Submodule Check . . . 22
- 5.3 Virtual PCS Are Free . . . 22
- 5.4 SNARKs over Rings . . . 23

6 Implementation and Evaluation **23**
- 6.1 Implementation and Settings . . . 23
- 6.2 Experimental Evaluations . . . 24
  - 6.2.1 Performance of Nenya and Comparison with Field SNARK . . . 25
  - 6.2.2 Improvement over Prior Work . . . 26
  - 6.2.3 Native Integer Computations . . . 27

A Instantiation of Multiplicative Packing **32**
- A.1 The Two-Layer Interpolation Construction . . . 32
- A.2 Efficient Reduction via Quotient Decomposition . . . 33

B Deferred Proofs **33**
B.1 Prime-Field Lift of Linear Codes 33
B.2 Proof of Theorem 5.1 34
B.3 Proof of Theorem 5.2 34
B.4 Proof of Theorem 5.6 35
B.5 Proof of Theorem 5.8 35
B.6 Proof of Theorem 5.12 35
B.7 Proof Sketch of Theorem 5.13 36

1 Introduction

Zero-knowledge succinct non-interactive arguments of knowledge, or zkSNARKs, have seen remarkable development in recent years. A zkSNARK enables a prover to convince a verifier of the correctness of a computation over a secret witness, while revealing nothing beyond the validity of the statement. zkSNARKs have found broad applications in areas such as blockchains [SCG+14, vit], machine learning [ZFZS20, LXZ21, WYX+21, GGJ+23, SLZ24, APKP24], and photo authenticity [NT16, DCB25]. However, most existing zkSNARK constructions model computations as arithmetic circuits with additions and multiplications over large prime fields or extension fields. In contrast, real-world programs written in high-level programming languages typically operate on fixed-width machine words, such as 32-bit and 64-bit integers. This mismatch can introduce substantial overhead when compiling such computations into zkSNARK-compatible circuits, as native integer operations must be emulated using field arithmetic, and may also lead to a loss of fidelity when computations are reformulated using finite-field arithmetic.

The major challenge of building zkSNARKs that support 32-bit and 64-bit integers natively lies in the small residue characteristic of these integer rings. Randomized testing and the Schwartz–Zippel lemma [Sch80, Zip79] are essential techniques in modern zkSNARKs. However, applying them over the integer ring $\mathbb{Z}/2^k\mathbb{Z}$ yields a large soundness error of $\frac{1}{2}$, compared to $\frac{1}{p}$ over a large prime field $\mathbb{F}_p$. One line of research addresses this issue using the technique of reverse multiplication-friendly embeddings (RMFEs) [JLX+25, BCGL22]. An RMFE packs multiple elements of $\mathbb{Z}/2^k\mathbb{Z}$ into a Galois ring $\mathrm{GR}(2^k, d)$ such that the soundness error of the Schwartz–Zippel lemma is reduced to $\frac{1}{2^d}$. However, existing schemes resulting from this research line remain impractical due to the high overhead of the reductions between constraints over $\mathbb{Z}/2^k\mathbb{Z}$ and $\mathrm{GR}(2^k, d)$, and the additional checks introduced by the reductions.

Our Contributions. In this paper, we develop the first concretely efficient SNARK scheme that supports native arithmetic over the integer rings $\mathbb{Z}/2^k\mathbb{Z}$ using RMFEs. To make this approach practical, we introduce the following technical contributions:

- First, we propose new reductions that efficiently transform rank-1 constraint systems (R1CSs) over $\mathbb{Z}/2^k\mathbb{Z}$ into constraints over $\mathrm{GR}(2^k, d)$ using RMFEs. We provide three variants tailored to different computation models: single-instruction-multiple-data (SIMD) computations, multiple-instruction-multiple-data (MIMD) computations, and general computations, in decreasing order of efficiency. Our reductions only require degree-2 RMFEs. Compared with existing reductions, such as those of [CG22], our reductions achieve improved packing parameters, leading to better concrete efficiency.

- Second, we propose an efficient polynomial commitment scheme (PCS) over Galois rings. The scheme is based on interleaved codes and follows the construction of [AHIV17, GLS+23]. The key technical innovation is that for suitable classes of error-correcting codes, such as those lifted from binary codes, the PCS supports image checks, kernel checks, and rotation checks on committed polynomials essentially for free. This eliminates a major bottleneck in previous reductions for R1CS over $\mathbb{Z}/2^k\mathbb{Z}$. This also enables sublinear-time SNARK verification in the holographic setting with preprocessing, as previous versions of these checks required linear verification time.

  For an R1CS of size n, the prover time of the resulting SNARK scheme is $O(n)$, the proof size is $O(\sqrt{n})$ and the verifier time is $O(\sqrt{n})$ (in the holographic setting). The scheme has a transparent setup and is plausibly post-quantum secure.

- Finally, we fully implement our scheme, Nenya, and benchmark its performance. For an R1CS of size 2^{25} for SIMD computations over 32-bit integer arithmetic, Nenya achieves a prover time of only 4.8 seconds with multi-threading. This is more than four orders of magnitude faster than the prior scheme in [JLX+25], and only 1.06× slower than a SNARK for an R1CS of

the same size over a field! For MIMD computations of size $n = 2^{25}$, the prover time of Nenya is 9.6 seconds, and for general computations it is 34.6 seconds. To further demonstrate the advantages of Nenya, we implement an image-filtering algorithm using native arithmetic modulo 2^8. For this computation, Nenya proves up to 12× faster than a SNARK over a prime field. Our code is available at `https://github.com/tnyuzg/nenya`.

1.1 Related Work

The most relevant prior works are [JLX+25, CG22, BCGL22]. Jia et al. [JLX+25] proposed a PCS over Galois rings. Combined with a constraint system from [CG22] that packs many $\mathbb{Z}/2^k\mathbb{Z}$ elements into a Galois ring through an RMFE, they built a SNARK for R1CS over $\mathbb{Z}/2^k\mathbb{Z}$. Our scheme improves upon these works in several aspects. First, the major overhead of prior schemes comes from the submodule checks on the committed witnesses, such as the image and kernel checks, as not every Galois ring element is a valid packing of the base ring elements via RMFE. Prior works [CG22] perform the checks at the PIOP level via random linear combinations, which add $O(\lambda n)$ work to both the prover and the verifier. Bootle et al. [BCGL22] reduced the verifier time of this approach from linear to sublinear using structured randomness, but the overhead on the prover remains the same. In our scheme, we shift these checks into our PCS (Section 5.2) and prove that they are essentially for free. This makes our prover time linear and the verifier sublinear in n, which are also concretely efficient.

Second, both schemes in [JLX+25, BCGL22] rely on the R1CS reduction in [CG22], which originally works over $\mathbb{Z}_2$, but can be adapted to work for $\mathbb{Z}/2^k\mathbb{Z}$. Our reductions significantly improve upon that in [CG22], with smaller commitment size and only degree-2 RMFEs (Section 4). Moreover, the reduction in [CG22] relies on a kernel check on the summation of the packed elements in $\mathbb{Z}/2^k\mathbb{Z}$. It is not clear how to perform the check efficiently. Our reduction uses a rotation check, which is supported by our PCS again for free. With these improvements and additional optimizations, we develop the first practical SNARK scheme for $\mathbb{Z}/2^k\mathbb{Z}$ using RMFEs, while these prior works were mainly of theoretical interest.

zkSNARKs over rings. In addition to these schemes using RMFEs, there are other prior works on building zkSNARKs over rings. For the information-theoretic part, the Schwartz–Zippel lemma was extended from fields to rings through the notion of unit-separated sets [BCPS18], which in turn allows the sumcheck and GKR protocols to be carried over to rings [CCKP19, SV22, BCS21]. On top of these, the standard PIOPs such as Libra, HyperPlonk, and Spartan have been restated over rings [WZD25a, HMZ25, JLX+25]. Most of these works remain theoretical, with no implementations or impractical efficiency.

For the cryptographic part, three families of commitments have been generalized to rings. Rinocchio [GNSV23] adapted the pairing-based SNARK of Pinocchio [PHGR13] to rings via cryptographic ring encodings. Lattice-based polynomial commitments or folding schemes [BS23, NS24, BC25] are native to rings, but mostly to specific polynomial rings with large residue characteristic, rather than the integer ring $\mathbb{Z}/2^k\mathbb{Z}$ we target. Hash-based polynomial commitments are the most relevant to our work. Proximity gaps are generalized from fields to rings for general linear codes [HMZ25, WZD25a] and for Reed–Solomon codes [JLX+25], with a new distance analysis of the code over Galois rings [WZD+25b]. We follow the same PCS based on interleaved codes over Galois rings, but propose submodule and rotation checks that are critical for building SNARKs using RMFEs.

Instead of building a SNARK over rings directly, some works reduce a relation over a ring to a relation over a field and then use a field-based SNARK. Such reductions translate cyclotomic-ring arithmetic to a field through the CRT [CCC+25], and integer and rational relations to a field using PIOPs [GWHD25]. These techniques are not directly comparable to ours, as we primarily focus on reductions utilizing RMFEs.

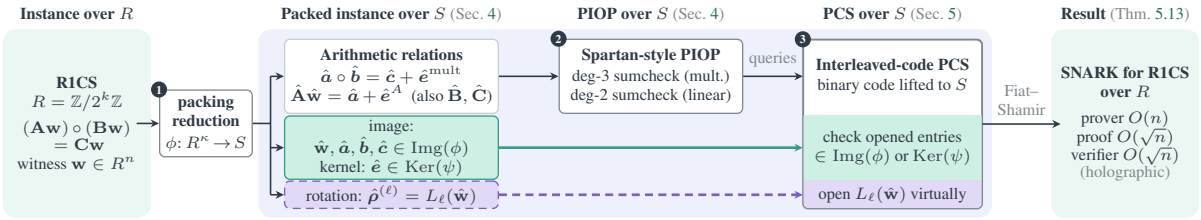


Figure 1: Overview of Nenya. Green regions live over $R = \mathbb{Z}/2^k\mathbb{Z}$ and the blue one over $S = \text{GR}(2^k, d)$. ❶ The packing reduction maps the original R1CS instance to a packed instance over S. ❷ Only its arithmetic relations go through the PIOP. ❸ The PCS checks the image and kernel relations on the opened columns and, for general circuits only, the rotation relation by a virtual opening of $L_\ell(\hat{\mathbf{w}})$, both at no extra cost (Sections 5.2 and 5.3). All three variants of Figure 3 share this pipeline.

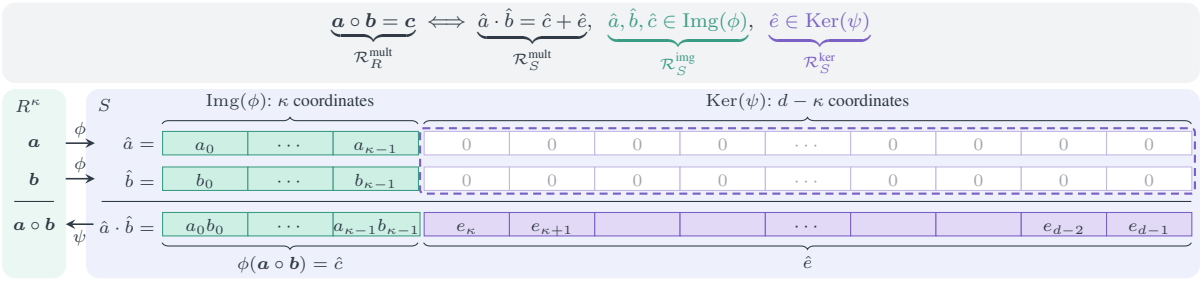


Figure 2: Reducing $\mathcal{R}_R^{\text{mult}}$ via any degree-2 multiplicative packing, with S drawn in a basis of $\text{Img}(\phi) \oplus \text{Ker}(\psi)$ in which ψ reads off the image coordinates. The dashed box marks the zeros enforced by $\mathcal{R}_S^{\text{img}}$.

2 Technical Overview

We will use bold lowercase for vectors ($\boldsymbol{a}$) and bold uppercase for matrices ($\mathbf{A}$). Throughout, the R1CS is defined over the base integer ring $R = \mathbb{Z}/2^k\mathbb{Z}$ and the proof system is built over the Galois ring $S = \text{GR}(2^k, d) = R[X]/(f)$, where f has degree d and is irreducible modulo 2; we write elements of S with a hat, as in $\hat{a}$.

Nenya proceeds in three steps, as shown in Figure 1. First, we reduce an R1CS of size n over R to a *packed* R1CS over S, i.e., an R1CS over S together with a few extra relations, via a multiplicative packing (RMFE [CCXY18]). The extra relations are specific to the packing: image and kernel relations and, for general R1CS, a rotation relation. Second, we reduce the packed R1CS to queries to the PCS via a Spartan-style polynomial IOP (PIOP) generalized to S, with sumchecks tailored to rings. Third, our PCS answers these queries and checks the image, kernel, and rotation relations directly on the committed values, at no extra cost.

Step 1: Reduction using Multiplicative Packing (Section 4). A degree-2 multiplicative packing is a pair of R-linear maps $\phi : R^\kappa \to S$ and $\psi : S \to R^\kappa$ with $\psi(\hat{a} \cdot \hat{b}) = \boldsymbol{a} \circ \boldsymbol{b}$ for $\hat{a} = \phi(\boldsymbol{a})$ and $\hat{b} = \phi(\boldsymbol{b})$: after packing, one multiplication in S acts as κ multiplications in R (our implementation uses $\kappa = 32$ and $d = 128$).

A multiplicative packing further requires $\phi(\mathbf{1}) = 1$, so $\psi \circ \phi$ is the identity, which implies $S = \text{Img}(\phi) \oplus \text{Ker}(\psi)$: every $s \in S$ splits into an image component and a kernel component. In a suitable basis (Figure 2), these components have κ and $d - \kappa$ coordinates, respectively. A packed element $\hat{a}$ has image coordinates $\boldsymbol{a}$ and zero kernel coordinates, i.e., it satisfies the *image relation* $\hat{a} \in \text{Img}(\phi)$. After multiplying by another packed element $\hat{b}$, the image coordinates become $\boldsymbol{a} \circ \boldsymbol{b}$, but the kernel component is in general nonzero; we call it the error $\hat{e}$, and the *kernel relation* requires $\hat{e} \in \text{Ker}(\psi)$.

Hence $\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c}$ becomes $\hat{a} \cdot \hat{b} = \hat{c} + \hat{e}$, with the image relation on $\hat{a}, \hat{b}, \hat{c}$, and the kernel relation on $\hat{e}$. Both are needed: without the kernel relation, any $\hat{c}$ passes, and without the image relation $\psi(\hat{a} \cdot \hat{b})$ need not equal $\psi(\hat{a}) \circ \psi(\hat{b})$.

An R1CS $(\mathbf{Aw}) \circ (\mathbf{Bw}) = \mathbf{Cw}$ divides into the multiplication relation $\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c}$ and the linear relations $\boldsymbol{a} = \mathbf{Aw}$, $\boldsymbol{b} = \mathbf{Bw}$, $\boldsymbol{c} = \mathbf{Cw}$. The multiplication relation is entry-wise, so it matches the packing directly and reduces as above. The linear relations, however, mix entries of $\mathbf{w}$, so how they carry over to S depends on which entries we pack together, giving three variants. For *SIMD* circuits (κ copies of one sub-circuit), we pack the κ copies of each witness entry; the sub-circuit matrix $\mathbf{A}_{\mathsf{sub}}$ is over R, so $\hat{\boldsymbol{a}} = \mathbf{A}_{\mathsf{sub}}\hat{\mathbf{w}}$ holds exactly. For *MIMD* circuits (κ different sub-circuits), we also pack the κ sub-circuit matrices, entry by entry, into a matrix $\hat{\mathbf{A}}$ over S, so each linear relation gets a kernel error: $\hat{\mathbf{A}}\hat{\mathbf{w}} = \hat{\boldsymbol{a}} + \hat{\boldsymbol{e}}^A$. For *general* circuits, we pack consecutive blocks of $\mathbf{w}$. Since a $\kappa \times \kappa$ block of $\mathbf{A}$ mixes slots, the diagonal method of Halevi and Shoup [HS14] rewrites it as slot-wise products with the κ rotations of the block, giving $\sum_\ell \hat{\mathbf{D}}_A^{(\ell)} \hat{\boldsymbol{\rho}}^{(\ell)} = \hat{\boldsymbol{a}} + \hat{\boldsymbol{e}}^A$ for packed diagonals $\hat{\mathbf{D}}_A^{(\ell)}$ and the *rotation relation* $\hat{\boldsymbol{\rho}}^{(\ell)} = L_\ell(\hat{\mathbf{w}})$ for a public R-linear map L_ℓ. Every product has at most two packed factors, so a degree-2 multiplicative packing suffices.

Step 2: PIOP (Section 4). The arithmetic relations, i.e., the multiplications and linear relations, are polynomial identities over S, so a Spartan-style PIOP [Set20] proves them, using challenges from the Teichmüller set of S. The image, kernel, and rotation relations are only R-linear and are not naturally captured by polynomial identities over S, so we defer them to the PCS, which handles them natively (Step 3).

Step 3: PCS (Section 5). An interleaved-code PCS [AHIV17, GLS+23] carries over to S with minor changes: the prover encodes the rows of the evaluation matrix and commits to the columns, and the verifier checks row combinations against t opened columns. Two obstacles remain: lifting a code to a Galois ring usually needs a new distance analysis and an encoder that multiplies in S, and the PCS must natively check the image, kernel, and rotation relations deferred from Step 2.

We resolve both in a unified way. We lift any binary code to S in a black-box way by applying its generator over S; the lifted code keeps the rate and relative distance δ of the binary code, and its encoder never multiplies in S, so encoding stays efficient. Moreover, the lifted code respects the structure of S: it keeps its distance not only in the usual Hamming sense but also with respect to any *complemented* submodule $M \subseteq S$, i.e., one with $S = M \oplus M'$ for some submodule M'. Since both $\mathrm{Img}(\phi)$ and $\mathrm{Ker}(\psi)$ are such submodules, the *submodule check* (Section 5.2) only tests that each opened entry lies in the submodule: a single entry outside it spreads to a δ fraction of the columns, so the t opened columns catch it. The lifted code also respects R-linear maps on S: since its generator matrix has entries in R, encoding commutes with applying any R-linear map L to each coordinate. Hence the commitment to $\hat{\mathbf{w}}$ also serves as a commitment to $L(\hat{\mathbf{w}})$, as the verifier can apply L to the opened entries before the consistency test. With the rotation maps L_ℓ of Step 1, this *virtual PCS* (Section 5.3) handles the rotation relation without committing to the rotated witnesses.

3 Preliminaries

Notation. We write $[n] := \{0, 1, \ldots, n-1\}$ for an index set, and $a \leftarrow E$ for a uniform draw from a finite set E. For an integer $i \in [2^k]$, $\mathrm{bin}(i) \in \{0,1\}^k$ denotes its k-bit binary representation. Vectors are bold lowercase $(\boldsymbol{a}, \mathbf{w}, \ldots)$ with i-th coordinate a_i in italics, and matrices bold uppercase $(\mathbf{A}, \mathbf{B}, \mathbf{C})$; $\boldsymbol{a} \circ \boldsymbol{b}$ is the entrywise (Hadamard) product and $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ the inner product. We write λ for the security parameter and call a function *negligible*, denoted $\mathtt{negl}(\lambda)$, if it decreases faster than any inverse polynomial in λ; $\mathcal{P}$ and $\mathcal{V}$ denote the prover and verifier.

3.1 Galois Rings

We work over Galois rings, a family of finite rings containing both the prime-power residue rings $\mathbb{Z}_{p^k}$ and the finite fields $\mathbb{F}_{p^d}$ as special cases. We focus on the case of $p = 2$, though our techniques apply to any prime p. Unlike fields, Galois rings have zero divisors when $k \geq 2$, which is the central obstacle when using field-based soundness arguments. We therefore replace the notion of "distinct points" in field arguments with the *unit-separated set*, realized concretely via the *Teichmüller set* over Galois rings.

Definition 3.1 (Galois ring)**.** Let p be a prime and $k, d \geq 1$. Let $f \in \mathbb{Z}_{p^k}[X]$ be monic of degree d with $\bar{f} := f \bmod p$ irreducible in $\mathbb{F}_p[X]$. The *Galois ring* of characteristic p^k and rank d is

$$\mathrm{GR}(p^k, d) \;:=\; \mathbb{Z}_{p^k}[X] \,/\, (f(X)).$$

Proposition 3.2 (Structure of Galois rings)**.** *Let* $\mathrm{GR} = \mathrm{GR}(p^k, d)$*. Then* GR *is a finite commutative local ring with unique maximal ideal* $\mathfrak{m} = (p)$ *and residue field* $\mathrm{GR}/\mathfrak{m} \cong \mathbb{F}_{p^d}$*. The units are* $\mathrm{GR}^\times = \mathrm{GR} \setminus \mathfrak{m}$*. As special cases,* $\mathrm{GR}(p, d) \cong \mathbb{F}_{p^d}$ *and* $\mathrm{GR}(p^k, 1) \cong \mathbb{Z}_{p^k}$*.*

3.1.1 Unit-Separated Sets

In field-based zkSNARKs, soundness depends on the size of the field where the random challenges are sampled. Over rings, the unit-separated set E plays this role, so the soundness depends on $|E|$.

Definition 3.3 (Unit-separated set)**.** A finite subset $E = \{e_1, \ldots, e_N\} \subseteq \mathrm{GR}$ is *unit-separated* (also called an *exceptional set* or *sampling set*) if $e_i - e_j \in \mathrm{GR}^\times$ for all $i \neq j$.

This notion is vacuous for fields since every two distinct elements are unit-separated, but zero divisors of rings make this nontrivial. The canonical unit-separated set of a Galois ring is:

Definition 3.4 (Teichmüller set)**.** The *Teichmüller set* of $\mathrm{GR} = \mathrm{GR}(p^k, d)$ is $\mathcal{T} \;:=\; \{\, x \in \mathrm{GR} : x^{p^d} = x \,\}$.

Proposition 3.5 (Properties of the Teichmüller set)**.** *Let* $\mathrm{GR} = \mathrm{GR}(p^k, d)$*, and let* $\mathcal{T} \subset \mathrm{GR}$ *denote its Teichmüller set. Then:* (1) $\mathcal{T} = \{0\} \cup \mathcal{T}^*$*, where the cyclic group* $\mathcal{T}^* = \langle \omega \rangle = \{1, \omega, \ldots, \omega^{p^d - 2}\}$ *has order* $p^d - 1$*, so* $|\mathcal{T}| = p^d$*;* (2) *the residue map* $R \twoheadrightarrow \mathbb{F}_{p^d}$ *restricts to a bijection* $\mathcal{T} \xrightarrow{\sim} \mathbb{F}_{p^d}$*;* (3) $\mathcal{T}$ *is unit-separated, and every unit-separated subset of* R *has size at most* $|\mathcal{T}| = p^d$*.*

3.1.2 Polynomials over Galois Rings

Polynomials over a Galois ring behave much like those over a field, except that zero divisors can collapse degrees; for example, $(2X)^2 = 0$ in $\mathbb{Z}_4[X]$. We use two kinds of interpolation: *univariate* Lagrange interpolation, which builds the packing of Section 3.3, and the *multilinear* extension, the arithmetization behind sumcheck and the commitment scheme. Both, like the bound on how many roots a polynomial can have, rely on dividing by differences $e_i - e_j$ of the evaluation points, so those points must be *unit-separated* — then the differences are units.

We first introduce Lagrange interpolation of univariate polynomials over Galois rings, which is used to instantiate the multiplicative packing.

Proposition 3.6 (Lagrange interpolation)**.** *Let* $E = \{e_1, \ldots, e_N\} \subseteq \mathrm{GR}$ *be unit-separated. For any values* $a_1, \ldots, a_N \in \mathrm{GR}$ *there is a unique* $f(X) \in \mathrm{GR}[X]$ *of degree* $< N$ *with* $f(e_i) = a_i$*, namely* $f(X) = \sum_{i=1}^{N} a_i \, \ell_i(X)$*, where the* Lagrange basis $\ell_i(X) := \prod_{j \neq i} \frac{X - e_j}{e_i - e_j}$ *is well-defined, since each* $e_i - e_j$ *is a unit, and satisfies* $\ell_i(e_j) = \delta_{ij}$*.*

We then introduce the multilinear extension over the Boolean cube, used to arithmetize the statement for sumcheck and the commitment scheme. Since $\{0,1\}$ is unit-separated in any ring ($1-0 \in \mathrm{GR}^\times$), interpolation on the cube $\{0,1\}^\mu$ is unconditional; a polynomial is *multilinear* if it has degree at most 1 in each variable.

Proposition 3.7 (Multilinear extension)**.** *Every function $f : \{0,1\}^\mu \to R$ has a unique multilinear extension $\tilde{f} \in \mathrm{GR}[X_1, \ldots, X_\mu]$ agreeing with f on $\{0,1\}^\mu$, namely $\tilde{f}(\boldsymbol{x}) = \sum_{\boldsymbol{h}\in\{0,1\}^\mu} f(\boldsymbol{h})\,\mathrm{eq}(\boldsymbol{x},\boldsymbol{h})$, where the* equality polynomial $\mathrm{eq}(\boldsymbol{x},\boldsymbol{h}) := \prod_{i=1}^{\mu}\big(h_i x_i + (1-h_i)(1-x_i)\big)$ *is multilinear in $\boldsymbol{x}$ and equals $\mathbf{1}[\boldsymbol{x}=\boldsymbol{h}]$ on $\{0,1\}^\mu$.*

When the context is clear, we drop the tilde and write $a(X)$ for the multilinear extension of a vector $\boldsymbol{a}$, and $A(X,Y)$ for the bilinear extension of a matrix $\mathbf{A}$.

Lemma 3.8 (Schwartz–Zippel over rings)**.** *Let $E \subseteq \mathrm{GR}$ be unit-separated, and let $0 \neq f \in \mathrm{GR}[X_1, \ldots, X_\mu]$ have total degree at most $\deg(f)$. Then $\Pr_{\boldsymbol{a}\leftarrow E^\mu}\big[f(\boldsymbol{a}) = 0\big] \le \frac{\deg(f)}{|E|}$.*

3.1.3 Modules and Linear Codes over Rings

A *module* over the ring GR is the ring analogue of a vector space over a field: an abelian group on which the base ring GR acts as scalars. It is *free* of *rank* k if it has a basis of k elements, that is, $M \cong \mathrm{GR}^k$. Galois rings are free over their base: $\mathrm{GR}(p^k, d)$ is a free $\mathbb{Z}_{p^k}$-module of rank d, and $\mathrm{GR}(p^k, d')$ is free of rank d'/d over $\mathrm{GR}(p^k, d)$ whenever $d \mid d'$. A *submodule* of M is a subset closed under addition and the GR-action, the analogue of a subspace.

A submodule is complemented if it can be cleanly split off from the module as an independent direct-sum piece.

Definition 3.9 (Complemented submodule)**.** A GR-submodule V of a GR-module M is *complemented* (a *direct summand*) if $M = V \oplus W$ for some GR-submodule W; that is, every $x \in M$ decomposes uniquely as $x = v + w$ with $v \in V$ and $w \in W$.

A *linear code* over GR is the ring analogue of a linear code over a field — a subspace of $\mathbb{F}^n$ there, a GR-submodule $\mathcal{C} \subseteq \mathrm{GR}^n$ here, presented as the image $\mathcal{C} = \{\mathbf{G}\boldsymbol{m} : \boldsymbol{m} \in \mathrm{GR}^k\}$ of a *generator matrix* $\mathbf{G} \in \mathrm{GR}^{n\times k}$ acting on messages $\boldsymbol{m} \in \mathrm{GR}^k$. The *Hamming weight* $\Delta(\boldsymbol{c})$ counts the nonzero coordinates of $\boldsymbol{c}$, the *minimum distance* is $d_{\min}(\mathcal{C}) := \min_{\mathbf{0}\neq\boldsymbol{c}\in\mathcal{C}} \Delta(\boldsymbol{c})$, and with length n and message length k the *rate* is $\rho := k/n$ and the *relative distance* is $\delta := d_{\min}(\mathcal{C})/n$.

Code-based polynomial commitment schemes rest on the *proximity gap* of a linear code: a collection of vectors is either jointly close to the code, or a random linear combination of them is far from the code with high probability, with no middle ground.

Theorem 3.10 (Proximity gap over rings [WZD25a])**.** *Let $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_\ell \in R^n$ and $e \ge 0$. If the tuple $(\boldsymbol{u}_i)_{i\in[\ell]}$ is more than e-far from the interleaved code $\mathcal{C}^\ell$, that is, $\Delta^\ell\big((\boldsymbol{u}_i)_{i\in[\ell]}, \mathcal{C}^\ell\big) > e$, then*

$$\Pr_{\boldsymbol{r}\leftarrow E^\ell}\Big[\Delta\big(\textstyle\sum_{i\in[\ell]} r_i\,\boldsymbol{u}_i,\, \mathcal{C}\big) \le e\Big] \;\le\; \tfrac{e+1}{|E|}.$$

For general linear codes, e can be up to $\frac{\delta n}{3}$.

3.2 zkSNARKs

A *zkSNARK* for an NP relation $\mathcal{R}$ is a succinct non-interactive argument of knowledge: complete, knowledge-sound, succinct (proof size and verification sublinear in the witness), and zero-knowledge.

Definition 3.11 (Zero-knowledge succinct non-interactive argument of knowledge (zkSNARK))**.** Let $\mathcal{R}$ be a relation with public instance x and private witness w. A SNARK for $\mathcal{R}$ consists of PPT algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ with the following properties:

- Completeness. For any instance $(x, w) \in \mathcal{R}$

$$\Pr\left[\begin{array}{l|l} \mathcal{V}(x, \mathbf{vk}, \pi) = 1 & (\mathbf{pk}, \mathbf{vk}) \leftarrow \mathcal{G}(1^\lambda), \\ & \pi \leftarrow \mathcal{P}(x, w, \mathbf{pk}) \end{array}\right] = 1$$

- Knowledge soundness. For any PPT adversary $\mathcal{A}$, there exists an expected PPT knowledge extractor $\mathcal{E}_\mathcal{A}$ such that for any instance x, the following probability is $\leq \mathtt{negl}(\lambda)$:

$$\Pr\left[\begin{array}{l|c} \mathcal{V}(x, \mathbf{vk}, \pi^*) = 1 & (\mathbf{pk}, \mathbf{vk}) \leftarrow \mathcal{G}(1^\lambda), \\ \wedge (x, w) \notin \mathcal{R} & (\pi^*; w) \leftarrow (\mathcal{A} || \mathcal{E}_\mathcal{A})(x, \mathbf{pk}) \end{array}\right].$$

- Zero knowledge. There exists a PPT simulator $\mathcal{S}$ such that for any PPT algorithm $\mathcal{V}^*$, $(x, w) \in \mathcal{R}$, $\mathbf{pk}$ output by $\mathcal{G}(1^\lambda)$, it holds that

$$\Pr[\pi \leftarrow \mathcal{P}(x, w, \mathbf{pk}) : \mathcal{V}^*(x, \mathbf{vk}, \pi) = 1] \approx \Pr[\pi \leftarrow \mathcal{S}(x, \mathbf{pk}) : \mathcal{V}^*(x, \mathbf{vk}, \pi) = 1],$$

  where $\mathcal{P}$ and $\mathcal{V}^*$ query a random oracle H, which $\mathcal{S}$ may reprogram.

- Succinctness. The proof size $|\pi|$ is sublinear in the size of the relation $\mathcal{R}$ and the witness $|w|$.

The standard recipe builds one from two components: a PIOP, which proves the statement using polynomial oracles, and a PCS, which realizes those oracles. We target R1CS relations. We primarily focus on building a SNARK for an R1CS over a base ring R (for example, $R = \mathbb{Z}/2^k\mathbb{Z}$). The scheme can be made zero-knowledge via standard compilers [AHIV17, CFS17, XZZ+19].

3.2.1 R1CS over Rings

The rank-1 constraint system (R1CS) is the standard NP-complete relation, indexed by a triple of matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in R^{n \times n}$ that encodes the circuit structure:

$$\mathcal{R}_R^{\mathrm{R1CS}} := \{\, \mathbf{w} \in R^n \mid (\mathbf{A} \cdot \mathbf{w}) \circ (\mathbf{B} \cdot \mathbf{w}) = \mathbf{C} \cdot \mathbf{w} \,\},$$

where each row in $\mathbf{A}, \mathbf{B}, \mathbf{C}$ is sparse due to the local structure of the circuit. An R1CS relation splits into two sub-relations. The *multiplication relation* captures the constraints between the left input, right input, and output of every multiplication gate:

$$\mathcal{R}_R^{\mathrm{mult}} := \{\, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in R^n \mid \boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c} \,\},$$

and the *linear relation* connects $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ to the witness $\mathbf{w}$ via the addition gates encoded in $\mathbf{A}, \mathbf{B}, \mathbf{C}$:

$$\mathcal{R}_R^{\mathrm{lin}} := \{\, \mathbf{w}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in R^n \mid \boldsymbol{a} = \mathbf{A} \cdot \mathbf{w},\ \boldsymbol{b} = \mathbf{B} \cdot \mathbf{w},\ \boldsymbol{c} = \mathbf{C} \cdot \mathbf{w} \,\}.$$

3.2.2 Polynomial Interactive Oracle Proof over Rings

In a polynomial interactive oracle proof (PIOP), the prover sends polynomial *oracles* that the verifier queries at points of its choice.

The central PIOP is sumcheck: it reduces a claim about the sum of a low-degree polynomial over the Boolean hypercube to a single evaluation at a random point, and it transfers to rings once challenges are drawn from a unit-separated set.

Theorem 3.12 (Sumcheck over rings)**.** *Let $g \in \mathrm{GR}[X_1, \ldots, X_\mu]$ have degree at most d in each variable, and let $E \subseteq \mathrm{GR}$ be unit-separated. The sumcheck protocol reduces a claim about $\sum_{\boldsymbol{h} \in \{0,1\}^\mu} g(\boldsymbol{h})$ to a single evaluation $g(\boldsymbol{r})$ at a random $\boldsymbol{r} \leftarrow E^\mu$ in time $O(\mu d)$, with soundness error at most $\mu d / |E|$.*

In round i, the prover sends the univariate polynomial $g_i(X) := \sum_{\boldsymbol{h}\in\{0,1\}^{\mu-i}} g(r_1, \ldots, r_{i-1}, X, \boldsymbol{h})$; the verifier checks $g_i(0) + g_i(1)$ against the claimed sum when $i = 1$ and against $g_{i-1}(r_{i-1})$ otherwise, then replies with a fresh challenge $r_i \leftarrow E$. After μ rounds, one evaluation of g at $(r_1, \ldots, r_\mu)$ settles the claim. The soundness error is the field bound with $|E|$ in place of the field size: each round fails with probability at most $d/|E|$ by the Schwartz–Zippel lemma over rings (Lemma 3.8).

3.2.3 Polynomial Commitment Scheme over Rings

A *polynomial commitment scheme (PCS)* over a ring R is a tuple $(\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$: $\mathsf{Gen}(1^\lambda)$ outputs public parameters; Commit maps a multilinear polynomial $f \in R[X_1, \ldots, X_\mu]$, given by its 2^μ evaluations on $\{0,1\}^\mu$, to a short commitment cm; Open verifies that cm is a commitment to a claimed f; and Eval is a succinct argument that $f(\boldsymbol{z}) = y$ at a verifier-chosen point $\boldsymbol{z} \in R^\mu$. Its completeness, knowledge soundness and zero-knowledge are defined similarly to those of a zkSNARK, and we omit the formal definitions. We use a PCS based on interleaved codes, and the protocol over Galois rings is in Section 5.

3.3 Multiplicative Packing

Multiplicative packing ϕ, also known as a (degree-D) reverse multiplication-friendly embedding (RMFE), packs multiple elements of the small ring into a single element of the large ring while preserving multiplicative structure to some extent.

Definition 3.13 (Degree-D multiplicative packing)**.** Let p be a prime and let $k, \kappa, d_0, d_1, D \geq 1$ be integers. A pair (ϕ, ψ) is called a degree-D multiplicative packing from $\mathrm{GR}(p^k, d_0)$ to $\mathrm{GR}(p^k, d_1)$ with packing size κ, if the packing

$$\phi : \mathrm{GR}(p^k, d_0)^\kappa \to \mathrm{GR}(p^k, d_1)$$

and the unpacking $\psi : \mathrm{GR}(p^k, d_1) \to \mathrm{GR}(p^k, d_0)^\kappa$ are two $\mathrm{GR}(p^k, d_0)$-linear maps such that

$$\psi\big(\phi(\boldsymbol{x}_1) \cdot \phi(\boldsymbol{x}_2) \cdots \phi(\boldsymbol{x}_D)\big) = \boldsymbol{x}_1 \circ \boldsymbol{x}_2 \circ \cdots \circ \boldsymbol{x}_D,$$

for all $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_D \in \mathrm{GR}(p^k, d_0)^\kappa$. Additionally, we require $\phi(\mathbf{1}) = 1$.

The definition of degree-D multiplicative packing can be extended to general algebraic objects, though we specify it over Galois rings for simplicity. Throughout the subsection, $R := \mathrm{GR}(p^k, d_0)$ denotes the small ring in which the witness in the original relation lives and $S := \mathrm{GR}(p^k, d_1)$ the large ring in which the packed witness in the reduced relation lives. We adopt the convention that hatted symbols denote elements of S and unhatted symbols denote elements of R; that is, $\phi : R^\kappa \to S$ maps $\boldsymbol{x} \mapsto \hat{x}$.

Moreover, the image of the packing and the kernel of the unpacking are complemented submodules:

Lemma 3.14 ([LXYY23])**.** *Let (ϕ, ψ) be a degree-D multiplicative packing from $\mathrm{GR}(p^k, d_0)$ to $\mathrm{GR}(p^k, d_1)$ with packing size κ. Then $\mathrm{GR}(p^k, d_1)$ is the direct sum of $\mathrm{Ker}(\psi)$ and $\mathrm{Img}(\phi)$. That is, $\mathrm{GR}(p^k, d_1) = \mathrm{Ker}(\psi) \oplus \mathrm{Img}(\phi)$.*

The packing rate $r_\phi := \frac{\kappa(d_0+1)}{d_1+1}$ characterizes the compactness of the packing, which can be asymptotically constant, as captured by the following lemma:

Lemma 3.15 ([EHL+23])**.** *There exists a family of degree-D multiplicative packings from $\mathbb{Z}_{2^k} = \mathrm{GR}(2^k, d_0 = 1)$ to $\mathrm{GR}(2^k, d_1)$ with packing size κ such that, for all $k \geq 1$, $r_\phi \to C_D$ as $\kappa \to \infty$, where C_D is a constant depending on the degree D.*

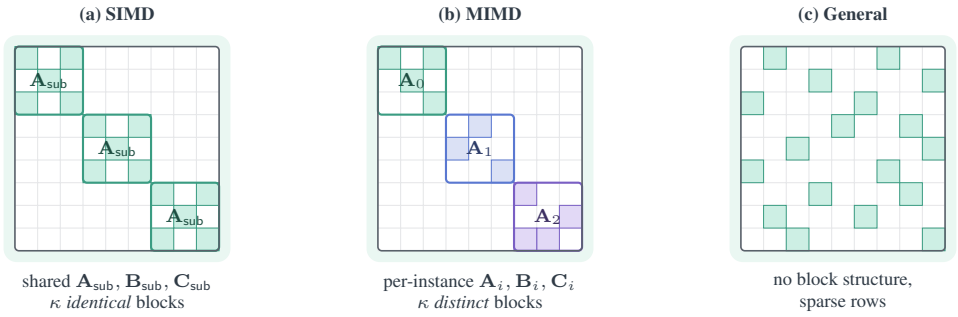


Figure 3: Sparsity structure of the R1CS matrix across the three computation models with packing size $\kappa = 3$.

3.3.1 Reduction of the Multiplication Relation

We typically consider only the degree-2 multiplicative packing, which we argue is the most efficient one for designing zkSNARKs. Assuming a degree-2 multiplicative packing (ϕ, ψ) with packing size κ and assuming $\kappa \mid n$, the multiplication relation, which is defined by the element-wise product:

$$\mathcal{R}_R^{\text{mult}} := \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in R^n \mid \boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c}\}$$

reduces to the conjunction of three relations over S: a multiplication relation with error, an image membership, and a kernel membership.

$$\mathcal{R}_S^{\text{mult}} := \left\{ \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{e}} \in S^{n/\kappa} \;\middle|\; \hat{\boldsymbol{a}} \circ \hat{\boldsymbol{b}} = \hat{\boldsymbol{c}} + \hat{\boldsymbol{e}} \right\},$$

$$\mathcal{R}_S^{\text{img}} := \left\{ \hat{\boldsymbol{x}} \in S^{n/\kappa} \;\middle|\; \hat{\boldsymbol{x}} \in \text{Img}(\phi)^{n/\kappa} \right\},$$

$$\mathcal{R}_S^{\text{ker}} := \left\{ \hat{\boldsymbol{e}} \in S^{n/\kappa} \;\middle|\; \hat{\boldsymbol{e}} \in \text{Ker}(\psi)^{n/\kappa} \right\}.$$

The full reduction results in $\mathcal{R}_S^{\text{mult}}$ together with $\mathcal{R}_S^{\text{img}}$ applied to each of $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}$ and $\mathcal{R}_S^{\text{ker}}$ applied to $\hat{\boldsymbol{e}}$.

4 Reducing R1CS Relations via Multiplicative Packing and PIOPs

Our SNARK schemes are constructed in three steps: (1) first, we reduce an R1CS relation over a small ring R to constraints over a large ring S via the multiplicative packing; (2) second, we build a PIOP over S to check the constraints; (3) finally, we compile the PIOP to a SNARK via a PCS over S, which also handles additional checks such as image and kernel checks on the committed witnesses.

We present the first two steps in this section, and the PCS in the next section. We consider three types of computations: SIMD, MIMD and general computations. The shapes of their R1CS matrices are illustrated in Figure 3. Throughout the paper, κ denotes the packing parameter of the multiplicative packing $\phi : R^\kappa \to S$, the number of small-ring elements packed into one large-ring element.

4.1 SIMD Circuits

The SIMD (*Single Instruction, Multiple Data*) R1CS relation captures κ identical sub-circuits evaluated in parallel, where κ is the degree of parallelism. The whole data-parallel circuit is an R1CS instance with block-diagonal matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in R^{n \times n}$, where $\mathbf{A} = \text{diag}(\mathbf{A}_{\mathsf{sub}}, \ldots, \mathbf{A}_{\mathsf{sub}})$

for a sub-circuit matrix $\mathbf{A}_{\mathsf{sub}} \in R^{m\times m}$ with $m := n/\kappa$ (likewise $\mathbf{B}, \mathbf{C}$). The witness $\mathbf{w} \in R^n$ splits into κ sub-witnesses $\mathbf{w}_i \in R^m$. Equivalently, we can write it as

$$\mathcal{R}_R^{\text{SIMD}} := \left\{ \{\mathbf{w}_i\} \subset R^m \;\middle|\; (\mathbf{A}_{\mathsf{sub}}\mathbf{w}_i) \circ (\mathbf{B}_{\mathsf{sub}}\mathbf{w}_i) = \mathbf{C}_{\mathsf{sub}}\mathbf{w}_i \right\}$$

which fixes shared sub-circuit matrices $\mathbf{A}_{\mathsf{sub}}, \mathbf{B}_{\mathsf{sub}}, \mathbf{C}_{\mathsf{sub}} \in R^{m\times m}$ across the κ different witnesses $\{\mathbf{w}_i\}_{i\in[\kappa]}$.

Note that we use the same parameter κ as the multiplicative packing parameter. However, our protocol can handle any number of sub-circuits. First, if the number is not a multiple of κ, we pad it to the nearest multiple, $\tau\kappa$, with dummy sub-circuits and witnesses. We then aggregate τ sub-circuits into one, resulting in κ sub-circuits in total with matrices $\mathbf{A}_{\mathsf{sub}}, \mathbf{B}_{\mathsf{sub}}, \mathbf{C}_{\mathsf{sub}}$ above. Our reduction only utilizes the parallelism of κ sub-circuits, but not their internal structures anymore. That is, any data-parallel circuit with $\tau\kappa$ sub-circuits can be trivially viewed as one with κ larger sub-circuits. The same convention applies to MIMD circuits.

4.1.1 Reduction of SIMD Circuits

Assuming the multiplicative packing $\phi : R^\kappa \to S$, we pack the κ sub-witnesses index-wise: for each index $j \in [m]$, we pack the elements $w_{i,j}$ at the same index across the κ sub-witnesses $\mathbf{w}_i$ into one large-ring element $\hat{w}_j := \phi(w_{0,j}, \ldots, w_{\kappa-1,j})$, resulting in the packed witness $\hat{\mathbf{w}} \in S^m$. We then reduce the multiplication and linear sub-relations in turn.

Multiplication relation. Let $\boldsymbol{a}_i := \mathbf{A}_{\mathsf{sub}} \cdot \mathbf{w}_i$, $\boldsymbol{b}_i := \mathbf{B}_{\mathsf{sub}} \cdot \mathbf{w}_i$, $\boldsymbol{c}_i := \mathbf{C}_{\mathsf{sub}} \cdot \mathbf{w}_i \in R^m$ for each $i \in [\kappa]$; the multiplication relation in $\mathcal{R}_R^{\text{SIMD}}$ is defined as $\boldsymbol{a}_i \circ \boldsymbol{b}_i = \boldsymbol{c}_i$ for $i \in [\kappa]$. We pack them index-wise into $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}} \in S^m$ in the same way we packed $\hat{\mathbf{w}}$. The elements at the same index across different instances form an element-wise product of length κ, $\boldsymbol{a}_{\cdot,j} \circ \boldsymbol{b}_{\cdot,j} = \boldsymbol{c}_{\cdot,j}$, which can be packed into a single large-ring product $\hat{a}_j \cdot \hat{b}_j = \hat{c}_j + \hat{e}_j^{\text{mult}}$, image membership $\hat{a}_j, \hat{b}_j, \hat{c}_j \in \text{Img}(\phi)$ and kernel membership $\hat{e}_j^{\text{mult}} \in \text{Ker}(\psi)$ via the reduction in Section 3.3. Applying it to all indices $j \in [m]$ yields

$$\hat{\boldsymbol{a}} \circ \hat{\boldsymbol{b}} = \hat{\boldsymbol{c}} + \hat{\boldsymbol{e}}^{\text{mult}} \text{ in } S^m.$$

This introduces a single auxiliary witness $\hat{\boldsymbol{e}}^{\text{mult}}$ and its kernel membership $\hat{\boldsymbol{e}}^{\text{mult}} \in \mathcal{R}_S^{\text{ker}}$. The image membership $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}} \in \mathcal{R}_S^{\text{img}}$ comes for free from the linear relation below.

Linear relation. Since ϕ is R-linear and $\mathbf{A}_{\mathsf{sub}}, \mathbf{B}_{\mathsf{sub}}, \mathbf{C}_{\mathsf{sub}}$ have entries in R, packing commutes with the matrices: $\hat{\boldsymbol{a}} = \mathbf{A}_{\mathsf{sub}} \cdot \hat{\mathbf{w}}, \hat{\boldsymbol{b}} = \mathbf{B}_{\mathsf{sub}} \cdot \hat{\mathbf{w}}, \hat{\boldsymbol{c}} = \mathbf{C}_{\mathsf{sub}} \cdot \hat{\mathbf{w}}$. No auxiliary witness is introduced in this step, but it requires one additional image membership $\hat{\mathbf{w}} \in \mathcal{R}_S^{\text{img}}$, which implies the image membership for $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}$ left open above.

4.1.2 PIOP for SIMD Circuits

Following the Spartan-style PIOP [Set20], we encode the length-m vectors as multilinear extensions over $\{0,1\}^\mu$ (so $m = 2^\mu$) and apply the sumcheck protocol to reduce the multiplication and linear relations to PCS queries. For the image membership of $\hat{\mathbf{w}}$ and the kernel membership of $\hat{\boldsymbol{e}}^{\text{mult}}$, we defer the checks to the PCS.

Protocol 4.1: PIOP for $\mathcal{R}_R^{\text{SIMD}}$

Public input: sub-circuit matrices $\mathbf{A}_{\mathsf{sub}}, \mathbf{B}_{\mathsf{sub}}, \mathbf{C}_{\mathsf{sub}} \in R^{m\times m}$, $m = n/\kappa$.
Witness: $\mathbf{w} \in R^n$.

1. *Reduction.* $\mathcal{P}$ computes $\hat{\mathbf{w}}$ and $\hat{\boldsymbol{e}}^{\text{mult}}$ and sends their polynomial oracles to $\mathcal{V}$.
2. *Multiplication check.* $\mathcal{V}$ sends a challenge $\boldsymbol{\tau} \leftarrow E^\mu$, and $\mathcal{P}$ and $\mathcal{V}$ run a degree-3 sumcheck (Theorem 3.12) on

$$g_1(x) = \text{eq}(\boldsymbol{\tau}, x)\big(\hat{a}(x)\hat{b}(x) - \hat{c}(x) - \hat{e}^{\text{mult}}(x)\big),$$

to reduce the claimed sum $\sum_x g_1(x) = 0$ to evaluations of $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{e}}^{\text{mult}}$ at $\boldsymbol{r}_x$, where $\boldsymbol{r}_x \leftarrow E^\mu$. $\mathcal{P}$ sends v_a, v_b, v_c as such claimed evaluations.

3. *Linear check.* $\mathcal{V}$ sends random batching challenges $\gamma_A, \gamma_B, \gamma_C \leftarrow E$, and $\mathcal{P}$ and $\mathcal{V}$ run a degree-2 sumcheck on

$$g_2(y) = \big(\gamma_A A_{\mathsf{sub}}(\boldsymbol{r}_x, y) + \gamma_B B_{\mathsf{sub}}(\boldsymbol{r}_x, y) + \gamma_C C_{\mathsf{sub}}(\boldsymbol{r}_x, y)\big)\, \hat{w}(y),$$

to reduce the claimed sum $\sum_y g_2(y) = \gamma_A v_a + \gamma_B v_b + \gamma_C v_c$ to evaluations of $\mathbf{A}_{\mathsf{sub}}, \mathbf{B}_{\mathsf{sub}}, \mathbf{C}_{\mathsf{sub}}$ at $(\boldsymbol{r}_x, \boldsymbol{r}_y)$ and $\hat{\mathbf{w}}$ at $\boldsymbol{r}_y$, where $\boldsymbol{r}_y \leftarrow E^\mu$.

4. *PCS queries.* $\mathcal{V}$ queries $\hat{e}^{\text{mult}}(\boldsymbol{r}_x)$ and $\hat{w}(\boldsymbol{r}_y)$ and evaluates $\mathbf{A}_{\mathsf{sub}}, \mathbf{B}_{\mathsf{sub}}, \mathbf{C}_{\mathsf{sub}}$ at $(\boldsymbol{r}_x, \boldsymbol{r}_y)$ to check whether the claimed evaluations are correct. The queries also check $\hat{\mathbf{w}} \in \mathrm{Img}(\phi)^m$ and $\hat{\boldsymbol{e}}^{\text{mult}} \in \mathrm{Ker}(\psi)^m$ by the free submodule check of Section 5.2.

Cost summary. The prover sends only two oracles, namely $\hat{\mathbf{w}}$ and the error $\hat{\boldsymbol{e}}^{\text{mult}}$ of size n/κ, then runs a degree-3 sumcheck for the multiplication check and a degree-2 sumcheck for the linear check, each over μ rounds, and answers the PCS openings. The verifier checks the two sumchecks and the PCS queries, evaluating the shared matrices $\mathbf{A}_{\mathsf{sub}}, \mathbf{B}_{\mathsf{sub}}, \mathbf{C}_{\mathsf{sub}}$ at $(\boldsymbol{r}_x, \boldsymbol{r}_y)$. Because the sub-circuits are identical, the matrices $\mathbf{A}_{\mathsf{sub}}, \mathbf{B}_{\mathsf{sub}}, \mathbf{C}_{\mathsf{sub}}$ are not packed via the multiplicative packing, so they need no preprocessing and the image membership of $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}$ comes for free, which makes SIMD the most efficient of the three variants.

4.2 MIMD Circuits

The MIMD (*Multiple Instruction, Multiple Data*) R1CS relation captures κ independent sub-circuits evaluated in parallel, where κ is the degree of parallelism. The whole circuit is an R1CS instance with block-diagonal matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in R^{n \times n}$, where $\mathbf{A} = \mathrm{diag}(\mathbf{A}_0, \ldots, \mathbf{A}_{\kappa-1})$ for distinct sub-circuit matrices $\mathbf{A}_i \in R^{m \times m}$ with $m := n/\kappa$ (likewise $\mathbf{B}, \mathbf{C}$). The witness $\mathbf{w} \in R^n$ splits into κ sub-witnesses $\mathbf{w}_i \in R^m$. Equivalently, we can write it as

$$\mathcal{R}_R^{\text{MIMD}} := \left\{ \{\mathbf{w}_i\} \subset R^m \;\middle|\; (\mathbf{A}_i \mathbf{w}_i) \circ (\mathbf{B}_i \mathbf{w}_i) = \mathbf{C}_i \mathbf{w}_i \right\}$$

for every instance $i \in [\kappa]$. Unlike SIMD, the sub-circuit matrices $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i$ differ across instances.

4.2.1 Reduction of MIMD Circuits

Using the multiplicative packing $\phi : R^\kappa \to S$, we pack the κ sub-witnesses index-wise exactly as in SIMD: for each index $j \in [m]$, $\hat{w}_j := \phi(w_{0,j}, \ldots, w_{\kappa-1,j})$, resulting in $\hat{\mathbf{w}} \in S^m$. Since the sub-circuits differ, we also pack the matrices index-wise: for each entry (j,k), $\hat{A}_{jk} := \phi\big((\mathbf{A}_0)_{jk}, \ldots, (\mathbf{A}_{\kappa-1})_{jk}\big) \in S$, giving $\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}} \in S^{m \times m}$. These are deterministic in the public matrices and precomputed, and they remain sparse: $\hat{A}_{jk} \neq 0$ only when $(\mathbf{A}_i)_{jk} \neq 0$ for some instance, so each row inherits the sparsity of the sub-circuits. We reduce the multiplication and linear sub-relations in turn.

Multiplication relation. The multiplication relation is identical to the SIMD case, as it depends only on the products $\boldsymbol{a}_i \circ \boldsymbol{b}_i = \boldsymbol{c}_i$, not on the matrices. With $\boldsymbol{a}_i := \mathbf{A}_i \mathbf{w}_i$, $\boldsymbol{b}_i := \mathbf{B}_i \mathbf{w}_i$, $\boldsymbol{c}_i := \mathbf{C}_i \mathbf{w}_i \in R^m$ packed index-wise into $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}} \in S^m$, collecting the per-index packings of Section 3.3 yields $\hat{\boldsymbol{a}} \circ \hat{\boldsymbol{b}} = \hat{\boldsymbol{c}} + \hat{\boldsymbol{e}}^{\text{mult}}$ in S^m. This introduces a single auxiliary witness $\hat{\boldsymbol{e}}^{\text{mult}}$ with kernel membership $\hat{\boldsymbol{e}}^{\text{mult}} \in \mathcal{R}_S^{\text{ker}}$.

Linear relation. Unlike SIMD, the packed matrices $\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}$ have entries in S, so each product $\hat{\mathbf{A}}\hat{\mathbf{w}}$ is a packed-times-packed product and incurs an error. Applying the reduction of Section 3.3

at each entry and summing over the inner index,

$$\hat{\mathbf{A}}\hat{\mathbf{w}} = \hat{\boldsymbol{a}} + \hat{\boldsymbol{e}}^A, \qquad \hat{\mathbf{B}}\hat{\mathbf{w}} = \hat{\boldsymbol{b}} + \hat{\boldsymbol{e}}^B, \qquad \hat{\mathbf{C}}\hat{\mathbf{w}} = \hat{\boldsymbol{c}} + \hat{\boldsymbol{e}}^C,$$

with $\hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C \in \mathcal{R}_S^{\ker}$. The prover therefore commits to the packed vectors $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}$ and the three auxiliary errors $\hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C$. The image membership $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}} \in \mathcal{R}_S^{\text{img}}$ is no longer free as in SIMD, because the matrices act over S rather than R; it must be checked alongside $\hat{\mathbf{w}} \in \mathcal{R}_S^{\text{img}}$.

4.2.2 PIOP for MIMD Circuits

The PIOP for MIMD circuits mirrors the SIMD one, with the linear check now over the packed matrices $\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}$ and three additional error oracles for $\hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C$ and three intermediate trace oracles $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}$ with corresponding membership checks. The full description is given in Protocol 4.2.

Protocol 4.2: PIOP for $\mathcal{R}_R^{\text{MIMD}}$

Public input: sub-circuit matrices $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i \in R^{m\times m}$ for $i \in [\kappa]$ (equivalently the packed $\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}} \in S^{m\times m}$), $m = n/\kappa$.
Witness: $\mathbf{w} \in R^n$.

1. *Reduction.* $\mathcal{P}$ computes $\hat{\mathbf{w}}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}$ and the errors $\hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C, \hat{\boldsymbol{e}}^{\text{mult}}$, and sends their polynomial commitments to $\mathcal{V}$.
2. *Multiplication check.* $\mathcal{V}$ sends a challenge $\boldsymbol{\tau} \leftarrow E^\mu$, and $\mathcal{P}$ and $\mathcal{V}$ run a degree-3 sumcheck (Theorem 3.12) on
$$g_1(\boldsymbol{x}) = \mathrm{eq}(\boldsymbol{\tau}, \boldsymbol{x})\big(\hat{a}(\boldsymbol{x})\hat{b}(\boldsymbol{x}) - \hat{c}(\boldsymbol{x}) - \hat{e}^{\text{mult}}(\boldsymbol{x})\big),$$
reducing the claimed sum $\sum_{\boldsymbol{x}} g_1(\boldsymbol{x}) = 0$ to a random $\boldsymbol{r}_x \leftarrow E^\mu$.
3. *Linear check.* $\mathcal{V}$ sends batching challenges $\gamma_A, \gamma_B, \gamma_C \leftarrow E$, and $\mathcal{P}$ and $\mathcal{V}$ run a degree-2 sumcheck on
$$g_2(\boldsymbol{y}) = \big(\gamma_A \hat{A}(\boldsymbol{r}_x, \boldsymbol{y}) + \gamma_B \hat{B}(\boldsymbol{r}_x, \boldsymbol{y}) + \gamma_C \hat{C}(\boldsymbol{r}_x, \boldsymbol{y})\big)\, \hat{w}(\boldsymbol{y}),$$
reducing the claimed sum $\sum_{\boldsymbol{y}} g_2(\boldsymbol{y}) = \gamma_A(\hat{a} + \hat{e}^A)(\boldsymbol{r}_x) + \gamma_B(\hat{b} + \hat{e}^B)(\boldsymbol{r}_x) + \gamma_C(\hat{c} + \hat{e}^C)(\boldsymbol{r}_x)$ to a random $\boldsymbol{r}_y \leftarrow E^\mu$.
4. *PCS queries.* $\mathcal{V}$ queries $\hat{a}, \hat{b}, \hat{c}, \hat{e}^A, \hat{e}^B, \hat{e}^C, \hat{e}^{\text{mult}}$ at $\boldsymbol{r}_x$ and $\hat{\mathbf{w}}$ at $\boldsymbol{r}_y$, evaluates $\hat{A}, \hat{B}, \hat{C}$ at $(\boldsymbol{r}_x, \boldsymbol{r}_y)$, and checks the two sumcheck claims. The queries also check $\hat{\mathbf{w}}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}} \in \mathrm{Img}(\phi)^m$ and $\hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C, \hat{\boldsymbol{e}}^{\text{mult}} \in \mathrm{Ker}(\psi)^m$ by the free submodule check of Section 5.2.

Cost summary. The prover sends eight oracles, namely $\hat{\mathbf{w}}$, $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}$, and the four errors $\hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C, \hat{\boldsymbol{e}}^{\text{mult}}$, all of size n/κ, then runs a degree-3 sumcheck for the multiplication check and a degree-2 sumcheck for the linear check, each over μ rounds, and answers the PCS openings. The verifier checks the two sumchecks and the PCS queries, evaluating $\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}$ at $(\boldsymbol{r}_x, \boldsymbol{r}_y)$. The packed matrices $\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}} \in S^{m\times m}$ are computed once in preprocessing by packing the public sub-circuit matrices index-wise.

4.3 General Circuits

The general R1CS relation $\mathcal{R}_R^{\text{R1CS}}$ has no parallel structure to exploit. We use the diagonal-order trick of Halevi–Shoup [HS14] to impose such structure, at the cost of κ additional rotated witnesses. This cost stays lightweight because we show that the rotated witnesses need not be

committed as their openings are derived virtually from the witness itself. Once the relation is cast in parallel form, the reduction of the element-wise product relation applies as before.

4.3.1 Reduction of General Circuits

Since the circuit has no native parallelism, κ now plays the role of a *block size*. Assuming $\kappa \mid n$ and writing $m := n/\kappa$, we divide the witness into m blocks of length κ and pack each block into one large-ring element, whereas in SIMD and MIMD we divide the witness into κ blocks of length m and pack the elements of the same index across different blocks. More specifically, writing the j-th block as $\mathbf{w}^{(j)} := (w_{j\kappa}, \ldots, w_{(j+1)\kappa-1}) \in R^\kappa$ for $j \in [m]$, we pack it as $\hat{w}_j := \phi(\mathbf{w}^{(j)}) \in S$, which produces a packed witness $\hat{\mathbf{w}} \in S^m$ exactly as in SIMD and MIMD but from a different grouping. This consecutive grouping comes at a price: the linear relation no longer commutes with packing, which we resolve with the diagonal-order trick of Halevi–Shoup [HS14]. We reduce the multiplication and linear sub-relations in turn.

Multiplication relation. Let $\boldsymbol{a} := \mathbf{A}\mathbf{w}$, $\boldsymbol{b} := \mathbf{B}\mathbf{w}$, $\boldsymbol{c} := \mathbf{C}\mathbf{w} \in R^n$. We pack them into $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}} \in S^m$ the same way we pack $\hat{\mathbf{w}}$, by grouping consecutive blocks together, which reduces $\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c}$ to $\hat{\boldsymbol{a}} \circ \hat{\boldsymbol{b}} = \hat{\boldsymbol{c}} + \hat{\boldsymbol{e}}^{\text{mult}}$ in S^m with $\hat{\boldsymbol{e}}^{\text{mult}} \in \mathcal{R}_S^{\text{ker}}$. Though the packing is different, this is still essentially the same as SIMD and MIMD since the multiplication relation is fully parallelizable.

Diagonal trick. For $\boldsymbol{x} \in R^\kappa$ and $\ell \in [\kappa]$, define the cyclic rotation $\text{rot}_\ell(\boldsymbol{x})_i := x_{(i+\ell) \bmod \kappa}$. For a $\kappa \times \kappa$ matrix $\mathbf{M}$, its ℓ-th cyclic diagonal $\boldsymbol{d}^{(\ell)} \in R^\kappa$ is $d_i^{(\ell)} := M_{i,\,(i+\ell) \bmod \kappa}$, and

$$\mathbf{M}\boldsymbol{x} = \sum\nolimits_{\ell \in [\kappa]} \boldsymbol{d}^{(\ell)} \circ \text{rot}_\ell(\boldsymbol{x}) \quad \text{in } R^\kappa, \tag{1}$$

since $(\mathbf{M}\boldsymbol{x})_i = \sum_{i' \in [\kappa]} M_{i,i'} x_{i'}$ and reindexing $i' = (i + \ell) \bmod \kappa$ rewrites the sum as $\sum_{\ell \in [\kappa]} d_i^{(\ell)} \cdot \text{rot}_\ell(\boldsymbol{x})_i$. The decomposition replaces κ^2 scalar products with κ length-κ element-wise products.

Linear relation. Partition $\mathbf{A} \in R^{n \times n}$ into a grid of blocks $\mathbf{A}^{(j,k)} \in R^{\kappa \times \kappa}$ for $j, k \in [m]$ with $A^{(j,k)}_{i,i'} := A_{j\kappa+i,\, k\kappa+i'}$, matching the block partition of $\mathbf{w}$. The j-th block-row of the product is then a sum of small block matrix-vector products $(\mathbf{A}\mathbf{w})_j = \sum_{k \in [m]} \mathbf{A}^{(j,k)} \mathbf{w}^{(k)}$, each a $\kappa \times \kappa$ block times a length-κ vector. We handle each block with the diagonal trick (1), which rewrites $\mathbf{A}^{(j,k)} \mathbf{w}^{(k)}$ as a sum over the κ diagonals of element-wise products $\boldsymbol{d}^{(\ell,j,k)} \circ \text{rot}_\ell(\mathbf{w}^{(k)})$, and then we pack each element-wise product into a single large-ring product plus an error using the reduction of Section 3.3.

For each diagonal index $\ell \in [\kappa]$, we pack the block diagonals into a *packed diagonal matrix* $\hat{\mathbf{D}}_A^{(\ell)} \in S^{m \times m}$ where $(\hat{\mathbf{D}}_A^{(\ell)})_{j,k} := \phi(\boldsymbol{d}^{(\ell,j,k)})$, and also pack the rotated witness into $\hat{\boldsymbol{\rho}}^{(\ell)} \in S^m$ where $(\hat{\boldsymbol{\rho}}^{(\ell)})_k := \phi(\text{rot}_\ell(\mathbf{w}^{(k)}))$. The linear relation then collapses to

$$\sum\nolimits_{\ell \in [\kappa]} \hat{\mathbf{D}}_A^{(\ell)} \cdot \hat{\boldsymbol{\rho}}^{(\ell)} = \hat{\boldsymbol{a}} + \hat{\boldsymbol{e}}^A \quad \text{in } S^m,$$

with $\hat{\boldsymbol{e}}^A \in \mathcal{R}_S^{\text{ker}}$. In effect, the original $n \times n$ matrix-vector product shrinks to an $m \times m$ one over the large ring S similar to MIMD, but at the price of an outer sum over κ such rotated matrices and witnesses. The similarly packed $\hat{\mathbf{D}}_B^{(\ell)}, \hat{\mathbf{D}}_C^{(\ell)}$ and the *same* rotation vectors $\hat{\boldsymbol{\rho}}^{(\ell)}$ give $\hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}$ with errors $\hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C$.

The packed diagonals are public and *sparse* since each nonzero of $\mathbf{A}$ lands in exactly one diagonal entry, resulting in the same sparsity.

Rotation relation. The rotation vectors must be the genuine block-wise rotations of $\hat{\mathbf{w}}$:

$$\mathcal{R}_S^{\text{rot}_\ell} := \left\{ (\hat{x}, \hat{y}) \in S \times S \;\middle|\; \hat{y} = \phi(\text{rot}_\ell(\psi(\hat{x}))) \right\},$$

applied to $(\hat{w}_k, (\hat{\boldsymbol{\rho}}^{(\ell)})_k)$ for every $k \in [m]$ and $\ell \in [\kappa]$.

4.3.2 PIOP for General Circuits

Following the Spartan-style PIOP, we encode the length-m vectors as multilinear extensions over $\{0,1\}^\mu$ (so $m = 2^\mu$) and apply the sumcheck protocol. As in MIMD, the image checks on $\hat{\mathbf{w}}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}$ and the kernel checks on $\hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C, \hat{\boldsymbol{e}}^{\text{mult}}$ are deferred to the PCS. Moreover, the polynomial oracle of each rotated witness $\hat{\boldsymbol{\rho}}^{(\ell)}$ is derived virtually from the oracle of the original witness $\hat{\mathbf{w}}$, so the prover never commits to any of them.

Protocol 4.3: PIOP for $\mathcal{R}_R^{\mathbf{R1CS}}$ (general circuit)

Public input: sparse packed diagonals $\hat{\mathbf{D}}_A^{(\ell)}, \hat{\mathbf{D}}_B^{(\ell)}, \hat{\mathbf{D}}_C^{(\ell)} \in S^{m\times m}$ for $\ell \in [\kappa]$, $m = n/\kappa$.
Witness: $\mathbf{w} \in R^n$.

1. *Reduction.* $\mathcal{P}$ computes $\hat{\mathbf{w}}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}$ and the errors $\hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C, \hat{\boldsymbol{e}}^{\text{mult}}$ and sends their polynomial oracles to $\mathcal{V}$; the rotation vectors $\hat{\boldsymbol{\rho}}^{(\ell)}$ are virtual, derived from $\hat{\mathbf{w}}$.
2. *Multiplication check.* $\mathcal{V}$ sends a challenge $\boldsymbol{\tau} \leftarrow E^\mu$, and $\mathcal{P}$ and $\mathcal{V}$ run a degree-3 sumcheck (Theorem 3.12) on
$$g_1(x) = \mathrm{eq}(\boldsymbol{\tau}, x)\big(\hat{a}(x)\hat{b}(x) - \hat{c}(x) - \hat{e}^{\text{mult}}(x)\big),$$
to reduce the claimed sum $\sum_x g_1(x) = 0$ to evaluations of $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{e}}^{\text{mult}}$ at $\boldsymbol{r}_x$, where $\boldsymbol{r}_x \leftarrow E^\mu$.
3. *Linear check.* $\mathcal{V}$ sends batching challenges $\gamma_A, \gamma_B, \gamma_C \leftarrow E$, and $\mathcal{P}$ and $\mathcal{V}$ run a degree-2 sumcheck on
$$g_2(y) = \sum\nolimits_{\ell\in[\kappa]} \big(\gamma_A \hat{D}_A^{(\ell)} + \gamma_B \hat{D}_B^{(\ell)} + \gamma_C \hat{D}_C^{(\ell)}\big)(\boldsymbol{r}_x, y)\, \hat{\rho}^{(\ell)}(y),$$
to reduce the claimed sum $\sum_y g_2(y) = \gamma_A(\hat{\boldsymbol{a}} + \hat{\boldsymbol{e}}^A)(\boldsymbol{r}_x) + \gamma_B(\hat{\boldsymbol{b}} + \hat{\boldsymbol{e}}^B)(\boldsymbol{r}_x) + \gamma_C(\hat{\boldsymbol{c}} + \hat{\boldsymbol{e}}^C)(\boldsymbol{r}_x)$ to evaluations of $\hat{\mathbf{D}}_A^{(\ell)}, \hat{\mathbf{D}}_B^{(\ell)}, \hat{\mathbf{D}}_C^{(\ell)}$ at $(\boldsymbol{r}_x, \boldsymbol{r}_y)$ and $\hat{\boldsymbol{\rho}}^{(\ell)}$ at $\boldsymbol{r}_y$, where $\boldsymbol{r}_y \leftarrow E^\mu$.
4. *PCS queries.* $\mathcal{V}$ queries $\hat{w}(\boldsymbol{r}_y)$ and queries each rotated $\hat{\rho}^{(\ell)}(\boldsymbol{r}_y)$ virtually, which also discharges the rotation relation; queries $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}}, \hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C, \hat{\boldsymbol{e}}^{\text{mult}}$ at $\boldsymbol{r}_x$; evaluates the sparse diagonals $\hat{\mathbf{D}}_A^{(\ell)}, \hat{\mathbf{D}}_B^{(\ell)}, \hat{\mathbf{D}}_C^{(\ell)}$ at $(\boldsymbol{r}_x, \boldsymbol{r}_y)$. The queries also check $\hat{\mathbf{w}}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}} \in \mathrm{Img}(\phi)^m$ and $\hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C, \hat{\boldsymbol{e}}^{\text{mult}} \in \mathrm{Ker}(\psi)^m$ by the free submodule check of Section 5.2.

Cost summary. The general reduction commits the same eight oracles as MIMD: as we show in Section 5, the κ rotation vectors $\hat{\boldsymbol{\rho}}^{(\ell)}$ are virtual, derived from $\hat{\mathbf{w}}$ without sending the actual polynomial oracles. The only extra work over MIMD is the linear check's outer sum over the κ packed diagonals $\hat{\mathbf{D}}^{(\ell)}$; but these are collectively as sparse as a single matrix, carrying only $\mathrm{nnz}(\mathbf{A}) = O(n)$ nonzeros, so verification stays near-linear in the circuit's sparsity.

4.4 Making the Verifier Sublinear

In the non-holographic setting, the verifier time is linear in the worst case, as it needs to process the circuit. Specifically, the verifier evaluates $\mathbf{A}, \mathbf{B}, \mathbf{C}$ at a random point $(\boldsymbol{r}_x, \boldsymbol{r}_y)$ in our schemes. The verifier can be made sublinear in the holographic setting with preprocessing. Following the approach in [Set20], since the circuit is of sparsity $t = O(n)$, the prover can commit to the sparse representations $\mathrm{val}(\cdot)$, $\mathrm{row}(\cdot)$ and $\mathrm{col}(\cdot)$ of $\mathbf{M} \in \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$ such that $M(x,y) = \sum_{i,j\in[m]} M_{i,j} \cdot \mathrm{eq}(\mathrm{bin}(i), x) \cdot \mathrm{eq}(\mathrm{bin}(j), y) = \sum_{k\in[t]} \mathrm{val}(k) \cdot \mathrm{eq}(\mathrm{bin}(\mathrm{row}(k)), x) \cdot \mathrm{eq}(\mathrm{bin}(\mathrm{col}(k)), y)$, where $\mathrm{val}(z)$ stores the z-th nonzero value of the sparse matrix $\mathbf{M}$ with $\mathrm{row}(z)$ denoting its row index and $\mathrm{col}(z)$ denoting its column index. Given this sum form, the prover can reduce the

verifier's opening of $M(\boldsymbol{r}_x, \boldsymbol{r}_y)$ by running the sumcheck protocol over rings for $\sum_{z\in[t]} \mathrm{val}(z) \cdot h_{\mathrm{col}}(z) \cdot h_{\mathrm{row}}(z)$, to the opening of $\mathrm{val}(z)$, $h_{\mathrm{row}}(z)$ and $h_{\mathrm{col}}(z)$ with additional memory checking constraints that $h_{\mathrm{row}}(z) = \mathrm{eq}(\mathrm{bin}(\mathrm{row}(z)), \boldsymbol{r}_x)$ and $h_{\mathrm{col}}(z) = \mathrm{eq}(\mathrm{bin}(\mathrm{col}(z)), \boldsymbol{r}_y)$. As observed by Wei et al. [WZD25a], permutation arguments based on the sumcheck protocol proposed in HyperPlonk [CBBZ23] can be extended to Galois rings. We can apply this approach to perform the memory checking, resulting in a sublinear-time verifier in the holographic setting.

5 Polynomial Commitment Scheme over Rings

Galois-ring SNARKs need a multilinear polynomial commitment scheme over Galois rings in order to compile Galois-ring PIOPs into succinct arguments. Because Galois rings retain much of the algebraic structure of finite fields, a PCS based on interleaved codes like [AHIV17,GLS+23, BCF+25] can be adapted to Galois rings with minor changes. We review this construction in Protocol 5.1, where the lines in black are the standard PCS over rings and the colored lines denote our additional checks that will be explained below.

Let $f \in S[X_1, \ldots, X_\mu]$ be the multilinear polynomial to commit, represented by its $N := 2^\mu$ evaluations on the Boolean cube $\{0,1\}^\mu$. Fix a factorization $N = k_1 k_2$ into a row count k_1 and a column count k_2, both powers of two; an $[n, k_2]$ linear code $\mathcal{C}$ over S of rate $\rho = k_2/n$ and relative distance $\delta = d_{\min}(\mathcal{C})/n$; and a query budget t. The prover reshapes the N evaluations of f into a row-major matrix $\mathbf{M} \in S^{k_1 \times k_2}$ where $M_{i,j} := f\big(\mathrm{bin}(i) \,\|\, \mathrm{bin}(j)\big)$ for $i \in [k_1],\ j \in [k_2]$, whose rows $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{k_1} \in S^{k_2}$ are the messages to be encoded.

Protocol 5.1: Interleaved-Code PCS over Rings

Public parameters: an $[n, k_2]$ linear code $\mathcal{C}$ over S; row/column counts k_1, k_2 with $k_1 k_2 = N$; query budget t.

Input: multilinear $f \in S[X_1, \ldots, X_\mu]$ given by its $N = 2^\mu$ evaluations.

Commit.

1. $\mathcal{P}$ arranges the evaluations of f into $\mathbf{M} \in S^{k_1 \times k_2}$ and encodes each row, $\boldsymbol{u}_i := \mathrm{Enc}(\boldsymbol{m}_i)$.
2. $\mathcal{P}$ sends an oracle to the n columns of $\mathbf{U} = (\boldsymbol{u}_i)_{i\in[k_1]}$.

Open.

1. $\mathcal{V}$ samples and sends $\boldsymbol{r} \leftarrow E^{k_1}$; $\mathcal{P}$ sends the message $\bar{\boldsymbol{m}}^r := \sum_{i\in[k_1]} r_i\, \boldsymbol{m}_i \in S^{k_2}$.
2. $\mathcal{V}$ parses the evaluation point $\boldsymbol{z} = (\boldsymbol{z}_0, \boldsymbol{z}_1)$ and sends $\boldsymbol{z}_0$; $\mathcal{P}$ sends $\bar{\boldsymbol{m}}^z := \sum_{i\in[k_1]} \mathrm{eq}(\mathrm{bin}(i), \boldsymbol{z}_0)\, L(\boldsymbol{m}_i) \in S^{k_2}$ where L is the intra-symbol linear operator.
3. $\mathcal{V}$ samples $Q \subseteq [n]$, $|Q| = t$, and queries the columns $\{U_{\cdot,j}\}_{j\in Q}$.
4. $\mathcal{V}$ re-encodes and checks $\mathrm{Enc}(\bar{\boldsymbol{m}}^r)_j = \sum_i r_i\, u_{i,j}$ and $\mathrm{Enc}(\bar{\boldsymbol{m}}^z)_j = \sum_i \mathrm{eq}(\mathrm{bin}(i), \boldsymbol{z}_0)\, L(u_{i,j})$ for all $j \in Q$.
5. Submodule check: $\mathcal{V}$ checks that each opened entry $U_{i,j} \in V$ for all $i \in [k_1]$, $j \in Q$ where $V \subseteq S$ is a complemented submodule.
6. $\mathcal{V}$ outputs the evaluation $\langle \bar{\boldsymbol{m}}^z, (\mathrm{eq}(\mathrm{bin}(j), \boldsymbol{z}_1))_{j\in[k_2]} \rangle$ if all checks pass.

Protocol 5.1 is constructed as an interactive oracle proof. The prover commits to the encoded matrix as an oracle that the verifier queries at t column positions. We instantiate the oracle by a Merkle-tree vector commitment over a collision-resistant hash, and apply the Fiat–Shamir transform to make it non-interactive.

However, two obstacles still stand in the way of building an efficient PCS over Galois rings

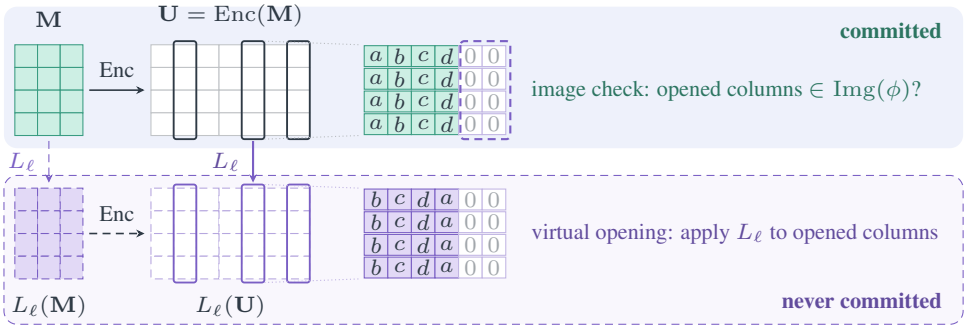


Figure 4: Image check and virtual opening. The prover commits only to $\mathbf{U} = \mathrm{Enc}(\mathbf{M})$. The image check tests whether opened columns lie in $\mathrm{Img}(\phi)$. Since $L_\ell \circ \mathrm{Enc} = \mathrm{Enc} \circ L_\ell$, applying L_ℓ to an opened column of $\mathbf{U}$ opens the never-committed $L_\ell(\mathbf{U})$.

and using it to compile our PIOPs into a full SNARK: the PCS's performance depends largely on the underlying linear code, while the full SNARK additionally requires image, kernel, and rotation checks on the committed values, as we discuss below:

- **Lifting linear codes to Galois rings is both intellectually and concretely costly.** Each lift, whether from binary to a field [AZ26] or from a field to a Galois ring [WZD25a], requires a new and nontrivial distance analysis. Properties of Galois rings such as zero divisors may introduce additional challenges to the analyses. Moreover, concretely speaking, the lifted encoder often relies on multiplications between Galois-ring elements, resulting in high encoding overhead.
- **Multiplicative packing hides structure from the PIOP layer.** Packed elements are constrained to lie in a fixed submodule of the ambient Galois ring such as an image, a kernel, or a rotation orbit. Since the PIOP layer treats each committed value as opaque and fails to utilize its inner properties, it can only enforce such constraints via additional checks with high overhead (e.g., the image check in [JLX+25, CG22]).

We resolve both issues in a unified paradigm. First, we give a black-box transformation that lifts any binary linear code to a Galois-ring code, preserving its rate and distance as well as avoiding any Galois-ring multiplication. Second, we observe that the PCS, unlike the PIOP layer, has direct access to the committed data. Combined with the structure of our lifted code, this access allows the PCS to prove the constraints of the packed elements essentially for free. That is, to check that all evaluations that define the polynomial are in $\mathrm{Img}(\phi)$ or $\mathrm{Ker}(\psi)$, the verifier just needs to check that this is true for every opened element, as in Step 5) of Protocol 5.1. To check a rotation on the packed elements, or any linear operator, the verifier simply performs this operation on the opened elements before running the consistency test, as in Steps 2) and 4) of Protocol 5.1. See Figure 4, Section 5.2 and Section 5.3.

5.1 Lifting Binary Linear Codes to Galois Rings

The lift from a binary linear code to a $\mathrm{GR}(2^k, d)$-linear code proceeds in two stages:

$$\mathbb{F}_2 \longrightarrow \mathbb{Z}_{2^k} \longrightarrow \mathrm{GR}(2^k, d).$$

The first stage lifts a binary linear code to a $\mathbb{Z}_{2^k}$-linear code, preserving its relative distance exactly, and the second stage extends it to $\mathrm{GR}(2^k, d)$ with the same parameters. The second stage also yields the submodule-distance preservation, see Theorem 5.6. The technique can be generalized to any linear code over a prime field $\mathbb{F}_p$ lifted to $\mathbb{Z}_{p^k}$ and then to $\mathrm{GR}(p^k, d)$. We show

the proof in Appendix B.1. In this section, we set $p = 2$, as binary linear codes are the most common class of error-correcting codes and match the integer ring $\mathbb{Z}/2^k\mathbb{Z}$ we aim to support.

From $\mathbb{F}_2$ to $\mathbb{Z}_{2^k}$. Given the generator matrix $\mathbf{G} \in \{0,1\}^{n\times k_2}$ of a binary code, we lift it to $\mathbb{Z}_{2^k}$ by viewing the values as 0s and 1s in $\mathbb{Z}_{2^k}$, and we abuse the notation and use $\mathbf{G} \in \mathbb{Z}_{2^k}^{n\times k_2}$ to denote the generator matrix. The encoder $\mathrm{Enc}_{2^k} : \mathbb{Z}_{2^k}^{k_2} \to \mathbb{Z}_{2^k}^{n}$ encodes $\boldsymbol{m}$ to $\mathbf{G}\,\boldsymbol{m} \bmod 2^k$, whose image is the code $\mathcal{C}_{2^k} := \mathrm{Img}(\mathrm{Enc}_{2^k})$.

Theorem 5.1 (Binary lift)**.** *If* Enc_2 *is injective, then for every* $k \geq 1$ *the encoder* Enc_{2^k} *is injective and*

$$d_{\min}(\mathcal{C}_{2^k}) = d_{\min}(\mathcal{C}_2),$$

so that $\mathcal{C}_{2^k}$ *has the same rate and relative distance as* $\mathcal{C}_2$*.*

From $\mathbb{Z}_{2^k}$ to $\mathrm{GR}(2^k, d)$. Set $R := \mathbb{Z}_{2^k}$ and $S := \mathrm{GR}(2^k, d)$, a free R-module of rank d (Section 3.1.3) with R-basis $\beta_1, \ldots, \beta_d$. Acting by the same generator $\mathbf{G} \in \mathbb{Z}^{n\times k_2}$ gives the S-linear encoder $\mathrm{Enc}_S : S^{k_2} \to S^n$ that encodes $\boldsymbol{m}$ to $\mathbf{G}\boldsymbol{m}$, with image the code $\mathcal{C}_S := \mathrm{Img}(\mathrm{Enc}_S)$.

Theorem 5.2 (Galois ring lift)**.** *If* Enc_{2^k} *is injective, then the encoder* Enc_S *is injective and*

$$d_{\min}(\mathcal{C}_S) = d_{\min}(\mathcal{C}_{2^k}),$$

so that $\mathcal{C}_S$ *has the same rate and relative distance as* $\mathcal{C}_{2^k}$*.*

Remark 5.3 (Relation to extension codes). Stage 2 can be viewed as a generalization of the extension-code construction of [DP25], which reuses the generator of a code over a field K to encode symbols in a K-vector space V. We apply it over a ring, with K replaced by $R = \mathbb{Z}_{2^k}$ and V by the free R-module S. Because S is free over R, the basis decomposition is unique, so the same slicing argument and distance bound carry over. Stage 1 is different, because $\mathbb{Z}_{2^k}$ is not an $\mathbb{F}_2$-vector space and the slicing argument does not apply, so there we use the 2-adic valuation argument of Theorem 5.1 instead.

Combining Theorems 5.1 and 5.2 gives the full lift. The proofs are presented in Appendix B.

Encoding timing. The lifted encoder $\mathrm{Enc}_S(\boldsymbol{m}) = \mathbf{G}\boldsymbol{m}$ runs the binary encoder over S, using only additions and R-scalar multiplications of S-elements, but not any Galois-ring multiplications. Each such operation is d operations over $R = \mathbb{Z}_{2^k}$, so a binary encoder of T operations lifts to Td operations over R. In particular, a linear-time binary encoder stays linear-time over S.

Instantiation. We instantiate the binary code with the Repeat-Accumulate-Accumulate (RAA) code of Blaze [BCF+25], whose encoder only involves repetitions, permutations and additions. By Corollary B.2, the lifted encoder is exactly this RAA algorithm over S in place of $\mathbb{F}_2$, inheriting the binary minimum distance. There is no ring-specific modification such as the alphabet randomization of prior lifts [AZ26]. It runs in linear time and uses neither a Galois-ring multiplication, unlike [WZD25a], nor an NTT, unlike [JLX+25].

5.2 Submodule Checks Are Free

Motivated by the kernel and image checks of Section 3.3, we introduce the submodule check for verifying that committed values lie in some $V \subseteq S$, where V is a submodule.

To utilize the error amplification from the linear code, we collapse V to zero via a quotient map, so that any value outside V becomes a nonzero coordinate of a *quotient code*. The minimum distance of that code then amplifies a single violation into δn violations, which a few random column queries detect. We show that for the lifted code of Section 5.1 the quotient code inherits the full minimum distance, so submodule membership can be checked for free by checking that the opened columns lie in V with only a minor soundness loss.

5.2.1 Quotient Code

For a vector $\boldsymbol{c} \in S^n$ and a subset $V \subseteq S$, define the *V-weight* as $\Delta_V(\boldsymbol{c}) := \#\{j \in [n] : c_j \notin V\}$. Taking $V = \{0\}$ recovers the ordinary Hamming weight. To analyze the V-weight $\Delta_V(\boldsymbol{c})$ using coding theory, we want to recast "coordinate lies in V" as "coordinate is zero in some companion alphabet." The quotient S/V supplies this companion where $c_j \in V$ if and only if $c_j + V = 0$ in S/V. Applied coordinate-wise, every codeword of $\mathcal{C}_S$ maps to a vector in $(S/V)^n$, and counting out-of-V coordinates of the original becomes counting nonzero coordinates of its image. A lower bound on the ordinary Hamming weight of these nonzero images then gives a lower bound on the V-weight, which we make precise in Lemma 5.5.

Definition 5.4 (Quotient code)**.** Let $V \subseteq S$ be an R-submodule of $S = \mathrm{GR}(2^k, d)$, with $R := \mathbb{Z}_{2^k}$, and let $\pi : S \to S/V$ be the quotient map with coordinate-wise extension $\pi^n : S^n \to (S/V)^n$. The *quotient code* of $\mathcal{C}_S$ by V is the image

$$\overline{\mathcal{C}}_V \;:=\; \pi^n(\mathcal{C}_S) \;=\; \left\{\, \pi^n(\boldsymbol{c}) : \boldsymbol{c} \in \mathcal{C}_S \,\right\} \;\subseteq\; (S/V)^n.$$

To make $\overline{\mathcal{C}}_V$ a code generated by the same $\mathbf{G}$, the encoder must respect the R-submodule V, that is, satisfy the closure condition $\mathbf{G}\, V^{k_2} \subseteq V^n$. This holds automatically for our lifted code, whose generator $\mathbf{G}$ has integer entries: each coordinate $(\mathbf{G}\boldsymbol{m})_i = \sum_j G_{ij}\, m_j$ is then an integer combination of the entries of $\boldsymbol{m}$, and a submodule is closed under integer combinations, so $\boldsymbol{m} \in V^{k_2}$ forces $\mathbf{G}\boldsymbol{m} \in V^n$. Closure lets Enc descend to the R-linear quotient encoder $\overline{\mathrm{Enc}} : (S/V)^{k_2} \to (S/V)^n$ with $\overline{\mathrm{Enc}}(\boldsymbol{m} + V^{k_2}) := \mathbf{G}\boldsymbol{m} + V^n$, again generated by $\mathbf{G}$ but over the quotient alphabet, and $\overline{\mathcal{C}}_V = \mathrm{Img}(\overline{\mathrm{Enc}})$.

We first demonstrate the usefulness of the quotient code by providing Lemma 5.5. Since $c_j \in V$ if and only if $c_j + V = 0$ in S/V, the V-weight equals the ordinary Hamming weight of the quotient image, $\Delta_V(\mathrm{Enc}(\boldsymbol{m})) = \Delta(\overline{\mathrm{Enc}}(\boldsymbol{m} + V^{k_2}))$ for every $\boldsymbol{m} \in S^{k_2}$. A lower bound on the V-weight is therefore the minimum distance of $\overline{\mathcal{C}}_V$.

Lemma 5.5 (V-weight lower bound)**.** *For every $\boldsymbol{m} \notin V^{k_2}$, if the quotient encoder is injective then $\overline{\mathrm{Enc}}(\boldsymbol{m} + V^{k_2}) \neq 0$ and*

$$\Delta_V(\mathrm{Enc}(\boldsymbol{m})) \;\geq\; d_{\min}(\overline{\mathcal{C}}_V).$$

We then show that the quotient code preserves the relative distance of our lifted code. The quotient can only reduce the minimum distance, $d_{\min}(\overline{\mathcal{C}}_V) \leq d_{\min}(\mathcal{C}_S)$, and in the worst case reduces it to zero: if some nonzero codeword of $\mathcal{C}_S$ lies entirely in V^n, then $d_{\min}(\overline{\mathcal{C}}_V) = 0$ and the submodule check is vacuous. No such collapse occurs for our lifted code once V is *complemented* (Definition 3.9), a direct summand $S = V \oplus W$ rather than a mere submodule.

Theorem 5.6 (Submodule-distance preservation)**.** *Let $V \subsetneq S$ be a complemented R-submodule of S, and let $\mathcal{C}_S$ be the lifted code of Section 5.1. Then*

$$d_{\min}(\overline{\mathcal{C}}_V) = d_{\min}(\mathcal{C}_S).$$

See Appendix B for the proof.

Since the multiplicative packing yields the direct-sum decomposition $S = \mathrm{Img}(\phi) \oplus \mathrm{Ker}(\psi)$ as R-modules shown in Lemma 3.14, both $\mathrm{Img}(\phi)$ and $\mathrm{Ker}(\psi)$ are complemented, giving the following corollary:

Corollary 5.7 (Distance preservation for the image and kernel submodules)**.** *For the lifted code $\mathcal{C}_S$, $d_{\min}(\overline{\mathcal{C}}_{\mathrm{Img}(\phi)}) = d_{\min}(\overline{\mathcal{C}}_{\mathrm{Ker}(\psi)}) = d_{\min}(\mathcal{C}_2)$. In particular, for $V \in \{\mathrm{Img}(\phi), \mathrm{Ker}(\psi)\}$, every $\mathrm{Enc}(\boldsymbol{m})$ with a coordinate outside V has at least δn coordinates outside V.*

5.2.2 Free Submodule Check

The submodule check in Step 5) of Protocol 5.1 comes at no cost. The verifier already opens t random columns of $\mathbf{U}$; to certify that every committed value lies in a complemented submodule V, it additionally tests $U_{i,j} \in V$ on each opened entry. In a basis adapted to V, this checks that the complementary (W-component) coordinates of each opened element vanish, so it uses no ring operations and adds nothing to the proof beyond the columns already opened.

Soundness combines the two halves of this subsection. The proximity gap (Theorem 3.10) first ensures that the committed columns are close to a unique codeword. Instantiating the V-weight bound of Lemma 5.5 at that codeword lower-bounds the number of its coordinates outside V by the quotient distance $d_{\min}(\overline{\mathcal{C}}_V)$, which Corollary 5.7 evaluates to δn. A committed row that leaves V therefore leaves it in at least δn coordinates, of which proximity hides at most e, so the t random columns still expose the violation with overwhelming probability. Here V is either the image $\mathrm{Img}(\phi)$ or the kernel $\mathrm{Ker}(\psi)$, the two memberships the PIOP layer needs; throughout, fix the proximity radius $e := \lfloor \delta n/3 \rfloor$.

Theorem 5.8 (Soundness of the interleaved-code PCS with submodule check)**.** *Let $\mathcal{C} := \mathcal{C}_S$ be the lifted $[n, k_2]$-linear code of Section 5.1 with relative distance δ, and let V be either* $\mathrm{Img}(\phi)$ *or* $\mathrm{Ker}(\psi)$*. For the interleaved-code PCS of Protocol 5.1 with the submodule check on V and challenges drawn from E^{k_1}, the testing, submodule, and evaluation checks have soundness errors*

$$\epsilon_{test} \le \left(1 - \tfrac{\delta}{3}\right)^t + \tfrac{n}{|E|}, \qquad \epsilon_{sub},\, \epsilon_{eval} \le \left(1 - \tfrac{2\delta}{3}\right)^t,$$

and the overall soundness error is $\epsilon \le \epsilon_{test} + \epsilon_{sub} + \epsilon_{eval}$.

We present the proof in Appendix B.

Remark 5.9 (Submodule check is free). In Theorem 5.8, the testing error $\epsilon_{\text{test}} = (1 - \delta/3)^t$ dominates. For the RAA code with $\delta = 0.19$ used in Section 6, the query budget is $t = 1060$. The testing error is $\epsilon_{\text{test}} \approx 2^{-100}$, while the submodule error $(1 - 2\delta/3)^t \approx 2^{-207}$ is much smaller and absorbed by it. The submodule check thus needs no extra column openings and is free.

5.3 Virtual PCS Are Free

The general-circuit reduction of Section 4.3 introduces the rotation relation $\mathcal{R}_S^{\text{rot}}$, whose naive treatment makes the prover commit to κ auxiliary rotated witnesses $\hat{\boldsymbol{\rho}}^{(\ell)}$ and prove their consistency. To avoid this, we provide a stronger tool that generalizes beyond rotations: the *virtual PCS*. For any public R-linear map $L : S \to S$, a single commitment to $\boldsymbol{u} \in S^n$ under Protocol 5.1 also acts as a commitment to $L(\boldsymbol{u})$, the vector obtained by applying L to each coordinate of $\boldsymbol{u}$, so the verifier can open $L(\boldsymbol{u})$ without the prover ever committing to it. Each rotated witness is then derived virtually from the single commitment to the original witness, at no extra cost.

The prover commits once to $\boldsymbol{u} \in S^n$ under Protocol 5.1. Fix a public R-linear map $L : S \to S$; fixing the R-basis $\beta_1, \ldots, \beta_d$ of S identifies L with its matrix $L \in R^{d \times d}$, which acts on a single symbol and on $\boldsymbol{u} \in S^n$ coordinatewise. The *virtual commitment* to $L(\boldsymbol{u})$ is the original commitment to $\boldsymbol{u}$ together with the public tag L. To open the j-th coordinate of $L(\boldsymbol{u})$, the verifier opens u_j from the original commitment and applies L locally, obtaining $L(u_j)$; the prover supplies no value of its own. Each virtual opening thus costs one original opening and one application of L.

Lemma 5.10 (Encoding commutes with R-linear transforms)**.** *For every $\boldsymbol{m} \in S^{k_2}$ and every R-linear map $L : S \to S$ applied coordinatewise, $L(\mathrm{Enc}_S(\boldsymbol{m})) = \mathrm{Enc}_S(L(\boldsymbol{m}))$.*

Proof. Decompose $\boldsymbol{m} = \sum_\ell \beta_\ell \, \boldsymbol{m}^{(\ell)}$ with $\boldsymbol{m}^{(\ell)} \in R^{k_2}$. Since $\mathbf{G}$ has integer entries, $\mathrm{Enc}_S(\boldsymbol{m}) = \sum_\ell \beta_\ell \, (\mathbf{G}\boldsymbol{m}^{(\ell)})$. Apply L coordinate-wise; writing $L(\beta_\ell) = \sum_s a_{s\ell} \, \beta_s$ with $a_{s\ell} \in R$, and using that L commutes with multiplication by R-scalars, $L(\mathrm{Enc}_S(\boldsymbol{m})) = \sum_\ell L(\beta_\ell) \, (\mathbf{G}\boldsymbol{m}^{(\ell)}) = \sum_s \beta_s \, \mathbf{G}(\sum_\ell a_{s\ell} \, \boldsymbol{m}^{(\ell)})$. The s-th basis slice of $L(\boldsymbol{m})$ is $(L(\boldsymbol{m}))^{(s)} = \sum_\ell a_{s\ell} \, \boldsymbol{m}^{(\ell)}$, so the right-hand side equals $\mathrm{Enc}_S(L(\boldsymbol{m}))$. □

Lemma 5.11 (Proximity transfer)**.** *Let $L : S \to S$ be an R-linear map applied coordinatewise. If $\Delta(\boldsymbol{u}, \boldsymbol{c}) \le e$ for $\boldsymbol{u}, \boldsymbol{c} \in S^n$, then $\Delta\big(L(\boldsymbol{u}), L(\boldsymbol{c})\big) \le e$.*

Proof. For every j, if $u_j = c_j$ then $L(u_j) = L(c_j)$. Hence $\mathrm{supp}\big(L(\boldsymbol{u}) - L(\boldsymbol{c})\big) \subseteq \mathrm{supp}(\boldsymbol{u} - \boldsymbol{c})$, and the bound on Hamming distance is preserved (no injectivity assumed). □

Theorem 5.12 (Virtual PCS soundness)**.** *Let $\mathcal{C}_S$ be the lifted code of Section 5.1 and $L : S \to S$ a public R-linear map applied coordinate-wise. Protocol 5.1 is a sound PCS for the virtual polynomial $L(f)$, obtained by applying L to every evaluation of f over the Boolean hypercube.*

See Appendix B for the proof. We use this to perform the rotation check of the general-circuit reduction. Block rotation is itself R-linear on the packed witness: let $L_\ell : S \to S$ be the fixed public R-linear map with $L_\ell(\phi(\boldsymbol{x})) = \phi(\mathrm{rot}_\ell(\boldsymbol{x}))$ for every block $\boldsymbol{x} \in R^\kappa$, so that the rotated witness is $\hat{\boldsymbol{\rho}}^{(\ell)} = L_\ell(\hat{\mathbf{w}})$. Applying Theorem 5.12 with $L = L_\ell$, the single commitment to $\hat{\mathbf{w}}$ already serves as a commitment to each $\hat{\boldsymbol{\rho}}^{(\ell)}$: when the PIOP queries $\hat{\rho}^{(\ell)}(\boldsymbol{r}_y)$, the verifier opens $\hat{\mathbf{w}}$ at the queried columns and applies L_ℓ locally. Since $\hat{\boldsymbol{\rho}}^{(\ell)}$ is bound to $L_\ell(\hat{\mathbf{w}})$ rather than chosen by the prover, it is the genuine block rotation of $\hat{\mathbf{w}}$, so the rotation relation $\mathcal{R}_S^{\mathrm{rot}}$ holds by construction and the κ rotation witnesses need no extra commitments.

5.4 SNARKs over Rings

Compiling the PIOPs of Section 4 with Protocol 5.1 yields our main theorem.

Theorem 5.13. *Let (ϕ, ψ) be a degree-2 multiplicative packing from $R = \mathbb{Z}/2^k\mathbb{Z}$ to $S = \mathrm{GR}(2^k, d)$ with $d = \Theta(\lambda)$, let $\mathcal{C}_S$ be the lift (Section 5.1) of a linear-time encodable binary code with constant rate and relative distance, and let H be a hash function modeled as a random oracle. Compiling Protocols 4.1, 4.2 and 4.3 with Protocol 5.1 over $\mathcal{C}_S$ via the BCS transformation [BSCS16], which uses H for both the Merkle-tree commitments and the Fiat–Shamir transform, yields transparent SNARKs for $\mathcal{R}_R^{SIMD}$, $\mathcal{R}_R^{MIMD}$ and $\mathcal{R}_R^{R1CS}$ in the random oracle model. For an R1CS of size n, the prover performs $O(n)$ operations over S and hash evaluations, and the proof size is $O_\lambda(\sqrt{n})$. After an $O(n)$-time holographic preprocessing (Section 4.4), the verifier runs in time $O_\lambda(\sqrt{n})$; without preprocessing, it runs in time $O(n)$. Standard zero-knowledge compilers [CFS17, XZZ+19, DHRR26] turn these SNARKs into zkSNARKs with the same asymptotic costs.*

We sketch the proof of Theorem 5.13 in Appendix B.

6 Implementation and Evaluation

We have fully implemented our SNARK system over integer rings, Nenya, and we benchmark its performance in this section.

6.1 Implementation and Settings

Nenya is implemented in Rust and consists of 9,875 lines of code, including the reductions and PIOPs in Section 4 and the PCS in Section 5 with image, kernel, and rotation checks. We implement most components ourselves, such as the ring arithmetic, but some of the implementations use arkworks [ac22] and Plonky3 [plo] as references. Experiments were executed on an AWS `c7i.48xlarge` machine (192 vCPUs), in both single- and multi-threaded configurations.

Ring arithmetic. Nenya operates over the Galois ring $S = \mathrm{GR}(2^k, 128)$ lifted from the base ring $R = \mathbb{Z}/2^k\mathbb{Z}$ with k chosen from $\{8, 16, 32, 64\}$ using a multiplicative packing with packing size $\kappa = 32$. This is not the best possible packing ratio for degree-2 RMFEs, but it is efficient and convenient to make κ and d powers of 2. We present the detailed construction of the

multiplicative packing in Appendix A. Base ring arithmetic over $\mathbb{Z}/2^k\mathbb{Z}$ is native to the hardware with k-bit integer types, thus additions and multiplications modulo 2^k incur no extra cost. Additions over S are implemented coordinate-wise over R and are cheap, whereas multiplications over S require polynomial products over R and thus are the dominating cost.

Compared with arithmetic over the degree-4 extension of the BabyBear field from the Plonky3 library [plo], one Galois-ring addition that processes all 32 packed instances of $\mathbb{Z}/2^{32}\mathbb{Z}$ at once is about 7.8× faster than 32 field additions, and one Galois-ring multiplication is about 1.2× faster than 32 field multiplications. See Table 1 for more details. Multiplications over S are implemented using the naive algorithm with complexity quadratic in the bitlength, which we find faster than the number-theoretic transform algorithm in this regime. The comparison with field arithmetic is a concrete sign of the hardware-friendliness that motivates $\mathbb{Z}/2^k\mathbb{Z}$ SNARKs, where the compiler and hardware can optimize ring computations directly.

Parameters. We target $\lambda = 100$ bits of security and use the RAA code at rate $\rho = 1/4$ with relative distance $\delta = 0.19$ [BCF+25]. The Merkle tree commitments and the Fiat–Shamir transform use BLAKE3. The witness is committed as a $k_1 \times k_2$ matrix with $k_1 k_2 = N$, and the verifier opens $t = 1060$ columns according to Theorem 5.8. The proof carries $k_2 + t\,k_1$ ring elements, minimized at $k_1 = \sqrt{N/t}$ and $k_2 = \sqrt{Nt}$; for $N = 2^{20}$ this rounds to a matrix of dimension $2^5 \times 2^{15}$.

Optimizations. In the R1CS reductions, the error terms are essentially quotients, which we compute faster using the quotient decomposition of [ZOZ26] with preprocessing. Polynomial commitments opened at the same point are stacked into a single commitment and opening [CBBZ23], as we observe that it can amortize the number of opened columns and concretely reduce the proof size of the PCS. Moreover, because the encoding preserves the 32-dimensional image space and the 96-dimensional kernel space, the committed matrices admit a sparse representation, so although MIMD and general conceptually commit six additional polynomials, they concretely pay for only three in proof size and prover time. For the sumchecks, we apply the optimization of [Gru24] so that the multiplication check sends only degree-2 polynomials in each round, similar to the linear check. We represent these polynomials by their evaluations at 0, 1, and ∞. The evaluation at ∞ is just the leading coefficient and thus is free. This helps more over rings than over fields: a field's cheap third point 2 is unavailable over S, where it must be replaced by an expensive evaluation at a $\mathcal{T}$ point.

6.2 Experimental Evaluations

We aim to answer the following three questions in the evaluations of Nenya:

Q1: What is the performance of Nenya under different parameters with our new techniques and optimizations?

Q2: How does Nenya compare to SNARKs over large finite fields and prior SNARK schemes over integer rings?

Q3: For computations that are native to fixed-width integers, what is the improvement of Nenya over SNARKs over finite fields?

Table 1: Speedup of one Galois-ring operation over the 32 field operations it replaces; columns are the base modulus 2^k.

	2^8	2^{16}	2^{32}	2^{64}
Add (×)	31	14	7.8	3.1
Mult (×)	2.0	2.3	1.2	0.39

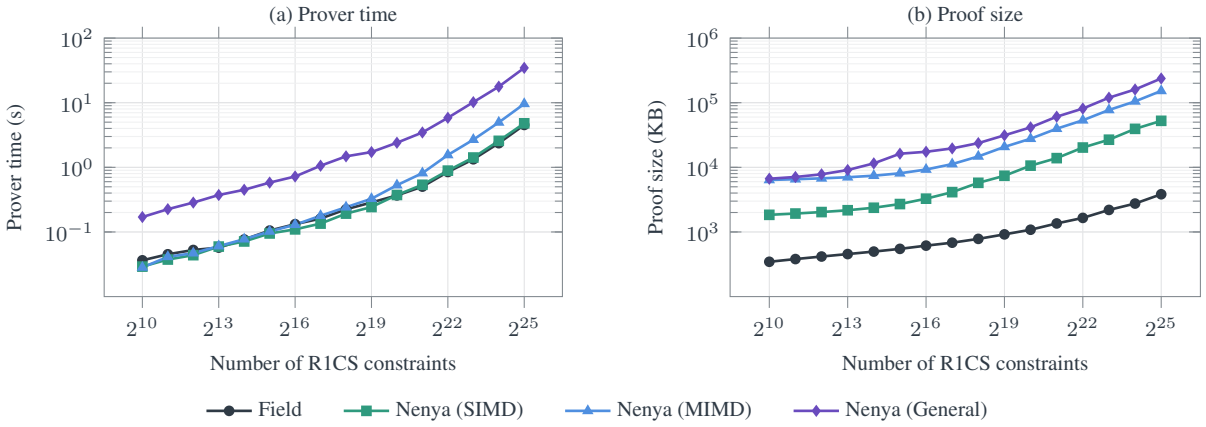


Figure 5: Multi-threaded end-to-end performance of Nenya across the three computation models versus a field SNARK of the same design.

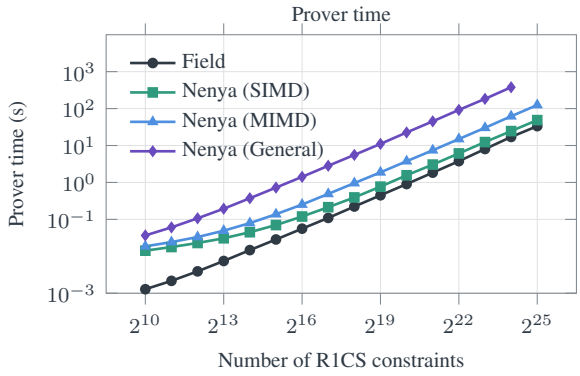


Figure 6: Single-threaded performance of Nenya.

6.2.1 Performance of Nenya and Comparison with Field SNARK

Figure 5 reports the multi-threaded prover time and the proof size from 2^{10} to 2^{25} R1CS constraints for the three computation models of Section 4 over 32-bit integers. We also compare the performance with a SNARK over a finite field of the same design—the same PIOP based on Spartan [Set20] and PCS using the RAA code over the degree-4 extension of the BabyBear field, without packing, submodule checks, or error terms[1]. At $n = 2^{25}$, for a SIMD computation, the prover of Nenya runs in 4.8 seconds, which is only 1.06× slower than the field SNARK on the same number of constraints. This is because our free submodule and rotation checks (Section 5.2) remove the dominant ring-specific cost—what remains is the wider Galois-ring arithmetic and the PCS of the error term, amortized over $\kappa = 32$ packed instances. The prover time for the general computation is 34.6 seconds, as it commits extra witnesses and performs the sumcheck protocol on a larger equation with rotations (Protocol 4.3). The prover time for a MIMD computation is 9.6 seconds. Single-threaded numbers are in Figure 6, which shows similar ordering and gaps.

The proof size of Nenya is significantly larger than that of the field-based SNARK, and this is completely due to the larger element size of $\mathrm{GR}(2^{32}, 128)$. For $n = 2^{20}$, the proof size of Nenya is 10× larger than that of the field-based SNARK on a SIMD computation, 25.6× larger for MIMD and 38.4× larger for a general computation. Improving the proof size of ring-based

[1]This field-based SNARK is for a general computation. The performance can be slightly improved for SIMD computations, but the gap is much smaller than for Nenya, as both versions have the same number and size of PCSs and sumchecks.

Table 2: Cost breakdown of Nenya over $\mathbb{Z}_{2^{32}}$ for $n = 2^{25}$ and the field-based SNARK (multi-threaded).

	Prover time (s)								Proof size (MB)			Verifier time (s)			
		PIOP			PCS							PIOP			
	Reduction	eval	linear	mult	encode	Merkle	open	Total	PIOP	PCS	Total	eval	sumcheck	PCS	Total
Field	—	1.26	0.37	2.23	0.46	0.07	0.13	**4.53**	0.003	3.83	**3.83**	0.98	1.46	0.11	**2.55**
SIMD	0.16	0.71	0.16	1.83	1.54	0.10	0.30	**4.80**	0.065	52.4	**52.4**	0.58	0.79	0.45	**1.82**
MIMD	1.00	0.90	0.14	2.35	4.53	0.26	0.41	**9.60**	0.065	152.3	**152.3**	0.69	0.80	0.89	**2.37**
General	12.70	2.58	5.36	4.76	7.17	0.38	1.63	**34.59**	0.097	237.0	**237.1**	2.87	2.31	1.92	**7.09**

Table 3: Effect of the base word size for $C = 2^{20}$ (multi-thread). "Field" is the BabyBear baseline (word-size independent); Nenya is instantiated over $\mathbb{Z}_{2^k}$ for $k \in \{8, 16, 32, 64\}$.

Model	Metric	Field	u8	u16	u32	u64
SIMD	Prover (s)	0.39	0.22	0.29	0.42	0.78
	Proof (MB)	1.08	3.39	5.81	10.62	20.22
MIMD	Prover (s)	0.39	0.37	0.52	0.81	1.46
	Proof (MB)	1.08	9.50	14.87	27.71	53.37
General	Prover (s)	0.39	1.93	1.90	2.33	3.99
	Proof (MB)	1.08	12.90	21.74	41.53	81.11

SNARKs using multiplicative packing is an interesting direction for future work.

Cost breakdowns. To better understand the performance of Nenya, Table 2 shows the cost breakdown for $n = 2^{25}$. In prior SNARKs over rings via multiplicative packing, the PCS dominates the prover time. However, thanks to our lifted error-correcting code together with the free submodule and rotation checks, our PCS is lightweight. Encoding, Merkle hashing and opening with these checks together constitute a small portion of the prover time. The prover time is dominated by the PIOPs (the linear and product checks) and, for the general model, by the packing reduction. By contrast, the proof size is dominated by the PCS in all schemes as its proof size is $O(\sqrt{n})$, while the proof size of the PIOP is $O(\log n)$.

We also present the verifier time of Nenya in Table 2. Note that we did not implement the variant with a sublinear verifier with preprocessing in the holographic setting. The verifier time of all schemes is linear in n. However, as shown in the table, the verifier time is completely dominated by the PIOP, namely evaluating polynomials defined by A, B, C at a random point. The result justifies that with preprocessing, the verifier time can be made sublinear and fast in practice, unlike prior works.

Moreover, we vary the base word size of the integer ring in Nenya and show the performance in Table 3. A smaller word size makes the Galois-ring operations cheaper, thus improving the prover time and the proof size. For R1CS over 8-bit integers for SIMD and MIMD computations, the prover is even faster than that of the field-based SNARK.

6.2.2 Improvement over Prior Work

Figure 7 compares Nenya with the SNARK scheme over rings using RMFE in [JLX+25]. The numbers are obtained from the paper, which sets $k = 16$. Due to the high overhead of enforcing the image and kernel checks and the inefficient reductions from $\mathbb{Z}/2^k\mathbb{Z}$ to the Galois ring, the scheme can only scale to $n = 2^{10}$ constraints. We estimate the performance for large R1CS via extrapolation with an $O(n \log n)$ prover time. As shown in the figure, Nenya's prover is at least

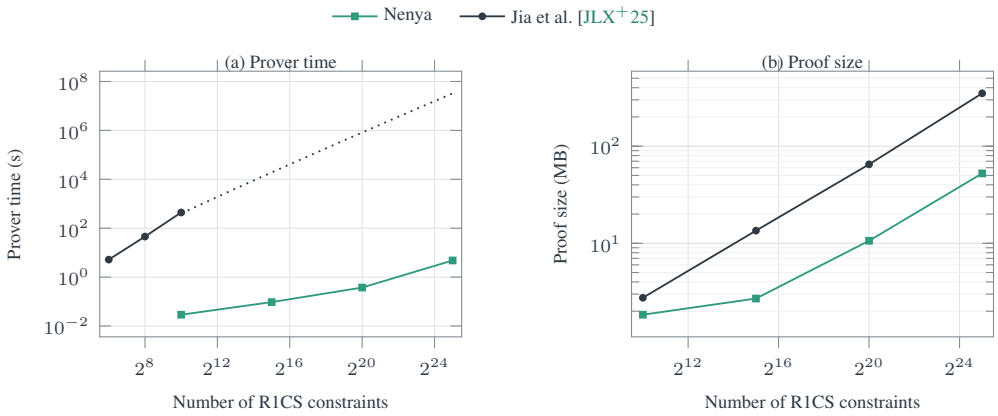


Figure 7: Nenya vs. the prior RMFE-based ring SNARK [JLX+25]. $k = 16$; dotted: extrapolated.

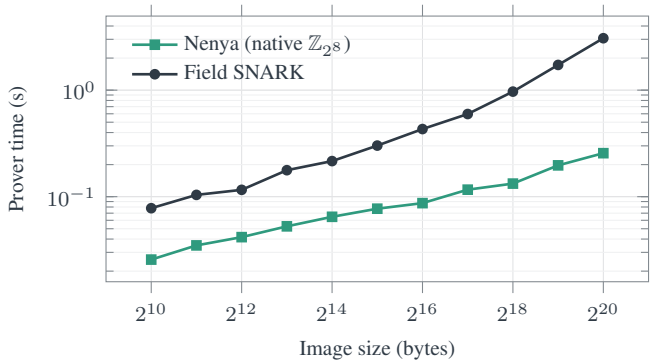


Figure 8: Prover time of Nenya and a field-based SNARK for PNG Sub/Up filtering over $\mathbb{Z}_{2^8}$.

four orders of magnitude faster than that of the scheme in [JLX+25]. Our proof size is about 6–7× smaller.

6.2.3 Native Integer Computations

Finally, we demonstrate the advantages of Nenya on a computation that is native over fixed-width integers. The computation consists of the PNG image filtering algorithms. The PNG *Sub* and *Up* filters predict each byte from a neighbor and store the residue $\mathrm{Filt}[x] = \mathrm{Orig}[x] - \mathrm{pred}[x] \bmod 256$, where pred is the byte to the left (Sub) or the byte above (Up). We prove that a batch of 32 grayscale images was correctly filtered—the raw bytes Orig private, the filtered bytes Filt public—through the inverse recurrence $\mathrm{Orig}[x] = \mathrm{Filt}[x] + \mathrm{pred}[x] \bmod 256$. The batch of 32 maps onto the SIMD model ($\kappa = 32$). The modular computation is implemented natively over $\mathbb{Z}_{2^8}$, thus each byte incurs a single constraint. By contrast, for a field-based SNARK, the R1CS has to emulate the modular reduction using bit-decompositions. For each byte, it must emulate the reduction with a borrow recurrence $\mathrm{Orig} + 256 \cdot \mathrm{borrow} = \mathrm{Filt} + \mathrm{pred}$, an 8-bit decomposition of Orig, and the attendant booleanity checks—about 11 constraints per byte. Lookup arguments can be used to improve the prover time, but the cost of each lookup is still significantly larger than one R1CS constraint.

Figure 8 shows the comparison of the prover time. Operating natively over $\mathbb{Z}_{2^8}$, Nenya's prover is 12× faster at an image size of 2^{20}.

Acknowledgments

This work is partially supported by the National Science Foundation (NSF) under Grant No. 2401481. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of these institutes.

References

[ac22] arkworks contributors. `arkworks` zkSNARK ecosystem. https://arkworks.rs, 2022.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2087–2104, 2017.

[APKP24] Kasra Abbaszadeh, Christodoulos Pappas, Jonathan Katz, and Dimitrios Papadopoulos. Zero-knowledge proofs of training for deep neural networks. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 4316–4330, 2024.

[AZ26] Pariya Akhiani and Yupeng Zhang. Distance of RAA codes over large finite fields (with applications in zkSNARKs and PCGs), 2026. Cryptology ePrint 2026/524.

[BC25] Dan Boneh and Binyi Chen. LatticeFold: A lattice-based folding scheme and its applications to succinct proof systems. In *Advances in Cryptology – ASIACRYPT 2025*, Lecture Notes in Computer Science. Springer, 2025. Cryptology ePrint 2024/257.

[BCF+25] Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D. Rothblum, and Hadas Zeilberger. Blaze: Fast SNARKs from interleaved RAA codes. In *Advances in Cryptology – EUROCRYPT 2025*, volume 15604 of *Lecture Notes in Computer Science*, pages 123–152. Springer, 2025. Cryptology ePrint 2024/1609, https://eprint.iacr.org/2024/1609.

[BCGL22] Jonathan Bootle, Alessandro Chiesa, Ziyi Guan, and Siqi Liu. Linear-time probabilistic proofs with sublinear verification for algebraic automata over every field, 2022. Cryptology ePrint 2022/1056.

[BCPS18] Anurag Bishnoi, Pete L. Clark, Aditya Potukuchi, and John R. Schmitt. On zeros of a polynomial in a finite grid. *Combinatorics, Probability and Computing*, 27(3):310–333, 2018.

[BCS21] Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Sumcheck arguments and their applications. In *Advances in Cryptology – CRYPTO 2021*, volume 12825 of *Lecture Notes in Computer Science*, pages 742–773. Springer, 2021. Cryptology ePrint 2021/333.

[BS23] Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In *Advances in Cryptology – CRYPTO 2023*, Lecture Notes in Computer Science. Springer, 2023. Cryptology ePrint 2022/1341, https://eprint.iacr.org/2022/1341.

[BSCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography Conference*, pages 31–60. Springer, 2016.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In *Advances in Cryptology – EUROCRYPT 2023*, Lecture Notes in Computer Science. Springer, 2023.

[CCC+25] Ignacio Cascudo, Anamaria Costache, Daniele Cozzo, Dario Fiore, Antonio Guimarães, and Eduardo Soria-Vazquez. Verifiable computation for approximate homomorphic encryption schemes. In *Advances in Cryptology – CRYPTO 2025*, volume 16006 of *Lecture Notes in Computer Science*. Springer, 2025. Cryptology ePrint 2025/286, https://eprint.iacr.org/2025/286.

[CCKP19] Shuo Chen, Jung Hee Cheon, Dongwoo Kim, and Daejun Park. Verifiable computing for approximate computation, 2019. Cryptology ePrint 2019/762.

[CCXY18] Ignacio Cascudo, Ronald Cramer, PI Xing, and Chen Yuan. Amortized complexity of information-theoretically secure MPC revisited. In *Advances in Cryptology – CRYPTO 2018*, volume 10993 of *Lecture Notes in Computer Science*, pages 395–426. Springer, 2018. Cryptology ePrint 2018/429.

[CFS17] Alessandro Chiesa, Michael A Forbes, and Nicholas Spooner. A zero knowledge sumcheck and its applications. *arXiv preprint arXiv:1704.02086*, 2017.

[CG22] Ignacio Cascudo and Emanuele Giunta. On interactive oracle proofs for Boolean R1CS statements. In *Financial Cryptography and Data Security – FC 2022*, volume 13411 of *Lecture Notes in Computer Science*, pages 230–247. Springer, 2022. Cryptology ePrint 2021/694.

[CRX21] Ronald Cramer, Matthieu Rambaud, and Chaoping Xing. Asymptotically-good arithmetic secret sharing over $\mathbb {Z}/p^{\ell }\mathbb {Z}$ with strong multiplication and its applications to efficient MPC. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, Lecture Notes in Computer Science, pages 656–686. Springer, 2021.

[DCB25] Trisha Datta, Binyi Chen, and Dan Boneh. Veritas: Verifying image transformations at scale. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 4606–4623. IEEE, 2025.

[DHRR26] Rahul Dalal, Tamir Hemo, Eugene Rabinovich, and Ron Rothblum. VEIL: Lightweight zero-knowledge for hash-based multilinear proof systems. Cryptology ePrint Archive, Paper 2026/683, 2026.

[DP25] Benjamin E. Diamond and Jim Posen. Succinct arguments over towers of binary fields. In *Advances in Cryptology – EUROCRYPT 2025*, volume 15604 of *Lecture Notes in Computer Science*, pages 93–122. Springer, 2025. Cryptology ePrint 2023/1784, https://eprint.iacr.org/2023/1784.

[EHL+23] Daniel Escudero, Cheng Hong, Hongqing Liu, Chaoping Xing, and Chen Yuan. Degree-D reverse multiplication-friendly embeddings: Constructions and applications. In *Advances in Cryptology – ASIACRYPT 2023*, volume 14438 of *Lecture Notes in Computer Science*, pages 106–138. Springer, 2023. Cryptology ePrint 2023/173, https://eprint.iacr.org/2023/173.

[GGJ+23] Sanjam Garg, Aarushi Goel, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Guru-Vamsi Policharla, and Mingyuan Wang. Experimenting with zero-knowledge proofs of training. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1880–1894, 2023.

[GLS+23] Alexander Golovnev, Jonathan Lee, Srinath Setty, Justin Thaler, and Riad S Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In *Annual International Cryptology Conference*, pages 193–226. Springer, 2023.

[GNSV23] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *Journal of Cryptology*, 36(4):41, 2023. Cryptology ePrint 2021/322, https://eprint.iacr.org/2021/322.

[Gru24] Angus Gruen. Some improvements for the PIOP for ZeroCheck. Cryptology ePrint Archive, Paper 2024/108, 2024.

[GWHD25] Albert Garreta, Hendrik Waldner, Katerina Hristova, and Luca Dall'Ava. Zinc: Succinct arguments with small arithmetization overheads from IOPs of proximity to the integers, 2025. Cryptology ePrint 2025/316.

[HMZ25] Mi-Ying (Miryam) Huang, Xinyu Mao, and Jiapeng Zhang. Sublinear proofs over polynomial rings, 2025. Cryptology ePrint 2025/199.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In *Advances in Cryptology – CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2014. Cryptology ePrint 2014/106.

[JLX+25] Yuhao Jia, Songsong Li, Chaoping Xing, Yizhou Yao, and Chen Yuan. Polynomial commitments for Galois rings and applications to SNARKs over $\mathbb{Z}_{2^k}$. In *Advances in Cryptology – CRYPTO 2025*, volume 16005 of *Lecture Notes in Computer Science*, pages 515–548. Springer, 2025.

[LXYY23] Fuchun Lin, Chaoping Xing, Yizhou Yao, and Chen Yuan. Amortized NISC over $\mathbb{Z}_{2^k}$ from RMFE. In *Advances in Cryptology – ASIACRYPT 2023*, volume 14438 of *Lecture Notes in Computer Science*, pages 38–70. Springer, 2023. Cryptology ePrint 2023/1363, https://eprint.iacr.org/2023/1363.

[LXZ21] Tianyi Liu, Xiang Xie, and Yupeng Zhang. zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2968–2985, 2021.

[NS24] Ngoc Khanh Nguyen and Gregor Seiler. Greyhound: Fast polynomial commitments from lattices. In *Advances in Cryptology – CRYPTO 2024*, Lecture Notes in Computer Science. Springer, 2024. Cryptology ePrint 2024/1293, https://eprint.iacr.org/2024/1293.

[NT16] Assa Naveh and Eran Tromer. PhotoProof: Cryptographic image authentication for any set of permissible transformations. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 255–271. IEEE, 2016.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252, 2013.

[plo] Plonky3. https://github.com/Plonky3/plonky3.

[SCG+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*, pages 459–474. IEEE, 2014.

[Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *Advances in Cryptology – CRYPTO 2020*, pages 704–737, 2020.

[SLZ24] Haochen Sun, Jason Li, and Hongyang Zhang. zkLLM: Zero knowledge proofs for large language models. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, 2024.

[SV22] Eduardo Soria-Vazquez. Doubly efficient interactive proofs over infinite and non-commutative rings. In *Theory of Cryptography – TCC 2022*, Lecture Notes in Computer Science, pages 497–525. Springer, 2022. Cryptology ePrint 2022/587, https://eprint.iacr.org/2022/587.

[vdZ] Jasper van der Zwet. Reverse multiplication friendly embeddings over ring extensions and their application to multiparty computation.

[vit] An Incomplete Guide to Rollups. https://vitalik.ca/general/2021/01/05/rollup.html.

[WYX+21] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 501–518, 2021.

[WZD25a] Yuanju Wei, Xinxuan Zhang, and Yi Deng. Transparent SNARKs over Galois rings. In *Public-Key Cryptography – PKC 2025*, Lecture Notes in Computer Science. Springer, 2025. Cryptology ePrint 2025/263.

[WZD+25b] Zhuo Wu, Xinxuan Zhang, Yi Deng, Yuanju Wei, Zhongliang Zhang, and Liuyu Yang. Polylogarithmic polynomial commitment scheme over Galois rings. In *Computer Security – ESORICS 2025*, Lecture Notes in Computer Science. Springer, 2025. Cryptology ePrint 2025/1767.

[XZZ+19] Tiacheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *Annual International Cryptology Conference*, pages 733–764. Springer, 2019.

[ZFZS20] Jiaheng Zhang, Zhiyong Fang, Yupeng Zhang, and Dawn Song. Zero knowledge proofs for decision tree predictions and accuracy. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 2039–2053, 2020.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation*, 1979.

[ZOZ26] Tianyu Zhang, Yupeng Ouyang, and Yupeng Zhang. Dynark: Making Groth16 dynamic. In *Advances in Cryptology – EUROCRYPT 2026*, pages 178–208. Springer, 2026.

A Instantiation of Multiplicative Packing

We instantiate the degree-2 multiplicative packing of Section 3.3 concretely as $\phi : \mathbb{Z}_{2^k}^{32} \hookrightarrow \mathrm{GR}(2^k, 128)$, with packing size $\kappa = 32$ and rate 1/4. A larger rate is achievable, for example $\phi : \mathbb{Z}_{2^k}^{42} \hookrightarrow \mathrm{GR}(2^k, 135)$ [vdZ,CRX21] via interpolation at the point at infinity, but it makes each ring operation costlier with no clear concrete gain. We give the construction in Section A.1 and analyze its image, kernel, and reduction in Section A.2.

A.1 The Two-Layer Interpolation Construction

A degree-2 multiplicative packing is built from Lagrange interpolation. We view a large-ring element as a polynomial with small-ring coefficients, taken modulo an irreducible g, and pack the small-ring elements as its evaluation values. Multiplication is then preserved on the evaluations: the product of two packings agrees with the packing of the element-wise product up to a quotient term that unpacking discards. This holds as long as the product stays below $\deg g$, so that reducing modulo g causes no wrap-around.

Theorem A.1 (Polynomial interpolation as multiplicative packing [CCXY18, vdZ])**.** *Let $E = \{e_i\}_{i\in[s]} \subset \mathrm{GR}(2^k, d)$ be unit-separated (so $s \le 2^d$), with Lagrange basis $\{\ell_i\}_{i\in[s]}$, and write the larger ring as $\mathrm{GR}(2^k, dm) = \mathrm{GR}(2^k, d)[X]/(g)$ for a monic g of degree $m \ge 2s-1$ irreducible modulo the maximal ideal. Then Lagrange interpolation gives a degree-2 multiplicative packing of size s. The packing $\phi : \mathrm{GR}(2^k, d)^s \hookrightarrow \mathrm{GR}(2^k, dm)$ writes its inputs as the Lagrange coefficients of the interpolating polynomial,*

$$\phi(a_0, \ldots, a_{s-1}) = \sum_{i\in[s]} a_i\, \ell_i(X) \bmod g(X),$$

and the unpacking $\psi : \mathrm{GR}(2^k, dm) \twoheadrightarrow \mathrm{GR}(2^k, d)^s$ evaluates the representative $\bar{c} \in \mathrm{GR}(2^k, d)[X]$ of c over E, $\psi(c) = (\bar{c}(e_0), \ldots, \bar{c}(e_{s-1}))$.

Starting from $\mathbb{Z}_{2^k}$, whose Teichmüller set $\{0, 1\}$ has size 2, a single layer packs only two elements, into a ring far too small for the proof system's soundness. We therefore compose two layers, lifting through an intermediate ring $\mathrm{GR}(2^k, 4)$ and packing again.

Theorem A.2 (Concatenation of Multiplicative Packing [CCXY18, vdZ])**.** *Given degree-2 multiplicative packings $\phi_0 : \mathrm{GR}(2^k, d_0)^{s_0} \to \mathrm{GR}(2^k, d_1)$ and $\phi_1 : \mathrm{GR}(2^k, d_1)^{s_1} \to \mathrm{GR}(2^k, d_2)$, the concatenated packing $\phi : \mathrm{GR}(2^k, d_0)^{s_0 s_1} \to \mathrm{GR}(2^k, d_2)$ constructed as $\phi(\boldsymbol{a}) := \phi_1\big(\phi_0(\boldsymbol{a}^{(1)}), \ldots, \phi_2(\boldsymbol{a}^{(s_1)})\big)$ where $\boldsymbol{a}^{(i)}$ are size-s_0 blocks in $\boldsymbol{a}$, is also a degree-2 multiplicative packing.*

The ring tower. We instantiate the concatenation with a size-2 packing over $\mathbb{Z}_{2^k}$ and a size-16 packing over the intermediate ring. This realizes $R = R_0$ and $S = R_2$ as the ends of a tower $R_0 \hookrightarrow R_1 \hookrightarrow R_2$,

$$\begin{aligned} R_0 &:= \mathrm{GR}(2^k, d_0) = \mathbb{Z}_{2^k}, && d_0 = 1, \\ R_1 &:= \mathrm{GR}(2^k, d_1) = R_0[\alpha] \,/\, (f_1(\alpha)), && d_1 = 4, \\ R_2 &:= \mathrm{GR}(2^k, d_2) = R_1[\beta] \,/\, (f_2(\beta)), && d_2 = 128, \end{aligned}$$

with f_1 monic of degree $d_1 = 4$ and f_2 of degree $d_2/d_1 = 32$, each irreducible modulo the maximal ideal. Given the unit-separated Teichmüller set $\mathcal{T}_0 \subset R_0$ that $|\mathcal{T}_0| = 2$, the first layer $\phi_0 : R_0^2 \to R_1$ packs elements as $\phi_0(a_0, a_1) := a_0\ell_0(\alpha) + a_1\ell_1(\alpha)$ where $\ell^0(\alpha)$,$\ell^1(\alpha)$ are Lagrange basis over $\mathcal{T}_0$. Though the packing size of this first layer is small, it serves to boost the size of unit-separated set $\mathcal{T}_1 \subset R_1$ that $|\mathcal{T}_1| = 16$ for the second layer $\phi_0 : R_1^{16} \to R_2$, where it packs elements as $\phi_1(b_0, \ldots, b_{15}) := \sum_{i\in[16]} b_i\ell_i(\beta)$. The degrees of irreducible polynomials are chosen as 4 and 32 respectively so that they are both larger than the degree-2 products at each layer, as well as making the final layer $\mathrm{GR}(2^k, 128)$ to achieve 128 security bits.

A.2 Efficient Reduction via Quotient Decomposition

We give explicit formulas for the error term, which is a quotient in our construction using quotient decomposition [ZOZ26], which is based on $\ell_i \ell_j = q_{ij} v_E + \delta_{ij} \ell_i$ with the vanishing polynomial $v_E := \prod_{t \in E}(X - t)$ for some q_{ij} of degree $\leq s - 2$.

Theorem A.3 (Quotient decomposition for R_2)**.** *Let* $\hat{a} = \phi(\boldsymbol{a}), \hat{b} = \phi(\boldsymbol{b}), \hat{c} = \phi(\boldsymbol{a} \circ \boldsymbol{b})$ *for* $\boldsymbol{a}, \boldsymbol{b} \in R_0^{32}$*. Then*

$$\hat{a} \cdot \hat{b} = \hat{c} + \hat{q}_0(\alpha, \beta)\hat{v}_0(\alpha) + \hat{q}_1(\alpha, \beta)\hat{v}_1(\beta). \tag{2}$$

where $\hat{q}_0(\alpha, \beta) = \sum_{i \in [16]} \left(\sum_{j,k \in [2]} (a_{i,j} b_{i,k}) q_{j,k}(\alpha) \right) \ell_i(\beta)$, $\hat{q}_1(\alpha, \beta) = \sum_{i,j \in [16]} \left(\sum_{k,l \in [2]} a_{i,k} b_{j,l} \ell_k(\alpha) \ell_l(\alpha) \right) q_{i,j}(\beta)$

Proof. All products below stay under $\deg f_2 = 32$ and $\deg f_1 = 4$, hence are computed in R_2 without reduction. Write the layer-one packings $\hat{a}_i := \phi_0(a_{i,0}, a_{i,1}) = \sum_{k \in [2]} a_{i,k}\, \ell_k(\alpha)$, so that $\hat{a} = \sum_{i \in [16]} \hat{a}_i\, \ell_i(\beta)$, and likewise $\hat{b}$. On $\mathcal{T}_1$, the identity $\ell_i(\beta)\ell_j(\beta) = q_{i,j}(\beta)\, \hat{v}_1(\beta) + \delta_{ij}\, \ell_i(\beta)$ gives $\hat{a}\, \hat{b} = \sum_{i,j \in [16]} \hat{a}_i \hat{b}_j\, \ell_i(\beta)\ell_j(\beta)$, which equals $\hat{q}_1(\alpha, \beta)\, \hat{v}_1(\beta) + \sum_{i \in [16]} \hat{a}_i \hat{b}_i\, \ell_i(\beta)$. On $\mathcal{T}_0$, the same identity gives $\hat{a}_i \hat{b}_i = \big(\sum_{j,k \in [2]} a_{i,j} b_{i,k}\, q_{j,k}(\alpha)\big)\hat{v}_0(\alpha) + \phi_0(a_{i,0} b_{i,0}, a_{i,1} b_{i,1})$; since $\sum_i \phi_0(a_{i,0} b_{i,0}, a_{i,1} b_{i,1})\, \ell_i(\beta) = \phi(\boldsymbol{a} \circ \boldsymbol{b}) = \hat{c}$, the tail $\sum_i \hat{a}_i \hat{b}_i\, \ell_i(\beta)$ equals $\hat{q}_0(\alpha, \beta)\, \hat{v}_0(\alpha) + \hat{c}$. Combining gives (2). □

Computing the reduction. With constant size preprocessing of $q_{j,k}(\alpha)\ell_i(\beta)$ and $q_{i,j}(\beta)\ell_k(\alpha)\ell_l(\alpha)$, the prover can compute the error terms using only Galois ring additions without any Galois ring multiplication or NTT, significantly improving the reduction time, which becomes one of the bottlenecks for the general circuit.

B Deferred Proofs

B.1 Prime-Field Lift of Linear Codes

We record the prime generalization of the two-step lift; the binary case (Theorems 5.1 and 5.2) is $p = 2$. Throughout, fix a prime p, integers $k, d \geq 1$, and an integer generator matrix $\mathbf{G} \in \mathbb{Z}^{n \times k_2}$ of message length k_2. Set $R := \mathbb{Z}_{p^k}$, $S := \mathrm{GR}(p^k, d)$, and let $\mathrm{Enc}_N(\boldsymbol{m}) := \mathbf{G}\boldsymbol{m} \bmod N$ over each of $\mathbb{F}_p$, R, and (under the R-action on S) S, with codes $\mathcal{C}_N := \mathrm{Img}(\mathrm{Enc}_N)$.

Theorem B.1 (Prime-field two-step lift)**.** *If* Enc_p *is injective, then so are* Enc_{p^k} *and* Enc_S*, and*

$$d_{\min}(\mathcal{C}_p) \;=\; d_{\min}(\mathcal{C}_{p^k}) \;=\; d_{\min}(\mathcal{C}_S).$$

Proof. The two stages generalize Theorem 5.1 and Theorem 5.2 respectively, replacing the prime 2 by p.

Stage 1 ($\mathbb{F}_p \to R$). This is the valuation-layer argument of Theorem 5.1, with "odd" replaced by "not divisible by p."

We first prove $d_{\min}(\mathcal{C}_{p^k})$ is lower bounded by $d_{\min}(\mathcal{C}_p)$. Let $\boldsymbol{m} \in R^{k_2}$ be a nonzero message, with integer representatives in $\{0, 1, \ldots, p^k - 1\}^{k_2}$. Let a be the largest integer such that p^a divides every coordinate of $\boldsymbol{m}$; since $\boldsymbol{m} \not\equiv \mathbf{0} \pmod{p^k}$, we have $0 \leq a \leq k - 1$. Write $\boldsymbol{m} = p^a \boldsymbol{m}'$ with at least one coordinate of $\boldsymbol{m}'$ not divisible by p, so that $\boldsymbol{x} := \boldsymbol{m}' \bmod p \in \mathbb{F}_p^{k_2}$ is nonzero. By integer linearity, $\mathrm{Enc}_{p^k}(\boldsymbol{m}) \equiv p^a\, \mathbf{G}\boldsymbol{m}' \pmod{p^k}$. Whenever the i-th coordinate of $\mathrm{Enc}_p(\boldsymbol{x})$ is nonzero, the integer $(\mathbf{G}\boldsymbol{m}')_i$ is not divisible by p, and multiplying it by p^a with $a < k$ cannot produce zero mod p^k. Hence $\mathrm{supp}(\mathrm{Enc}_p(\boldsymbol{x})) \subseteq \mathrm{supp}(\mathrm{Enc}_{p^k}(\boldsymbol{m}))$. Since $\boldsymbol{x} \neq \mathbf{0}$ and Enc_p is injective, $\Delta(\mathrm{Enc}_{p^k}(\boldsymbol{m})) \geq \Delta(\mathrm{Enc}_p(\boldsymbol{x})) \geq d_{\min}(\mathcal{C}_p) \geq 1$. In particular $\mathrm{Enc}_{p^k}(\boldsymbol{m}) \neq \mathbf{0}$, so Enc_{p^k} is injective; taking the minimum over nonzero $\boldsymbol{m}$ gives $d_{\min}(\mathcal{C}_{p^k}) \geq d_{\min}(\mathcal{C}_p)$.

We then prove $d_{\min}(\mathcal{C}_{p^k})$ is upper bounded by $d_{\min}(\mathcal{C}_p)$. Choose $\boldsymbol{x} \in \mathbb{F}_p^{k_2}$ with $\Delta(\mathrm{Enc}_p(\boldsymbol{x})) = d_{\min}(\mathcal{C}_p)$, lift it coordinate-wise to $\{0, \ldots, p-1\}^{k_2}$, and set $\boldsymbol{m} := p^{k-1}\boldsymbol{x} \in R^{k_2}$. Then $\mathrm{Enc}_{p^k}(\boldsymbol{m}) \equiv$

$p^{k-1}\mathbf{G}\boldsymbol{x} \pmod{p^k}$, and for any integer t we have $p^{k-1}t \not\equiv 0 \pmod{p^k}$ if and only if $p \nmid t$. Hence the nonzero coordinates of $\mathrm{Enc}_{p^k}(\boldsymbol{m})$ are exactly those of $\mathrm{Enc}_p(\boldsymbol{x})$, so $\Delta(\mathrm{Enc}_{p^k}(\boldsymbol{m})) = d_{\min}(\mathcal{C}_p)$ and $d_{\min}(\mathcal{C}_{p^k}) \le d_{\min}(\mathcal{C}_p)$.

Stage 2 ($R \to S$). This is the slicing argument of Theorem 5.2, unchanged. Fix an R-basis $\beta_1, \dots, \beta_d$ of the free rank-d module S. Every $\boldsymbol{m} \in S^{k_2}$ decomposes uniquely as $\boldsymbol{m} = \sum_{\ell=1}^{d} \beta_\ell \, \boldsymbol{m}^{(\ell)}$ with $\boldsymbol{m}^{(\ell)} \in R^{k_2}$, and since $\mathbf{G}$ has integer entries, encoding commutes with the decomposition: $\mathbf{G}\boldsymbol{m} = \sum_\ell \beta_\ell \, (\mathbf{G}\boldsymbol{m}^{(\ell)})$. By uniqueness of the basis expansion, $(\mathbf{G}\boldsymbol{m})_j = 0$ if and only if $(\mathbf{G}\boldsymbol{m}^{(\ell)})_j = 0$ for every ℓ, so $\mathrm{supp}(\mathbf{G}\boldsymbol{m}) = \bigcup_\ell \mathrm{supp}(\mathbf{G}\boldsymbol{m}^{(\ell)})$. In particular $\mathbf{G}\boldsymbol{m} = \mathbf{0}$ forces every $\boldsymbol{m}^{(\ell)} = \mathbf{0}$, hence Enc_S is injective.

We first prove the lower bound. If $\mathbf{G}\boldsymbol{m} \neq \mathbf{0}$, pick ℓ^* with $\mathbf{G}\boldsymbol{m}^{(\ell^*)} \neq \mathbf{0}$; then $\boldsymbol{m}^{(\ell^*)} \in R^{k_2}$ is nonzero, so $\Delta(\mathbf{G}\boldsymbol{m}) \ge \Delta(\mathbf{G}\boldsymbol{m}^{(\ell^*)}) \ge d_{\min}(\mathcal{C}_{p^k})$, giving $d_{\min}(\mathcal{C}_S) \ge d_{\min}(\mathcal{C}_{p^k})$. For the matching upper bound, pick $\boldsymbol{m}^* \in R^{k_2}$ with $\Delta(\mathbf{G}\boldsymbol{m}^*) = d_{\min}(\mathcal{C}_{p^k})$ and set $\boldsymbol{m} := \beta_1 \, \boldsymbol{m}^* \in S^{k_2}$. Then $\mathbf{G}\boldsymbol{m} = \beta_1 \, (\mathbf{G}\boldsymbol{m}^*)$, whose basis decomposition places $\mathbf{G}\boldsymbol{m}^*$ in the first slot and zero elsewhere, so $\mathrm{supp}(\mathbf{G}\boldsymbol{m}) = \mathrm{supp}(\mathbf{G}\boldsymbol{m}^*)$ and $d_{\min}(\mathcal{C}_S) \le d_{\min}(\mathcal{C}_{p^k})$. □

The key invariants are the lowest nonzero p-adic layer of the message (Stage 1) and the basis decomposition of S over R (Stage 2). Neither uses $p = 2$.

B.2 Proof of Theorem 5.1

Proof. We first prove $d_{\min}(\mathcal{C}_{2^k})$ is lower bounded by $d_{\min}(\mathcal{C}_2)$. Let $\boldsymbol{m} \in \mathbb{Z}_{2^k}^{k_2}$ be a nonzero message, with values in $\{0, 1, \dots, 2^k - 1\}^{k_2}$. Let a be the largest integer such that 2^a divides every coordinate of $\boldsymbol{m}$; since $\boldsymbol{m} \not\equiv \mathbf{0} \pmod{2^k}$, we have $0 \le a \le k-1$. Write $\boldsymbol{m} = 2^a \boldsymbol{m}'$ with at least one coordinate of $\boldsymbol{m}'$ odd, so that $\boldsymbol{x} := \boldsymbol{m}' \bmod 2 \in \mathbb{F}_2^{k_2}$ is nonzero. By linearity, $\mathrm{Enc}_{2^k}(\boldsymbol{m}) \equiv 2^a \, \mathbf{G}\boldsymbol{m}' \pmod{2^k}$. Whenever the i-th coordinate of $\mathrm{Enc}_2(\boldsymbol{x})$ equals 1, the integer $(\mathbf{G}\boldsymbol{m}')_i$ is odd, and multiplying an odd integer by 2^a with $a < k$ cannot produce zero mod 2^k. Hence $\mathrm{supp}(\mathrm{Enc}_2(\boldsymbol{x})) \subseteq \mathrm{supp}(\mathrm{Enc}_{2^k}(\boldsymbol{m}))$. Since $\boldsymbol{x} \neq \mathbf{0}$ and Enc_2 is injective, $\Delta(\mathrm{Enc}_{2^k}(\boldsymbol{m})) \ge \Delta(\mathrm{Enc}_2(\boldsymbol{x})) \ge d_{\min}(\mathcal{C}_2) \ge 1$. In particular $\mathrm{Enc}_{2^k}(\boldsymbol{m}) \neq \mathbf{0}$, so Enc_{2^k} is injective; taking the minimum over nonzero $\boldsymbol{m}$ gives $d_{\min}(\mathcal{C}_{2^k}) \ge d_{\min}(\mathcal{C}_2)$.

We then prove $d_{\min}(\mathcal{C}_{2^k})$ is upper bounded by $d_{\min}(\mathcal{C}_2)$. Choose $\boldsymbol{x} \in \mathbb{F}_2^{k_2}$ with $\Delta(\mathrm{Enc}_2(\boldsymbol{x})) = d_{\min}(\mathcal{C}_2)$, lift it coordinate-wise to a $\{0,1\}$-integer vector, and set $\boldsymbol{m} := 2^{k-1}\boldsymbol{x} \in \mathbb{Z}_{2^k}^{k_2}$. Then $\mathrm{Enc}_{2^k}(\boldsymbol{m}) \equiv 2^{k-1} \, \mathbf{G}\boldsymbol{x} \pmod{2^k}$, and for any integer b, $2^{k-1}b \not\equiv 0 \pmod{2^k}$ if and only if b is odd. Hence the nonzero coordinates of $\mathrm{Enc}_{2^k}(\boldsymbol{m})$ are exactly the 1-coordinates of $\mathrm{Enc}_2(\boldsymbol{x})$, so $\Delta(\mathrm{Enc}_{2^k}(\boldsymbol{m})) = d_{\min}(\mathcal{C}_2)$ and $d_{\min}(\mathcal{C}_{2^k}) \le d_{\min}(\mathcal{C}_2)$. □

B.3 Proof of Theorem 5.2

Proof. Every $\boldsymbol{m} \in S^{k_2}$ decomposes uniquely as $\boldsymbol{m} = \sum_{\ell=1}^{d} \beta_\ell \, \boldsymbol{m}^{(\ell)}$ with $\boldsymbol{m}^{(\ell)} \in R^{k_2}$. Since $\mathbf{G}$ has integer entries acting on S via its R-module structure, encoding commutes with the decomposition: $\mathbf{G}\boldsymbol{m} = \sum_{\ell=1}^{d} \beta_\ell \, (\mathbf{G}\boldsymbol{m}^{(\ell)})$. By uniqueness of the basis expansion, for each coordinate j, $(\mathbf{G}\boldsymbol{m})_j = 0 \iff (\mathbf{G}\boldsymbol{m}^{(\ell)})_j = 0$ for every ℓ, so $\mathrm{supp}(\mathbf{G}\boldsymbol{m}) = \bigcup_\ell \mathrm{supp}(\mathbf{G}\boldsymbol{m}^{(\ell)})$. In particular $\mathbf{G}\boldsymbol{m} = \mathbf{0}$ forces every $\mathbf{G}\boldsymbol{m}^{(\ell)} = \mathbf{0}$, hence every $\boldsymbol{m}^{(\ell)} = \mathbf{0}$ since Enc_{2^k} is injective, so Enc_S is injective.

We first prove $d_{\min}(\mathcal{C}_S)$ is lower bounded by $d_{\min}(\mathcal{C}_{2^k})$. If $\mathbf{G}\boldsymbol{m} \neq \mathbf{0}$, pick ℓ^* with $\mathbf{G}\boldsymbol{m}^{(\ell^*)} \neq \mathbf{0}$; then $\boldsymbol{m}^{(\ell^*)} \in R^{k_2}$ is nonzero, so $\Delta(\mathbf{G}\boldsymbol{m}) \ge \Delta(\mathbf{G}\boldsymbol{m}^{(\ell^*)}) \ge d_{\min}(\mathcal{C}_{2^k})$, giving $d_{\min}(\mathcal{C}_S) \ge d_{\min}(\mathcal{C}_{2^k})$.

We then prove $d_{\min}(\mathcal{C}_S)$ is upper bounded by $d_{\min}(\mathcal{C}_{2^k})$. Pick $\boldsymbol{m}^* \in R^{k_2}$ with $\Delta(\mathbf{G}\boldsymbol{m}^*) = d_{\min}(\mathcal{C}_{2^k})$ and set $\boldsymbol{m} := \beta_1 \, \boldsymbol{m}^* \in S^{k_2}$. Then $\mathbf{G}\boldsymbol{m} = \beta_1 \, (\mathbf{G}\boldsymbol{m}^*)$, and for any $c \in R$ the basis decomposition of $\beta_1 c$ has c in the first slot and 0 elsewhere, so $\beta_1 c = 0 \iff c = 0$. Hence $\mathrm{supp}(\mathbf{G}\boldsymbol{m}) = \mathrm{supp}(\mathbf{G}\boldsymbol{m}^*)$ and $d_{\min}(\mathcal{C}_S) \le d_{\min}(\mathcal{C}_{2^k})$. □

Corollary B.2 (Binary code to Galois-ring code)**.** *If* Enc_2 *is injective, then for* $k, d \geq 1$ *and* $S = \mathrm{GR}(2^k, d)$*,*

$$d_{\min}(\mathcal{C}_2) = d_{\min}(\mathcal{C}_{2^k}) = d_{\min}(\mathcal{C}_S),$$

so $\mathcal{C}_S$ *has the same rate and relative distance as the binary code* $\mathcal{C}_2$*.*

B.4 Proof of Theorem 5.6

Proof. Since V is complemented, fix an R-basis $\beta_1, \ldots, \beta_d$ of S adapted to V: $V = \mathrm{span}_R\{\beta_\ell : \ell \in I\}$ for some $I \subsetneq [d]$, so $S/V \cong R^{d-|I|}$ via the complementary basis. Decompose $\boldsymbol{m} \in S^{k_2}$ as $\boldsymbol{m} = \sum_{\ell=1}^{d} \beta_\ell \, \boldsymbol{m}^{(\ell)}$ with $\boldsymbol{m}^{(\ell)} \in R^{k_2}$; because $\mathbf{G}$ has integer entries, $\mathbf{G}\boldsymbol{m} = \sum_\ell \beta_\ell \, (\mathbf{G}\boldsymbol{m}^{(\ell)})$, so $\overline{\mathrm{Enc}}(\bar{\boldsymbol{m}})_j = 0$ if and only if $(\mathbf{G}\boldsymbol{m}^{(\ell)})_j = 0$ for every $\ell \notin I$.

We first prove $d_{\min}(\overline{\mathcal{C}}_V)$ is lower bounded by $d_{\min}(\mathcal{C}_S) = d_{\min}(\mathcal{C}_2)$ (Corollary B.2). If $\bar{\boldsymbol{m}} \neq \mathbf{0}$ in $(S/V)^{k_2}$, then $\boldsymbol{m}^{(\ell^*)} \neq \mathbf{0}$ in R^{k_2} for some $\ell^* \notin I$, so $\Delta(\mathbf{G}\boldsymbol{m}^{(\ell^*)}) \geq d_{\min}(\mathcal{C}_2)$ by Theorem 5.1, and hence $\Delta(\overline{\mathrm{Enc}}(\bar{\boldsymbol{m}})) \geq d_{\min}(\mathcal{C}_2)$.

For the matching upper bound, pick any $\ell^* \notin I$ and $\boldsymbol{m}^* \in R^{k_2}$ with $\Delta(\mathbf{G}\boldsymbol{m}^*) = d_{\min}(\mathcal{C}_2)$, and set $\boldsymbol{m} := \beta_{\ell^*} \, \boldsymbol{m}^* \in S^{k_2}$. Then $\mathbf{G}\boldsymbol{m} = \beta_{\ell^*} \, (\mathbf{G}\boldsymbol{m}^*)$, whose basis decomposition places $\mathbf{G}\boldsymbol{m}^*$ in the ℓ^*-th slot and zero elsewhere, so $\mathrm{supp}(\overline{\mathrm{Enc}}(\bar{\boldsymbol{m}})) = \mathrm{supp}(\mathbf{G}\boldsymbol{m}^*)$ and $\Delta(\overline{\mathrm{Enc}}(\bar{\boldsymbol{m}})) = d_{\min}(\mathcal{C}_2)$. □

B.5 Proof of Theorem 5.8

Proof. The verifier accepts only when the testing, submodule, and evaluation checks of Protocol 5.1 all pass. We bound the prover's success in three exhaustive cases and take a union bound; throughout, $x_{0,i} := \mathrm{eq}(\mathrm{bin}(i), \boldsymbol{z}_0)$ are the evaluation weights of Protocol 5.1.

Case 1: the interleaved matrix is far from the code, $\Delta^{k_1}(\{\boldsymbol{u}_i\}, \mathcal{C}^{k_1}) > e$. By the proximity gap (Theorem 3.10), the combination $\sum_i r_i \boldsymbol{u}_i$ is e-far from $\mathcal{C}$ except with probability $(e+1)/|E|$. The verifier re-encodes $\bar{\boldsymbol{m}}^r$ to $\mathrm{Enc}(\bar{\boldsymbol{m}}^r) \in \mathcal{C}$, which is then e-far from $\sum_i r_i \boldsymbol{u}_i$, so the column check $\mathrm{Enc}(\bar{\boldsymbol{m}}^r)_j = \sum_i r_i u_{i,j}$ fails on at least $e+1$ positions and a random column catches it with probability $\geq (e+1)/n$. Hence $\epsilon_{\mathrm{test}} \leq (1-(e+1)/n)^t + (e+1)/|E|$.

In the remaining cases $\{\boldsymbol{u}_i\}$ is e-close to a unique codeword tuple $\{\boldsymbol{c}_i\} \subset \mathcal{C}$, and we split on the submodule check.

Case 2.1: some $\boldsymbol{c}_i \notin V^n$. Its message $\boldsymbol{m}_i$ has a coordinate outside V, so Lemma 5.5 lower-bounds the number of coordinates of $\boldsymbol{c}_i = \mathrm{Enc}(\boldsymbol{m}_i)$ outside V by the quotient distance $d_{\min}(\overline{\mathcal{C}}_V)$, which Corollary 5.7 evaluates to δn. Closeness $\Delta(\boldsymbol{u}_i, \boldsymbol{c}_i) \leq e$ leaves $\boldsymbol{u}_i$ with at least $\delta n - e$ such coordinates, each caught by a random column with probability $\geq \delta - e/n$. Hence $\epsilon_{\mathrm{sub}} \leq (1 - \delta + e/n)^t$.

Case 2.2: all $\boldsymbol{c}_i \in V^n$. The verifier re-encodes $\bar{\boldsymbol{m}}^z$ to $\bar{\boldsymbol{u}}^z := \mathrm{Enc}(\bar{\boldsymbol{m}}^z) \in \mathcal{C}$. If $\bar{\boldsymbol{u}}^z \neq \sum_i x_{0,i} \boldsymbol{c}_i$, the two codewords disagree on at least δn positions, while matrix-vs-codeword closeness gives $\sum_i x_{0,i} \boldsymbol{u}_i = \sum_i x_{0,i} \boldsymbol{c}_i$ on at least $n - e$ positions. The column check $\bar{u}^z_j = \sum_i x_{0,i} u_{i,j}$ then holds on at most $n - \delta n + e$ positions, and a random column catches the discrepancy with probability $\geq \delta - e/n$. Hence $\epsilon_{\mathrm{eval}} \leq (1 - \delta + e/n)^t$.

Substituting $e = \lfloor \delta n/3 \rfloor$ gives $1 - (e+1)/n \leq 1 - \delta/3$ and $1 - \delta + e/n \leq 1 - 2\delta/3$, with $(e+1)/|E| \leq n/|E|$, which are the bounds claimed. □

For the knowledge soundness of the PCS, if the code has efficient decoding, the witness can be decoded as the message of the unique codeword that is close to the interleaved matrix. Otherwise, one can apply the rewinding technique of [GLS+23] to extract the witness. Our submodule and virtual PCS checks do not affect the extraction.

B.6 Proof of Theorem 5.12

Proof. As in Case 1 of the proof of Theorem 5.8, the matrix $\mathbf{U}$ is e-close to an interleaved code, except with probability $\varepsilon_{\mathrm{test}}$. Let $\mathbf{M}^*$ be the message of the unique codeword that is e-close to

U. If $\bar{\boldsymbol{m}}^z \neq \sum_{i\in[k_1]} \mathrm{eq}(\mathrm{bin}(i), \boldsymbol{z}_0)\, L(\boldsymbol{m}_i^*)$, then $\mathrm{Enc}(\bar{\boldsymbol{m}}^z)$ is at least δ-far from $\mathrm{Enc}(L(\bar{\boldsymbol{m}}^{*z})) = L(\mathrm{Enc}(\bar{\boldsymbol{m}}^{*z}))$ by Lemma 5.10. Then the probability that $\mathrm{Enc}(\bar{\boldsymbol{m}}^z)_j = \sum_i \mathrm{eq}(\mathrm{bin}(i), \boldsymbol{z}_0)\, L(u_{i,j})$ for all $j \in Q$ is at most $(1-\delta+e/n)^t \leq (1-2\delta/3)^t$.

□

B.7 Proof Sketch of Theorem 5.13

The correctness of the protocols follows from the definition of the multiplicative packing, the correctness of the sumcheck protocol and the correctness of the PCS in a straightforward way. The proof of knowledge soundness proceeds in three steps: (1) extracting witnesses in the Galois ring from the PCS with image, kernel and rotation constraints; (2) proving the soundness of the PIOP (sumcheck protocol) over the Galois ring; (3) unpacking the witnesses to obtain the witnesses over the base ring.

SIMD. By Theorem 5.8, there exists an extractor that extracts $\hat{\mathbf{w}} \in \mathrm{Img}(\phi), \hat{\boldsymbol{e}}^{\mathrm{mult}} \in \mathrm{Ker}(\psi)$ such that their polynomial evaluations are consistent with those queried by the verifier from the polynomial oracles, except with probability $\varepsilon_{\mathsf{pcs}}$. Then by the soundness of the sumcheck protocol (Theorem 3.12) over equations in Steps 2) and 3) of Protocol 4.1, $\mathbf{A}\hat{\mathbf{w}} \circ \mathbf{B}\hat{\mathbf{w}} = \mathbf{C}\hat{\mathbf{w}} + \hat{\boldsymbol{e}}^{\mathrm{mult}}$, except with probability $\varepsilon_{\mathsf{piop}}$. Finally, by applying ψ to the equation, $\psi(\mathbf{A}\hat{\mathbf{w}} \circ \mathbf{B}\hat{\mathbf{w}}) = \psi(\mathbf{C}\hat{\mathbf{w}})$. Since $\mathbf{A}, \mathbf{B}, \mathbf{C} \in R^{n\times n}$ are scalars, it implies $\mathbf{A}\mathbf{w} \circ \mathbf{B}\mathbf{w} = \mathbf{C}\mathbf{w}$ where $\mathbf{w} \in R^n$ is the image of $\hat{\mathbf{w}}$ by the definition of the multiplicative packing. $\mathbf{w}$ is the extracted witness that satisfies $\mathcal{R}_R^{\mathrm{SIMD}}$, except with a negligible probability of $\varepsilon_{\mathsf{pcs}} + \varepsilon_{\mathsf{piop}}$.

MIMD. By Theorem 5.8, there exists an extractor that extracts $\hat{\mathbf{w}}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}} \in \mathrm{Img}(\phi), \hat{\boldsymbol{e}}^{\mathrm{mult}}, \hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C \in \mathrm{Ker}(\psi)$ such that their polynomial evaluations are consistent with those queried by the verifier from the polynomial oracles, with overwhelming probability. Then by the soundness of the sumcheck protocol over equations in Steps 2) and 3) of Protocol 4.2, $\hat{\boldsymbol{a}} \circ \hat{\boldsymbol{b}} = \hat{\boldsymbol{c}} + \hat{\boldsymbol{e}}^{\mathrm{mult}}$, and $\hat{\mathbf{A}}\hat{\mathbf{w}} = \hat{\boldsymbol{a}} + \hat{\boldsymbol{e}}^A, \hat{\mathbf{B}}\hat{\mathbf{w}} = \hat{\boldsymbol{b}} + \hat{\boldsymbol{e}}^B, \hat{\mathbf{C}}\hat{\mathbf{w}} = \hat{\boldsymbol{c}} + \hat{\boldsymbol{e}}^C$, with overwhelming probability. Finally, by applying ψ to the equations, $\psi(\hat{\mathbf{A}}\hat{\mathbf{w}}) = \psi(\hat{\boldsymbol{a}})$, which implies $\mathbf{A}_i\mathbf{w}_i = \boldsymbol{a}_i$ for $i \in [\kappa]$ where $\mathbf{A}_i, \mathbf{w}_i, \boldsymbol{a}_i$ are the images. Likewise, $\mathbf{B}_i\mathbf{w}_i = \boldsymbol{b}_i$ and $\mathbf{C}_i\mathbf{w}_i = \boldsymbol{c}_i$, and $\boldsymbol{a}_i \circ \boldsymbol{b}_i = \boldsymbol{c}_i$. Therefore, $\mathbf{w}$ is the extracted witness that satisfies $\mathcal{R}_R^{\mathrm{MIMD}}$, except with negligible probability.

General R1CS. By Theorem 5.8, there exists an extractor that extracts $\hat{\mathbf{w}}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{c}} \in \mathrm{Img}(\phi), \hat{\boldsymbol{e}}^{\mathrm{mult}}, \hat{\boldsymbol{e}}^A, \hat{\boldsymbol{e}}^B, \hat{\boldsymbol{e}}^C \in \mathrm{Ker}(\psi)$ such that their polynomial evaluations are consistent with those queried by the verifier from the polynomial oracles, with overwhelming probability. Then by the soundness of the sumcheck protocol over equations in Steps 2) and 3) of Protocol 4.3, $\hat{\boldsymbol{a}} \circ \hat{\boldsymbol{b}} = \hat{\boldsymbol{c}} + \hat{\boldsymbol{e}}^{\mathrm{mult}}$, and $\sum_\ell \hat{\mathbf{D}}_A^{(\ell)} \hat{\boldsymbol{\rho}}^{(\ell)} = \hat{\boldsymbol{a}} + \hat{\boldsymbol{e}}^A$ with overwhelming probability (similar equations hold for $\mathbf{B}$ and $\mathbf{C}$). By Theorem 5.12, the virtual PCS guarantees $\hat{\boldsymbol{\rho}}^{(\ell)} = L_\ell(\hat{\mathbf{w}})$. Finally, by applying ψ to the equations and due to the diagonal identity in Equation (1), the left side unpacks to $\mathbf{A}\mathbf{w}$ while the right unpacks to $\psi(\hat{\boldsymbol{a}})$, such that $\hat{\boldsymbol{a}} = \phi(\mathbf{A}\mathbf{w})$. Similarly, $\hat{\boldsymbol{b}} = \phi(\mathbf{B}\mathbf{w}), \hat{\boldsymbol{c}} = \phi(\mathbf{C}\mathbf{w})$. Therefore, the witness $\mathbf{w}$ extracted from $\hat{\mathbf{w}}$ satisfies $(\mathbf{A}\mathbf{w}) \circ (\mathbf{B}\mathbf{w}) = \mathbf{C}\mathbf{w}$, which is $\mathcal{R}_R^{\mathrm{R1CS}}$, except with negligible probability.