Symphony: Scalable SNARKs in the Random Oracle Model from Lattice-Based High-Arity Folding

Binyi Chen

Stanford University

October 25, 2025

Abstract

Folding schemes are a powerful tool for building scalable proof systems. However, existing folding-based SNARKs require embedding hash functions (modeled as random oracles) into SNARK circuits, introducing both security concerns and significant proving overhead.

We re-envision how to use folding, and introduce Symphony, the first folding-based SNARK that avoids embedding hashes in SNARK circuits. It is memory-efficient, parallelizable, streaming-friendly, plausibly post-quantum secure, with polylogarithmic proof size and verification, and a prover dominated by committing to the input witnesses.

As part of our construction, we introduce a new lattice-based folding scheme that compresses *a large number* of NP-complete statements into one *in a single shot*, which may be of independent interest. Furthermore, we design a generic compiler that converts a folding scheme into a SNARK without embedding the Fiat-Shamir circuit into proven statements. Our evaluation shows its concrete efficiency, making Symphony a promising candidate for applications such as zkVM, proof of learning, and post-quantum aggregate signatures.

Contents

1 Introduction

Succinct non-interactive arguments of knowledge (SNARKs) allow a prover to convince weak devices that a computation was performed correctly using only a short proof. It has enabled new applications such as scaling and securing blockchain [Whi18; Xio+23], digital provenance [NT16; DB22; KHSS22; DCB25], verifiable machine learning [CWSK24; YCBC24], verifiable delay functions [BBBF18], etc. While many efficient SNARKs were introduced [BBHR18; AHIV17; Ben+19; Gol+23; ZCF24; Bre+25; COS20; BS23; ACFY25], they require massive memory for proving large statements. A memory-efficient framework is to split the computation into small steps and prove each step separately. Two main strategies exist: streaming provers [BCHO22; PP24; Baw+24] and incrementally verifiable computation (IVCs) or proof of carrying data (PCDs). Early IVC/PCD systems rely on recursive SNARKs [Val08; BCTV14], which embed a SNARK verifier into each statement. The recursion imposes prohibitive overhead and remained only of theoretical interest.

Recently, a new framework called accumulation or folding was introduced in Halo [BGH19] and further developed in [BCMS20; Bün+21; BDFG21; KST22]. Folding is a public-coin protocol that reduces the task of checking multiple uniform statements to checking one, and can be made non-interactive via the Fiat-Shamir heuristic. Though weaker than SNARKs without the full succinctness, folding is more concretely efficient and still suffices for IVC/PCDs [Bün+21; KST22].

Before our work, the standard (and the only) approach to use folding to obtain succinct proof systems is by folding *recursive* statements. E.g., in an IVC, at each iteration step, the prover folds an accumulated statement and an online statement to obtain a new accumulated statement, and it generates a new online statement that checks a step function and the correctness of the previous folding step. This technique extends to folding trees (or DAGs) to obtain PCDs. Since folding verifiers are much simpler than SNARK verifiers, this achieves significant speedups over recursive SNARKs. Compared to monolithic SNARKs, folding is more streaming-friendly and allows one to start proving without knowing all witnesses upfront. However, this strategy faces a few key limitations.

Instantiating random oracles. Existing folding schemes require Fiat-Shamir for non-interactivity, and hash-based folding schemes [BMNW25a; BMNW25b; BCFW25] invoke even more hashes for Merkle openings. This leads to both efficiency and security concerns.

Efficiency-wise, the folding verifier logic embeds heavy hashing gadgets: A standard hash gadget takes thousands of R1CS constraints; even a SNARK-friendly hash takes hundreds of constraints. To justify, the Fiat-Shamir circuit is dominant in lattice-based folding schemes [BC24; BC25; NS25]; the verifier for hash-based 2-to-1 folding schemes already takes tens of millions of R1CS constraints.

Security-wise, hash-based SNARKs and the Fiat-Shamir transform are only proven secure in the random oracle model. If we instantiate the oracle with concrete hash functions and embed it into proven statements, the scheme's security relies only on heuristic conjectures. Even worse, attacks exist [KRS25] for GKR-based SNARKs if we allow the proven statement to compute the Fiat-Shamir hash function itself.

Limited folding arity. Due to the cost of hashing gadgets, folding typically takes only 2 or 3 inputs per step. Batch proving more input statements then requires deeper folding trees, which degrades parallelism and security. For example, IVCs/PCDs from rewinding-based folding is secure only if the folding depth is a constant, with known attacks to certain schemes for super-logarithmic depths [LS24; BMNW25a]. Some schemes propose folding more inputs per step [Ngu+24; RZ22] but the embedded folding verification circuit becomes more expensive. We thus ask,

> *Can we obtain memory-efficient folding-based SNARKs with streaming provers, without deep folding trees or instantiating random oracles in SNARK circuits?*

1.1 Our Contributions

We re-envision how to use folding. For the first time, we move away from recursive folding and advocate a *high-arity* folding approach. While previously deemed impractical, we introduce a new framework that makes high-arity folding feasible and avoids instantiating random oracles inside SNARK circuits.

First, we construct a new lattice-based *high-arity* folding scheme. It is plausibly post-quantum secure, with a prover only dominated by computing the input witness commitments, and the verifier is mainly just combining the commitments using a random vector generated by the Fiat-Shamir heuristic.

Second, we develop a generic framework that converts a broad class of folding schemes, including all group-based and lattice-based ones, into SNARKs in the *random oracle model*. Each input statement depends only on the application logic, not the folding verifier. We do need an extra proof ensuring the correctness of folding, but it does *not* embed random oracle calls inside the proven statement. In Section 8, we further extend the technique to support higher folding depths.

Evaluation and applications. In Section 7, we present a candidate instantiation for our folding-based SNARKs. It supports efficient proof generation for 2^{16} standard R1CS statements over a 64-bit field, each with over 2^{16} constraints. We expect the resulting proofs to be under 200KB (and under 50KB if post-quantum security is not required), with verification in tens of milliseconds. The prover is memory-efficient, parallelizable, and streaming-friendly, i.e., it can start working as soon as some proven statements are known. (See Remark 4.1 for more details.) The prover cost is dominated only by witness commitments, which takes about $3 \cdot 2^{32}$ multiplications between arbitrary elements and low-norm elements over $R_q := \mathbb{Z}_q[X]/\langle X^{64}+1\rangle$. Further speedup is possible if the R1CS witness already has a low ℓ_∞-norm. For instance, if the witnesses are 8-bit signed integers, we expect another 8x speedup.

Our folding framework is useful in applications where computation decomposes into many uniform statements. Examples include video editing provenance, post-quantum aggregate signatures, zkVMs, and proof of machine learning.

1.2 Technical Overview

Lattice-based high-arity folding. We first review a standard framework for constructing folding schemes from linearly homomorphic commitments. Each input is a **committed R1CS statement**[1]: the instance consists of a public input $\mathbf{X}_{\mathsf{in}}$ and a commitment c; the relation checks that a message-opening pair (m, o) is valid for c and that m satisfies the R1CS constraints. Our lattice-based folding scheme compresses $\ell_{\mathsf{np}} > 1$ R1CS statements in three steps:

1. **Commitment.** The prover commits to the witnesses of the ℓ_{np} statements using a ring/module-based Ajtai commitment [Ajt96; PR06; LM06].
2. **Sumcheck reduction.** Standard techniques [Set20; CBBZ23] transform the R1CS statements to a sumcheck claim. The prover and verifier then run a sumcheck protocol [LFKN92] that reduces the claim to ℓ_{np} linear evaluation statements.
3. **Random linear combination.** The verifier samples a low-norm vector $\boldsymbol{\beta}$. The prover combines the witnesses using $\boldsymbol{\beta}$, and the verifier similarly combines the instances. By linearity of the evaluation statements and the Ajtai commitments, the committed linear relation is preserved after the random linear combination.

The key challenge, however, is in proving the *low-norm* property of the witness openings. This guarantee is essential for the binding property of Ajtai commitments and for proving the knowledge soundness of the folding scheme.

We combine random projection [GHL22; BS23; KLNO25] with the monomial embedding technique [BC25] to obtain a range proof with near-optimal complexity:

Random projection. LaBRADOR [BS23] adapts the technique from [GHL22], reducing the norm-check of a long vector $\mathbf{w}$ to that of a shorter vector $\mathbf{J}\mathbf{w} \bmod q$, where $\mathbf{J} \in \{0, \pm 1\}^{\lambda_{\mathsf{pj}} \times n}$ is a random matrix. Yet, the verifier time is linear in the witness size for generating $\mathbf{J}$. Recently, Klooß et al. [KLNO25] introduces a refinement with sublinear verifiers by using a *structured* projection $\mathbf{J} := \mathbf{I}_{n/\ell_h} \otimes \mathbf{J}'$, where $\mathbf{J} \in \{0, \pm 1\}^{\lambda_{\mathsf{pj}} \times \ell_h}$ is a narrower random matrix. While elegant, the resulting protocol has a sub-optimal prover and the verifier checks certain relations *over integers* that are cumbersome to represent as a circuit over finite fields.

Monomial embedding lookup. To prove that the projected vector $\mathbf{w}' = \mathbf{J}\mathbf{w} \bmod q$ is low-norm, we use the exact range proof from LatticeFold+ [BC25]. Let $R := \mathbb{Z}[X]/\langle X^d + 1\rangle$ be a power-of-two cyclotomic ring, $R_q := R/qR$, and define a table

[1]An R1CS relation is NP-complete and can capture arbitrary computation.

lookup polynomial $t(X) := \sum_{i\in[1,d/2)} i \cdot (X^{-i} + X^i) \in R_q$. On input instance a commitment to an integer vector $\mathbf{f} \in (-d/2, d/2)^n$, the range proof reduces the norm check of $\mathbf{f}$ to a simple linear relation: the prover sends a commitment to a monomial vector $\mathbf{g} \in R_q^n$ (where for $i \in [n]$, $\mathbf{g}_i \approx X^{\mathbf{f}_i}$ embeds $\mathbf{f}_i$ onto exponent) and proves that the constant term of $\mathbf{g}_i \cdot t(X)$ equals $\mathbf{f}_i$ for all $i \in [n]$. The prover cost is approximately $O(n)$ R_q-additions.

LatticeFold+ actually proves the norm of a *ring* vector $\mathbf{w} \in R^n$, which corresponds to an integer coefficient matrix $\mathbf{W} \in \mathbb{Z}^{n\times d}$. A naive extension would require sending d monomial commitments, leading to large communication and expensive verification. LatticeFold+ addresses this via a *commitment transformation* technique, which is highly complex and requires two extra sumchecks that cannot be batched with the one already used for R1CS.

Our simplification. In our setting, we only need to check the norm of a *projected* vector $\mathbf{w}' = \mathbf{J}\mathbf{w} \bmod q \in R_q^{n/\ell_h}$, whose coefficient matrix $\mathbf{W}' \in ([-q/2, q/2) \cap \mathbb{Z})^{(n/\ell_h)\times d}$ has smaller dimensions.[2] By flattening $\mathbf{W}'$ into a vector, we directly apply the monomial embedding protocol *without commitment transformation*. Moreover, the prover only computes $O(1)$ instead of d monomial commitments. After integrating the range proof with the three-step framework above, the prover is mainly for computing witness commitments ($O(n)$ R_q-multiplications per input). The verifier circuit complexity is dominated by the Fiat-Shamir heuristic and the random linear combination of the input instances. However, the Fiat-Shamir circuit size is still large with high folding arity ℓ_{np}. This makes the standard IVC/PCD compiler infeasible.

Memory friendliness. As discussed in Remark 4.1, the folding prover can be implemented in a memory-efficient way, requiring space roughly the same as the witness size n of each input statement. As a tradeoff, the algorithm requires $2 + \log\log(n)$ passes over the input data. Designing a one-pass streaming algorithm with comparable prover complexity in the non-recursive setting remains an open problem.

From folding to SNARKs: The commit-and-prove compiler. We present a new compiler that eliminates the Fiat-Shamir circuit. Before presenting the full scheme, we consider a warm-up that compiles a single-shot folding protocol Π_{fold} into a SNARK:

1. Apply Fiat–Shamir to make the public-coin Π_{fold} non-interactive.
2. The prover executes the non-interactive folding scheme $\mathsf{FS}[\Pi_{\mathsf{fold}}]$ to obtain a folding proof π and a reduced instance $\mathbb{x}_o$. Using the corresponding witness $\mathbb{w}_o$, it generates a SNARK proof π_{snark} for the reduced relation $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_o$ and outputs $(\pi, \mathbb{x}_o, \pi_{\mathsf{snark}})$.
3. The SNARK verifier checks the SNARK proof π_{snark} against $\mathbb{x}_o$ and runs the folding verifier of $\mathsf{FS}[\Pi_{\mathsf{fold}}]$ to check proof π against $\mathbb{x}_o$ and input $\mathbb{x}$.

[2]As a tradeoff, we only achieve approximate range proofs. But this is sufficient in our framework where the folding depth is a small constant.

This naive approach is simple but inefficient for large ℓ_{np}. For $\ell_{\mathsf{np}} = 1000$, the folding proof π easily exceeds 30MB in size with lattice or hash-based schemes. An alternative is embedding the folding verification circuit into the SNARK relation, which is impratical either. For $\ell_{\mathsf{np}} = 1000$, the Fiat-Shamir transform circuit alone blows up to millions of 2-to-1 hashes over cryptographic fields, with hash-based folding schemes being even worse.

We overcome the bottleneck using commit-and-prove SNARKs (CP-SNARKs) [Kil89; CLOS02; CFQ19], a special class of SNARKs proving that a witness satisfies an NP-relation and is a valid opening to the instance commitments. Crucially, the SNARK statement does *not* encode the commitment-opening relation. Our final scheme below is inspired by Protostar [BC23], but the underlying commitments Π_{cm} need not be linearly homomorphic.

1. Convert Π_{fold} into an interactive protocol $\mathsf{CM}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{fold}}]$, where in each round, instead of sending message m_i, the prover sends a commitment $c_{\mathsf{fs},i} = \Pi_{\mathsf{cm}}.\mathsf{Commit}(m_i)$. At the end, the prover further sends the openings $(m_i)_i$. The verifier checks that $(m_i)_i$ are valid openings to the commitments and simulates the verifier of Π_{fold} w.r.t. $(m_i)_i$.

2. Apply Fiat–Shamir to make the public-coin $\mathsf{CM}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{fold}}]$ non-interactive. Let $\mathsf{FS}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{fold}}]$ denote the resulting scheme.

3. Convert $\mathsf{FS}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{fold}}]$ into a SNARK as follows:

   - The prover computes commitments $\{c_{\mathsf{fs},i}\}_i$, folded instance $\mathbb{x}_o$, and a SNARK proof π_{snark} for $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_o$. Let π_{cp} be a CP-SNARK proof of knowledge for messages $\{m_i\}_i$ checking that $[(c_{\mathsf{fs},i})_i, (m_i)_i]$ is a valid folding proof (w.r.t. $\mathsf{FS}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{fold}}]$) for $\mathbb{x}$ and $\mathbb{x}_o$. The prover sends $\{c_{\mathsf{fs},i}\}_i$, $\mathbb{x}_o$, π_{snark}, and π_{cp}.
   - The verifier checks π_{snark} against $\mathbb{x}_o$. It then derives folding verifier challenges $\{r_i\}$ from the FS-transcript $(\mathbb{x}, \{c_{\mathsf{fs},i}\}_i)$ and checks π_{cp} against $(\mathbb{x}, \mathbb{x}_o, \{c_{\mathsf{fs},i}, r_i\}_i)$.

For Π_{cm}, we can use Merkle commitments as in hash-based CP-SNARKs or KZG commitments as in pairing-based CP-SNARKs. Compared to the warm-up, this scheme compresses $>$ 30MB-sized folding proofs into $\log(n)$ Merkle (or KZG) commitments, under 1KB for typical statement sizes.

Crucially, given the property of CP-SNARKs, the CP-SNARK statement embeds no Fiat-Shamir circuits instantiating random oracles, nor checking that $\{m_i\}_i$ open correctly to $\{c_{\mathsf{fs},i}\}_i$. Instead, it only proves $O(\ell_{\mathsf{np}})$ multiplications over R_q (for combining Ajtai commitments). For large input instance $\mathbb{x}$, verifying π_{cp} can be costly. Remark 6.1 describes an optimization that further compresses $\mathbb{x}$.

Higher folding depth without recursive circuits. Our instantiation folds 2^{10} R1CS statements over $R_q = \mathbb{Z}_q[X]/\langle X^{64} + 1\rangle$, equivalent to 2^{16} statements over $\mathbb{Z}_q$. Recent hash-based SNARKs can easily prove more than 2^{24} R1CS constraints in a few seconds. So we can handle more than 2^{40} constraints in total. In principle, even larger arity is possible: e.g., a 2^{14}-way folding batches over a million statements, enough to prove *a billion RISC-V instructions* if each statement covers a thousand instructions. However, higher arity increases norm blowup, forcing larger moduli or lattice dimensions, and requires the CP-SNARK to prove more R_q-multiplications. For extremely large number of statements, we might want to increase folding depth modestly (e.g., depth two).

Our scheme extends to higher folding depths. After obtaining the first layer CP-SNARK proof and the reduced statement $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_o$, we further split $(\mathbb{x}_o, \mathbb{w}_o)$ to multiple uniform NP statements. Our high-arity folding scheme, combined with the commit-and-prove compiler, then reduces these uniform statements again into a CP-SNARK and a SNARK proof. The final output is two CP-SNARK proofs plus one SNARK proof. Importantly, we still avoid embedding Fiat-Shamir heuristics in circuits. In Section 8, we show how to split $(\mathbb{x}_o, \mathbb{w}_o)$ when the Ajtai commitment parameter satisfies a structural property. For general cases, we might use Mangrove's *uniformization technique* [Ngu+24] and we leave the detailed construction to future work.

1.3 Additional Related Work

The most relevant works are LatticeFold+[BC25] and the recent lattice-based SNARK of [KLNO25] discussed in Section 1.1. We summarize other prior work below.

There are two main categories of plausibly post-quantum secure folding schemes. Hash-based folding schemes [BMNW25a; BMNW25b; BCFW25] achieve asymptotically fast provers but have expensive folding verifiers due to many Merkle openings. Lattice-based folding schemes [BC24; BC25; NS25; FKNP24] have smaller verifier circuit, but the circuit for the Fiat-Shamir transform is still a bottleneck, which forces them to use low folding arity under the standard IVC/PCD compiler [Bün+21; KST22]. Unfortunately, standard IVC/PCD compilers only allow for constant folding depth, meaning that the resulting schemes can fold at most $O(1)$ statements. Recently, Mangrove [Ngu+24] introduces a more efficient SNARK compiler from folding schemes, but it still requires embedding the entire folding verifier circuit (including FS) into each statement. Chiesa et. al. [CGSY24] provides a tighter security bound for the PCD compiler from recursive SNARKs, though it only applies to SNARKs with straightline extractors.

Several works [CCS22; Che+23] show how to construct PCD in oracle models without instantiating the oracles. However, they rely on the existence of relativized SNARKs, which do not exist in the random oracle model [BCG24].

1.4 Organization

We provide necessary background in Section 2. In Section 3, we introduce a toolbox of protocols that serve as building blocks for our folding scheme. Section 4 presents our high-arity folding scheme that compresses a large number of R1CS statements. Next, Section 6 presents a generic compiler that converts the high-arity folding scheme into a succinct argument. Section 7 gives a candidate instantiation. In Section 8, we extend our scheme to support folding depth two.

2 Preliminaries

In Section 2.1, we provide necessary background about cyclotomic rings, norms, and the techniques of monomial embedding and random projection. Section 2.2 reviews the notion of binding commitments and the instantiations over lattices. Section 2.3 reviews the tensor-of-rings framework, which is useful for interleaving between sumcheck and folding operations. Section 2.4 reviews the notion of reduction of knowledge that captures both SNARKs and folding schemes. Section 2.5 interprets the sumcheck protocol as a special reduction of knowledge. Finally, Section 2.6 recalls (commit-and-prove) SNARKs.

Notation. $\lambda \in \mathbb{N}$ is the security parameter. A function $f(\lambda)$ is $\mathsf{poly}(\lambda)$ if there exists a $c \in \mathbb{N}$ such that $f(\lambda) = O(\lambda^c)$. If $f(\lambda) = o(\lambda^{-c})$ for all $c \in \mathbb{N}$, we say $f(\lambda)$ is in $\mathsf{negl}(\lambda)$ and is **negligible**. A probability that is $1 - \mathsf{negl}(\lambda)$ is **overwhelming**. We use logarithm $\log(\cdot)$ to denote $\log_2(\cdot)$. For $l, r \in \mathbb{Z}$, $l < r$, (l, r) denotes the set $\{l+1, l+2, \ldots, r-1\}$. We denote by $[l, r) := (l-1, r)$, $[l, r] := [l, r+1)$, and $[n] := [1, n]$. $\mathbb{Z}_q$ denotes the ring of integers modulo q with unique representatitves in $[-q/2, q/2) \cap \mathbb{Z}$. For a set S that supports element subtraction, $S - S$ is the set of differences between any two *distinct* elements in S. An **indexed relation** is a set of triples $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ where the index $\mathbb{i}$ is fixed at the setup phase, $\mathbb{x}$ is the online instance and $\mathbb{w}$ is the witness. We omit $\mathbb{i}$ when clear in context.

2.1 Algebra Background

We follow the notation from [BC25] and provide necessary algebra background.

Vectors and matrices. A vector is a column vector by default and a ring is always commutative. For a vector $\mathbf{f}$ of length n and every $i \in [n]$, we denote by $\mathbf{f}_i$ or $\mathbf{f}[i]$ the ith element of $\mathbf{f}$. For vectors $\mathbf{f}, \mathbf{g}$ of the same length, their inner product is denoted as $\langle \mathbf{f}, \mathbf{g} \rangle$. $\mathbf{I}_n$ denotes the identity matrix of rank n. Let $\bar{R}$ be a ring. For vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k \in \bar{R}^n$, we denote by $[\mathbf{u}_1, \ldots, \mathbf{u}_k] \in \bar{R}^{n \times k}$ and $(\mathbf{u}_1, \ldots, \mathbf{u}_k) \in \bar{R}^{nk}$ the horizontal and vertical concatenations. Row vector concatenations are defined similarly. For a matrix $\mathbf{M} \in \bar{R}^{n \times m}$, $\{\mathbf{M}_{i,*} \in \bar{R}^m\}_{i \in [n]}$ and $\{\mathbf{M}_{*,j} \in \bar{R}^n\}_{j \in [m]}$ denote the rows and columns of $\mathbf{M}$, respectively. $\mathsf{flt}(\mathbf{M}) := (\mathbf{M}_{1,*}, \ldots, \mathbf{M}_{n,*}) \in \bar{R}^{nm}$ denotes the vertical concatenation of its rows.

Cyclotomic rings. $R := \mathbb{Z}[X]/\langle X^d + 1 \rangle$ denotes the power-of-two cyclotomic ring of dimension $d = 2^k$. Let $q > 2$ be a prime, $R_q := R/qR = \mathbb{Z}_q[X]/\langle X^d + 1 \rangle$ denotes the residual ring of R with respect to q. For $f = \sum_{i \in [d]} f_i X^{i-1} \in R_q$, we use $\mathsf{cf}(f) := (f_1, \ldots, f_d) \in \mathbb{Z}_q^d$ to denote its coefficient vector, and $\mathsf{ct}(f) := f_1$ is the constant term. For $\mathbf{f} \in R_q^n$, we define $\mathsf{cf}(\mathbf{f}) := (\mathsf{cf}(\mathbf{f}_1)^\top, \ldots, \mathsf{cf}(\mathbf{f}_n)^\top) \in \mathbb{Z}_q^{n \times d}$, and

$\mathsf{ct}(\mathbf{f}) = (\mathsf{ct}(\mathbf{f}_1), \ldots, \mathsf{ct}(\mathbf{f}_n)) \in \mathbb{Z}_q^n$ is its first column. Conversely, given $\mathbf{f} \in \mathbb{Z}_q^d$ and $\mathbf{F} \in \mathbb{Z}_q^{n \times d}$, $\mathsf{cf}^{-1}(\mathbf{f}) \in R_q$ and $\mathsf{cf}^{-1}(\mathbf{F}) \in R_q^n$ are the ring element and vector such that $\mathsf{cf}(\mathsf{cf}^{-1}(\mathbf{f})) = \mathbf{f}$ and $\mathsf{cf}(\mathsf{cf}^{-1}(\mathbf{F})) = \mathbf{F}$.

Norms. The ℓ_2-norm and ℓ_∞-norm of a matrix $\mathbf{F} \in \mathbb{Z}^{n \times m}$ are defined as

$$\|\mathbf{F}\|_\infty := \max_{i \in [n],\, j \in [m]} (|\mathbf{F}_{i,j}|), \qquad \|\mathbf{F}\|_2 := \left(\sum_{i \in [n],\, j \in [m]} \mathbf{F}_{i,j}^2 \right)^{1/2} .$$

The norm of a ring vector $\mathbf{f} \in R^n$ is the norm of $\mathsf{cf}(\mathbf{f}) \in \mathbb{Z}^{n \times d}$. The norm of $\mathbf{F} \in R^{n \times m}$ is the same as the norm of $\mathsf{flt}(\mathbf{F}) \in R^{nm}$. For ease of exposition, we abuse the notation a bit and define **norms for matrices over R_q (and $\mathbb{Z}_q$)** as the norms of their canonical lifting matrices from R_q (and $\mathbb{Z}_q$) to R (and $\mathbb{Z}$). The operator norm of $a \in R$ is

$$\|a\|_{\mathsf{op}} := \sup_{y \in R} \frac{\|a \cdot y\|_2}{\|y\|_2} , \tag{1}$$

and the operator norm $\|\mathcal{S}\|_{\mathsf{op}}$ for a set $\mathcal{S} \subseteq R$ is $\|\mathcal{S}\|_{\mathsf{op}} := \max_{a \in \mathcal{S}} \|a\|_{\mathsf{op}}$. Similarly, we define **the operator norm for set $\mathcal{S}$ over R_q**, where for $a \in \mathcal{S} \subseteq R_q$, $y \in R$, we denote by $a \cdot y \in R$ the multiplication over R between y and the canonical embedding of a in R.

In our instantiation, we use the **folding challenge set $\mathcal{S}$** in LaBRADOR [BS23], where $R_q := \mathbb{Z}_q[X]/\langle X^{64} + 1\rangle$, and each element in $\mathcal{S}$ has coefficients in $\{0, \pm 1, \pm 2\}$ and operator norm at most 15. Moreover, elements in $\mathcal{S} - \mathcal{S}$ are invertible over R_q [LS18, Corollary 1.2].

Gadget decomposition. Let $q \in \mathbb{N}$ be a modulus and $2 \leq b < q$ an integer base. Set $k := 1 + \lfloor \log_b(q) \rfloor$. For a vector $\mathbf{f} \in \mathbb{Z}_q^n$, we define the base-$b$ decomposition $\mathsf{decomp}_{b,k}(\cdot)$ of $\mathbf{f}$ as follows: for each $i \in [n]$, decompose $\mathbf{f}_i \in \mathbb{Z}_q$ as $\mathbf{g}_i \in \mathbb{Z}_q^k$ such that $\|\mathbf{g}_i\|_\infty \leq b/2$ and $\mathbf{f}_i = \langle \mathbf{g}_i, (1, b, \ldots, b^{k-1}) \rangle$. Then set $\mathsf{decomp}_{b,k}(\mathbf{f}) := (\mathbf{g}_1, \ldots, \mathbf{g}_n) \in \mathbb{Z}_q^{nk}$.

Monomial embedding. We review the monomial embedding framework from [BC25], which encodes a bounded integer to a monomial. Define the **d-monomial set**

$$\mathcal{M} := \left\{0, 1, X, \ldots, X^{d-1}\right\} \subseteq \mathbb{Z}_q[X] . \tag{2}$$

Sometimes we also write $\mathcal{M} \subseteq R_q$ to denote the natural embedding of $\left\{0, 1, X, \ldots, X^{d-1}\right\}$ to the ring R_q. And $a \in \mathcal{M}$ denotes that $a \in R_q$ is in the embedding set of $\mathcal{M}$ in R_q.

Note that $X^d = -1$ over R_q. We define the **table polynomial**

$$t(X) := \sum_{i \in [1, d/2)} i \cdot (X^i + X^{-i}) = \sum_{i \in [1, d/2)} i \cdot (X^i - X^{d-i}) \in R_q . \tag{3}$$

For $a \in (-d/2, d/2) \subseteq \mathbb{Z}_q$, $\mathsf{sgn}(a) \in \{-1, 0, 1\}$ is the sign of a and $\mathsf{sgn}(0) := 0$. We define $\mathsf{Exp}(a) := \mathsf{sgn}(a)X^a \in \mathcal{M} \subseteq R_q$ for $a \neq 0$ and $\mathsf{Exp}(0) := \{0, 1, X^{d/2}\}$. For $\mathbf{M} \in (-d/2, d/2)^{m \times n}$, define

$$\mathsf{Exp}(\mathbf{M}) := (\mathsf{Exp}(\mathbf{M}_{i,j}))_{i \in [m],\, j \in [n]} \in \mathcal{M}^{m \times n}\,. \tag{4}$$

We review the following lemma.

Lemma 2.1 (Lemma 2.2 [BC25])**.** *For all $a \in (-d/2, d/2)$ and $b \in \mathsf{Exp}(a) \subseteq \mathcal{M}$, we have $\mathsf{ct}(b \cdot t(X)) = a$. Conversely, for all $a \in \mathbb{Z}_q$, if $\mathsf{ct}(b \cdot t(X)) = a$ for some $b \in \mathcal{M}$, then $a \in (-d/2, d/2)$.*

Random projection. We review the random projection lemma from [BS23; GHL22], which states that the ℓ_2-norm of a vector is preserved after random projection.

Lemma 2.2 (Corollary 3.3 of [GHL22], Lemma 4.2 of [BS23])**.** *Let χ be a distribution over $\{0, \pm 1\}$ where $\Pr[\chi = 0] = 1/2$ and $\Pr[\chi = -1] = \Pr[\chi = 1] = 1/4$. For all $\mathbf{v} \in \mathbb{Z}^n$,*

$$\Pr_{\mathbf{u} \leftarrow \chi^n}\left[|\langle \mathbf{u}, \mathbf{v} \rangle| > 9.5\|\mathbf{v}\|_2\right] \lesssim 2^{-141}\,. \tag{5}$$

Moreover, let $q \in \mathbb{N}$. For all $B \leq q/125$ and vector $\mathbf{v} \in [-q/2, q/2)^n$ with $\|v\|_2 > B$,

$$\Pr_{\mathbf{J} \leftarrow \chi^{256 \times n}}\left[\|\mathbf{J}\mathbf{v} \bmod q\|_2 \leq \sqrt{30}B\right] \lesssim 2^{-128}\,. \tag{6}$$

Coordinate-wise special soundness. Fix $\ell \in \mathbb{N}$ and let $\mathcal{S}$ be a finite set. For two vectors $\mathbf{a}, \mathbf{b} \in \mathcal{S}^\ell$, we say that $\mathbf{a} \equiv_i \mathbf{b}$ for $i \in [\ell]$ if $\mathbf{a}_i \neq \mathbf{b}_i$ and $\mathbf{a}_j = \mathbf{b}_j$ for all $j \in [\ell] \setminus \{i\}$.

Lemma 2.3 ([FMN24], Lemma 7.1)**.** *Define challenge space $\mathcal{U} := \mathcal{S}^\ell$ and output space $\mathcal{Y}$. Let $\Psi : \mathcal{U} \times \mathcal{Y} \to \{0, 1\}$ be any predicate. For every probabilistic algorithm $\mathcal{A} : \mathcal{U} \to \mathcal{Y}$, let*

$$\epsilon_\Psi(\mathcal{A}) := \Pr_{\mathbf{u} \xleftarrow{R} \mathcal{U}}\left[\Psi(\mathbf{u}, \mathcal{A}(\mathbf{u})) = 1\right].$$

There is an oracle algorithm $\mathcal{E}$ such that $\mathcal{E}^{\mathcal{A}}(\mathbf{u}_0, y_0)$, on input $\mathbf{u}_0 \xleftarrow{R} \mathcal{U}$, $y_0 \leftarrow \mathcal{A}(\mathbf{u}_0)$, with probability at least

$$\epsilon_\Psi(\mathcal{A}) - \ell/|\mathcal{S}|\,, \tag{7}$$

outputs $\ell + 1$ pairs $(\mathbf{u}_i, y_i)_{i=0}^{\ell}$ such that $\Psi(\mathbf{u}_i, y_i) = 1$ for all $i \in [0, \ell]$, and $\mathbf{u}_i \equiv_i \mathbf{u}_0$ for all $i \in [\ell]$. $\mathcal{E}$ calls $\mathcal{A}$ for $1 + \ell$ times in expectation.

2.2 Lattice-based Binding Commitments

We recall the notion of binding commitment schemes. Later, we instantiate it with the module-variant of the Ajtai binding commitment [Ajt96; PR06; LM06].

Definition 2.1 (Binding Commitment)**.** *A binding commitment* $\mathsf{CM} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{RVfyOpen})$ *with message space $\mathcal{M}^*$, commitment space $\mathbb{C}$ and opening space $\mathcal{O}$ consists of algorithms:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp_{cm}}$: *on input security parameter* 1^λ*, output parameter* $\mathsf{pp_{cm}}$*.*
- $\mathsf{Commit}(\mathsf{pp_{cm}}, m) \to (c, o)$: *on input parameter* $\mathsf{pp_{cm}}$ *and a message* $m \in \mathcal{M}^*$*, output a commitment* $c \in \mathbb{C}$ *and an opening state* $o \in \mathcal{O}$*.*
- $\mathsf{RVfyOpen}(\mathsf{pp_{cm}}, c, m, o) \to b$: *on input the parameter* $\mathsf{pp_{cm}}$*, commitment* $c \in \mathbb{C}$*, message* $m \in \mathcal{M}^*$*, opening* $o \in \mathcal{O}$*, output a bit* b *indicating whether* (m, o) *is an opening of* c *w.r.t.* $\mathsf{pp_{cm}}$*.*

CM *satisfies the following properties:*

Perfect Completeness: *For all* λ *and* $m \in \mathcal{M}^*$*,*

$$\Pr\left[\begin{array}{c}\mathsf{pp_{cm}} \leftarrow \mathsf{Setup}(1^\lambda),\\ (c,o) \leftarrow \mathsf{Commit}(\mathsf{pp_{cm}}, m)\end{array} : \mathsf{RVfyOpen}(\mathsf{pp_{cm}}, c, m, o) = 1\right] = 1\,.$$

Binding: *For all expected poly-time adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c}\mathsf{pp_{cm}} \leftarrow \mathsf{Setup}(1^\lambda),\\ \left(c, \begin{array}{c}(m_1,o_1),\\(m_2,o_2)\end{array}\right) \leftarrow \mathcal{A}(\mathsf{pp_{cm}})\end{array} : \begin{array}{c} m_1 \neq m_2 \ \wedge \forall i \in [2] :\\ \mathsf{RVfyOpen}(\mathsf{pp_{cm}}, c, m_i, o_i) = 1\end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

We instantiate the binding commitment below.

Construction 2.1. Fix set $\mathcal{S} \subseteq R_q$, a bound $B_{\mathsf{bnd}} \in \mathbb{R}$ to be specified later and let $B_{\mathsf{rbnd}} := 2B_{\mathsf{bnd}}$. Set commitment space $\mathbb{C} := R_q^\kappa$ where $\kappa = \kappa(\lambda)$, opening space $\mathcal{O} := \mathcal{O}'_{B_{\mathsf{rbnd}}} \times (\mathcal{S} - \mathcal{S})$ where

$$\mathcal{O}'_{B_{\mathsf{rbnd}}} := \left\{\mathbf{f} \in R_q^n \ : \ \|\mathbf{v}\|_2 \leq B_{\mathsf{rbnd}}\right\} \tag{8}$$

and let the message space be

$$\mathcal{M}^* := \left\{\mathbf{m} \in R_q^n \ : \ \exists(\mathbf{f}, s) \in \mathcal{O} \ : \ s \cdot \mathbf{m} = \mathbf{f}\right\}. \tag{9}$$

The commitment[3] $\mathsf{CM} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{RVfyOpen}, \mathsf{VfyOpen})$ works as follows:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp_{cm}}$: Output $\mathsf{pp_{cm}} := \mathbf{A} \leftarrow\!\!\$\ R_q^{\kappa \times n}$.
- $\mathsf{Commit}(\mathsf{pp_{cm}}, \mathbf{m} = \mathbf{f}/s \in \mathcal{M}^*) \to c$: Output $c := \mathbf{A}\mathbf{m}$ and $o := (\mathbf{f}, s)$.
- $\mathsf{RVfyOpen}(\mathsf{pp_{cm}}, c, \mathbf{m}, o = (\mathbf{f}, s)) \to b$: Output 1 if and only if

$$\mathbf{A}\mathbf{f} = s \cdot c \ \wedge \ o \in \mathcal{O} \ \wedge s \cdot \mathbf{m} = \mathbf{f}\,. \tag{10}$$

- $\mathsf{VfyOpen}(\mathsf{pp_{cm}}, c, \mathbf{m}, o = (\mathbf{f}, s)) \to b$: Output 1 if and only if $s = 1$ and

$$\mathbf{A}\mathbf{f} = c \ \wedge \ \|\mathbf{f}\|_2 < B_{\mathsf{bnd}} \ \wedge \mathbf{m} = \mathbf{f}\,. \tag{11}$$

We simply write $\mathsf{VfyOpen}(\mathsf{pp_{cm}}, c, \mathbf{f}) = 1$ in this case.

[3]We add a *strict* opening verification interface $\mathsf{VfyOpen}(\cdot)$ to distinguish *strict* and *relaxed* openings.

Completeness is straightforward. Let T denote the operator norm of $\mathcal{S}$. As noted in [BS23], the scheme is (relaxed) binding if the Module-SIS assumption $\mathsf{MSIS}_{q,\kappa,n,4TB_{\mathsf{rbnd}}}$ holds (Definition 2.2). In the following, assuming that $\mathsf{MSIS}_{q,\kappa,n,\beta_{\mathsf{SIS}}}$ holds for some β_{SIS}, we fix

$$B_{\mathsf{rbnd}} := \beta_{\mathsf{SIS}}/(4T)\,. \tag{12}$$

Definition 2.2 (Module SIS [LS15])**.** *Let $q = q(\lambda)$, $\kappa = \kappa(\lambda)$, $n = n(\lambda)$ and $\beta_{\mathsf{SIS}} = \beta_{\mathsf{SIS}}(\lambda)$. The $\mathsf{MSIS}_{q,\kappa,n,\beta_{\mathsf{SIS}}}$ assumption states that for all expected polynomial-time adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} \mathbf{A} \xleftarrow{\mathrm{R}} R_q^{\kappa\times n} \\ \mathbf{x}\in R_q^n \leftarrow \mathcal{A}(\mathbf{A}) \end{array} \;:\; (\mathbf{A}\mathbf{x}=\mathbf{0}) \;\wedge\; (0 < \|\mathbf{x}\|_2 \le \beta_{\mathsf{SIS}})\right] < \mathsf{negl}(\lambda)\,.$$

Fine-grained commitment opening relation. Sometimes we need a more fine-grained check on opening vectors. Let auxiliary parameters be $\ell_h \in \mathbb{N}$ and $B \in \mathbb{R}$, with $\ell_h \mid n$. We say that $\mathsf{VfyOpen}_{\ell_h,B}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1$ if

$$\mathbf{A}\mathbf{f} = c \quad \wedge \quad \forall (i,j) \in [n/\ell_h] \times [d] \,:\, \|\mathbf{F}_{i,j}\|_2 \le B\,. \tag{13}$$

Here $\mathbf{F}_{i,j} \in \mathbb{Z}_q^{\ell_h\times 1}$ is the submatrix of $\mathsf{cf}(\mathbf{f}) \in \mathbb{Z}_q^{n\times d}$. That is,

$$\mathsf{cf}(\mathbf{f}) = \begin{bmatrix} \mathbf{F}_{1,1} & \dots & \mathbf{F}_{1,d} \\ \vdots & \ddots & \vdots \\ \mathbf{F}_{n/\ell_h,1} & \dots & \mathbf{F}_{n/\ell_h,d} \end{bmatrix}.$$

Note that $\mathsf{VfyOpen}_{\ell_h,B}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1 \implies \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1$ if $B \cdot \sqrt{nd/\ell_h} \le B_{\mathsf{bnd}}$.

2.3 Tensor of Rings

We review the **Tensor-of-Rings** framework from [BC25; DP24; NS25]. Given an extension field $\mathbb{K} = \mathbb{F}_{q^t}$ and a ring $R_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$, define the tensor

$$\mathbb{E} := \mathbb{K} \otimes_{\mathbb{F}_q} R_q\,. \tag{14}$$

An element e in $\mathbb{E}$ can be represented as a matrix over $\mathbb{Z}_q^{t\times d}$. Moreover, there are two alternative ways to understand $e \in \mathbb{E}$:

- By viewing each column of e as a $\mathbb{K}$-element, $e = [e_1, \dots, e_d] \in \mathbb{K}^{1\times d}$ is an element in the vector space $\mathbb{K}^d$. Thus we can perform scalar multiplication between $a \in \mathbb{K}$ and $e \in \mathbb{E}$, that is, $a \cdot [e_1, \dots, e_d] = [a \cdot e_1, \dots, a \cdot e_d]$.
- By viewing each row of $\mathbb{E}$ as an R_q-element, $e = (e'_1, \dots, e'_t) \in R_q^t$ is an element in the R_q-module[4] of dimension t. Thus we can perform scalar multiplication between $e \in \mathbb{E}$ and $b \in R_q$, that is, $(e'_1, \dots, e'_t) \cdot b = (b \cdot e'_1, \dots, b \cdot e'_t)$.

Looking ahead, the $\mathbb{K}$-vector space interpretation will be convenient for running sum-checks over $\mathbb{K}$; the R_q-module interpretation will be convenient for folding witnesses via low-norm challenges over $\mathcal{S} \subseteq R_q$.

[4]Modules are generalizations of vector spaces where scalars are ring elements.

Multiplication between R_q and $\mathbb{K}$. Given the tensor $\mathbb{E}$ defined above, we define multiplication between an R_q-element and a $\mathbb{K}$-element: Take $a \in \mathbb{K}$ with coefficient vector $\mathsf{cf}(a) \in \mathbb{Z}_q^t$ and $b \in R_q$ with coefficient vector $\mathsf{cf}(b) \in \mathbb{Z}_q^d$, we define $a \cdot b \in \mathbb{E}$ as the $\mathbb{E}$-element represented by the matrix

$$\mathsf{cf}(a) \otimes \mathsf{cf}(b)^\top \in \mathbb{Z}_q^{t \times d} .$$

There are two ways to interpret the multiplication above:

- We lift $b \in R_q$ to $e_b := \begin{bmatrix} b \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{E}$ (where the 1st row is b and the following $t-1$ rows are zeros) and perform scalar multiplication between $a \in \mathbb{K}$ and e_b (by viewing e_b as an element in the $\mathbb{K}$-vector space).
- We lift $a \in \mathbb{K}$ to $e_a := [a, 0, \ldots, 0] \in \mathbb{E}$ (where the 1st column is a and the next $d-1$ columns are zeros) and perform scalar multiplication between e_a and $b \in R_q$ (by viewing e_a as an element in the R_q-module).

2.4 Generalized Reduction of Knowledge

We generalize the notion of reduction of knowledge (RoK) [KP23]. The completeness requirement is relaxed to allow honest provers to fail with negligible probability. The knowledge soundness is modified to allow witness extraction for a relaxed input relation. We follow the notation from [BC25] verbatim.

Definition 2.3 (Generalized Reduction of Knowledge)**.** *Let $\mathcal{R}_1$, $\mathcal{R}_1'$, $\mathcal{R}_2$ be indexed relations. A reduction of knowledge Π from $\mathcal{R}_1$ to $\mathcal{R}_2$ (with relaxed input relation $\mathcal{R}_1'$) consists of PPT algorithms below:*

- $\mathcal{G}(1^\lambda) \to \mathbb{i}$*: on input security parameter λ output index $\mathbb{i}$.*
- $\mathsf{P}(\mathbb{i}, \mathbb{x}_1, \mathbb{w}_1) \to (\mathbb{x}_2, \mathbb{w}_2)$*: take index $\mathbb{i}$, a statement $(\mathbb{x}_1, \mathbb{w}_1)$ such that $(\mathbb{i}, \mathbb{x}_1, \mathbb{w}_1) \in \mathcal{R}_1$, interact with the verifier, and output a statement $(\mathbb{x}_2, \mathbb{w}_2)$ such that $(\mathbb{i}, \mathbb{x}_2, \mathbb{w}_2) \in \mathcal{R}_2$.*
- $\mathsf{V}(\mathbb{i}, \mathbb{x}_1) \to \mathbb{x}_2$*: take index $\mathbb{i}$, an instance $\mathbb{x}_1$ for $\mathcal{R}_1$, interacts with the prover, and output an instance $\mathbb{x}_2$ for relation $\mathcal{R}_2$.* V *outputs $\perp$ if rejects early.*

We denote by $\langle \mathsf{P}(\mathbb{w}_1), \mathsf{V} \rangle [\mathbb{i}, \mathbb{x}_1] \to (\mathbb{x}_2, \mathbb{w}_2)$ the interaction between P *and* V *with common input $(\mathbb{i}, \mathbb{x}_1)$. By default, (i) the reduced instances output by* P *and* V *are the same, and (ii) $\perp \notin \mathcal{L}(\mathcal{R}_2)$. For notational convenience, we omit index $\mathbb{i}$ when clear in context.*

The reduction of knowledge satisfies the properties below.

Definition 2.4 (ϵ-Completeness)**.** *For all PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} \mathbb{i} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{x}_1, \mathbb{w}_1) \leftarrow \mathcal{A}(\mathbb{i}) \\ (\mathbb{x}_2, \mathbb{w}_2) \leftarrow \langle \mathsf{P}(\mathbb{w}_1), \mathsf{V} \rangle [\mathbb{i}, \mathbb{x}_1] \end{array} : \begin{array}{c} (\mathbb{i}, \mathbb{x}_1, \mathbb{w}_1) \in \mathcal{R}_1 \wedge \\ (\mathbb{i}, \mathbb{x}_2, \mathbb{w}_2) \notin \mathcal{R}_2 \end{array} \right] \le \epsilon .$$

Definition 2.5 (κ-Knowledge Soundness w.r.t. $\mathcal{R}_1'$)**.** *There exists a function $\kappa(\cdot)$ such that for all expected polynomial time adversaries P^* with non-negligible success probability, there is an expected polynomial time extractor Ext, given $\mathbb{i} \leftarrow \mathcal{G}(1^\lambda)$, $(\mathbb{x}_1, \mathsf{st}) \leftarrow \mathsf{P}^*(\mathbb{i})$,*

$$\left| \Pr\left[(\mathbb{i}, \langle \mathsf{P}^*(\mathsf{st}), \mathsf{V}\rangle [\mathbb{i}, \mathbb{x}_1]) \in \mathcal{R}_2\right] - \Pr\left[(\mathbb{i}, \mathbb{x}_1, \mathsf{Ext}^{\mathsf{P}^*}(\mathbb{i}, \mathbb{x}_1, \mathsf{st})) \in \mathcal{R}_1'\right] \right| \leq \kappa(\lambda)\,.$$

When $\mathcal{R}_1' = \mathcal{R}_1$, we simply say that it satisfies κ-knowledge soundness.

Definition 2.6 (Public reducibility.)**.** *There is a deterministic polynomial time algorithm f such that for all PPT adversary $\mathcal{A}$ and expected polynomial time adversary P^*:*

$$\Pr\left[\begin{array}{c} \mathbb{i} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{x}_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathbb{i}) \\ (\mathsf{tr}, \mathbb{x}_2, \mathbb{w}_2) \leftarrow \langle \mathsf{P}^*(\mathsf{st}), \mathsf{V}\rangle [\mathbb{i}, \mathbb{x}_1] \end{array} \quad : \quad f(\mathbb{i}, \mathbb{x}_1, \mathsf{tr}) = \mathbb{x}_2 \right] = 1\,.$$

Here tr denotes the transcript of the interaction $\langle \mathsf{P}^*(\mathsf{st}), \mathsf{V}\rangle [\mathbb{i}, \mathbb{x}_1]$.

2.5 The Sumcheck Protocol

Let $\mathbb{K} = \mathbb{F}_{q^t}$ be an extension field. For vector $\mathbf{r} \in \mathbb{K}^k$, we define its tensor $\mathsf{ts}(\mathbf{r})$ as

$$\mathsf{ts}(\mathbf{r}) := (eq_b(\mathbf{r}))_{b \in \{0,1\}^k} \in \mathbb{K}^{2^k} \tag{15}$$

where $eq_b(\mathbf{r}) := \prod_{i \in [k]} \left[(1 - b_i)(1 - \mathbf{r}_i) + b_i \mathbf{r}_i\right]$. For a multilinear polynomial[5] $f(\mathbf{X}) \in \mathbb{K}[X_1, \ldots, X_{\log(n)}]$, we denote by its evaluation vector $\mathbf{f}$ as

$$\mathbf{f} := (f(b))_{b \in \{0,1\}^{\log(n)}} \in \mathbb{K}^n\,.$$

Observe that the evaluation $f(\mathbf{r})$ at point $\mathbf{r} \in \mathbb{K}^{\log(n)}$ satisfies that

$$f(\mathbf{r}) := \sum_{b \in \{0,1\}^{\log(n)}} f(b) \cdot eq_b(\mathbf{r}) = \langle \mathbf{f}, \mathsf{ts}(\mathbf{r}) \rangle\,.$$

Let $g(\mathbf{X}) \in \mathbb{K}[X_1, \ldots, X_{\log(n)}]$ denote a multivariate polynomial over $\mathbb{K}$ of constant degree D. We use $c = \mathsf{Commit}(g)$ to denote that c is a "virtual commitment" to the polynomial g. The classical sumcheck protocol [LFKN92] can be understood as a special reduction of knowledge from

$$\mathcal{R}_{\mathsf{sum}} := \left\{ (\mathbb{x}, \mathbb{w}) \;:\; \begin{array}{l} \mathbb{x} = (c,\, v \in \mathbb{K}), \\ \mathbb{w} = g(\mathbf{X}) \quad \text{s.t.} \\ c = \mathsf{Commit}(g) \;\wedge\; \sum_{b \in \{0,1\}^{\log(n)}} g(b) = v \end{array} \right\}. \tag{16}$$

to the relation

$$\mathcal{R}_{\mathsf{eval}} := \left\{ (\mathbb{x}, \mathbb{w}) \;:\; \begin{array}{l} \mathbb{x} = (c,\, \mathbf{r} \in \mathbb{K}^{\log(n)}, v \in \mathbb{K}), \\ \mathbb{w} = g(\mathbf{X}) \quad \text{s.t.} \\ c = \mathsf{Commit}(g) \;\wedge\; g(\mathbf{r}) = v \end{array} \right\}. \tag{17}$$

[5]A multilinear polynomial is a multivariate polynomial with individual degree ≤ 1.

The protocol has linear-time prover and polylogarithmic verifier. The knowledge error is $\epsilon_{\mathsf{sum}} := D\log(n)/|\mathbb{K}| + \epsilon_{\mathsf{bind}}$ where ϵ_{bind} is the binding error of the commitment.

When $g(\mathbf{X})$ is of the special form $g(\mathbf{X}) = h(f_1(\mathbf{X}), \ldots, f_k(\mathbf{X}))$ for some polynomial h and *multilinear* polynomials $f_1, \ldots, f_k$, we instantiate the commitment c as the commitments to the evaluation vectors $(\mathbf{f}_i)_{i=1}^k$, and checking $g(\mathbf{r}) = v$ is equivalent to:

$$(\forall i \in [k] : \langle \mathbf{f}_i, \mathsf{ts}(\mathbf{r})\rangle = u_i) \ \wedge \ h(u_1, \ldots, u_k) = v\,. \tag{18}$$

By letting the verifier check $h(u_1, \ldots, u_k) = v$, $\mathcal{R}_{\mathsf{eval}}$ can be simplified to a linear relation

$$\mathcal{R}'_{\mathsf{eval}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = ((c_i)_{i=1}^k, \mathbf{r} \in \mathbb{K}^{\log(n)}, (u_i \in \mathbb{K})_{i=1}^k), \\ \mathbb{w} = (\mathbf{f}_i)_{i=1}^k \quad \text{s.t.} \\ \forall i \in [k] : \langle \mathbf{f}_i, \mathsf{ts}(\mathbf{r})\rangle = u_i) \ \wedge \ c_i = \mathsf{Commit}(\mathbf{f}_i) \end{array} \right\}. \tag{19}$$

Sumcheck batching. Given k sumcheck statements over $\mathbb{K}$ w.r.t. polynomials $g_1, \ldots, g_k \in \mathbb{K}[X_1, \ldots, X_{\log(n)}]$, we can reduce them to a single sumcheck statement for the polynomial $\sum_{i=1}^k g_i \cdot \alpha^{i-1}$, where $\alpha \xleftarrow{R} \mathbb{K}$ is a challenge sampled by the verifier.

2.6 (Commit-and-Prove) SNARKs

We review and generalize the notion of SNARKs and commit-and-prove SNARKs.

Definition 2.7 (SNARKs). *Let* $\mathsf{GenR}(1^\lambda) \to (\mathcal{R}, \mathcal{R}', \mathbb{i})$ *be an algorithm that outputs the description of a relation* $\mathcal{R}$*, a relaxed relation* $\mathcal{R}'$*, and an index* $\mathbb{i}$ *on input the security parameter. We simply write* $\mathsf{GenR}(1^\lambda) \to (\mathcal{R}, \mathbb{i})$ *when* $\mathcal{R} = \mathcal{R}'$*. A succinct non-interactive argument of knowledge (SNARK)* Π *w.r.t.* GenR *consists of algorithms:*

$\mathsf{Setup}(\mathcal{R}, \mathbb{i}) \to (\mathsf{pk}, \mathsf{vk})$: *take relation description* $\mathcal{R}$ *and index* $\mathbb{i}$*, output proving parameter* pk *and verifier parameter* vk*. We omit input* $\mathbb{i}$ *when clear in context.*

$\mathsf{Prove}(\mathsf{pk}, \mathcal{R}, \mathbb{x}, \mathbb{w}) \to \pi$: *take proving parameter* pk*, relation* $\mathcal{R}$*, instance-witness pair* $(\mathbb{x}, \mathbb{w})$*, output a proof* π*.*

$\mathsf{Vf}(\mathsf{vk}, \mathcal{R}, \mathbb{x}, \pi) \to b$: *take verifier parameter* vk*, relation* $\mathcal{R}$*, instance* $\mathbb{x}$*, and proof* π*, output a bit* b *indicating accept* $(b = 1)$ *or reject.*

A SNARK satisfies properties below:

Completeness: *For all PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c} (\mathcal{R}, \mathcal{R}', \mathbb{i}) \leftarrow \mathsf{GenR}(1^\lambda),\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(\mathcal{R}, \mathbb{i}) \\ (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(\mathcal{R}, \mathbb{i}),\ \pi \leftarrow \mathsf{Prove}(\mathsf{pk}, \mathcal{R}, \mathbb{x}, \mathbb{w}) \end{array} : \begin{array}{c} (\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R} \ \wedge \\ \mathsf{Vf}(\mathsf{vk}, \mathcal{R}, \mathbb{x}, \pi) = 0 \end{array}\right] \le \mathsf{negl}(\lambda)\,.$$

Succinctness: *The bitlength of proof* π *is* $\mathsf{poly}(\lambda, \log(|\mathbb{w}|))$ *and the verifier time is* $\mathsf{poly}(\lambda, |\mathbb{x}|, \log(|\mathbb{w}|))$*.*

Knowledge soundness w.r.t. $\mathcal{R}'$: *For all PPT adversary* P^**, there exists an expected PPT extractor* Ext *such that*

$$\Pr\left[\begin{array}{c}(\mathcal{R}, \mathcal{R}', \mathbb{i}) \leftarrow \mathsf{GenR}(1^\lambda),\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(\mathcal{R}, \mathbb{i}) \\ (\mathbb{x}, \pi) \leftarrow \mathsf{P}^*(\mathsf{pk}, \mathcal{R}),\ \mathbb{w} \leftarrow \mathsf{Ext}(\mathsf{pk}, \mathsf{vk}, \mathcal{R}, \mathcal{R}', \mathbb{i})\end{array} : \begin{array}{c}(\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin \mathcal{R}' \wedge \\ \mathsf{Vf}(\mathsf{vk}, \mathcal{R}, \mathbb{x}, \pi) = 1\end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

Next, we review the notion of commit-and-prove SNARKs [Kil89; CLOS02; CFQ19]. Informally, given an instance $\mathbb{x}$ and a commitment c, it proves knowledge of $\mathbb{w} := (\mathbb{w}_1, \mathbb{w}_2)$ such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and $\mathbb{w}_1$ is an opening to the commitment c.

Definition 2.8 (CP-SNARKs [Kil89; CLOS02; CFQ19])**.** *Let* $\mathsf{GenR}(1^\lambda) \rightarrow (\mathcal{R}, \mathbb{i})$ *be a relation generator as in Definition 2.7 where the witness space is* $\mathcal{M}_1^* \times \cdots \times \mathcal{M}_\ell^* \times \mathcal{M}_0^*$*. Let* $\mathsf{CM} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{RVfyOpen})$ *be a commitment scheme. Define relation generator* $\mathsf{GenR}_c(1^\lambda)$*:*

- *Run* $(\mathcal{R}, \mathbb{i}) \leftarrow \mathsf{GenR}(1^\lambda)$ *and* $\mathsf{pp_{cm}} \leftarrow \mathsf{Setup}(1^\lambda)$.
- *Output* $\mathbb{i}' := (\mathsf{pp_{cm}}, \mathbb{i})$ *and relation*

$$\mathcal{R}' := \left\{ \begin{array}{l} \mathbb{i}' = (\mathsf{pp_{cm}}, \mathbb{i}), \\ \mathbb{x}' := (\mathbb{x}, (c_i)_{i=1}^{\ell}), \\ \mathbb{w}' = ((\mathbb{w}_i, o_i)_{i=1}^{\ell}, \mathbb{w}^*) \end{array} : \begin{array}{l} (\mathbb{i}, \mathbb{x}, \mathbb{w} = (\mathbb{w}_1, \ldots, \mathbb{w}_\ell, \mathbb{w}^*)) \in \mathcal{R} \wedge \\ \forall i \in [\ell] : \mathsf{CM.RVfyOpen}(\mathsf{pp_{cm}}, c_i, \mathbb{w}_i, o_i) = 1 \end{array} \right\}. \tag{20}$$

A Commit-and-Prove SNARK (CP-SNARK) for CM *and* GenR *is a SNARK w.r.t.* GenR_c*. In practice, its instantiation is roughly as efficient as a SNARK for* GenR.

3 A Toolbox of Reduction of Knowledge

In this section, we introduce several reductions of knowledge that will serve as building blocks for our folding schemes.

Relation parameters. All relations in this paper share the public parameters:

$$\mathbb{i} := (\mathsf{pp_{cm}}, R_q, \mathbb{K}, \mathbb{E}) \tag{21}$$

where $R_q := \mathbb{Z}_q[X]/\langle X^d + 1\rangle$ for a prime q, $\mathsf{pp_{cm}} = \mathbf{A} \in R_q^{\kappa \times n}$ is the MSIS matrix for the lattice commitment CM (with commitment space $\mathbb{C} := R_q^\kappa$), $\mathbb{K} := \mathbb{F}_{q^t}$, and $\mathbb{E} := R_q \otimes_{\mathbb{Z}_q} \mathbb{K}$ is the tensor ring. In the following, we omit index $\mathbb{i}$ in relations for simplicity.

3.1 Generic Committed Linear Relation

We introduce **generic committed linear relations**, which serves as an anchor between our folding scheme and a SNARK. Later, we will see how the folding scheme reduces multiple R1CS statements into two efficiently provable statements in this linear relation.

Set parameters $\mathsf{aux} := (n_{\mathsf{in}} \in \mathbb{N}, (\mathbf{M}_i \in \mathbb{Z}_q^{m_i \times n})_{i=1}^{k_x}),\ M := \max_{i \in [k_x]}\{m_i\}$ where $(m_i)_{i=1}^{k_x}$ are powers of two and $n \geq n_{\mathsf{in}}$. Define relation

$$\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = (c \in \mathbb{C},\ \mathbf{x} \in R_q^{n_{\mathsf{in}}},\ \mathbf{r} \in \mathbb{K}^{\log M},\ \mathbf{v} \in \mathbb{E}^{k_x}) \\ \mathbb{w} = \mathbf{f} \in R_q^n \quad \text{s.t.} \\ \mathbf{x} = \mathbf{f}[1..n_{\mathsf{in}}] \quad \wedge \quad \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1 \\ \wedge\ \forall i \in [k_x] \ : \ \langle \mathsf{ts}(\mathbf{r})[1..m_i],\ \mathbf{M}_i \mathbf{f} \rangle = \mathbf{v}_i \end{array} \right\}. \tag{22}$$

In words, the witness $\mathbf{f}$ is a valid opening to the commitment c and matches the public input $\mathbf{x}$; moreover, for $i \in [k_x]$, $\mathbf{v}_i$ equals the inner product between $\mathbf{M}_i\mathbf{f}$ and $\mathsf{ts}(\mathbf{r})$'s prefix, where the tensor vector $\mathsf{ts}(\mathbf{r}) \in \mathbb{K}^M$ is defined in Eq. (15).

3.2 RoKs for Hadamard Relations

We reduce checking a batched Hadamard product relation (common in R1CS) to a generic linear relation. The idea is similar to that in Neo [NS25] and LatticeFold+[BC25]. Fix parameters $\mathsf{aux} := (n_{\mathsf{in}} = 0, (\mathbf{M}_i \in \mathbb{Z}_q^{m \times n})_{i=1}^3)$ where m is a power-of-two. The input relation is

$$\mathcal{R}_{\mathsf{had}}^{\mathsf{aux}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = c \in \mathbb{C}, \quad \mathbb{w} = \mathbf{F} \in \mathbb{Z}_q^{n \times d} \quad \text{s.t.} \\ (\mathbf{M}_1\mathbf{F}) \circ (\mathbf{M}_2\mathbf{F}) = \mathbf{M}_3\mathbf{F} \ \wedge \ \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathsf{cf}^{-1}(\mathbf{F})) = 1 \end{array} \right\}, \tag{23}$$

where $\circ$ denotes element-wise multiplication. The output relation is[6]

$$\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = (c \in \mathbb{C},\ \mathbf{r} \in \mathbb{K}^{\log m}, \mathbf{v} \in \mathbb{E}^3), \\ \mathbb{w} = \mathbf{f} \in R_q^n \quad \text{s.t.} \\ \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1 \ \wedge \ \forall i \in [3] : \langle (\mathbf{M}_i\mathbf{f}),\ \mathsf{ts}(\mathbf{r}) \rangle = \mathbf{v}_i \end{array} \right\}. \tag{24}$$

The reduction protocol Π_{had} is described in Figure 1.

Proposition 3.1. *Π_{had} in Figure 1 is an RoK from $\mathcal{R}_{\mathsf{had}}^{\mathsf{aux}}$ (Eq. (23)) to $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$ (Eq. (24)).*

Proof. We show that Π_{had} is perfectly complete and $((d+\log(m))/|\mathbb{K}| + \epsilon_{\mathsf{sum}})$-knowledge sound, where ϵ_{sum} is the knowledge error of the (committed) sumcheck RoK in Section 2.5.
Completeness. If the input $(\mathbb{x} = c, \mathbb{w} = \mathbf{F}) \in \mathcal{R}_{\mathsf{had}}^{\mathsf{aux}}$, the sumcheck claim in Eq. (25) holds. By the completeness of the sumcheck RoK, the verifier's check in Eq. (26) passes, and the output is in $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$.
Knowledge soundness. Consider an adversary P^*. The extractor simulates the protocol with P^*, and from P^*'s output $\mathbb{w}_o = \mathbf{f}$, it returns $\mathbf{F} := \mathsf{cf}(\mathbf{f})$.

Suppose $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$, i.e., $\mathsf{cf}^{-1}(\mathbf{F})$ is a valid opening to c and the linear evaluation statements hold. If $\mathbf{F}$ does not satisfy the Hadamard product check in $\mathcal{R}_{\mathsf{had}}^{\mathsf{aux}}$, then exists

[6]When computing $\mathbf{M}_i\mathbf{f} \in R_q^m$, $\mathbf{M}_i$'s entries are interpreted as R_q-elements.

Parameters: $\mathsf{aux} := (n_{\mathsf{in}} = 0, (\mathbf{M}_i \in \mathbb{Z}_q^{m\times n})_{i=1}^3)$
Input: $\mathbb{x} = c \in \mathbb{C}$ and $\mathbb{w} = \mathbf{F} \in \mathbb{Z}_q^{n\times d}$
Output: $\mathbb{x}_o = (c, \mathbf{r} \in \mathbb{K}^{\log m}, \mathbf{v} \in \mathbb{E}^3)$ and $\mathbb{w}_o = \mathbf{f} \in R_q^n$
The protocol $\langle \mathsf{P}(\mathbb{x};\mathbb{w}); \mathsf{V}(\mathbb{x})\rangle$:

1. $\mathsf{V} \to \mathsf{P}$: Send challenges $\mathbf{s} \leftarrow\!\!\$\ \mathbb{K}^{\log m}$ and $\alpha \leftarrow\!\!\$\ \mathbb{K}$
2. $\mathsf{P} \leftrightarrow \mathsf{V}$: Run a sumcheck protocol for the claim

$$\sum_{b\in\{0,1\}^{\log m}} \left(\sum_{j=1}^{d} \alpha^{j-1} \cdot f_j(b)\right) = 0 \tag{25}$$

where the polynomial $f_j(\mathbf{X}) \in \mathbb{K}[X_1, \ldots, X_{\log m}]$ $(1 \le j \le d)$ is defined as

$$f_j(\mathbf{X}) = eq(\mathbf{s}, \mathbf{X}) \cdot (g_{1,j}(\mathbf{X}) \cdot g_{2,j}(\mathbf{X}) - g_{3,j}(\mathbf{X}))$$

and $g_{i,j}(\mathbf{X})$ $(1 \le i \le 3)$ is the multilinear extension of $\mathbf{M}_i\mathbf{F}_{*,j} \in \mathbb{Z}_q^m$. The protocol reduces to an evaluation claim

$$\sum_{j=1}^{d} \alpha^{j-1} \cdot f_j(\mathbf{r}) = e$$

where $\mathbf{r} \leftarrow\!\!\$\ \mathbb{K}^{\log m}$ is the sumcheck challenge and $e \in \mathbb{K}$

3. $\mathsf{P} \to \mathsf{V}$: Send values $\mathbf{U} \in \mathbb{K}^{3\times d}$ where $\mathbf{U}_{i,j} := g_{i,j}(\mathbf{r})$ for $i \in [3]$ and $j \in [d]$
4. V : Compute $eq(\mathbf{s}, \mathbf{r}) \in \mathbb{K}$ and check

$$\sum_{j=1}^{d} \alpha^{j-1} \cdot eq(\mathbf{s}, \mathbf{r}) \cdot (\mathbf{U}_{1,j} \cdot \mathbf{U}_{2,j} - \mathbf{U}_{3,j}) \overset{?}{=} e\,. \tag{26}$$

Abort if it fails, otherwise output $\mathbb{x}_o := (c, \mathbf{r}, \mathbf{v} \in \mathbb{E}^3)$ where[a] for $i \in [3]$

$$\mathbf{v}_i := \sum_{j=1}^{d} (X^{j-1}) \cdot \mathbf{U}_{i,j} \in \mathbb{E}\,.$$

5. P : output $\mathbb{w}_o := \mathsf{cf}^{-1}(\mathbf{F})$

[a] Here $X^{j-1} \in \mathcal{M} \subseteq R_q$ for $j \in [d]$. The multiplication between R_q and $\mathbb{K}$ is defined in Section 2.3.

Figure 1: The protocol Π_{had} to reduce $\mathcal{R}_{\mathsf{had}}^{\mathsf{aux}}$ to $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$.

a column $j \in [d]$ where the Hadamard check fails for $\mathbf{F}_{*,j}$. Thus, with probability at least $1 - (\log(n)/|\mathbb{K}|)$ (over random $\mathbf{s}$),

$$\sum_{b \in \{0,1\}^{\log(m)}} f_j(b) \neq 0\,.$$

Conditioned on this, with probability at least $1-(d/|\mathbb{K}|)$ (over random α), the sumcheck claim in Eq. (25) is false. By the soundness of the *committed* sumcheck RoK, the knowledge error is bounded by $\epsilon_{\mathsf{sum}} + (d + \log(n))/|\mathbb{K}|$. $\square$

Proposition 3.2. *Π_{had} in Figure 1 runs a single degree-3 sumcheck protocol over $\mathbb{K}$ of size m. The additional complexity beyond the sumcheck is as follows:*

- ***Prover cost** $T_p^{\mathsf{had}}(m)$: $3d$ inner products between $\mathbb{Z}_q^m \subseteq \mathbb{K}^m$ and $\mathbb{K}^m$ (for computing $\mathbf{U}$), and the cost for computing[7] $(\mathbf{M}_i\mathbf{F})_{i=1}^3$.*
- ***Verifier cost** $T_v^{\mathsf{had}}(m)$: $O(d + \log(m))$ $\mathbb{K}$-ops.*
- ***Verifier randomness** $\Gamma_v^{\mathsf{had}}(m)$: $O(\log(m))$ $\mathbb{K}$-elements.*

3.3 RoKs for Monomial Relations

LatticeFold+[BC25] introduces an RoK for monomials, which checks that each commitment opening is a monomial vector, i.e., each entry lies in the monomial set $\mathcal{M} := \{0, 1, X, \ldots, X^{d-1}\} \subseteq R_q$ (Eq. (2)).

Lemma 3.1 (Lemma 4.2 of [BC25])**.** *For parameters $k_g, n \in \mathbb{N}$ where n is a power-of-two, there exists a reduction of knowledge Π_{mon} from relation*

$$\mathcal{R}_{\mathsf{mon}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = (c^{(i)} \in \mathbb{C})_{i=1}^{k_g}, \quad \mathbb{w} = (\mathbf{g}^{(i)} \in R_q^n)_{i=1}^{k_g} \quad \text{s.t.} \\ \forall i \in [k_g] \ : \ \mathsf{VfyOpen}(\mathsf{pp_{cm}}, c^{(i)}, \mathbf{g}^{(i)}) = 1 \ \wedge \mathbf{g}^{(i)} \in \mathcal{M}^n \end{array} \right\}, \tag{27}$$

to relation[8]

$$\mathcal{R}_{\mathsf{batchlin}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = (\mathbf{r} \in \mathbb{K}^{\log n}, [c^{(i)} \in \mathbb{C},\ \mathbf{u}^{(i)} \in \mathbb{E}]_{i=1}^{k_g}), \\ \mathbb{w} = (\mathbf{g}^{(i)} \in R_q^n)_{i=1}^{k_g} \quad \text{s.t.} \\ \forall i \in [k_g] \ : \ \mathsf{VfyOpen}(\mathsf{pp_{cm}}, c^{(i)}, \mathbf{g}^{(i)}) = 1 \ \wedge \ \langle \mathbf{g}^{(i)}, \mathsf{ts}(\mathbf{r}) \rangle = \mathbf{u}^{(i)} \end{array} \right\}. \tag{28}$$

Besides a single degree-3 sumcheck over $\mathbb{K}$ of size n, the prover's complexity $T_p^{\mathsf{mon}}(k_g, n)$ is $O(nk_g)$ $\mathbb{K}$-additions (for computing $(\mathbf{u}^{(i)})_{i=1}^{k_g}$) and $O(n)$ $\mathbb{K}$-ops. The verifier's complexity $T_v^{\mathsf{mon}}(k_g, n)$ is $O(k_g d + \log(n))$ $\mathbb{K}$-ops.

3.4 Approximate Range Proofs for Ring Vectors

We reduce checking the norms of a ring vector to statements in generic linear relations. The protocol combines the exact-norm proof idea from [BC25] and the random projection

[7]It takes $O((m + n)d)$ $\mathbb{Z}_q$-multiplications when $\mathbf{M}_i$'s are sparse R1CS matrices.
[8]The multiplication between R_q and $\mathbb{K}$ is defined in Section 2.3.

techniques from [BS23; KLNO25] to achieve near-optimal efficiency. As a tradeoff, the protocol provides only an approximate range proof, i.e., the extracted witness may have a slightly larger norm than specified in the completeness relation. Looking ahead, this is sufficient in our setting, where the folding depth is a small constant (e.g., 1 or 2).

Let $n \in \mathbb{N}$ denote the witness length, and $\ell_h \in \mathbb{N}$, $\lambda_{\mathsf{pj}} := 256$ denote the projection input and output length, with $\ell_h \mid n$. Set norm bound $B \in \mathbb{R}$ and $d' := d - 2$. Let k_g be the minimal integer such that

$$B_{d,k_g} := (d'/2) \cdot (1 + d' + \cdots + d'^{k_g - 1}) \geq 9.5B\,. \tag{29}$$

Define input relation

$$\mathcal{R}_{\mathsf{rg}}^{\ell_h, B} := \left\{ (\mathbb{x}, \mathbb{w}) \;:\; \begin{array}{l} \mathbb{x} = c \in \mathbb{C} \quad \mathbb{w} = \mathbf{f} \in R_q^n \quad \text{s.t.} \\ \mathsf{VfyOpen}_{\ell_h, B}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1 \text{ (Eq. (13))} \end{array} \right\}. \tag{30}$$

Denote by $\mathsf{aux}_J := (n_{\mathsf{in}} = 0, \mathbf{M}_J := \mathbf{I}_{n/\ell_h} \otimes \mathbf{J} \in \mathbb{Z}_q^{(n\lambda_{\mathsf{pj}}/\ell_h) \times n})$ for some random $\mathbf{J} \leftarrow \chi^{\lambda_{\mathsf{pj}} \times \ell_h}$. For simplicity, assume that $m := n\lambda_{\mathsf{pj}}/\ell_h$ is a power of two and $md = n$. (The scheme naturally extends to more general parameter choices.) The output relation is $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_J} \times \mathcal{R}_{\mathsf{batchlin}}$ where $\mathcal{R}_{\mathsf{batchlin}}$ is defined in Eq. (28) and

$$\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_J} := \left\{ (\mathbb{x}, \mathbb{w}) \;:\; \begin{array}{l} \mathbb{x} = (c \in \mathbb{C},\, \mathbf{r} \in \mathbb{K}^{\log m}, \mathbf{v} \in \mathbb{E}), \\ \mathbb{w} = \mathbf{f} \in R_q^n \quad \text{s.t.} \\ \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1 \;\wedge\; \langle (\mathbf{M}_J \mathbf{f}), \mathsf{ts}(\mathbf{r}) \rangle = \mathbf{v} \end{array} \right\}. \tag{31}$$

The reduction protocol Π_{rg} is described in Figure 2.

Remark 3.1. Compared to LatticeFold+[BC25], our approximate range proofs require fewer monomial commitments and therefore avoid the added complexity of folding the so-called *double commitments*. Compared to LaBRADOR [BS23], our construction improves the verifier complexity from linear to polylogarithmic.

Theorem 3.1. *Π_{rg} in Figure 2 is a RoK from $\mathcal{R}_{\mathsf{rg}}^{\ell_h, B}$ to $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_J} \times \mathcal{R}_{\mathsf{batchlin}}$ w.r.t. relaxed input relation $\mathcal{R}_{\mathsf{rg}}^{\ell_h, B'}$ where $B' = 16 B_{d,k_g}/\sqrt{30}$. The completeness error is $\epsilon \approx n\lambda_{\mathsf{pj}} d/(\ell_h \cdot 2^{141})$.*

Proof. We show that Π_{rg} is ϵ-complete and knowledge sound w.r.t. $\mathcal{R}_{\mathsf{rg}}^{\ell_h, B'}$.

Completeness. If the input $(\mathbb{x} = c, \mathbb{w} = \mathbf{f}) \in \mathcal{R}_{\mathsf{rg}}^{\ell_h, B}$, it suffices to show that

- P^* aborts in Step 2 with probability at most ϵ. This holds because $\|\mathbf{H}\|_\infty \leq 9.5B \leq B_{d,k_g}$ with probability $1 - \epsilon$, by Eq. (29) and a union bound on Eq. (5).
- In Step 5, $ut_1^{(i)} = \langle \mathsf{ts}(\mathbf{s}), \mathbf{v}^{(i)} \rangle$ for all $i \in [k_g]$. This holds because
  - $\mathbf{v}^{(i)} = \mathbf{H}^{(i)\top} \mathsf{ts}(\mathbf{r})$ by definition.

Parameters: $\ell_h \in \mathbb{N}$, $B \in \mathbb{R}$, $d' := d-2$, $m := n\lambda_{\mathsf{pj}}/\ell_h$ with $md = n$, B_{d,k_g} in Eq. (29), $t(X) \in R_q$ in Eq. (3), and index $\mathbb{i}$ in Eq. (21)
Input: $\mathbb{x} = c \in \mathbb{C}$ and $\mathbb{w} = \mathbf{f} \in R_q^n$
Output instance: $\mathbb{x}_o = (\mathbb{x}^*, \mathbb{x}_{\mathsf{bat}})$ where $\mathbb{x}^* = (c, \mathbf{r} \in \mathbb{K}^{\log m}, v \in \mathbb{E})$, and

$$\mathbb{x}_{\mathsf{bat}} = (\mathbf{r}' := (\mathbf{r}, \mathbf{s}) \in \mathbb{K}^{\log n}, [c^{(i)} \in \mathbb{C},\, u^{(i)} \in \mathbb{E}]_{i=1}^{k_g})\,. \tag{32}$$

Output witness: $\mathbb{w}_o = (\mathbf{f} \in R_q^n,\, (\mathbf{g}^{(i)} \in \mathcal{M}^n)_{i=1}^{k_g})$
The protocol $\langle \mathsf{P}(\mathbb{x}; \mathbb{w}); \mathsf{V}(\mathbb{x}) \rangle$**:**

1. $\mathsf{V} \to \mathsf{P}$: Send projection matrix $\mathbf{J} \leftarrow\!\!\$\ \chi^{\lambda_{\mathsf{pj}} \times \ell_h}$
2. P : Let $\mathbf{H} := (\mathbf{I}_{n/\ell_h} \otimes \mathbf{J}) \times \mathsf{cf}(\mathbf{f}) \in \mathbb{Z}_q^{m \times d}$. Abort if $\|\mathbf{H}\|_\infty > B_{d,k_g}$. Otherwise, decompose $\mathbf{H}$ as
$$\mathbf{H} = \mathbf{H}^{(1)} + d'\mathbf{H}^{(2)} + \cdots + d'^{k_g-1}\mathbf{H}^{(k_g)} \tag{33}$$
where $\left\|\mathbf{H}^{(i)}\right\|_\infty \le d'/2$ for all $i \in [k_g]$. Flatten $(\mathbf{H}^{(i)})_{i=1}^{k_g}$ to $(\mathbf{h}^{(i)} := \mathsf{flt}(\mathbf{H}^{(i)}))_{i=1}^{k_g}$
3. $\mathsf{P} \to \mathsf{V}$: For $i \in [k_g]$, send $c^{(i)} := \mathbf{A} \times \mathbf{g}^{(i)} \in \mathbb{C}$ where $\mathbf{g}^{(i)} := \mathsf{Exp}(\mathbf{h}^{(i)}) \in \mathcal{M}^n$
4. $\mathsf{P} \leftrightarrow \mathsf{V}$: Run protocol Π_{mon} in Lemma 3.1 on input $((c^{(i)})_{i=1}^{k_g}, (\mathbf{g}^{(i)})_{i=1}^{k_g})$
   - At the end of round $\log(m)$, upon receiving the sumcheck challenge $\mathbf{r} \leftarrow\!\!\$\ \mathbb{K}^{\log(m)}$, P sends $\mathbf{v}^{(i)} := \mathbf{H}^{(i)\top}\mathsf{ts}(\mathbf{r}) \in \mathbb{K}^d$ for $i \in [k_g]$
   - Let $\mathbf{s} \leftarrow\!\!\$\ \mathbb{K}^{\log d}$ be the last $\log(d)$ challenges, and for $i \in [k_g]$, let $u^{(i)} := \langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathbf{g}^{(i)} \rangle \in \mathbb{E}$. Π_{mon} reduces to checking
   $$\left(\mathbb{x}_{\mathsf{bat}} = \left((\mathbf{r}, \mathbf{s}), [c^{(i)}, u^{(i)}]_{i=1}^{k_g}\right), (\mathbf{g}^{(i)})_{i=1}^{k_g}\right) \in \mathcal{R}_{\mathsf{batchlin}}\,. \qquad \text{(Eq. (28))}$$
5. V : For $i \in [k_g]$, compute $u^{(i)} \cdot t(X) \in \mathbb{E}$ and write it as the $\mathbb{K}$-vector space form
$$\left[ut_1^{(i)}, \ldots, ut_d^{(i)}\right] \in \mathbb{K}^{1 \times d}\,.$$
Reject if there exists $i \in [k_g]$ such that $ut_1^{(i)} \neq \langle \mathsf{ts}(\mathbf{s}), \mathbf{v}^{(i)} \rangle$, where $\mathbf{v}^{(i)}$ is the value received at Step 4 (that is supposed to be $\mathbf{H}^{(i)\top}\mathsf{ts}(\mathbf{r})$)
6. V : Set $\mathbb{x}^* := (c, \mathbf{r}, v \in \mathbb{E})$ where v's $\mathbb{K}$-vector space representation is
$$v := \left(\mathbf{v}^{(1)} + d'\mathbf{v}^{(2)} + \cdots d'^{k_g-1}\mathbf{v}^{(k_g)}\right)^\top \in \mathbb{K}^{1 \times d}\,. \tag{34}$$
Output $\mathbb{x}_o := (\mathbb{x}^*, \mathbb{x}_{\mathsf{bat}})$
7. P : output $\mathbb{w}_o = (\mathbf{f} \in R_q^n,\, (\mathbf{g}^{(i)} \in \mathcal{M}^n)_{i=1}^{k_g})$

Figure 2: The protocol Π_{rg} to reduce $\mathcal{R}_{\mathsf{rg}}^{\ell_h, B}$ to $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_J} \times \mathcal{R}_{\mathsf{batchlin}}$.

- $u^{(i)} = \langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathsf{Exp}(\mathbf{h}^{(i)})\rangle \in \mathbb{E}$ by definition. Hence, by viewing $\mathbb{E}$ as an R_q-module, $u^{(i)} \cdot t(X)$ equals $\langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathsf{Exp}(\mathbf{h}^{(i)}) \cdot t(X)\rangle$.
- By viewing $\mathbb{E}$ as a $\mathbb{K}$-vector space, the first column $ut_1^{(i)}$ of $u^{(i)} \cdot t(X)$ equals $\langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathsf{ct}(\mathsf{Exp}(\mathbf{h}^{(i)}) \cdot t(X))\rangle$.
- By Lemma 2.1, for all $i \in [k_g]$, $\mathsf{ct}(\mathsf{Exp}(\mathbf{h}^{(i)}) \cdot t(X)) = \mathsf{flt}(\mathbf{H}^{(i)})$. Thus,

$$\begin{aligned} ut_1^{(i)} &= \langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathsf{ct}(\mathsf{Exp}(\mathbf{h}^{(i)}) \cdot t(X))\rangle = \langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathsf{flt}(\mathbf{H}^{(i)})\rangle \\ &= \langle \mathsf{ts}(\mathbf{s}), \mathbf{H}^{(i)\top}\mathsf{ts}(\mathbf{r})\rangle = \langle \mathsf{ts}(\mathbf{s}), \mathbf{v}^{(i)}\rangle . \end{aligned}$$

- $((c, \mathbf{r}, v), \mathbf{f}) \in \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_J}$. Since c is the commitment to $\mathbf{f}$, it suffices to prove the linear statement $\langle (\mathbf{M}_J\mathbf{f}), \mathsf{ts}(\mathbf{r})\rangle = v$ where $\mathbf{M}_J := \mathbf{I}_{n/\ell_h} \otimes \mathbf{J}$. Towards this, for $\mathbf{H} := (\mathbf{I}_{n/\ell_h} \otimes \mathbf{J}) \times \mathsf{cf}(\mathbf{f})$, it suffices to check $\mathbf{H}^\top\mathsf{ts}(\mathbf{r}) = v^\top \in \mathbb{K}^d$ (by viewing $v \in \mathbb{E}$ as an element in $\mathbb{K}^{1\times d}$). This holds because $\mathbf{v}^{(i)} = \mathbf{H}^{(i)\top}\mathsf{ts}(\mathbf{r})$ for all $i \in [k_g]$ by Step 4. Then $v^\top = \mathbf{H}^\top\mathsf{ts}(\mathbf{r})$ follows from Eq. (33) and Eq. (34).
- $(\mathbb{x}_{\mathsf{bat}}, (\mathbf{g}^{(i)})_{i=1}^{k_g}) \in \mathcal{R}_{\mathsf{batchlin}}$. This holds by assignments of $c^{(i)}$, $u^{(i)}$ in Step 3, 4.

Knowledge soundness. Consider an adversary P^*. The extractor simulates the protocol with P^*, and from P^*'s output $\mathbb{w}_o = (\mathbf{f}, (\mathbf{g}^{(i)})_{i=1}^{k_g})$, it returns $\mathbf{f}$. Let E denote the event the verifier checks pass and $(\mathbb{x}_o, \mathbb{w}_o)$ is in the output relation, but $(c, \mathbf{f}) \notin \mathcal{R}_{\mathsf{rg}}^{\ell_h, B'}$ where c is the input commitment chosen by P^*. Observe that E occurs only if at least one of the following bad events occurs.

- The norm check in $\mathsf{VfyOpen}_{\ell_h,B'}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f})$ (Eq. (13)) fails for $B' = 16B_{d,k_g}/\sqrt{30}$, but the random projection $\mathbf{H} := (\mathbf{I}_{n/\ell_h} \otimes \mathbf{J}) \times \mathsf{cf}(\mathbf{f})$ has infinity norm $\|\mathbf{H}\|_\infty \le B_{d,k_g}$. Recall $\lambda_{\mathsf{pj}} := 256$ and write

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{1,1} & \dots & \mathbf{H}_{1,d} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{m/\lambda_{\mathsf{pj}},1} & \dots & \mathbf{H}_{m/\lambda_{\mathsf{pj}},d} \end{bmatrix}$$

  where $\mathbf{H}_{i,j} \in \mathbb{Z}_q^{\lambda_{\mathsf{pj}}}$ for $i \in [m/\lambda_{\mathsf{pj}}], j \in [d]$. $\|\mathbf{H}\|_\infty \le B_{d,k_g}$ implies that $\|\mathbf{H}_{i,j}\|_2 \le \sqrt{\lambda_{\mathsf{pj}}} \cdot B_{d,k_g} = 16B_{d,k_g} = \sqrt{30}B'$ for all $i \in [m/\lambda_{\mathsf{pj}}], j \in [d]$. Since $\mathbf{f}$ is bound to c (as $(\mathbb{x}_o, \mathbb{w}_o)$ is valid by assumption), by Eq. (6) of Lemma 2.2, this event happens with probability at most 2^{-128}.

- P^*'s output $(\mathbf{g}^{(i)})_{i=1}^{k_g}$ are not monomial vectors. By Lemma 3.1, this happens with negligibile probability.

- The random projection $\mathbf{H}$ has infinity norm $\|\mathbf{H}\|_\infty > B_{d,k_g}$ and $(\mathbf{g}^{(i)})_{i=1}^{k_g}$ are monomial vectors. For $i \in [k_g]$, define $\mathbf{H}^{(i)} \in \mathbb{Z}_q^{m\times d}$ such that

$$\mathsf{flt}(\mathbf{H}^{(i)}) = \mathsf{ct}(\mathbf{g}^{(i)} \cdot t(X)) \in \mathbb{Z}_q^{md} . \tag{35}$$

By Lemma 2.1, $\left\|\mathbf{H}^{(i)}\right\|_\infty \le d'/2$ for all $i \in [k_g]$. Since $\|\mathbf{H}\|_\infty > B_{d,k_g}$,

$$\mathbf{H} \ne \mathbf{H}^{(1)} + d'\mathbf{H}^{(2)} + \cdots + d'^{k_g-1}\mathbf{H}^{(k_g)}\,. \tag{36}$$

We first claim that $\langle \mathsf{ts}(\mathbf{s}), \mathbf{H}^{(i)\top}\mathsf{ts}(\mathbf{r})\rangle = \langle \mathsf{ts}(\mathbf{s}), \mathbf{v}^{(i)}\rangle$ for all $i \in [k_g]$:

- Since $(\mathbb{x}_o, \mathbb{w}_o)$ is in the output relation, $u^{(i)} = \langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathbf{g}^{(i)}\rangle$ for $i \in [k_g]$.
- By viewing $\mathbb{E}$ as an R_q-module, $u^{(i)} \cdot t(X)$ equals $\langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathbf{g}^{(i)} \cdot t(X)\rangle$.
- By viewing $\mathbb{E}$ as a $\mathbb{K}$-vector space, the 1st column $ut_1^{(i)}$ is $\langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathsf{ct}(\mathbf{g}^{(i)} \cdot t(X))\rangle$, which is $\langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathsf{flt}(\mathbf{H}^{(i)})\rangle = \langle \mathsf{ts}(\mathbf{s}), \mathbf{H}^{(i)\top}\mathsf{ts}(\mathbf{r})\rangle$ by Eq. (35).
- $ut_1^{(i)} = \langle \mathsf{ts}(\mathbf{s}), \mathbf{v}^{(i)}\rangle$ by the check at Step 5. Thus the claim holds.

Thus, with overwhelming probability over $\mathbf{s}$, we have $\mathbf{H}^{(i)\top}\mathsf{ts}(\mathbf{r}) = \mathbf{v}^{(i)}$ for all $i \in [k_g]$. Since $((c, \mathbf{r}, v), \mathbf{f}) \in \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_J}$ by assumption, $\mathbf{H}^\top\mathsf{ts}(\mathbf{r}) = v^\top$ for $\mathbf{H} := (\mathbf{I}_{n/\ell_h} \otimes \mathbf{J}) \times \mathsf{cf}(\mathbf{f})$, and by the verifier check in Eq. (34), we have

$$\begin{aligned}
\mathbf{H}^\top\mathsf{ts}(\mathbf{r}) = v^\top &= \left(\mathbf{v}^{(1)} + d'\mathbf{v}^{(2)} + \cdots d'^{k_g-1}\mathbf{v}^{(k_g)}\right) \\
&= \left(\mathbf{H}^{(1)\top}\mathsf{ts}(\mathbf{r}) + d'\mathbf{H}^{(2)\top}\mathsf{ts}(\mathbf{r}) + \cdots d'^{k_g-1}\mathbf{H}^{(k_g)\top}\mathsf{ts}(\mathbf{r})\right) \\
&= \left(\mathbf{H}^{(1)} + d'\mathbf{H}^{(2)} + \cdots d'^{k_g-1}\mathbf{H}^{(k_g)}\right)^\top \mathsf{ts}(\mathbf{r})\,.
\end{aligned}$$

Note that $\mathbf{g}^{(i)}$ $(1 \le i \le k_g)$ is bound to $c^{(i)}$, which is sent before sampling $\mathbf{r}$. By Eq. (36), the above equation holds with negligible probability over $\mathbf{r}$.

Thus, E occurs with negligible probability and knowledge soundness holds. □

Proposition 3.3. *Π_{rg} in Figure 2 runs a single degree-3 sumcheck protocol over $\mathbb{K}$ of size n. The additional complexity beyond the sumcheck is as follows:*

- ***Prover cost** $T_p^{\mathsf{rg}}(k_g, n)$**:** $nd\lambda_{\mathsf{pj}}$ low-norm $\mathbb{Z}$-additions (Step 2), $O(k_g\kappa n)$ R_q-additions (Step 3), $O(n)$ $\mathbb{K}$-ops (Step 4), and $T_p^{\mathsf{mon}}(k_g, n)$.*
- ***Verifier cost** $T_v^{\mathsf{rg}}(k_g, n)$**:** $k_g t$ R_q-ops and d $\mathbb{K}$-ops (Step 5), $k_g d$ scalar multiplications between $\mathbb{Z}_q$ and $\mathbb{K}$ (Step 6), and $T_v^{\mathsf{mon}}(k_g, n)$.*
- ***Verifier randomness** $\Gamma_v^{\mathsf{rg}}(k_g, n)$**:** $\lambda_{\mathsf{pj}}\ell_h$ bits plus the randomness for Π_{mon}.*

4 High-Arity Folding for NP-Complete Relations

In this section, we present a high-arity folding scheme for committed NP statements. Section 4.1 introduces the generalized committed R1CS relation that captures real-world computations. Section 4.2 then describes a folding scheme that compresses a large number (e.g. 1024) of generalized R1CS statements.

4.1 Generalized Committed R1CS Relation

In this section, we define the **generalized committed R1CS relation** $\mathcal{R}_{\mathsf{gr1cs}}$. The definition naturally extends to customizable constraint systems (CCS) [STW23].

Formally, setting auxiliary parameters

$$\mathsf{aux} := (n_{\mathsf{in}}, n_w, n := n_{\mathsf{in}} + n_w, \ell_h \in \mathbb{N}, B \in \mathbb{R}, (\mathbf{M}_i \in \mathbb{Z}_q^{m \times n})_{i=1}^{3}) \,. \tag{37}$$

The generalized R1CS relation is defined as follows:

$$\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}} := \left\{ (\mathbb{x}, \mathbb{w}) \; : \; \begin{array}{l} \mathbb{x} = (c \in \mathbb{C}, \mathbf{X}_{\mathsf{in}} \in \mathbb{Z}_q^{n_{\mathsf{in}} \times d}), \quad \mathbb{w} = \mathbf{W} \in \mathbb{Z}_q^{n_w \times d} \quad \text{s.t.} \\ \mathbf{F}^\top := [\mathbf{X}_{\mathsf{in}}^\top, \mathbf{W}^\top] \in \mathbb{Z}_q^{d \times n} \quad \text{s.t.} \\ (\mathbf{M}_1 \times \mathbf{F}) \circ (\mathbf{M}_2 \times \mathbf{F}) = \mathbf{M}_3 \times \mathbf{F} \\ \wedge \; \mathsf{VfyOpen}_{\ell_h, B}(\mathsf{pp}_{\mathsf{cm}}, c, \mathsf{cf}^{-1}(\mathbf{F})) = 1 \end{array} \right\} . \tag{38}$$

We can view this as checking d R1CS statements over $\mathbb{Z}_q$ in batch. However, the witnesses in $\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}}$ must be low-norm. To handle standard R1CS statements over $\mathbb{Z}_q$ with *arbitrary* witnesses, we apply the following standard trick.

Let q be a prime, choose a base $b \in \mathbb{N}$, and set $k_{\mathsf{cs}} := 1 + \lfloor \log_b(q) \rfloor$. Given the original R1CS matrices $(\bar{\mathbf{M}}_i \in \mathbb{Z}_q^{m \times \bar{n}})_{i=1}^{3}$, we define the new R1CS matrices as

$$(\mathbf{M}_i := \bar{\mathbf{M}}_i \otimes [1, b, \ldots, b^{k_{\mathsf{cs}}-1}] \in \mathbb{Z}_q^{m \times n})_{i=1}^{3}$$

where $n := \bar{n} k_{\mathsf{cs}}$. Given the original instance-witness pairs $(x_{\mathsf{in}}^{(i)} \in \mathbb{Z}_q^{\bar{n}_{\mathsf{in}}}, \mathbf{w}^{(i)} \in \mathbb{Z}_q^{\bar{n}_w})_{i=1}^{d}$, an honest prover sets the converted instance $\mathbb{x}$ and witness $\mathbb{w}$ as

$$\mathbb{x} := \left(c, \; \mathbf{X}_{\mathsf{in}} := \left[\mathsf{decomp}_{b,k_{\mathsf{cs}}}(x_{\mathsf{in}}^{(1)}) || \ldots || \mathsf{decomp}_{b,k_{\mathsf{cs}}}(x_{\mathsf{in}}^{(d)}) \right] \in \mathbb{Z}_q^{n_{\mathsf{in}} \times d} \right)$$
$$\mathbb{w} := \mathbf{W} = \left[\mathsf{decomp}_{b,k_{\mathsf{cs}}}(\mathbf{w}^{(1)}) || \ldots || \mathsf{decomp}_{b,k_{\mathsf{cs}}}(\mathbf{w}^{(d)}) \right] \in \mathbb{Z}_q^{n_w \times d} ,$$

where $(n_{\mathsf{in}}, n_w) = (\bar{n}_{\mathsf{in}}, \bar{n}_w) \cdot k_{\mathsf{cs}}$ and c is the commitment to $\mathbf{F}$ as in Eq. (38). It then proves the relation $\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}}$ where the parameter aux is defined as

$$n_{\mathsf{in}} = \bar{n}_{\mathsf{in}} k_{\mathsf{cs}}, \; n_w = \bar{n}_w k_{\mathsf{cs}}, \; n = n_{\mathsf{in}} + n_w, \; \ell_h \in \mathbb{N}, \; B = 0.5 b \sqrt{\ell_h}, \; (\mathbf{M}_i)_{i=1}^{3} \,.$$

First, if the prover is honest, the converted statement will be in $\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}}$ because

1. Each entry of $\mathbf{F}$ is bounded by $b/2$, so the commitment opening relation holds for bound B.
2. Denote by $\mathbf{f}^{(j)} := (x_{\mathsf{in}}^{(j)}, \mathbf{w}^{(j)})$ for all $j \in [d]$. The Hadamard product condition preserves since for every $i \in [3]$, $j \in [d]$,

$$\bar{\mathbf{M}}_i \times \mathbf{f}^{(j)} = (\bar{\mathbf{M}}_i \otimes [1, b, \ldots, b^{k_{\mathsf{cs}}-1}]) \times \mathsf{decomp}_{b,k_{\mathsf{cs}}}(\mathbf{f}^{(j)}) = \mathbf{M}_i \times \mathbf{F}_{*,j} \,.$$

On the other hand, if we know a witness[9] $\mathbf{W}$ for $(c, \mathbf{X}_{\mathsf{in}})$, where the Hadamard product condition (under $\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}}$) holds and the opening $\mathbf{F}^\top := [\mathbf{X}_{\mathsf{in}}^\top, \mathbf{W}^\top]$ is bound to commitment c, then we can recover a witness for the original R1CS statements by combining every k_{cs} rows of $\mathbf{W}$ with the coefficients $[1, b, \ldots, b^{k_{\mathsf{cs}}-1}]$.

[9]The low-norm $\mathbf{W}$ is not necessarily of ℓ_∞-norm less than $b/2$.

4.2 The High-Arity Folding Scheme

Parameters: $(\mathbf{M}_i \in \mathbb{Z}_q^{m\times n})_{i=1}^3$, $m_J := n\lambda_{\mathsf{pj}}/\ell_h$ with $m_J \le m \le n$
Input: $\mathbb{x} = (c \in \mathbb{C}, \mathbf{X}_{\mathsf{in}} \in \mathbb{Z}_q^{n_{\mathsf{in}}\times d})$, $\quad \mathbb{w} = \mathbf{W} \in \mathbb{Z}_q^{n_w \times d}$
Output instance: $\mathbb{x}_o = (\mathbb{x}^*, \mathbb{x}_{\mathsf{bat}})$ where

$$\mathbb{x}^* = (c,\, x_{\mathsf{in}} \in R_q^{n_{\mathsf{in}}}, \mathbf{r} := (\bar{\mathbf{r}} \in \mathbb{K}^{\log(m_J)}, \bar{\mathbf{s}}) \in \mathbb{K}^{\log(m)}, \mathbf{v} \in \mathbb{E}^4) \tag{39}$$

$$\mathbb{x}_{\mathsf{bat}} = ((\bar{\mathbf{r}}, \bar{\mathbf{s}}, \mathbf{s}) \in \mathbb{K}^{\log n},\, [c^{(i)} \in \mathbb{C},\, u^{(i)} \in \mathbb{E}]_{i=1}^{k_g}) \tag{40}$$

Output witness: $\mathbb{w}_o = (\mathbf{f} \in R_q^n,\, (\mathbf{g}^{(i)} \in \mathcal{M}^n)_{i=1}^{k_g})$
The protocol $\langle \mathsf{P}(\mathbb{x};\mathbb{w}); \mathsf{V}(\mathbb{x})\rangle$**:**

1. $\mathsf{V} \to \mathsf{P}$: Send challenges below:
   - $\mathbf{J} \leftarrow\!\!\$\; \chi^{\lambda_{\mathsf{pj}}\times \ell_h}$ in Step 1 of Π_{rg} (Figure 2)
   - $\mathbf{s}' \leftarrow\!\!\$\; \mathbb{K}^{\log(m)}$, $\alpha \leftarrow\!\!\$\; \mathbb{K}$ in Step 1 of Π_{had} (Figure 1)
2. $\mathsf{P} \to \mathsf{V}$: Send helper commitments $(c^{(i)} := \mathbf{A} \times \mathbf{g}^{(i)})_{i=1}^{k_g}$ in Step 3 of Π_{rg} (Figure 2)
3. $\mathsf{P} \leftrightarrow \mathsf{V}$: Run two sumcheck protocols in parallel[a] for
   - the sumcheck claim from Eq. (25) in Step 2 of Π_{had}
   - the sumcheck claim from Π_{mon} in Step 4 of Π_{rg}

   The first sumcheck has $\log(m)$ rounds, the second has $\log(n)$ rounds. The two protocols share the sumcheck challenge

   $$(\bar{\mathbf{r}} \in \mathbb{K}^{\log(m_J)},\, \bar{\mathbf{s}} \in \mathbb{K}^{\log(m/m_J)},\, \mathbf{s} \in \mathbb{K}^{\log(n/m)})$$

4. $\mathsf{P} \leftrightarrow \mathsf{V}$: Execute the rest of Π_{had} and Π_{rg}.
   - Π_{had} outputs

   $$\mathbb{x}_{\mathsf{had},o} = (c,\, \mathbf{r} := (\bar{\mathbf{r}}, \bar{\mathbf{s}}), \mathbf{v}' \in \mathbb{E}^3), \qquad \mathbb{w}_{\mathsf{had},o} = \mathbf{f} \in R_q^n \tag{41}$$

   - Π_{rg} outputs

   $$\mathbb{x}_{\mathsf{rg},o} = \left[\mathbb{x} = (c, \bar{\mathbf{r}}, v \in \mathbb{E}),\, \mathbb{x}_{\mathsf{bat}} = ((\bar{\mathbf{r}}, \bar{\mathbf{s}}, \mathbf{s}),\, [c^{(i)},\, u^{(i)} \in \mathbb{E}]_{i=1}^{k_g})\right] \tag{42}$$

   $$\mathbb{w}_{\mathsf{rg},o} = (\mathbf{f}, (\mathbf{g}^{(i)} \in \mathcal{M}^n)_{i=1}^{k_g}) \tag{43}$$

5. V : Output $\mathbb{x}_o = (\mathbb{x}^*, \mathbb{x}_{\mathsf{bat}})$ where $\mathbb{x}^* = (c, x_{\mathsf{in}} := \mathsf{cf}^{-1}(\mathbf{X}_{\mathsf{in}}), \mathbf{r}, \mathbf{v} := (\mathbf{v}', v))$ and $\mathbb{x}_{\mathsf{bat}}$ is in Eq. (42)
6. P : Output $\mathbb{w}_o = \mathbb{w}_{\mathsf{rg},o}$

[a]We can further combine the two sumcheck instances to one via random linear combination. We omit this optimization for simplicity.

Figure 3: The protocol Π_{gr1cs} to reduce $\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}}$ to $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux_{cs}}} \times \mathcal{R}_{\mathsf{batchlin}}$.

We present a folding scheme that compresses $\ell_{\mathsf{np}} = \mathsf{poly}(\lambda)$ (generalized) R1CS statements.

Single-instance reduction. To build intuition, we first describe a reduction for a single R1CS instance before presenting the full folding scheme.

Fix R1CS parameter aux from Eq. (37). We construct an RoK from $\mathcal{R}^{\mathsf{aux}}_{\mathsf{gr1cs}}$ (Eq. (38)) to $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin}}$ where $\mathcal{R}_{\mathsf{batchlin}}$ is defined in Eq. (28). The parameter $\mathsf{aux}_{\mathsf{cs}}$ is given by:

$$\mathsf{aux}_{\mathsf{cs}} := (n_{\mathsf{in}}, (\mathbf{M}_i)_{i=1}^{4}), \tag{44}$$

where $n_{\mathsf{in}}, (\mathbf{M}_i)_{i=1}^{3}$ are the same as in aux, and $\mathbf{M}_4 := \mathbf{I}_{n/\ell_h} \otimes \mathbf{J}$ for a random $\mathbf{J} \leftarrow\$ \chi^{\lambda_{\mathsf{pj}} \times \ell_h}$. Denote by m and m_J the number of rows of $(\mathbf{M}_i)_{i=1}^{3}$ and $\mathbf{M}_4$, respectively. WLOG, we assume that they are all powers-of-two and $m_J \leq m \leq n$. The scheme naturally extends to more general choices of parameters.

The protocol Π_{gr1cs} is described in Figure 3. Informally, it interleaves the approximate range proof Π_{rg} (Figure 2) with the Hadamard proof Π_{had} (Figure 1).

Lemma 4.1. *Let* aux *be the R1CS parameter defined in Eq. (37). Define* aux' *as* aux *except that the norm bound B is replaced with $B' = 16B_{d,k_g}/\sqrt{30}$.* Π_{gr1cs} *in Figure 3 is a reduction of knowledge from* $\mathcal{R}^{\mathsf{aux}}_{\mathsf{gr1cs}}$ *(Eq. (38)) to* $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin}}$*, with the relaxed input relation* $\mathcal{R}^{\mathsf{aux}'}_{\mathsf{gr1cs}}$*. Here,* $\mathcal{R}_{\mathsf{batchlin}}$ *is from Eq. (28) and* $\mathsf{aux}_{\mathsf{cs}}$ *is defined in Eq. (44). The completeness error ϵ matches that of Theorem 3.1.*

Proof. We show that Π_{gr1cs} is ϵ-complete and knowledge sound w.r.t. $\mathcal{R}^{\mathsf{aux}'}_{\mathsf{gr1cs}}$.

Completeness. Let $(\mathbb{x} := (c, \mathbf{X}_{\mathsf{in}}), \mathbb{w} := \mathbf{W})$ be the input. Set $\mathbf{F}^\top := [\mathbf{X}_{\mathsf{in}}^\top, \mathbf{W}^\top]$ and $\mathbf{f} := \mathsf{cf}^{-1}(\mathbf{F})$. Denote by the output as

$$\mathbb{x}_o := (\mathbb{x}^* = (c, \mathsf{cf}^{-1}(\mathbf{X}_{\mathsf{in}}), \mathbf{r}, \mathbf{v} := (\mathbf{v}' \in \mathbb{E}^3, v \in \mathbb{E})), \mathbb{x}_{\mathsf{bat}}), \qquad \mathbb{w}_o := (\mathbf{f}, (\mathbf{g}^{(i)})_{i=1}^{k_g}).$$

If $(\mathbb{x}, \mathbb{w})$ is in $\mathcal{R}^{\mathsf{aux}}_{\mathsf{gr1cs}}$, then $(c, \mathbf{F})$ is in $\mathcal{R}^{\mathsf{aux}}_{\mathsf{had}}$ and $(c, \mathbf{f})$ is in $\mathcal{R}^{\ell_h, B}_{\mathsf{rg}}$. By the perfect completeness of Π_{had} (Proposition 3.1) and ϵ-completeness of Π_{rg} (Theorem 3.1), with probability $1-\epsilon$, it holds that $((c, \mathbf{r}, \mathbf{v}'), \mathbf{f})$ is in $\mathcal{R}^{\mathsf{aux}}_{\mathsf{lin}}$ (Eq. (24)) and $(((c, \mathbf{r}, v), \mathbb{x}_{\mathsf{bat}}), \mathbb{w}_o)$ is in $\mathcal{R}^{\mathsf{aux}_J}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin}}$ (Eq. (31) and Eq. (28)). This implies that the output is in $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin}}$ as desired.

Knowledge soundness. Consider an adversary P^*. The extractor simulates the protocol with P^*, and from P^*'s output $\mathbb{w}_o = (\mathbf{f}, (\mathbf{g}^{(i)})_{i=1}^{k_g})$, it parses $\mathsf{cf}(\mathbf{f})$ as $\mathsf{cf}(\mathbf{f})^\top = [\mathbf{X}^\top, \mathbf{W}^\top]$ and returns $\mathbf{W} \in \mathbb{Z}_q^{n_w \times d}$.

Let $(\mathbb{x}_o, \mathbb{w}_o)$ denote the output instance and witness. If $(\mathbb{x}_o, \mathbb{w}_o)$ is in $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin}}$, then by the knowledge soundness of Π_{had} (Proposition 3.1), Π_{rg} (Theorem 3.1), and by definition of $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}}$ (Eq. (44)), with overwhelming probability, we have

$$(c, \mathbf{f}) \in \mathcal{R}^{\ell_h, B'}_{\mathsf{rg}} \;\wedge\; ((c, \mathsf{cf}(\mathbf{f})) \in \mathcal{R}^{\mathsf{aux}}_{\mathsf{had}}) \;\wedge\; (\mathbf{X} = \mathbf{X}_{\mathsf{in}}).$$

Thus, the extractor outputs a valid witness for $\mathcal{R}^{\mathsf{aux}'}_{\mathsf{gr1cs}}$ as desired. $\square$

Proposition 4.1. Π_{gr1cs} *in Figure 3 runs two degree-3 sumcheck protocols over $\mathbb{K}$, one of size n and the other of size m. The additional complexity beyond the sumcheck is:*

- ***Prover cost:*** $T_p^{\mathsf{gr1cs}}(k_g, n, m) := T_p^{\mathsf{had}}(m) + T_p^{\mathsf{rg}}(k_g, n)$.
- ***Verifier cost:*** $T_v^{\mathsf{gr1cs}}(k_g, n, m) := T_v^{\mathsf{had}}(m) + T_v^{\mathsf{rg}}(k_g, n)$.
- ***Verifier randomness:*** $\Gamma_v^{\mathsf{gr1cs}}(k_g, n, m) := \Gamma_v^{\mathsf{had}}(m) + T_v^{\mathsf{rg}}(k_g, n)$.

$T_p^{\mathsf{had}}(m)$, $T_v^{\mathsf{had}}(m)$, $\Gamma_v^{\mathsf{had}}(m)$ are defined in Proposition 3.2; $T_p^{\mathsf{rg}}(k_g, n)$, $T_v^{\mathsf{rg}}(k_g, n)$, $\Gamma_v^{\mathsf{rg}}(k_g, n)$ are defined in Proposition 3.3.

Multi-instances reduction. Figure 4 describes the high-arity folding scheme Π_{fold}:

Step 1-3: Π_{fold} executes ℓ_{np} parallel instances of Π_{gr1cs} (Figure 3) with shared randomness. To improve efficiency, the $2\ell_{\mathsf{np}}$ sumcheck instances are merged into two via random linear combination.

Step 4-6: The verifier samples a low-norm challenge vector $\boldsymbol{\beta} \leftarrow\$ \mathcal{S}^{\ell_{\mathsf{np}}}$. The commitments, evaluations, and witnesses are then folded with respect to $\boldsymbol{\beta}$.

Remark 4.1 (Memory Efficiency). We describe a prover algorithm for Π_{fold} where the memory cost is about the same for running a single Π_{gr1cs} instance with witness size n. As a tradeoff, the prover takes $2 + \log\log(n)$ passes to the input data.

1. The algorithm starts by computing the ℓ_{np} input commitments in a stream and obtains the first-round random challenges.
2. Once getting the random combiner α, the prover executes the sumcheck using the algorithm from Section 4 of [Baw+25]. For each of the $\log\log(n)$ passes over the input data, the prover computes the sumcheck evaluation table by linearly combining the table of each instance. The sumcheck prover takes time $O(n \log\log(n))$ plus the cost of combining the ℓ_{np} evaluation tables, where the latter is dominant.
3. After executing the sumcheck and deriving the folding challenge $\boldsymbol{\beta}$, the prover streams the input witnesses again and combines them into a single folded witness.

Theorem 4.1. *Let* aux, $\mathsf{aux}_{\mathsf{cs}}$ *be the parameters defined in Eq. (37) and Eq. (44). Define* aux' *as* aux *except that the norm bound B is replaced with $B' = 16B_{d,k_g}/\sqrt{30}$. Assume that Construction 2.1 is relaxed binding w.r.t. opening space $\mathcal{O}'_{B_{\mathsf{rbnd}}} \times (\mathcal{S} - \mathcal{S})$ (Eq. (8)), where $|\mathcal{S}| = \omega(\mathsf{poly}(\lambda))$, $\mathcal{S} - \mathcal{S}$ is invertible over R_q, and*

$$B_{\mathsf{rbnd}}/2 = B_{\mathsf{bnd}} \geq \ell_{\mathsf{np}} \cdot \|\mathcal{S}\|_{\mathsf{op}} \cdot \max(B \cdot \sqrt{nd/\ell_h}, \sqrt{n})\,. \tag{50}$$

For $\ell_{\mathsf{np}} = \mathsf{poly}(\lambda)$, Π_{fold} in Figure 4 is an RoK from $(\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}})^{\ell_{\mathsf{np}}}$ (Eq. (38)) to $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin}}$ (Eq. (28)), with the relaxed input relation $(\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}'})^{\ell_{\mathsf{np}}}$. The completeness error is $\ell_{\mathsf{np}} \cdot \epsilon$ with ϵ being the completeness error of Π_{gr1cs} (Lemma 4.1).

Remark 4.2. Eq. (50) is a worst-case bound. In practice, the folding challenge $\boldsymbol{\beta}$ is sampled from a symmetric distribution, the folded witness is expected to have a much lower norm. We leave the concrete probability analysis for future work.

Proof of Theorem 4.1. We show that Π_{fold} is $\ell_{\mathsf{np}}{\cdot}\epsilon$-complete and knowledge sound w.r.t. $(\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}'})^{\ell_{\mathsf{np}}}$.

Parameters: $(\mathbf{M}_i \in \mathbb{Z}_q^{m\times n})_{i=1}^3$, $m_J := n\lambda_{\mathsf{pj}}/\ell_h$ with $m_J \le m \le n$

Input: $\mathbb{x} := \left\{\mathbb{x}_\ell = (c_\ell \in \mathbb{C}, \mathbf{X}_{\mathsf{in}}^\ell \in \mathbb{Z}_q^{n_{\mathsf{in}}\times d})\right\}_{\ell=1}^{\ell_{\mathsf{np}}}$, $\quad \mathbb{w} := \left\{\mathbb{w}_\ell = \mathbf{W}^\ell \in \mathbb{Z}_q^{n_w\times d}\right\}_{\ell=1}^{\ell_{\mathsf{np}}}$

For $\ell \in [\ell_{\mathsf{np}}]$, denote by $\mathbf{f}^\ell := \mathsf{cf}^{-1}\left([\mathbf{X}_{\mathsf{in}}^{\ell\top}, \mathbf{W}^{\ell\top}]^\top\right) \in R_q^n$.

Output: $\mathbb{x}_o = (\mathbb{x}^*, \mathbb{x}_{\mathsf{bat}})$ (defined in Eq. (39)) and $\mathbb{w}_o = (\mathbf{f}^* \in R_q^n, (\mathbf{g}^{(i)} \in \mathcal{M}^n)_{i=1}^{k_g})$

The protocol $\langle \mathsf{P}(\mathbb{x};\mathbb{w}); \mathsf{V}(\mathbb{x})\rangle$**:**

1. $\mathsf{P} \leftrightarrow \mathsf{V}$: Run ℓ_{np} parallel instances of Π_{gr1cs} (Figure 3), where the ℓ-th ($1 \le \ell \le \ell_{\mathsf{np}}$) instance takes input $(\mathbb{x}_\ell, \mathbb{w}_\ell)$. It further applies optimizations below:
   - The parallel executions reuse the same challenges $\mathbf{J}, \mathbf{s}', \alpha$ in Step 1 of Π_{gr1cs}
   - The $2\ell_{\mathsf{np}}$ parallel sumcheck claims are merged into 2 claims:
     - The 1st claim is of the form
     $$\sum_{b\in\{0,1\}^{\log m}} \left(\sum_{\ell=1}^{\ell_{\mathsf{np}}}\sum_{j=1}^{d} \alpha^{(\ell-1)\cdot d+j-1}\cdot f^{\ell,j}(b)\right) = 0 \tag{45}$$
     where $f^{\ell,j}(\mathbf{X}) \in \mathbb{K}[X_1,\dots,X_{\log m}]$ ($1 \le j \le d$) is defined as
     $$f^{\ell,j}(\mathbf{X}) = eq(\mathbf{s}', \mathbf{X})\cdot\left(g_1^{\ell,j}(\mathbf{X})\cdot g_2^{\ell,j}(\mathbf{X}) - g_3^{\ell,j}(\mathbf{X})\right)$$
     and $g_i^{\ell,j}(\mathbf{X})$ ($1 \le i \le 3$) is the multilinear extension of $\mathbf{M}_i \times \mathsf{cf}(\mathbf{f}^\ell)_{*,j} \in \mathbb{Z}_q^m$
     - The 2nd claim is the random linear combination of the ℓ_{np} sumcheck claims in the monomial check Π_{mon} in Step 4 of Π_{rg}, with powers of α being the random combiners

     The 1st sumcheck has $\log(m)$ rounds, the 2nd has $\log(n)$ rounds.
     The shared sumcheck challenge is
     $$(\bar{\mathbf{r}} \in \mathbb{K}^{\log(m_J)}, \bar{\mathbf{s}} \in \mathbb{K}^{\log(m/m_J)}, \mathbf{s} \in \mathbb{K}^{\log(n/m)})$$
   - The 1st and the 2nd sumcheck protocols reduce to evaluation values $e^* \in \mathbb{E}$ and $u^* \in \mathbb{E}$, respectively
2. $\mathsf{P} \leftrightarrow \mathsf{V}$: The ℓ-th ($\ell \in [\ell_{\mathsf{np}}]$) execution of Π_{gr1cs} reduces to $(\mathbb{x}_{o,\ell} := (\mathbb{x}_\ell^*, \mathbb{x}_{\mathsf{bat},\ell}), \mathbb{w}_{o,\ell})$:
   $$\mathbb{x}_\ell^* := (c_\ell, \mathsf{cf}^{-1}(\mathbf{X}_{\mathsf{in}}^\ell), (\bar{\mathbf{r}}, \bar{\mathbf{s}}), \mathbf{v}^\ell \in \mathbb{E}^4) \quad \mathbb{x}_{\mathsf{bat},\ell} := ((\bar{\mathbf{r}}, \bar{\mathbf{s}}, \mathbf{s}), [c_\ell^{(i)}, u_\ell^{(i)} \in \mathbb{E}]_{i=1}^{k_g}) \tag{46}$$
   $$\mathbb{w}_{o,\ell} := (\mathbf{f}^\ell, (\mathbf{g}^{i,\ell})_{i=1}^{k_g}) \tag{47}$$
   where $c_\ell^{(i)}$ ($1 \le i \le k_g$) is the monomial commitment to $\mathbf{g}^{i,\ell}$ sent by the prover
3. V : check that evaluations $(\mathbf{v}^\ell)_{\ell=1}^{\ell_{\mathsf{np}}}$ and $(u_\ell^{(i)})_{i\in[k_g],\ell\in[\ell_{\mathsf{np}}]}$ are consistent with e^*, u^* according to Eq. (45) and the randomly-combined claim for Π_{mon} above

Figure 4: The protocol Π_{fold} to reduce $(\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}})^{\ell_{\mathsf{np}}}$ to $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux_{cs}}} \times \mathcal{R}_{\mathsf{batchlin}}$: Part I.

4. $\mathsf{V} \to \mathsf{P}$: Send low-norm challenge $\boldsymbol{\beta} \leftarrow\!\$\; \mathcal{S}^{\ell_{\mathsf{np}}}$
5. V : Set $\mathbb{x}^* := (c^* \in \mathbb{C}, x^*_{\mathsf{in}} \in R_q^{n_{\mathsf{in}}}, (\bar{\mathbf{r}}, \bar{\mathbf{s}}) \in \mathbb{K}^{\log(m)}, \mathbf{v}^* \in \mathbb{E}^4)$ where

$$(c^*, x^*_{\mathsf{in}}, \mathbf{v}^*) := \sum_{\ell=1}^{\ell_{\mathsf{np}}} \boldsymbol{\beta}_\ell \cdot (c_\ell, \mathsf{cf}^{-1}(\mathbf{X}^\ell_{\mathsf{in}}), \mathbf{v}^\ell).$$

Set $\mathbb{x}_{\mathsf{bat}} := ((\bar{\mathbf{r}}, \bar{\mathbf{s}}, \mathbf{s}), [c^{(i)} \in \mathbb{C}, u^{(i)} \in \mathbb{E}]_{i=1}^{k_g})$ where for $i \in [k_g]$,

$$(c^{(i)},\, u^{(i)}) := \sum_{\ell=1}^{\ell_{\mathsf{np}}} \boldsymbol{\beta}_\ell \cdot (c_\ell^{(i)},\, u_\ell^{(i)})\,. \tag{48}$$

Output $\mathbb{x}_o := (\mathbb{x}^*, \mathbb{x}_{\mathsf{bat}})$
6. P : Output $\mathbb{w}_o = (\mathbf{f}^* \in R_q^n, (\mathbf{g}^{(i)} \in R_q^n)_{i=1}^{k_g})$ where

$$\mathbf{f}^* := \sum_{\ell=1}^{\ell_{\mathsf{np}}} \boldsymbol{\beta}_\ell \cdot \mathbf{f}^\ell, \qquad \mathbf{g}^{(i)} := \sum_{\ell=1}^{\ell_{\mathsf{np}}} \boldsymbol{\beta}_\ell \cdot \mathbf{g}^{i,\ell} \tag{49}$$

Figure 4: The protocol Π_{fold} to reduce $(\mathcal{R}^{\mathsf{aux}}_{\mathsf{gr1cs}})^{\ell_{\mathsf{np}}}$ to $\mathcal{R}^{\mathsf{aux_{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin}}$: Part II.

Completeness. By the ϵ-completeness of Π_{gr1cs} (Lemma 4.1) and the perfect completenes of sumcheck batching, with probability at least $1 - \ell_{\mathsf{np}} \cdot \epsilon$, Step 3 passes, and each statement in $(\mathbb{x}_{o,\ell}, \mathbb{w}_{o,\ell})_{\ell=1}^{\ell_{\mathsf{np}}}$ (Step 2) is in $\mathcal{R}^{\mathsf{aux_{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin}}$.
Next, we show that the folded statement is also in $\mathcal{R}^{\mathsf{aux_{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin}}$:

- After linearly-combining the instances and the witnesses using $\boldsymbol{\beta}$, the linear relations under $\mathcal{R}^{\mathsf{aux_{cs}}}_{\mathsf{lin}}$ and $\mathcal{R}_{\mathsf{batchlin}}$ still holds.

- The folded witnesses are valid openings to the folded commitments: Each input witness $\mathbf{f}^\ell$ $(1 \le \ell \le \ell_{\mathsf{np}})$ has ℓ_2-norm $\le B \cdot \sqrt{nd/\ell_h}$; each monomial vector $\mathbf{g}^{i,\ell}$ $(i \in [k_g], \ell \in [\ell_{\mathsf{np}}])$ has ℓ_2-norm $\le \sqrt{n}$. Thus, the norms of the folded witnesses $\mathbf{f}^*$ and $(\mathbf{g}^{(i)})_{i=1}^{k_g}$ are bounded by $\ell_{\mathsf{np}} \cdot \|\mathcal{S}\|_{\mathsf{op}} \cdot B \cdot \sqrt{nd/\ell_h}$ and $\ell_{\mathsf{np}} \cdot \|\mathcal{S}\|_{\mathsf{op}} \cdot \sqrt{n}$, respectively. By Eq. (50), the folded witnesses are valid (strict) openings w.r.t. bound B_{bnd}.

Knowledge soundness. We first describe the extractor. WLOG, assume that the adversary P^* is deterministic. Define folding challenge space $\mathcal{U} := \mathcal{S}^{\ell_{\mathsf{np}}}$. Let $\mathcal{Y}$ be the domain of $(\mathsf{tr}, (\mathbb{x}_o, \mathbb{w}_o))$ where tr is the protocol transcript and $(\mathbb{x}_o, \mathbb{w}_o)$ is the output instance-witness pair. Define predicate $\Psi(\mathbf{u} \in \mathcal{U}, \mathbf{y} := (\mathsf{tr}, (\mathbb{x}_o, \mathbb{w}_o)))$ that outputs 1 if and only if (i) tr is an accepting transcript (with folding challenge $\boldsymbol{\beta} = \mathbf{u}$) and (ii) $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}^{\mathsf{aux_{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin}}$. For folding challenge $\mathbf{u}$ and the extra verifier randomness $\mathbf{r}_v$, define $\mathcal{A}_{\mathbf{r}_v}(\mathbf{u})$ as the adversary that simulates the execution of P^* with verifier randomness $\mathbf{r}_v$ and folding challenge $\mathbf{u}$. Let Ext_{cwss} be the oracle extractor from Lemma 2.3.

Extractor $\mathsf{Ext}^{\mathcal{A}^*}$:

1. Sample folding challenge $\mathbf{u}_0 \leftarrow\!\!\$\ \mathcal{S}^{\ell_{\mathsf{np}}}$ and extra verifier randomness $\mathbf{r}_v$
2. Run extractor $\mathsf{Ext}_{cwss}^{\mathcal{A}_{\mathbf{r}_v}(\mathbf{u}_0)}$. Abort if it fails. Otherwise, let the output be
$$\left(\mathbf{u}_\ell, \mathbf{y}_\ell := (\mathsf{tr}_\ell, \mathbb{x}_o^\ell, \mathbb{w}_o^\ell)\right)_{\ell=0}^{\ell_{\mathsf{np}}},$$
where $\Psi(\mathbf{u}_\ell, \mathbf{y}_\ell) = 1$ for all $\ell \in [0, \ell_{\mathsf{np}}]$, and $\mathbf{u}_\ell \equiv_\ell \mathbf{u}_0$ for all $\ell \in [\ell_{\mathsf{np}}]$
3. For every $\ell \in [\ell_{\mathsf{np}}]$, parse $\mathbb{w}_o^\ell = (\mathbf{f}^{*,\ell}, (\mathbf{g}^{(i,\ell)})_{i=1}^{k_g})$, compute[10]
$$\mathbf{f}^\ell := (\mathbf{f}^{*,\ell} - \mathbf{f}^{*,0})/(\mathbf{u}_\ell[\ell] - \mathbf{u}_0[\ell]) \in R_q^n\,, \tag{51}$$
$$\forall i \in [k_g]\ :\ \mathbf{h}^{i,\ell} := (\mathbf{g}^{(i,\ell)} - \mathbf{g}^{(i,0)})/(\mathbf{u}_\ell[\ell] - \mathbf{u}_0[\ell]) \in R_q^n \tag{52}$$
Parse $\mathbf{f}^\ell$ as $\mathbf{f}^\ell := \mathsf{cf}^{-1}\left([\mathbf{X}^{\ell\top}, \mathbf{W}^{\ell\top}]^\top\right) \in R_q^n$, abort if $\mathbf{X}^\ell \neq \mathbf{X}_{\mathsf{in}}^\ell$
4. Output $\mathbb{w} := \left\{\mathbb{w}_\ell = \mathbf{W}^\ell \in \mathbb{Z}_q^{n_w \times d}\right\}_{\ell=1}^{\ell_{\mathsf{np}}}$

Success probability. Fix verifier randomness $\mathbf{r}_v$ (that excludes folding challenge $\boldsymbol{\beta}$). By Lemma 2.3, Step 2 succeeds with probability $\epsilon_\Psi(\mathcal{A}_{\mathbf{r}_v}) - \ell_{\mathsf{np}}/|\mathcal{S}|$ where
$$\epsilon_\Psi(\mathcal{A}_{\mathbf{r}_v}) := \Pr_{\mathbf{u} \xleftarrow{\mathrm{R}} \mathcal{U}}\left[\Psi(\mathbf{u}, \mathcal{A}_{\mathbf{r}_v}(\mathbf{u})) = 1\right].$$

Let $(c_\ell)_{\ell=1}^{\ell_{\mathsf{np}}}$ denote the input commitments and $(c_\ell^{(i)})_{i\in[k_g],\ell\in[\ell_{\mathsf{np}}]}$ the monomial commitments. The extracted openings $(\mathbf{f}^\ell, (\mathbf{h}^{i,\ell})_{i=1}^{k_g})_{\ell=1}^{\ell_{\mathsf{np}}}$ satisfy:

- For $\ell \in [\ell_{\mathsf{np}}]$, $\mathbf{f}^\ell$ is bound to c_ℓ, with relaxed opening $(\mathbf{f}^{*,\ell} - \mathbf{f}^{*,0}, \mathbf{u}_\ell[\ell] - \mathbf{u}_0[\ell])$ (Eq. (10)).
- For $i \in [k_g], \ell \in [\ell_{\mathsf{np}}]$, $\mathbf{h}^{i,\ell}$ is bound to $c_\ell^{(i)}$, with opening $(\mathbf{g}^{(i,\ell)} - \mathbf{g}^{(i,0)}, \mathbf{u}_\ell[\ell] - \mathbf{u}_0[\ell])$.
- For $\ell \in [\ell_{\mathsf{np}}]$, let $\mathbb{x}_{o,\ell}$ be the reduced instance from Eq. (46), and set $\mathbb{w}_{o,\ell} := (\mathbf{f}^\ell, (\mathbf{h}^{i,\ell})_{i=1}^{k_g})_{\ell=1}^{\ell_{\mathsf{np}}}$. Define $\hat{\mathcal{R}}_{\mathsf{lin}}^{\mathsf{auxcs}}$, $\hat{\mathcal{R}}_{\mathsf{batchlin}}$ as the relaxed relations of $\mathcal{R}_{\mathsf{lin}}^{\mathsf{auxcs}}$, $\mathcal{R}_{\mathsf{batchlin}}$ where the strict opening check $\mathsf{VfyOpen}$ is replaced with the relaxed ones from Eq. (10). Since $\Psi(\mathbf{u}_\ell, \mathbf{y}_\ell) = \Psi(\mathbf{u}_0, \mathbf{y}_0) = 1$, and $\mathbf{u}_\ell \equiv_\ell \mathbf{u}_0$, we get that $(\mathbb{x}_{o,\ell}, \mathbb{w}_{o,\ell}) \in \hat{\mathcal{R}}_{\mathsf{lin}}^{\mathsf{auxcs}} \times \hat{\mathcal{R}}_{\mathsf{batchlin}}$.

Let ϵ_{P^*} be the prover P^*'s success probability. Thus, with probability at least $\epsilon_{\mathsf{P}^*} - (\ell_{\mathsf{np}}/|\mathcal{S}|)$ (over $\mathbf{r}_v$, $\mathbf{u}_0$ and Ext_{cwss}'s randomness), the extractor outputs $(\mathbb{w}_{o,\ell})_{\ell\in[\ell_{\mathsf{np}}]}$ at Step 3 satisfying the relaxed relation $(\hat{\mathcal{R}}_{\mathsf{lin}}^{\mathsf{auxcs}} \times \hat{\mathcal{R}}_{\mathsf{batchlin}})_{\mathsf{np}}^\ell$ w.r.t. $(\mathbb{x}_{o,\ell})_{\ell=1}^{\ell_{\mathsf{np}}}$. By the same soundness argument as Lemma 4.1 (relying on relaxed binding, Eq. (10), instead of strict binding, Eq. (11), of Construction 2.1), and by a union bound, the extractor succeeds with probability
$$\epsilon_{\mathsf{P}^*} - (\ell_{\mathsf{np}}/|\mathcal{S}|) - \ell_{\mathsf{np}} \cdot \mathsf{negl}(\lambda) = \epsilon_{\mathsf{P}^*} - \mathsf{negl}(\lambda)\,,$$
and produces a witness $\mathbb{w} := \left\{\mathbb{w}_\ell = \mathbf{W}^\ell\right\}_{\ell=1}^{\ell_{\mathsf{np}}}$ in $(\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}'})^{\ell_{\mathsf{np}}}$ as required. $\square$

[10] $\mathbf{f}^\ell$ is well-defined as $\mathbf{u}_\ell[\ell] - \mathbf{u}_0[\ell] \in \mathcal{S} - \mathcal{S}$ is invertible over R_q by assumption.

Proposition 4.2. Π_{fold} *(Figure 4) runs two degree-3 sumcheck protocols*[11] *over $\mathbb{K}$, with size n and m respectively. Additional complexity beyond sumchecks is:*

- ***Prover cost*** $T_p^{\mathsf{fold}}(\ell_{\mathsf{np}}, k_g, n, m)$***:***
  - $n\ell_{\mathsf{np}}$ *multiplications between $\mathcal{S}$ and R_q (to compute $\mathbf{f}^*$),*
  - $k_g n\ell_{\mathsf{np}}$ *multiplications between $\mathcal{S}$ and $\mathcal{M} \subseteq R_q$ (to compute $(\mathbf{g}^{(i)})_{i=1}^{k_g}$),*
  - $\ell_{\mathsf{np}} \cdot T_p^{\mathsf{gr1cs}}(k_g, n, m)$*.*
- ***Verifier cost*** $T_v^{\mathsf{fold}}(\ell_{\mathsf{np}}, n_{\mathsf{in}}, k_g, n, m)$***:***
  - $(1 + k_g)\ell_{\mathsf{np}}$ *multiplications between $\mathcal{S}$ and $\mathbb{C} := R_q^\kappa$ (for commitments),*
  - $\ell_{\mathsf{np}} \cdot n_{\mathsf{in}}$ *multiplications between $\mathcal{S}$ and R_q (for public inputs),*
  - $(4 + k_g)t\ell_{\mathsf{np}}$ *multiplications between $\mathcal{S}$ and R_q (for evaluations over $\mathbb{E}$),*
  - $\ell_{\mathsf{np}} \cdot T_v^{\mathsf{gr1cs}}(k_g, n, m)$*.*
- ***Verifier randomness:*** $\Gamma_v^{\mathsf{fold}}(\ell_{\mathsf{np}}, k_g, n, m) = \Gamma_v^{\mathsf{gr1cs}}(k_g, n, m) + \ell_{\mathsf{np}} \log(|\mathcal{S}|)$*.*

$T_p^{\mathsf{gr1cs}}(k_g, n, m)$*,* $T_v^{\mathsf{gr1cs}}(k_g, n, m)$*, and* $\Gamma_v^{\mathsf{gr1cs}}(k_g, n, m)$ *are defined in Proposition 4.1.*

5 The Fiat-Shamir Transform

Π_{fold} can be made non-interactive via the Fiat-Shamir transform. More precisely, we transform an RoK into a non-interactive argument in two steps.

The Commit-and-Open transformation. Let Π_{rok} be a $(2\mathsf{rnd} + 1)$-message public-coin RoK from $\mathcal{R}$ to an output relation $\mathcal{R}_o$, with relaxed input relation $\mathcal{R}'$. Let Π_{cm} be a binding commitment scheme with public parameter $\mathsf{pp}_{\mathsf{cm}}$. (Later in Section 6, Π_{cm} will be the polynomial (or vector) commitment scheme used in the commit-and-prove SNARK.) We denote by $\mathsf{CM}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ a variant of Π_{rok}, where each prover message m_i ($i \in [\mathsf{rnd}]$) is replaced with the commitment $c_i := \Pi_{\mathsf{cm}}.\mathsf{Commit}(\mathsf{pp}_{\mathsf{cm}}, m_i)$, and at the end of the protocol, the prover further sends the opening messages $(m_i)_{i=1}^{\mathsf{rnd}}$. The verifier (i) runs the original RoK verifier w.r.t. $(m_i)_{i=1}^{\mathsf{rnd}}$ and its own verifier challenges, and (ii) checks that $(m_i)_{i=1}^{\mathsf{rnd}}$ are the valid openings to the commitments $(c_i)_{i=1}^{\mathsf{rnd}}$. With a similar argument as in Protostar [BC23], it is clear that $\mathsf{CM}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ is an RoK if Π_{rok} is, where the input, output and relaxed input relations are the same.

The Fiat-Shamir transform. Next, we denote by $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ the Fiat-Shamir transformation of $\mathsf{CM}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$, where verifier challenges are derived via a hash function H modeled as a random oracle. The transcript is initialized with $\mathbb{x}$, and the challenge r_1 is set to $\mathsf{H}(\mathbb{x})$. For each round $i \in [\mathsf{rnd}]$, the challenge r_i and the commitment $c_i = \Pi_{\mathsf{cm}}.\mathsf{Commit}(\mathsf{pp}_{\mathsf{cm}}, m_i)$ are appended to the transcript, where m_i is the prover message. The next challenge r_{i+1} is then derived from the transcript. Note that in practice, instead of taking the entire transcript as an oracle input, we can use the Merkle-Damgård framework to fix the input length of the hash function.

[11]The sumcheck protocol complexity is independent of ℓ_{np} after batching.

We emphasize that the folding proof of $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ contains opening messages $\{m_i\}_{i=1}^{\mathsf{rnd}}$ to enable folding verifications. But later in Section 6, the SNARK proofs in Construction 6.1 do not include $\{m_i\}_{i=1}^{\mathsf{rnd}}$.

Next, we present a variant of Theorem 4.1, which shows the security of $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{fold}}]$.

Theorem 5.1. *Let Π_{cm} be a binding commitment scheme that is further straightline extractable.*[12] *Let H be a hash function modeled as a random oracle. Define aux, $\mathsf{aux}_{\mathsf{cs}}$, aux' and other parameters as in Theorem 4.1. $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ is an RoK from $(\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}})^{\ell_{\mathsf{np}}}$ (Eq. (38)) to $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin}}$ (Eq. (28)), with the relaxed input relation $(\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}'})^{\ell_{\mathsf{np}}}$.*

Proof. The proof is similar to that of Theorem 4.1. Intuitively, the statement holds because (i) the *committed* sumcheck protocols are still sound after the Fiat-Shamir transform above [Can+19; CMS19] (formally proved in Lemma A.2), and (ii) we can replace Lemma 2.3 with a variant in the random oracle model, adapted from Section 8.2 and Figure 13 of [FMN24]. We defer the full proof to Appendix A. □

Remark 5.1. Compared to Π_{fold}, the knowledge error of $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ is increased by a factor of Q – the number of random oracle queries. This is common in known security proofs of the Fiat-Shamir transform. To achieve the same security level, one can enlarge the extension field $\mathbb{K} = \mathbb{F}_{q^t}$ and the folding challenge set $\mathcal{S}$. This has only a minor impact on the efficiency of the batched sumcheck protocol and the random linear combinations of the R_q-witness vectors, which are insignificant compared to the cost of committing to the ℓ_{np} input witnesses.

6 SNARKs in the ROM from High-Arity Folding

In this section, given the Fiat-Shamir transform (Section 5) of the high-arity folding scheme from Section 4, and a Commit-and-Prove-SNARK (Definition 2.8), we construct a succinct proof system in the random oracle model that batch-proves many R1CS statements. Unlike prior folding-based IVC/SNARKs, our approach avoids instantiating random oracles in SNARK circuits and requires only a *single* folding step. The scheme inherits plausible post-quantum security from the underlying SNARKs. When instantiating with a hash-based (or pairing-based) SNARK, the proof size is independent of the number of R1CS statements and poly-logarithmic to the witness size *per statement*.

From (Non-Succinct) RoKs to SNARKs. We start with a modular compiler that transforms a reduction of knowledge into a SNARK. Let $\mathsf{GenR}(1^\lambda) \to (\mathcal{R}, \mathcal{R}', \mathbb{i})$ be a relation generator that outputs relation $\mathcal{R}$, relaxed relation $\mathcal{R}'$, and index $\mathbb{i}$ on input 1^λ. We omit index $\mathbb{i}$ when clear in context.

[12]Straightline extractability means that an extractor can derive the opening simply from the commitment and the adversary transcript (that includes the queries to the idealized oracle).

Let Π_{rok} be a $(2\mathsf{rnd}+1)$-message public-coin RoK from $\mathcal{R}$ to an output relation $\mathcal{R}_o$, with relaxed input relation $\mathcal{R}'$. Denote by $(\mathbb{x}, \mathbb{w})$, $(\mathbb{x}_o, \mathbb{w}_o)$ the input and output instance-witness pairs, respectively. Here $\mathsf{rnd} = \mathsf{poly}(\log(\lambda))$ and

$$|\mathbb{x}_o| = \mathsf{poly}(\lambda, |\mathbb{x}|, \log(|\mathbb{w}|)) \,. \tag{53}$$

Denote the i-th prover message by $m_i \in \mathcal{M}^*_{p,i}$ $(1 \le i \le \mathsf{rnd})$ and the i-th verifier challenge by $r_i \in \mathcal{M}^*_{v,i}$ $(1 \le i \le \mathsf{rnd}+1)$. By the public reducibility of RoKs, there is a deterministic function f such that

$$\mathbb{x}_o := f\left(\mathbb{x}, (m_i)_{i=1}^{\mathsf{rnd}}, (r_i)_{i=1}^{\mathsf{rnd}+1}\right) . \tag{54}$$

Define the CP-SNARK relation $\mathcal{R}_{\mathsf{cp}}$ with input

$$\mathbb{x}_{\mathsf{cp}} := (\mathbb{x}, (r_i)_{i=1}^{\mathsf{rnd}+1}, (c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}, \mathbb{x}_o), \quad \mathbb{w} := (\mathbb{w}_{\mathsf{cp}} := (m_i)_{i=1}^{\mathsf{rnd}}, \mathbb{w}_e) \tag{55}$$

which checks Eq. (54) and that $c_{\mathsf{fs},i} = \Pi_{\mathsf{cm}}.\mathsf{Commit}(\mathsf{pp}_{\mathsf{cm}}, m_i)$ for all $i \in [\mathsf{rnd}]$. The commitment scheme Π_{cm} must be straightline extractable but need not be linearly homomorphic. E.g., Merkle commitments used in hash-based CP-SNARKs and KZG commitments used in pairing-based CP-SNARKs both suffice.

Denote by $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ the Fiat-Shamir transformation of $\mathsf{CM}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ defined in Section 5. Again, we emphasize that the folding proof of $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ contains prover messages $\{m_i\}_{i=1}^{\mathsf{rnd}}$ to enable folding verifications, but the SNARK proofs in Construction 6.1 do not include $\{m_i\}_{i=1}^{\mathsf{rnd}}$.

Construction 6.1. Let H be a hash function modeled as a random oracle. We construct a SNARK Π^* for $\mathcal{R}$ (with relaxed relation $\mathcal{R}'$) using (i) a SNARK Π_{snark} for $\mathcal{R}_o$, and (ii) a CP-SNARK Π_{cp} for $\mathcal{R}_{\mathsf{cp}}$ w.r.t. the commitment Π_{cm}.

Setup: $\mathsf{Setup}(\mathcal{R}) \to (\mathsf{pk}^*, \mathsf{vk}^*)$:

1. $\mathsf{pp}_{\mathsf{cm}} \leftarrow \Pi_{\mathsf{cm}}.\mathsf{Setup}(1^\lambda)$
2. $(\mathsf{pk}_{\mathsf{cp}}, \mathsf{vk}_{\mathsf{cp}}) \leftarrow \Pi_{\mathsf{cp}}.\mathsf{Setup}(\mathsf{pp}_{\mathsf{cm}}, \mathcal{R}_{\mathsf{cp}})$
3. $(\mathsf{pk}, \mathsf{vk}) \leftarrow \Pi_{\mathsf{snark}}.\mathsf{Setup}(\mathcal{R}_o)$
4. Output $\mathsf{pk}^* := (\mathsf{pp}_{\mathsf{cm}}, \mathsf{pk}_{\mathsf{cp}}, \mathsf{pk})$, $\mathsf{vk}^* := (\mathsf{pp}_{\mathsf{cm}}, \mathsf{vk}_{\mathsf{cp}}, \mathsf{vk})$

Prover: $\mathsf{Prove}^{\mathsf{H}}(\mathsf{pk}^*, \mathcal{R}, \mathbb{x}, \mathbb{w}) \to \pi$:

1. Initialize transcript $\mathsf{tr} := \mathbb{x}$
2. For $i \in [\mathsf{rnd}]$:
   - Derive challenge r_i from tr and the hash function H
   - Run Π_{rok} and obtain prover message m_i
   - Append tr with $(r_i, c_{\mathsf{fs},i} := \Pi_{\mathsf{cm}}.\mathsf{Commit}(\mathsf{pp}_{\mathsf{cm}}, m_i))$
3. Derive $r_{\mathsf{rnd}+1}$ and run Π_{rok} to obtain $(\mathbb{x}_o, \mathbb{w}_o)$
4. Compute $\mathbb{w}_e$ such that $(\mathbb{x}_{\mathsf{cp}}, (\mathbb{w}_{\mathsf{cp}}, \mathbb{w}_e)) \in \mathcal{R}_{\mathsf{cp}}$ with $(\mathbb{x}_{\mathsf{cp}}, \mathbb{w}_{\mathsf{cp}})$ defined in Eq. (55)
5. $\pi_{\mathsf{cp}} \leftarrow \Pi_{\mathsf{cp}}.\mathsf{Prove}(\mathsf{pk}_{\mathsf{cp}}, \mathcal{R}_{\mathsf{cp}}, \mathbb{x}_{\mathsf{cp}}, \mathbb{w}_{\mathsf{cp}}, \mathbb{w}_e)$
6. $\pi \leftarrow \Pi_{\mathsf{snark}}.\mathsf{Prove}(\mathsf{pk}, \mathcal{R}_o, \mathbb{x}_o, \mathbb{w}_o)$

7. Output $\pi^* := (\pi_{\mathsf{cp}}, \pi, (c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}, \mathbb{x}_o)$

Verifier: $\mathsf{Vf}^{\mathsf{H}}(\mathsf{vk}^*, \mathcal{R}, \mathbb{x}, \pi^*) \to b$:

1. Parse $\pi^* = (\pi_{\mathsf{cp}}, \pi, (c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}, \mathbb{x}_o)$
2. Recompute $(r_i)_{i=1}^{\mathsf{rnd}+1}$ from $\mathbb{x}, (c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}$ and H
3. Derive $\mathbb{x}_{\mathsf{cp}} = (\mathbb{x}, (r_i)_{i=1}^{\mathsf{rnd}+1}, (c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}, \mathbb{x}_o)$ as in Eq. (55)
4. Output $\Pi_{\mathsf{cp}}.\mathsf{Vf}(\mathsf{vk}_{\mathsf{cp}}, \mathcal{R}_{\mathsf{cp}}, \mathbb{x}_{\mathsf{cp}}, \pi_{\mathsf{cp}}) \ \wedge \ \Pi_{\mathsf{snark}}.\mathsf{Vf}(\mathsf{vk}, \mathcal{R}_o, \mathbb{x}_o, \pi)$

Remark 6.1 (Instance Compression). If $\mathbb{x}$ is large, verifying π_{cp} against $\mathbb{x}$ might be costly. To improve efficiency, we replace $\mathbb{x}$ with a vector commitment $c_{\mathsf{fs},0} := \Pi_{\mathsf{cm}}.\mathsf{Commit}(\mathsf{pp}_{\mathsf{cm}}, \mathbb{x})$. The CP-SNARK, when proving Eq. (54), treats $c_{\mathsf{fs},0}$ as the instance and treats $\mathbb{x}$ as a witness. When verifying the CP-SNARK proof, the verifier replaces $\mathbb{x}$ (in $\mathbb{x}_{\mathsf{cp}}$) with $c_{\mathsf{fs},0}$. $c_{\mathsf{fs},0}$ is only 32 bytes with Merkle commitments and 48 bytes with KZG commitments. In Π_{fold} where $\mathbb{x}$ contains commitments $(c_\ell)_{\ell=1}^{\ell_{\mathsf{np}}}$ and public inputs $(\mathbf{X}_{\mathsf{in}}^{\ell})_\ell$, the verifier also checks the consistency between $(\mathbf{X}_{\mathsf{in}}^{\ell})_\ell$ and $c_{\mathsf{fs},0}$ *outside* the circuit.

Remark 6.2 (Optimizing proof sizes). To achieve smaller proof sizes, one option is to generate a single CP-SNARK proof that proves both $(\mathbb{x}_{\mathsf{cp}}, (\mathbb{w}_{\mathsf{cp}}, \mathbb{w}_e)) \in \mathcal{R}_{\mathsf{cp}}$ and $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_o$, thus removing the need of an extra SNARK proof π.

Theorem 6.1. *If the RoK Π_{rok} satisfies that $\sum_{i=1}^{\mathsf{rnd}+1} |r_i| = \mathsf{poly}(\lambda, |\mathbb{x}|, \log(|\mathbb{w}|))$ where r_i's are the verifier challenges, then Construction 6.1 is a SNARK w.r.t. the relation generator* GenR *(Definition 2.7).*

Proof. We show the succinctness, completeness, and knowledge soundness of Π^*.

Succinctness. The proof is $\pi^* := (\pi_{\mathsf{cp}}, \pi, (c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}, \mathbb{x}_o)$. We first scrutinize the proof size: (i) π_{cp}, π are succinct by the succinctness of the SNARKs Π_{cp}, Π_{snark}. (ii) The set of commitments $(c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}$ is succinct since rnd is polylogarithmic and the commitments are succinct. (iii) $\mathbb{x}_o$ is succinct by assumption (Eq. (53)).

Next, we check verifier complexity: (i) computing $(r_i)_{i=1}^{\mathsf{rnd}+1}$ takes a transcript of sublinear size since commitments are succinct and $\sum_{i=1}^{\mathsf{rnd}+1} |r_i|$ is sublinear by assumption. (ii) Verifying π_{cp}, π takes sublinear time by the succinctness of Π_{cp}, Π_{snark}.

Completeness. By the SNARK completeness of Π_{snark} and the completeness of Π_{rok}, with overwhelming probability, we have $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_o$ and $\Pi_{\mathsf{snark}}.\mathsf{Vf}(\mathsf{vk}, \mathcal{R}_o, \mathbb{x}_o, \pi) = 1$. Likewise, by the SNARK completeness of Π_{cp} and Eq. (54), $(\mathbb{x}_{\mathsf{cp}}, (\mathbb{w}_{\mathsf{cp}}, \mathbb{w}_e)) \in \mathcal{R}_{\mathsf{cp}}$ and $\Pi_{\mathsf{cp}}.\mathsf{Vf}(\mathsf{vk}_{\mathsf{cp}}, \mathcal{R}_{\mathsf{cp}}, \mathbb{x}_{\mathsf{cp}}, \pi_{\mathsf{cp}}) = 1$. Thus, the verifier accepts with overwhelming probability if the prover is honest.

Knowledge soundness. Let P^* be a malicious prover for Π^*. We first construct an adversary $\mathcal{A}$ against the RoK $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ (where $\mathcal{A}$ depends on P^*):

1. Run the extractor of Π_{cp} to obtain witness $(\mathbb{w}_{\mathsf{cp}}, \mathbb{w}_e)$. Let $\mathbb{x}_{\mathsf{cp}} := (\mathbb{x}, (r_i)_{i=1}^{\mathsf{rnd}+1}, (c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}, \mathbb{x}_o)$ be the CP-SNARK instance, where $\mathbb{x}$ is the SNARK instance, $(c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}, \mathbb{x}_o$ are parts of the SNARK proof π^*, all output by P^*. The challenges $(r_i)_{i=1}^{\mathsf{rnd}+1}$ are derived from $\mathbb{x}, (c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}$ and the hash function H. Abort if $(\mathbb{x}_{\mathsf{cp}}, (\mathbb{w}_{\mathsf{cp}}, \mathbb{w}_e)) \notin \mathcal{R}_{\mathsf{cp}}$

2. Run the extractor of Π_{snark} to obtain witness $\mathbb{w}_o$ w.r.t. the instance $\mathbb{x}_o$ underlying $\mathbb{x}_{\mathsf{cp}}$. Abort if $(\mathbb{x}_o, \mathbb{w}_o) \notin \mathcal{R}_o$
3. Output instance $\mathbb{x}_o$, witness $\mathbb{w}_o$, the round commitments $(c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}$, and the opening prover messages $(m_i)_{i=1}^{\mathsf{rnd}}$ underlying $\mathbb{w}_{\mathsf{cp}}$

Let ϵ_{P^*} be the success probability of P^*. We show that $\mathcal{A}$ is an expected polynomial time adversary for $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ with success probability $\epsilon_{\mathsf{P}^*} - \mathsf{negl}(\lambda)$: Step 1 and 2 succeed with probability $\epsilon_{\mathsf{P}^*} - \mathsf{negl}(\lambda)$ by knowledge soundness of Π_{cp} and Π_{snark}. Therefore,

$$((\mathbb{x}_{\mathsf{cp}}, (\mathbb{w}_{\mathsf{cp}}, \mathbb{w}_e)) \in \mathcal{R}_{\mathsf{cp}}) \quad \wedge \quad ((\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_o) \,,$$

which implies that, with probability $\epsilon_{\mathcal{A}} := \epsilon_{\mathsf{P}^*} - \mathsf{negl}(\lambda)$, the folding proof $[(c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}, (m_i)_{i=1}^{\mathsf{rnd}}]$ (for $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$) is valid w.r.t. the input instance $\mathbb{x}$ and output instance $\mathbb{x}_o$, and $\mathbb{w}_o$ is a valid witness for $\mathbb{x}_o$ w.r.t. $\mathcal{R}_o$.

Let $\mathsf{Ext}_{\mathsf{rok}}^{\mathcal{A}}$ be the RoK extractor for $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ w.r.t. adversary $\mathcal{A}$. The SNARK extractor for Π^* simply runs $\mathsf{Ext}_{\mathsf{rok}}^{\mathcal{A}}$. By knowledge soundness of $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$, with probability $\epsilon_{\mathcal{A}} - \mathsf{negl}(\lambda)$, it outputs a valid witness $\mathbb{w}$ for $\mathbb{x}$ w.r.t. the relaxed relation $\mathcal{R}'$. □

Scalable SNARKs from High-Arity Folding. By plugging the RoK Π_{fold} (Figure 4) and its Fiat-Shamir transform $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{fold}}]$ (Section 5) into Construction 6.1, and instantiating Π_{cp} and Π_{snark} with SNARKs in the random oracle model (or algebraic group model if we use pairing-based SNARKs), we obtain the following corollary.

Corollary 6.2. *Let* aux, aux' *be the R1CS parameters defined in Theorem 4.1. For* $\ell_{\mathsf{np}} = \mathsf{poly}(\lambda)$ *that satisfies Eq. (50) in Theorem 4.1, there exists a SNARK (in the random oracle model) for relation* $(\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}})^{\ell_{\mathsf{np}}}$ *with relaxed relation* $(\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}'})^{\ell_{\mathsf{np}}}$.

Proposition 6.1. *Let* $\mathbb{C}_{\mathsf{FS}}$ *and* $\mathbb{C}_{\mathsf{fold}}$ *be the commitment domains used in the Fiat-Shamir heuristic and folding. The SNARK from Corollary 6.2 achieves:*

Prover time:
- *Folding cost* $T_p^{\mathsf{fold}}(\ell_{\mathsf{np}}, k_g, n, m)$ *from Proposition 4.2.*
- *SNARK proving cost for* Π_{cp} *and* Π_{snark}.
- *Commitment cost (over* $\mathbb{C}_{\mathsf{FS}}$*) for prover messages* $m_1, \ldots, m_{\mathsf{rnd}}$.

Verifier time:
- *SNARK verification for* Π_{cp} *and* Π_{snark}.
- *Fiat-Shamir transformation cost with transcript being* $\log(n) + O(1)$ $\mathbb{C}_{\mathsf{FS}}$*-elements and randomness* $\Gamma_v^{\mathsf{fold}}(\ell_{\mathsf{np}}, k_g, n, m)$ *from Proposition 4.2.*

Proof size: $\log(n) + O(1)$ $\mathbb{C}_{\mathsf{FS}}$*-elements,* $1 + k_g$ $\mathbb{C}_{\mathsf{fold}}$*-elements,* $4 + k_g$ $\mathbb{E}$*-elements,* n_{in} R_q*-elements, and two SNARK proofs* π_{cp}, π_{snark}.

7 Instantiation

Table 1 presents a candidate instantiation for the folding-based SNARKs in Corollary 6.2, with MSIS parameters set according to the lattice estimator [APS15] at 117-bit security.

Variable	Description	Instantiation
q	prime modulus	64-bit prime
d	ring dimension & R1CS batching factor	64
κ	MSIS rank	12
β_{SIS}	ℓ_2-norm bound for MSIS hardness	2^{37}
$\mathbb{K}$	extension field for sumcheck	$\mathbb{F}_{q^2}$
ℓ_{np}	folding arity	2^{10}
ℓ_h	projection input length	2^{14}
λ_{pj}	projection output length	2^8
$\bar{n}$	R1CS witness length per instance	2^{16}
m	number of R1CS constraints per instance	2^{16}
k_{cs}	decomposition factor for R1CS witnesses	16
b	decomposition base for R1CS witnesses	2^4
n	generalized R1CS witness length	$\bar{n}k_{\mathsf{cs}} = 2^{20}$
$\mathcal{S}$	folding challenge set	as in [BS23]
B_{rbnd}	norm bound for relaxed openings	$\beta_{\mathsf{SIS}}/(4\|\mathcal{S}\|_{\mathsf{op}}) \approx 2^{31}$
B_{bnd}	norm bound for strict openings s.t. Eq. (50) holds	$B_{\mathsf{rbnd}}/2 = 2^{30}$
k_g	number of monomial vectors in the range proof	3
B_{d,k_g}	ℓ_∞-norm bound for range proof	121117 (Eq. (29))
B	input witness norm bound, $0.5b\sqrt{\ell_h} \leq B \leq B_{d,k_g}/9.5$	2^{10}
B'	relaxed input norm bound, $16B_{d,k_g}/\sqrt{30} \leq B' \leq \beta_{\mathsf{SIS}}/(\sqrt{nd/\ell_h})$	353806

Table 1: Parameters and candidate instantiations

The sizes of the extension field $\mathbb{K}$ and the folding challenge set $\mathcal{S}$ are set to be larger than[13] 2^{128}. We leave the concrete implementation as future work, but we provide rough estimate for prover/verifier complexity and proof size to highlight its efficiency.

Prover complexity. The commitment opening relation is not embedded in the CP-SNARK circuit by definition. If we instantiate the SNARK Π_{snark} with a lattice-based scheme (e.g. [BS23; NS24]), the corresponding SNARK statement does not embed the lattice commitment opening relation either.

By Proposition 4.2, the CP-SNARK circuit is dominated by the folding verifier cost $T_v^{\mathsf{fold}}(\ell_{\mathsf{np}}, n_{\mathsf{in}}, k_g, n, m)$. Under Table 1 parameters, this requires about $2^{16} \sim 2^{17}$ multiplications between $\mathcal{S}$ and R_q.

The SNARK circuit for Π_{snark}, defined by $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin}}$, is dominated by k_g inner products between $\mathcal{M}^n$ and $\mathbb{K}^n$ (Eq. (28)), 3 inner products between R_q^m and $\mathbb{K}^m$, and 1 inner product between $R_q^{m_J}$ and $\mathbb{K}^{m_J}$, where $m_J := n\lambda_{\mathsf{pj}}/\ell_h$. With Table 1 parameters, this is about 2^{25} constraints over field $\mathbb{Z}_q$. Effectively, they are just 7 polynomial evaluation proofs, whose structure might enable more direct and efficient

[13]We need to scale the set sizes by a factor of Q (the number of random oracle queries) to match the security bounds achieved by non-interactive folding schemes.

constructions.

By Proposition 6.1, the proving cost is dominated by the high-arity folding scheme, whose cost is dominated by the committing cost for input witnesses. Under Table 1, this is $\kappa n \ell_{\mathsf{np}} = 3 \cdot 2^{32}$ multiplications between arbitrary R_q-elements and bounded elements (with ℓ_∞-norm at most $0.5b = 8$).

Overall, with the proving cost above, we can batchly prove $\ell_{\mathsf{np}} \cdot d = 2^{16}$ R1CS statements over $\mathbb{Z}_q$, each with 2^{16} constraints. This is about 2^{32} total R1CS constraints.

Remark 7.1. When the original R1CS witness already has a low-norm (e.g., in large language models), the decomposition factor k_{cs} can be smaller, leading to further speedup. For example, if the witness consists of 8-bit signed integers, we obtain an extra 8x speedup.

Proof size. For simplicity, assume the public input length $n_{\mathsf{in}} = 0$. By Proposition 6.1, with Table 1 parameters, the proof size is dominated by two SNARK proofs, 4 elements over R_q^κ, and 7 elements over $\mathbb{E}$. The most succinct post-quantum SNARKs (e.g. WHIR [ACFY25], LaBRADOR [BS23]) have proof sizes in range 50-100KB. Thus, we estimate the proof size to be below 200KB. When post-quantum security is not needed, we can use pairing-based SNARKs such as Hyperplonk+KZG [CBBZ23], which leads to a proof size below 50KB.

Verifier complexity. By Proposition 6.1, the verifier is dominated by the two SNARK verifications and the Fiat-Shamir transform. With Table 1 parameters, the Fiat-Shamir randomness is dominated by the random projection matrix that requires no more than $2 \cdot \ell_h \cdot \lambda_{\mathsf{pj}} = 2^{23}$ random bits ($\approx$ 1MB). Generating this randomness with a standardized hash function such as SHA-3 typically takes less than a millisecond.

8 Two-layer Folding by Splitting Linear Statements

In this section, under a slightly stronger MSIS assumption, we reduce a generic linear statement of witness size $n = n' \cdot \ell$ into ℓ generic linear statements, each with witness size n'. Here both n' and ℓ are powers-of-two. As shown in Section 1.2, this reduction allows us to further split Π_{fold}'s (Figure 4) output statement (that lies in the linear relation $\mathcal{R}_o := \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin}}$) into multiple uniform linear statements, so that we can apply the high-arity folding scheme again.

Let $\mathbb{i} := (\mathbf{A}, R_q, \mathbb{K}, \mathbb{E})$ be the index defined in Eq. (21). We further assume that the MSIS matrix $\mathbf{A} \in R_q^{\kappa \times n}$ has the structure

$$\mathbf{A} = [r_1 \cdot \mathbf{A}', \ldots, r_\ell \cdot \mathbf{A}'] \tag{56}$$

for some random $r_1, \ldots, r_\ell \leftarrow\!\!\$\ R_q$ and $\mathbf{A}' \leftarrow\!\!\$\ R_q^{\kappa \times n'}$.

Consider parameter $\mathsf{aux} := (n_{\mathsf{in}} \in \mathbb{N}, (\mathbf{M}_i \in \mathbb{Z}_q^{m_i \times n})_{i=1}^{k_x})$ and a generic linear relation $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$ defined in Eq. (22). Our goal is to reduce a statement in $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$ into multiple statements in a simpler linear relation. For simplicity, we assume $n_{\mathsf{in}} = 0$, $k_x = 1$ and denote $\mathbf{M} := \mathbf{M}_1$. Our approach naturally extends to the general case.

WLOG, assume that $\mathbf{M} \in \mathbb{Z}_q^{(m'\ell)\times(n'\ell)}$ is of the form $\mathbf{M} = \mathbf{I}_\ell \otimes \mathbf{M}'$ for some $\mathbf{M}' \in \mathbb{Z}_q^{m'\times n'}$, where m' is a power-of-two. (Looking ahead, in our two-layer folding-based SNARK, $\mathbf{M}'$ will correspond to a R1CS (or projection) matrix for the 2nd-layer witnesses, which can no longer be split.) We can write $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$ as

$$\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = (c \in \mathbb{C},\ (\mathbf{r} \in \mathbb{K}^{\log \ell}, \mathbf{s} \in \mathbb{K}^{\log m'}),\ v \in \mathbb{E}) \\ \mathbb{w} = \mathbf{f} := (\mathbf{f}_1, \ldots, \mathbf{f}_\ell) \in R_q^{n'\ell} \quad \text{s.t.} \\ \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1 \ \wedge \ \langle \mathsf{ts}(\mathbf{r}||\mathbf{s}),\, \mathbf{M}\mathbf{f} \rangle = v \end{array} \right\}. \tag{57}$$

Define relation

$$\mathcal{R}' := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = (c \in \mathbb{C},\ \mathbf{s} \in \mathbb{K}^{\log m'},\ v \in \mathbb{E}) \\ \mathbb{w} = \mathbf{f} \in R_q^{n'} \quad \text{s.t.} \\ \mathsf{VfyOpen}(\mathbf{A}', c, \mathbf{f}) = 1 \ \wedge \ \langle \mathsf{ts}(\mathbf{s}),\, \mathbf{M}'\mathbf{f} \rangle = v \end{array} \right\}. \tag{58}$$

A statement $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$ can be reduced to ℓ statements in $\mathcal{R}'$ via the following reduction of knowledge:

1. $\mathsf{P} \to \mathsf{V}$: For all $i \in [\ell]$, send $c_i := \mathbf{A}'\mathbf{f}_i$ and $v_i := \langle \mathsf{ts}(\mathbf{s}), \mathbf{M}'\mathbf{f}_i \rangle$
2. V : Check that $c = \sum_{i=1}^{\ell} r_i \cdot c_i$ and $\langle \mathsf{ts}(\mathbf{r}), (v_1, \ldots, v_\ell) \rangle = v$. Abort if it fails
3. V : For all $i \in [\ell]$, output $\mathbb{x}_i = (c_i, \mathbf{s}, v_i)$
4. P : For all $i \in [\ell]$, output $\mathbb{w}_i = \mathbf{f}_i$

Two-layer folding. Using the splitting RoK above, we describe a two-layer folding-based SNARK that batch-proves $\ell_{\mathsf{np}} \cdot \ell$ statements in the generalized R1CS relation (Eq. (38)).

- Each individual statement has witness length n' and R1CS matrices $(\mathbf{M}_i' \in \mathbb{Z}_q^{m'\times n'})_{i=1}^{3}$. In the 1st layer, every ℓ consecutive witness vectors $(\mathbf{f}_1, \ldots, \mathbf{f}_\ell)$ is packed into a single witness vector. The MSIS matrix $\mathbf{A}$ is defined as in Eq. (56) and for $i \in [3]$, the R1CS matrix $\mathbf{M}_i$ is set to $\mathbf{M}_i := \mathbf{I}_\ell \otimes \mathbf{M}_i'$.
- Let $\mathcal{R}_o := \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux_{cs}}} \times \mathcal{R}_{\mathsf{batchlin}}$, and define $\mathcal{R}_o'$ as identical to $\mathcal{R}_o$ except that (i) each witness has length n' instead of $n' \cdot \ell$, (ii) the MSIS matrix becomes $\mathbf{A}'$, and (iii) $(\mathbf{M}_i = \mathbf{I}_\ell \otimes \mathbf{M}_i')_i$ are replaced with $(\mathbf{M}_i')_i$. After finishing the 1st-layer folding, we split the folded statement $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_o$ into ℓ linear statements in $\mathcal{R}_o'$ using the splitting RoK above.
- The witnesses of the ℓ linear statements may have relatively large norms after the 1st-layer folding. Directly folding them in the 2nd layer would require larger moduli or higher lattice dimensions. To address this, we use the decomposition RoK from [BC24; BC25] to reduce the ℓ statements into $\ell \cdot k_b$ statements with

much lower norms. In practice, $k_b \leq 4$ suffices. Now, each of the $\ell \cdot k_b$ statements lies in $\mathcal{R}'$, which is almost identical to $\mathcal{R}'_o$ except that the commitment opening check $\mathsf{VfyOpen}(\cdot)$ is replaced with a more fine-grained check $\mathsf{VfyOpen}_{\ell_h, B'}(\cdot)$ for a lower norm bound B'.

- Given the $\ell \cdot k_b$ linear statements in $\mathcal{R}'$, we use the high-arity folding scheme and the commit-and-prove compiler again to compress them into a CP-SNARK proof and a SNARK proof. Since the input statements are already linear, the high-arity folding scheme does not need to run the Hadamard protocol (Figure 1). Moreover, in addition to the folding-verifier logic, the CP-SNARK relation also ensures that the verifier checks in both the splitting RoK and the decomposition RoK pass. With the commit-and-prove technique, the CP-SNARK circuits only checks linear combinations, not the Fiat-Shamir heuristic or commitment opening relations.

Compared to the single-layer folding-based SNARK, the CP-SNARK statements in the two-layer scheme only checks $O(\ell_{\mathsf{np}} + \ell)$ R_q-multiplications, rather than $O(\ell_{\mathsf{np}} \cdot \ell)$ multiplications as in the single-layer scheme. The norm blowup is also significantly smaller, enabling smaller moduli and lower lattice dimensions. As a tradeoff, the two-layer scheme relies on a slightly stronger assumption where the MSIS assumption holds even if the matrix $\mathbf{A}$ has the form of Eq. (56). Furthermore, although most 1st-layer sub-protocols remain memory-efficient and streaming-friendly, the sumchecks become ℓ-times larger. Naively running them would incur either high computational cost or high memory cost. Fortunately, again, we can use the technique from Section 4 of [Baw+25] to run the sumcheck prover in time $O(n'\ell \log\log(n'\ell))$, space $O((n'\ell)^{1/k})$ with $O(k + \log\log(n'\ell))$ passes over the input data.

Acknowledgments. The author acknowledges the use of generative AI tools for grammar revision. No scientific material in this paper was originated from AI. We thank Prof. Dan Boneh for the valuable feedback, and thank Dr. Srinath Setty for discussions about memory-efficient and streaming-friendly sumcheck protocols. This work was funded by NSF, DARPA, the Simons Foundation, and UBRI. Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of DARPA.

References

[ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. “WHIR: Reed-Solomon Proximity Testing with Super-Fast Verification”. In: *EUROCRYPT 2025, Part IV*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15604. LNCS. Springer, Cham, May 2025, pp. 214–243. DOI: 10.1007/978-3-031-91134-7_8.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. “Ligero: Lightweight Sublinear Arguments Without a Trusted Setup”. In: *ACM CCS 2017*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, 2017, pp. 2087–2104. DOI: 10.1145/3133956.3134104.

[Ajt96] Miklós Ajtai. “Generating Hard Instances of Lattice Problems (Extended Abstract)”. In: *28th ACM STOC*. ACM Press, May 1996, pp. 99–108. DOI: 10.1145/237814.237838.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. *On The Concrete Hardness Of Learning With Errors*. Cryptology ePrint Archive, Report 2015/046. 2015. URL: https://eprint.iacr.org/2015/046.

[Baw+24] Anubhav Baweja, Pratyush Mishra, Tushar Mopuri, Karan Newatia, and Steve Wang. *Scribe: Low-memory SNARKs via Read-Write Streaming*. Cryptology ePrint Archive, Report 2024/1970. 2024. URL: https://eprint.iacr.org/2024/1970.

[Baw+25] Anubhav Baweja, Alessandro Chiesa, Elisabetta Fedele, Giacomo Fenzi, Pratyush Mishra, Tushar Mopuri, and Andrew Zitek-Estrada. *Time-Space Trade-Offs for Sumcheck*. Cryptology ePrint Archive, Paper 2025/1473. 2025. URL: https://eprint.iacr.org/2025/1473.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. “Verifiable Delay Functions”. In: *CRYPTO 2018, Part I*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. LNCS. Springer, Cham, Aug. 2018, pp. 757–788. DOI: 10.1007/978-3-319-96884-1_25.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. *Scalable, transparent, and post-quantum secure computational integrity*. Cryptology ePrint Archive, Report 2018/046. 2018. URL: https://eprint.iacr.org/2018/046.

[BC23] Benedikt Bünz and Binyi Chen. “Protostar: Generic Efficient Accumulation/Folding for Special-Sound Protocols”. In: *ASIACRYPT 2023, Part II*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14439. LNCS. Springer, Singapore, Dec. 2023, pp. 77–110. DOI: 10.1007/978-981-99-8724-5_3.

[BC24] Dan Boneh and Binyi Chen. *LatticeFold: A Lattice-based Folding Scheme and its Applications to Succinct Proof Systems*. Cryptology ePrint Archive, Report 2024/257. 2024. URL: https://eprint.iacr.org/2024/257.

[BC25] Dan Boneh and Binyi Chen. “LatticeFold+: faster, simpler, shorter lattice-based folding for succinct proof systems”. In: *Annual International Cryptology Conference (CRYPTO)*. Springer. 2025, pp. 327–361.

[BCFW25] Benedikt Bünz, Alessandro Chiesa, Giacomo Fenzi, and William Wang. *Linear-Time Accumulation Schemes*. Cryptology ePrint Archive, Paper 2025/753. 2025. URL: https://eprint.iacr.org/2025/753.

[BCG24] Annalisa Barbara, Alessandro Chiesa, and Ziyi Guan. *Relativized Succinct Arguments in the ROM Do Not Exist.* Cryptology ePrint Archive, Report 2024/728. 2024. URL: https://eprint.iacr.org/2024/728.

[BCHO22] Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. "Gemini: Elastic SNARKs for Diverse Environments". In: *EUROCRYPT 2022, Part II.* Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13276. LNCS. Springer, Cham, 2022, pp. 427–457. DOI: 10.1007/978-3-031-07085-3_15.

[BCMS20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. "Recursive Proof Composition from Accumulation Schemes". In: *TCC 2020, Part II.* Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12551. LNCS. Springer, Cham, Nov. 2020, pp. 1–18. DOI: 10.1007/978-3-030-64378-2_1.

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. "Scalable Zero Knowledge via Cycles of Elliptic Curves". In: *CRYPTO 2014, Part II.* Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. LNCS. Springer, Berlin, Heidelberg, Aug. 2014, pp. 276–294. DOI: 10.1007/978-3-662-44381-1_16.

[BDFG21] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. "Halo Infinite: Proof-Carrying Data from Additive Polynomial Commitments". In: *CRYPTO 2021, Part I.* Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 649–680. DOI: 10.1007/978-3-030-84242-0_23.

[Ben+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. "Aurora: Transparent Succinct Arguments for R1CS". In: *EUROCRYPT 2019, Part I.* Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11476. LNCS. Springer, Cham, May 2019, pp. 103–128. DOI: 10.1007/978-3-030-17653-2_4.

[BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. *Halo: Recursive Proof Composition without a Trusted Setup.* Cryptology ePrint Archive, Report 2019/1021. 2019. URL: https://eprint.iacr.org/2019/1021.

[BMNW25a] Benedikt Bünz, Pratyush Mishra, Wilson Nguyen, and William Wang. "Accumulation Without Homomorphism". In: *ITCS 2025.* Ed. by Raghu Meka. Vol. 325. LIPIcs, Jan. 2025, 23:1–23:25. DOI: 10.4230/LIPIcs.ITCS.2025.23.

[BMNW25b] Benedikt Bünz, Pratyush Mishra, Wilson Nguyen, and William Wang. "Arc: Accumulation for reed–solomon codes". In: *Annual International Cryptology Conference (CRYPTO).* Springer. 2025, pp. 128–160.

[Bre+25] Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D. Rothblum, and Hadas Zeilberger. "Blaze: Fast SNARKs from Interleaved RAA Codes". In: *EUROCRYPT 2025, Part IV*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15604. LNCS. Springer, Cham, May 2025, pp. 123–152. DOI: 10.1007/978-3-031-91134-7_5.

[BS23] Ward Beullens and Gregor Seiler. "LaBRADOR: Compact Proofs for R1CS from Module-SIS". In: *CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. LNCS. Springer, Cham, Aug. 2023, pp. 518–548. DOI: 10.1007/978-3-031-38554-4_17.

[Bün+21] Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. "Proof-Carrying Data Without Succinct Arguments". In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 681–710. DOI: 10.1007/978-3-030-84242-0_24.

[Can+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N Rothblum, Ron D Rothblum, and Daniel Wichs. "Fiat-Shamir: from practice to theory". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 1082–1090.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. "HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates". In: *EUROCRYPT 2023, Part II*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14005. LNCS. Springer, Cham, Apr. 2023, pp. 499–530. DOI: 10.1007/978-3-031-30617-4_17.

[CCS22] Megan Chen, Alessandro Chiesa, and Nicholas Spooner. "On Succinct Non-interactive Arguments in Relativized Worlds". In: *EUROCRYPT 2022, Part II*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13276. LNCS. Springer, Cham, 2022, pp. 336–366. DOI: 10.1007/978-3-031-07085-3_12.

[CFQ19] Matteo Campanelli, Dario Fiore, and Anaïs Querol. "LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs". In: *ACM CCS 2019*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM Press, Nov. 2019, pp. 2075–2092. DOI: 10.1145/3319535.3339820.

[CGSY24] Alessandro Chiesa, Ziyi Guan, Shahar Samocha, and Eylon Yogev. "Security Bounds for Proof-Carrying Data from Straightline Extractors". In: *TCC 2024, Part II*. Ed. by Elette Boyle and Mohammad Mahmoody. Vol. 15365. LNCS. Springer, Cham, Dec. 2024, pp. 464–496. DOI: 10.1007/978-3-031-78017-2_16.

[Che+23] Megan Chen, Alessandro Chiesa, Tom Gur, Jack O'Connor, and Nicholas Spooner. "Proof-Carrying Data from Arithmetized Random Oracles". In: *EUROCRYPT 2023, Part II*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14005. LNCS. Springer, Cham, Apr. 2023, pp. 379–404. DOI: 10.1007/978-3-031-30617-4_13.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. "Universally composable two-party and multi-party secure computation". In: *34th ACM STOC*. ACM Press, May 2002, pp. 494–503. DOI: 10.1145/509907.509980.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. "Succinct Arguments in the Quantum Random Oracle Model". In: *TCC 2019, Part II*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11892. LNCS. Springer, Cham, Dec. 2019, pp. 1–29. DOI: 10.1007/978-3-030-36033-7_1.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. "Fractal: Post-quantum and Transparent Recursive Proofs from Holography". In: *EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Cham, May 2020, pp. 769–793. DOI: 10.1007/978-3-030-45721-1_27.

[CWSK24] Bing-Jyue Chen, Suppakit Waiwitlikhit, Ion Stoica, and Daniel Kang. "ZKML: An Optimizing System for ML Inference in Zero-Knowledge Proofs". In: *Proceedings of the Nineteenth European Conference on Computer Systems, EuroSys 2024, Athens, Greece, April 22-25, 2024*. ACM, 2024, pp. 560–574. DOI: 10.1145/3627703.3650088. URL: https://doi.org/10.1145/3627703.3650088.

[DB22] Trisha Datta and Dan Boneh. *Using ZK Proofs to Fight Disinformation*. link. 2022.

[DCB25] Trisha Datta, Binyi Chen, and Dan Boneh. "VerITAS: Verifying Image Transformations at Scale". In: *2025 IEEE Symposium on Security and Privacy*. Ed. by Marina Blanton, William Enck, and Cristina Nita-Rotaru. IEEE Computer Society Press, May 2025, pp. 4606–4623. DOI: 10.1109/SP61157.2025.00097.

[DP24] Benjamin E. Diamond and Jim Posen. *Polylogarithmic Proofs for Multilinears over Binary Towers*. Cryptology ePrint Archive, Report 2024/504. 2024. URL: https://eprint.iacr.org/2024/504.

[FKNP24] Giacomo Fenzi, Christian Knabenhans, Ngoc Khanh Nguyen, and Duc Tu Pham. "Lova: Lattice-Based Folding Scheme from Unstructured Lattices". In: *ASIACRYPT 2024, Part IV*. Ed. by Kai-Min Chung and Yu Sasaki. Vol. 15487. LNCS. Springer, Singapore, Dec. 2024, pp. 303–326. DOI: 10.1007/978-981-96-0894-2_10.

[FMN24] Giacomo Fenzi, Hossein Moghaddas, and Ngoc Khanh Nguyen. “Lattice-Based Polynomial Commitments: Towards Asymptotic and Concrete Efficiency”. In: *Journal of Cryptology* 37.3 (July 2024), p. 31. DOI: 10.1007/s00145-024-09511-8.

[GHL22] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. “Practical Non-interactive Publicly Verifiable Secret Sharing with Thousands of Parties”. In: *EUROCRYPT 2022, Part I*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13275. LNCS. Springer, Cham, 2022, pp. 458–487. DOI: 10.1007/978-3-031-06944-4_16.

[Gol+23] Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. “Brakedown: Linear-Time and Field-Agnostic SNARKs for R1CS”. In: *CRYPTO 2023, Part II*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14082. LNCS. Springer, Cham, Aug. 2023, pp. 193–226. DOI: 10.1007/978-3-031-38545-2_7.

[KHSS22] Daniel Kang, Tatsunori Hashimoto, Ion Stoica, and Yi Sun. *ZK-IMG: Attested Images via Zero-Knowledge Proofs to Fight Disinformation*. 2022. eprint: 2211.04775.

[Kil89] Joe Kilian. “Uses of randomness in algorithms and protocols”. PhD thesis. Massachusetts Institute of Technology, 1989.

[KLNO25] Michael Klooß, Russell W. F. Lai, Ngoc Khanh Nguyen, and Michał Osadnik. *RoK and Roll – Verifier-Efficient Random Projection for $\tilde{O}(\lambda)$-size Lattice Arguments*. Cryptology ePrint Archive, Paper 2025/1220. 2025. URL: https://eprint.iacr.org/2025/1220.

[KP23] Abhiram Kothapalli and Bryan Parno. “Algebraic Reductions of Knowledge”. In: *CRYPTO 2023, Part IV*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14084. LNCS. Springer, Cham, Aug. 2023, pp. 669–701. DOI: 10.1007/978-3-031-38551-3_21.

[KRS25] Dmitry Khovratovich, Ron D Rothblum, and Lev Soukhanov. “How to Prove False Statements: Practical Attacks on Fiat-Shamir”. In: *Annual International Cryptology Conference*. Springer. 2025, pp. 3–26.

[KST22] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. “Nova: Recursive Zero-Knowledge Arguments from Folding Schemes”. In: *CRYPTO 2022, Part IV*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13510. LNCS. Springer, Cham, Aug. 2022, pp. 359–388. DOI: 10.1007/978-3-031-15985-5_13.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. “Algebraic methods for interactive proof systems”. In: *Journal of the ACM (JACM)* 39.4 (1992), pp. 859–868.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. “Generalized Compact Knapsacks Are Collision Resistant”. In: *ICALP 2006, Part II*. Ed. by Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener. Vol. 4052. LNCS. Springer, Berlin, Heidelberg, July 2006, pp. 144–155. DOI: 10.1007/11787006_13.

[LS15] Adeline Langlois and Damien Stehlé. “Worst-case to average-case reductions for module lattices”. In: *DCC* 75.3 (2015), pp. 565–599. DOI: 10.1007/s10623-014-9938-4.

[LS18] Vadim Lyubashevsky and Gregor Seiler. “Short, Invertible Elements in Partially Splitting Cyclotomic Rings and Applications to Lattice-Based Zero-Knowledge Proofs”. In: *EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. LNCS. Springer, Cham, 2018, pp. 204–224. DOI: 10.1007/978-3-319-78381-9_8.

[LS24] Hyeonbum Lee and Jae Hong Seo. *On the Security of Nova Recursive Proof System*. Cryptology ePrint Archive, Report 2024/232. 2024. URL: https://eprint.iacr.org/2024/232.

[Ngu+24] Wilson D. Nguyen, Trisha Datta, Binyi Chen, Nirvan Tyagi, and Dan Boneh. “Mangrove: A Scalable Framework for Folding-Based SNARKs”. In: *CRYPTO 2024, Part X*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14929. LNCS. Springer, Cham, Aug. 2024, pp. 308–344. DOI: 10.1007/978-3-031-68403-6_10.

[NS24] Ngoc Khanh Nguyen and Gregor Seiler. “Greyhound: Fast Polynomial Commitments from Lattices”. In: *CRYPTO 2024, Part X*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14929. LNCS. Springer, Cham, Aug. 2024, pp. 243–275. DOI: 10.1007/978-3-031-68403-6_8.

[NS25] Wilson Nguyen and Srinath Setty. *Neo: Lattice-based folding scheme for CCS over small fields and pay-per-bit commitments*. Cryptology ePrint Archive, Report 2025/294. 2025. URL: https://eprint.iacr.org/2025/294.

[NT16] Assa Naveh and Eran Tromer. “PhotoProof: Cryptographic Image Authentication for Any Set of Permissible Transformations”. In: *2016 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2016, pp. 255–271. DOI: 10.1109/SP.2016.23.

[PP24] Christodoulos Pappas and Dimitrios Papadopoulos. “Sparrow: Space-Efficient zkSNARK for Data-Parallel Circuits and Applications to Zero-Knowledge Decision Trees”. In: *ACM CCS 2024*. Ed. by Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie. ACM Press, Oct. 2024, pp. 3110–3124. DOI: 10.1145/3658644.3690318.

[PR06] Chris Peikert and Alon Rosen. "Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices". In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. LNCS. Springer, Berlin, Heidelberg, Mar. 2006, pp. 145–166. DOI: 10.1007/11681878_8.

[RZ22] Carla Ràfols and Alexandros Zacharakis. *Folding Schemes with Selective Verification*. Cryptology ePrint Archive, Report 2022/1576. 2022. URL: https://eprint.iacr.org/2022/1576.

[Set20] Srinath Setty. "Spartan: Efficient and General-Purpose zkSNARKs Without Trusted Setup". In: *CRYPTO 2020, Part III*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12172. LNCS. Springer, Cham, Aug. 2020, pp. 704–737. DOI: 10.1007/978-3-030-56877-1_25.

[STW23] Srinath Setty, Justin Thaler, and Riad Wahby. *Customizable constraint systems for succinct arguments*. Cryptology ePrint Archive, Report 2023/552. 2023. URL: https://eprint.iacr.org/2023/552.

[Val08] Paul Valiant. "Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency". In: *TCC 2008*. Ed. by Ran Canetti. Vol. 4948. LNCS. Springer, Berlin, Heidelberg, Mar. 2008, pp. 1–18. DOI: 10.1007/978-3-540-78524-8_1.

[Whi18] Barry Whitehat. *Roll up token*. link. 2018.

[Xio+23] Luoyuan Xiong, Binyi Chen, Zhenfei Zhang, Benedikt Bünz, Ben Fisch, Fernando Krell, and Philippe Camacho. "VeriZexe: Decentralized Private Computation with Universal Setup". In: *32nd USENIX Security Symposium (USENIX Security 23)*. 2023, pp. 4445–4462.

[YCBC24] Chhavi Yadav, Amrita Roy Chowdhury, Dan Boneh, and Kamalika Chaudhuri. "FairProof : Confidential and Certifiable Fairness for Neural Networks". In: *ICML 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=EKye56rLuv.

[ZCF24] Hadas Zeilberger, Binyi Chen, and Ben Fisch. "BaseFold: Efficient Field-Agnostic Polynomial Commitment Schemes from Foldable Codes". In: *CRYPTO 2024, Part X*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14929. LNCS. Springer, Cham, Aug. 2024, pp. 138–169. DOI: 10.1007/978-3-031-68403-6_5.

A Proof of Theorem 5.1

The completeness proof is identical to that of Theorem 4.1. We focus on the knowledge soundness proof.

We first review a variant of Lemma 2.3 in the random oracle model, adapted from Section 8.2 and Figure 13 of [FMN24]. For ease of exposition, we assume that there are two random oracles H, H_{fd}, where $\mathsf{H}_{\mathsf{fd}} : \{0,1\}^L \to \mathcal{S}^{\ell_{\mathsf{np}}}$ is used to sample the folding challenge $\boldsymbol{\beta}$ given the L-bit transcript prefix, while H is used to generate other verifier challenges. In practice, one can use a single hash function to instantiate both H and H_{fd} via domain separation.

Lemma A.1 ([FMN24], Section 8.2)**.** *Define $\mathcal{U} := \mathcal{S}^{\ell}$, output space $\mathcal{Y}$, predicate $\Psi : \mathcal{U} \times \mathcal{Y} \to \{0,1\}$ as in Lemma 2.3. Define $\mathsf{prefix}_L : \{0,1\}^* \to \{0,1\}^L$ as the function that outputs the first L bits of its input. Let H and $\mathsf{H}_{\mathsf{fd}} : \{0,1\}^L \to \mathcal{S}^{\ell_{\mathsf{np}}}$ be random oracles.*

For every oracle algorithm[14] $\mathcal{A}^{\mathsf{H},\mathsf{H}_{\mathsf{fd}}}$ making at most Q oracle queries and outputting $y \in \mathcal{Y}$, define

$$\epsilon_{\Psi}(\mathcal{A}) := \Pr\big[y \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{H}_{\mathsf{fd}}},\ u := \mathsf{H}_{\mathsf{fd}}(\mathsf{prefix}_L(y)) \ :\ \Psi(u,y) = 1\big]\,,$$

where the randomness is over the random oracles.

There exists an oracle algorithm $\mathcal{E}^{\mathcal{A}}$, which invokes $\mathcal{A}$ at most $1+\ell$ times in expectation, such that with probability at least

$$\epsilon_{\Psi}(\mathcal{A}) - (Q+1)\cdot \ell/|\mathcal{S}|\,, \tag{59}$$

it outputs a tuple $(y_L \in \{0,1\}^L, (\mathbf{u}_i,\, y_{R,i})_{i=0}^{\ell})$ satisfying that (i) $\mathbf{u}_i \equiv_i \mathbf{u}_0$ for all $i \in [\ell]$, and (ii) for all $i \in [0,\ell]$:

$$(\Psi(\mathbf{u}_i, y_i := (y_L, y_{R,i})) = 1) \ \wedge\ (y_i = \mathcal{A}^{\mathsf{H},\mathsf{H}_{\mathsf{fd}}[y_L,\mathbf{u}_i]})\,.$$

Here, $\mathsf{H}_{\mathsf{fd}}[y_L, \mathbf{u}_0] = \mathsf{H}_{\mathsf{fd}}$ is the original sampled oracle, and $\mathsf{H}_{\mathsf{fd}}[y_L, \mathbf{u}_i]$ $(1 \le i \le \ell)$ denotes the programmed oracle of H_{fd} where $\mathsf{H}_{\mathsf{fd}}(y_L) := \mathbf{u}_i$.

We also need a non-interactive variant of Lemma 4.1 in the random oracle model. The proof is non-trivial, because compared to the proof of Lemma 4.1, we can no longer use the soundness of the *interactive* sumcheck protocol to argue the soundness of the Fiat-Shamir transformed protocol.

Our proof relies on the notion of **round-by-round soundness** [Can+19; CMS19]. A protocol is round-by-round sound if there is a state function over partial transcripts such that (i) if the instance x is not in the language, then the initial state is doomed; (ii) in each round, if the current transcript is in a doomed state, then for any next prover message, with overwhelming probability over the verifier's challenge, the next state will still be doomed; and (iii) if the final state w.r.t. the entire transcript is doomed, the verifier rejects.

Lemma A.2. *Define parameters aux, aux' as in Lemma 4.1. Let Π_{gr1cs} be the RoK in Figure 3. Let Π_{cm} be a binding commitment scheme that is further straightline extractable. Let H be a hash function modeled as a random oracle. $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{gr1cs}}]$ is an RoK from $\mathcal{R}^{\mathsf{aux}}_{\mathsf{gr1cs}}$ (Eq. (38)) to $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin}}$, with the relaxed input relation $\mathcal{R}^{\mathsf{aux}'}_{\mathsf{gr1cs}}$. Here, $\mathcal{R}_{\mathsf{batchlin}}$ is from Eq. (28) and $\mathsf{aux}_{\mathsf{cs}}$ is defined in Eq. (44).*

[14]WLOG, assume that $\mathcal{A}$ has no internal randomness, as we can convert a randomized $\mathcal{A}$ into a deterministic one (by choosing the optimal randomness) such that $\epsilon_{\Psi}(\mathcal{A})$ does not decrease.

Proof. Let Π'_{gr1cs} be the interactive proof version of Π_{gr1cs} (where every instance and prover commitment also attaches a message opening). As shown in [Can+19; CMS19], sumcheck protocols are *round-by-round sound.* Thus, it is straightforward to argue that Π'_{gr1cs} is also round-by-round sound. Let State, $\epsilon_{\mathsf{rbr}} = \mathsf{negl}(\lambda)$ be the state function and round-by-round soundeness error of Π'_{gr1cs}.

Next, we reduce the adaptive soundness of $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{gr1cs}}]$ to the round-by-round soundness of Π'_{gr1cs} and the security of Π_{cm}. Suppose for contradiction that a cheating PPT prover $\mathcal{A}$ for $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{gr1cs}}]$, with non-negligible probability ϵ, outputs an instance $\mathbb{x}$ and a tuple $(\mathsf{tr}, (\mathbb{x}_o, \mathbb{w}_o))$ such that (i) $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_o$ is the output instance-witness pair, (ii) $\mathbb{x}$ and the witness opening $\mathbb{w}$ (deterministically derived from $\mathbb{w}_o$) is not in the input relation, and (iii) $\mathsf{tr} = [(c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}, (r_i)_{i=1}^{\mathsf{rnd}+1}, (m_i)_{i=1}^{\mathsf{rnd}}]$ is an accepting transcript, where $(c_{\mathsf{fs},i})_i$ are the round commitments, $(r_i)_i$ are the hashed challenges, and $(m_i)_i$ are the opening prover messages. Note that $\mathbb{w}_o$ also includes valid openings $(\mathbf{g}^{(i)} \in \mathcal{M}^n)_{i=1}^{k_g}$ to the prover commitments $(c^{(i)})_{i=1}^{k_g}$ sent in Π_{gr1cs}.

From $\mathcal{A}$, we obtain a cheating PPT prover $\mathcal{A}'$ with same success probability. The adversary $\mathcal{A}'$ simply simulates the execution of $\mathcal{A}$ and for each prover commitment $c^{(i)}$ ($i \in [k_g]$) included in prover message m_1, $\mathcal{A}'$ also attaches the corresponding opening $\mathbf{g}^{(i)}$ from $\mathbb{w}_o$. Let m'_1 denote the updated message. We say that $\mathcal{A}'$ succeeds if $\mathcal{A}$ succeeds and $[(\mathbb{x}, \mathbb{w}), (r_1|m'_1|r_2|m_2|\cdots|m_{\mathsf{rnd}}|r_{\mathsf{rnd}+1}), (\mathbb{x}_o, \mathbb{w}_o)]$ is a valid transcript for Π'_{gr1cs}.

By the binding property of Π_{cm} and the definition of round-by-round soundness, either we break the binding of Π_{cm}, or there is an index $i \in [\mathsf{rnd}]$, such that with probability at least $\epsilon' = (\epsilon - \mathsf{negl}(\lambda))/\mathsf{rnd}$, the output of $\mathcal{A}'$ satisfies that

$$\mathsf{State}((\mathbb{x}, \mathbb{w}), \mathsf{tr}_i) = \mathsf{reject} \qquad \text{and} \qquad \mathsf{State}((\mathbb{x}, \mathbb{w}), \mathsf{tr}_{i+1}) = \mathsf{accept}\,, \tag{60}$$

where $\mathsf{tr}_i := (r_1|m'_1|r_2|m_2|\ldots|m_{i-1}|r_i)$ and $\mathsf{tr}_{i+1} := (r_1|m'_1|r_2|m_2|\ldots|r_i|m_i|r_{i+1})$. We argue that ϵ' is at most $\approx Q \cdot \epsilon_{\mathsf{rbr}} = \mathsf{negl}(\lambda)$, contradicting with the assumption that ϵ is non-negligible.

Claim 1. $\epsilon' \le Q \cdot (\epsilon_{\mathsf{rbr}} + \mathsf{negl}(\lambda))$.

Proof. Note that $\mathcal{A}$ above must have queried H on input $(r_1|c_{\mathsf{fs},1}|\ldots|r_i|c_{\mathsf{fs},i})$, otherwise it can never predict r_{i+1} with non-negligible probability. WLOG, we assume $\mathcal{A}$ always outputs the same input instance $\mathbb{x}$. Given index i, and the Q-query adversary $\mathcal{A}$, consider a mental experiment as follows: (The experiment can be simulated in polynomial time.)

1. Simulate the execution of $\mathcal{A}$ until it succeeds. Let $\mathbb{w}$ be the input witness deterministically derived from the vector $\mathbf{f} \in R_q^n$ in the output witness $\mathbb{w}_o$

2. Sample a query index $q \leftarrow\$ [Q]$

3. Simulate the execution of $\mathcal{A}$ again with fresh randomness. Given the qth oracle query input s, the experiment aborts if s has been queried before or s is not of the form $(r_1|c_{\mathsf{fs},1}|\ldots|r_i|c_{\mathsf{fs},i})$; otherwise, it extracts the message m_j underlying $c_{\mathsf{fs},j}$ for $j \in [i]$

4. Given challenge r_1 (that includes random projection matrix $\mathbf{J}$), compute the monomial vectors $(\mathbf{g}^{(i)})_{i=1}^{k_g}$ from the projected matrix $(\mathbf{I}_{n/\ell_h} \otimes \mathbf{J}) \times \mathsf{cf}(\mathbf{f})$ and attach $(\mathbf{g}^{(i)})_{i=1}^{k_g}$ into m_1 to obtain updated message m_1'
5. Sample uniformly random r_{i+1} afterward[15]
6. Outputs 1 if and only if Eq. (60) holds w.r.t. $\mathsf{tr}_i := (r_1|m_1'|r_2|m_2|\dots|m_{i-1}|r_i)$ and $\mathsf{tr}_{i+1} := (r_1|m_1'|r_2|m_2|\dots|r_i|m_i|r_{i+1})$

By definition of round-by-round soundness, the success probability of the experiment is at most ϵ_{rbr}. On the other hand, by definition of ϵ' and the binding property of the lattice commitment, the experiment outputs 1 with probability at least $\epsilon'/Q - \mathsf{negl}(\lambda)$. Thus, $\epsilon' \le Q \cdot (\epsilon_{\mathsf{rbr}} + \mathsf{negl}(\lambda))$ as desired. □

In sum, $\mathcal{A}$'s success probability ϵ must be negligible, and thus $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{gr1cs}}]$ is adaptively sound. This also implies knowledge soundness as the extracted witness opening is deterministically derived from the adversary's output. □

Remark A.1. The straightline extractability requirement of Π_{cm} seems to be a proof artifact. We conjecture that the binding property suffices and leave it as an open problem.

Next, equipped with Lemma A.1 and Lemma A.2, we prove the knowledge soundness of $\mathsf{FS}^{\mathsf{H},\mathsf{H}_{\mathsf{fd}}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{fold}}]$.

Knowledge soundness. We first describe the extractor. WLOG, assume that the adversary P^* is deterministic. Define folding challenge space $\mathcal{U} := \mathcal{S}^{\ell_{\mathsf{np}}}$. Let $\mathcal{Y}$ be the domain of $(\pi, (\mathbb{x}_o, \mathbb{w}_o))$ where $(\mathbb{x}_o, \mathbb{w}_o)$ is the output instance-witness pair, and $\pi = [(c_{\mathsf{fs},i}, m_i)_{i=1}^{\mathsf{rnd}}]$ is the folding proof where $c_{\mathsf{fs},i}$'s are the round commitments and m_i's are the opening prover messages. Define predicate $\Psi(\mathbf{u} \in \mathcal{U}, \mathbf{y} := (\pi, (\mathbb{x}_o, \mathbb{w}_o)))$ that outputs 1 if and only if (i) π is an accepting proof (with derived folding challenge $\boldsymbol{\beta} = \mathbf{u}$) and (ii) $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_{\mathsf{lin}}^{\mathsf{auxcs}} \times \mathcal{R}_{\mathsf{batchlin}}$. For folding challenge $\mathbf{u}$ and the extra verifier randomness $\mathbf{r}_v$, define $\mathcal{A}_{\mathbf{r}_v}(\mathbf{u})$ as the adversary that simulates the execution of P^* with verifier randomness $\mathbf{r}_v$ and folding challenge $\mathbf{u}$. Let Ext_{cwss} be the oracle extractor from Lemma A.1.

<u>**Extractor $\mathsf{Ext}^{\mathcal{A}^*}$:**</u>

1. Sample folding challenge $\mathbf{u}_0 \leftarrow\!\!\$\ \mathcal{S}^{\ell_{\mathsf{np}}}$ and extra verifier randomness $\mathbf{r}_v$
2. Run extractor $\mathsf{Ext}_{cwss}^{\mathcal{A}_{\mathbf{r}_v}(\mathbf{u}_0)}$. Abort if it fails. Otherwise, let the output be

$$\left(\mathbf{u}_\ell, \mathbf{y}_\ell := (\pi_\ell, \mathbb{x}_o^\ell, \mathbb{w}_o^\ell)\right)_{\ell=0}^{\ell_{\mathsf{np}}},$$

where for all $\ell \in [\ell_{\mathsf{np}}]$, $\mathbf{u}_\ell \equiv_\ell \mathbf{u}_0$ and

$$\left(\Psi(\mathbf{u}_\ell, y_\ell := (y_L, y_{R,\ell})) = 1\right) \;\wedge\; \left(y_\ell = \mathcal{A}^{\mathsf{H},\mathsf{H}_{\mathsf{fd}}[y_L, \mathbf{u}_\ell]}\right).$$

[15]It is important to sample r_{i+1} after extracting m_1' and $(m_j)_{j=2}^{i}$, otherwise r_{i+1} may be correlated with the prover messages.

3. For every $\ell \in [\ell_{\mathsf{np}}]$, parse $\mathbb{w}_o^\ell = (\mathbf{f}^{*,\ell}, (\mathbf{g}^{(i,\ell)})_{i=1}^{k_g})$, compute

$$\mathbf{f}^\ell := (\mathbf{f}^{*,\ell} - \mathbf{f}^{*,0})/(\mathbf{u}_\ell[\ell] - \mathbf{u}_0[\ell]) \in R_q^n\,, \tag{61}$$

$$\forall i \in [k_g] \,:\, \mathbf{h}^{i,\ell} := (\mathbf{g}^{(i,\ell)} - \mathbf{g}^{(i,0)})/(\mathbf{u}_\ell[\ell] - \mathbf{u}_0[\ell]) \in R_q^n \tag{62}$$

   Parse $\mathbf{f}^\ell$ as $\mathbf{f}^\ell := \mathsf{cf}^{-1}\left([\mathbf{X}^{\ell\top}, \mathbf{W}^{\ell\top}]^\top\right) \in R_q^n$, abort if $\mathbf{X}^\ell \neq \mathbf{X}_{\mathsf{in}}^\ell$
4. Output $\mathbb{w} := \left\{\mathbb{w}_\ell = \mathbf{W}^\ell \in \mathbb{Z}_q^{n_w \times d}\right\}_{\ell=1}^{\ell_{\mathsf{np}}}$

Success probability. Fix verifier randomness $\mathbf{r}_v$ (that excludes folding challenge $\boldsymbol{\beta}$). By Lemma A.1, Step 2 succeeds with probability $\epsilon_\Psi(\mathcal{A}_{\mathbf{r}_v}) - (Q+1) \cdot \ell_{\mathsf{np}}/|\mathcal{S}|$ where

$$\epsilon_\Psi(\mathcal{A}_{\mathbf{r}_v}) := \Pr\left[y \leftarrow \mathcal{A}_{\mathbf{r}_v}^{\mathsf{H}, \mathsf{H}_{\mathsf{fd}}},\ u := \mathsf{H}_{\mathsf{fd}}(\mathsf{prefix}_L(y)) \,:\, \Psi(u, y) = 1\right].$$

Let $(c_\ell)_{\ell=1}^{\ell_{\mathsf{np}}}$ denote the input commitments and $(c_\ell^{(i)})_{i \in [k_g], \ell \in [\ell_{\mathsf{np}}]}$ the monomial commitments. The extracted openings $(\mathbf{f}^\ell, (\mathbf{h}^{i,\ell})_{i=1}^{k_g})_{\ell=1}^{\ell_{\mathsf{np}}}$ satisfy:

- For $\ell \in [\ell_{\mathsf{np}}]$, $\mathbf{f}^\ell$ is bound to c_ℓ, with relaxed opening $(\mathbf{f}^{*,\ell} - \mathbf{f}^{*,0}, \mathbf{u}_\ell[\ell] - \mathbf{u}_0[\ell])$ (Eq. (10)).
- For $i \in [k_g], \ell \in [\ell_{\mathsf{np}}]$, $\mathbf{h}^{i,\ell}$ is bound to $c_\ell^{(i)}$, with opening $(\mathbf{g}^{(i,\ell)} - \mathbf{g}^{(i,0)}, \mathbf{u}_\ell[\ell] - \mathbf{u}_0[\ell])$.
- For $\ell \in [\ell_{\mathsf{np}}]$, let $\mathbb{x}_{o,\ell}$ be the reduced instance from Eq. (46), and set $\mathbb{w}_{o,\ell} := (\mathbf{f}^\ell, (\mathbf{h}^{i,\ell})_{i=1}^{k_g})_{\ell=1}^{\ell_{\mathsf{np}}}$. Define $\hat{\mathcal{R}}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}}$, $\hat{\mathcal{R}}_{\mathsf{batchlin}}$ as the relaxed relations of $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}}$, $\mathcal{R}_{\mathsf{batchlin}}$ where the strict opening check $\mathsf{VfyOpen}$ is replaced with the relaxed ones from Eq. (10). Since $\Psi(\mathbf{u}_\ell, \mathbf{y}_\ell) = \Psi(\mathbf{u}_0, \mathbf{y}_0) = 1$, and $\mathbf{u}_\ell \equiv_\ell \mathbf{u}_0$, we get that $(\mathbb{x}_{o,\ell}, \mathbb{w}_{o,\ell}) \in \hat{\mathcal{R}}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \hat{\mathcal{R}}_{\mathsf{batchlin}}$.

Let ϵ_{P^*} be the prover P^*'s success probability. Thus, with probability at least $\epsilon_{\mathsf{P}^*} - ((Q+1) \cdot \ell_{\mathsf{np}}/|\mathcal{S}|)$ (where the randomness is over $\mathbf{r}_v$, $\mathbf{u}_0$, H, H_{fd}, and Ext_{cwss}), the extractor outputs $(\mathbb{w}_{o,\ell})_{\ell \in [\ell_{\mathsf{np}}]}$ at Step 3 satisfying the relaxed relation $(\hat{\mathcal{R}}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \hat{\mathcal{R}}_{\mathsf{batchlin}})_{\mathsf{np}}^\ell$ w.r.t. $(\mathbb{x}_{o,\ell})_{\ell=1}^{\ell_{\mathsf{np}}}$. By the same soundness argument as Lemma A.2 (relying on relaxed binding, Eq. (10), instead of strict binding, Eq. (11), of Construction 2.1), and by a union bound, the extractor succeeds with probability

$$\epsilon_{\mathsf{P}^*} - ((Q+1) \cdot \ell_{\mathsf{np}}/|\mathcal{S}|) - \ell_{\mathsf{np}} \cdot Q \cdot \mathsf{negl}(\lambda) = \epsilon_{\mathsf{P}^*} - \mathsf{negl}(\lambda)\,,$$

and produces a witness $\mathbb{w} := \left\{\mathbb{w}_\ell = \mathbf{W}^\ell\right\}_{\ell=1}^{\ell_{\mathsf{np}}}$ in $(\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}'})^{\ell_{\mathsf{np}}}$ as required. $\square$