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

Binyi Chen

Tsinghua University*

November 21, 2026

Abstract

Folding/Accumulation schemes are a powerful tool for building scalable proof systems. However, standard approaches for leveraging folding schemes, such as IVCs or PCDs, require embedding hash functions (modeled as random oracles) into recursive circuits, introducing both security concerns and significant proving overhead.

We re-envision how to use folding schemes, and introduce Symphony, the first proving paradigm that leverages folding schemes as a black box without embedding hashes in SNARK circuits. It is memory-efficient, parallelizable, and plausibly post-quantum secure, with polylogarithmic proof size and verification. The (low-memory) prover requires $O(\log \log n)$ passes of the input data and its computation is dominated by committing to the input witnesses.

A core component of our construction is a new lattice-based folding scheme that compresses *a large number* of NP-complete statements into one *in a single shot*. Furthermore, we design a generic compiler that converts a folding scheme into a SNARK without embedding the Fiat-Shamir circuit into proven statements.

*Partial work done while at Stanford University.

Contents

1 Introduction **3**
- 1.1 Our Contributions . . . 4
- 1.2 Technical Overview . . . 5
- 1.3 Comparison with Monolithic SNARKs . . . 8
- 1.4 Other Related Work . . . 9
- 1.5 Organization . . . 10

2 Preliminaries **10**
- 2.1 Algebra Background . . . 10
- 2.2 Lattice-based Binding Commitments . . . 13
- 2.3 Tensor of Rings . . . 15
- 2.4 Generalized Reduction of Knowledge . . . 15
- 2.5 The Sumcheck Protocol . . . 16
- 2.6 (Commit-and-Prove) SNARKs . . . 17

3 A Toolbox of Reduction of Knowledge **19**
- 3.1 Generic Committed Linear Relation . . . 19
- 3.2 RoKs for Hadamard Relations . . . 19
- 3.3 RoKs for Monomial Relations . . . 22
- 3.4 Approximate Range Proofs for Ring Vectors . . . 23

4 High-Arity Folding for NP-Complete Relations **28**
- 4.1 Generalized Committed R1CS Relation . . . 29
- 4.2 The High-Arity Folding Scheme . . . 30

5 The Fiat-Shamir Transform **37**

6 SNARKs in the ROM from High-Arity Folding **38**

7 Instantiation **42**

8 Two-layer Folding by Splitting Linear Statements **44**

A The Monomial RoK from LatticeFold+ **54**

B Proof of Theorem 5.1 **55**

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 splits the computation into small steps and proves 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 based on accumulation/folding schemes was introduced in Halo [BGH19] and further developed in [BCMS20; Bün+21; BDFG21; KST22]. Folding/Accumulation schemes are 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. By removing the *full succinctness* requirement, folding schemes allow more efficient provers and can still serve as a building block for constructing IVC/PCDs [Bün+21; KST22].

Before our work, the standard (and the only) framework to leverage folding schemes[1] is to construct IVCs/PCDs 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. Folding-based IVCs/PCDs is streaming-friendly, that is, one can start proving without knowing all witnesses upfront. Moreover, they achieve significant speedups over IVCs/PCDs built from recursive SNARKs, because the folding verifier circuit is much simpler than a SNARK verifier. However, this strategy faces a few limitations.

Instantiating random oracles in SNARK circuits. 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 efficiency and security concerns for IVC/PCD constructions.

Efficiency-wise, the folding verifier logic embeds heavy hashing gadgets: A SHA-256 circuit takes tens of thousands of R1CS constraints [Pro]; even a SNARK-friendly hash takes hundreds of R1CS constraints. Hence, the Fiat-Shamir circuit is dominant in lattice-based folding schemes [BC24; BC25b; NS25] as it requires hashing elements over

[1] Monolithic SNARKs that use "folding-like" techniques do exist, but they do not explicitly use folding/accumulation schemes as a black box.

R_q that are much larger than $\mathbb{Z}_q$-elements; hash-based folding verifier circuits take even more constraints to check Merkle opening paths.

Security-wise, hash-based SNARKs and non-interactive folding schemes 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 resulting IVC/PCDs (based on folding or recursive SNARKs) can be secure only under heuristic conjectures. Notably, practical attack exist [KRS25] for GKR-based SNARKs if we allow the proven statement to compute the Fiat-Shamir hash function itself. Although this does not directly affect folding-based IVC/PCD schemes, it is still a warning sign.

Limited folding arity. Due to the circuit cost of hashing, the folding schemes *underlying IVCs/PCDs* typically take only 2 or 3 inputs per step. Batch proving more input statements then requires deeper folding trees, which degrades parallelism and security of IVC/PCD schemes.[2] For the schemes that fold more inputs per step [Ngu+24; RZ22], the embedded folding verification circuit becomes more expensive. We thus ask,

> *Rather than using folding schemes for IVC/PCDs, can we leverage folding schemes in a different way to obtain memory-efficient proof systems?*

1.1 Our Contributions

We re-envision how to use folding schemes introduced in [BGH19; BCMS20; Bün+21]. For the first time, we move away from IVC/PCDs and advocate a *high-arity* folding approach. This leads to a new *folding-based* memory-efficient proving framework *without instantiating random oracles in proven statements.*

A new high-arity folding scheme. Our core technical contribution is a new lattice-based *high-arity* folding scheme (Theorem 4.1). It is plausibly post-quantum secure, the proving time is dominated by committing to the input witnesses and running a degree-3 sumcheck protocol over an extension field, which are standard baseline operations existing in all prior folding schemes. In comparison, prior schemes such as LatticeFold [BC25a], Neo [NS25], and concurrent works such as Cyclo [GLLO26] and SuperNeo [NS26] all rely on decomposing witnesses for range proofs, which requires generating more commitments or running higher-degree sumchecks; LatticeFold+ [BC25b] avoids witness-decomposition in range proofs but incurs extra costs for double-commitment transformations. These extra mechanisms significantly increase prover cost and recursive verifier circuit sizes (even if we exclude the Fiat-Shamir circuit) when the folding arity grows.

SNARKs from high-arity folding. As a second contribution, we develop a generic framework that converts the high-arity folding scheme into a succinct argument in the

[2] IVCs/PCDs from rewinding-based folding is secure only if the folding depth is a constant, with attacks on certain schemes for super-logarithmic depths [LS24; BMNW25a].

random oracle model (Theorem 6.1). Similar to IVC/PCDs, we do need a proof ensuring the correctness of folding, but it does *not* embed random oracle calls inside the proven statement. In Section 8, we extend the framework to support higher folding depths.

Targeted applications and evaluation. Symphony excels in proving *gigantic* statements that can be decomposed into many *mid-sized* uniform statements, where each uniform statement requires certain range proofs by default. This is the case for applications such as video editing provenance, post-quantum aggregate signatures, zkVMs, and verifiable machine learning. E.g., 0xPARC [0xP26] leverages Symphony to verify Llama 3 inferences; Syscoin [Sys26] uses Symphony to help shrink the SP1 verifier circuits.

In Section 7, we provide a concrete parameter instantiation and estimate Symphony's efficiency in terms of prover operations, SNARK circuit and proof sizes. To prove 2^{16} standard R1CS statements (each with 2^{16} constraints over a 64-bit field), we expect the resulting proofs to be under 200KB (and under 50KB if post-quantum security is not required). The prover memory size is independent of the number of R1CS statements, and the prover time is dominated by approximately $3 \cdot 2^{32}$ multiplications between arbitrary elements and low-norm elements over $R_q := \mathbb{Z}_q[X]/\langle X^{64} + 1\rangle$.

Limitations. Symphony is designed as a memory-efficient proof system, not a standard IVC/PCD scheme. Our main goal is to use folding schemes as a black box to achieve practical batch-proving while avoiding the heavy recursion overhead in IVC/PCD schemes. That said, Symphony does not support fully online streaming as in IVC/PCDs, but only supports small-memory proving via streaming over a known batch of statements. Furthermore, while concretely efficient, the verification complexity of Symphony is linear in the number of proven statements.

To achieve full succinctness and streaming, Symphony can be combined with recursive SNARKs: we partition the online statements into blocks, where each block of ℓ_{np} statements is folded using Symphony, then we apply a recursive SNARK to further absorb Symphony's verification logic. This hybrid approach still instantiates random oracles inside recursive statements, but it is much cheaper than the naive IVC/PCD schemes.

1.2 Technical Overview

Lattice-based high-arity folding. We review a standard folding paradigm from lattice commitments. Consider $\ell_{\mathsf{np}} > 1$ **committed R1CS statements**[3], where each statement consists of a public input $\mathbf{X}_{\mathsf{in}}$ and a commitment c (instantiated with module-based Ajtai hash functions [Ajt96; PR06; LM06]); the relation checks that a message-opening pair (m, o) is valid for c and $(\mathbf{X}_{\mathsf{in}}, m)$ satisfies the corresponding R1CS constraints. We fold the ℓ_{np} statements via steps below:

1. **Sumcheck reduction.** Standard techniques [Set20; CBBZ23] transform the R1CS statements to a sumcheck claim. The prover and verifier then run a sumcheck protocol [LFKN92] to reduce the claims into ℓ_{np} linear evaluation statements.

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

2. **Random linear combination.** The verifier samples a low-norm vector $\boldsymbol{\beta}$ and linearly combines the instances via $\boldsymbol{\beta}$. The prover similarly combines the witnesses. By linearity of the evaluation statements and the Ajtai hash function, the committed linear relation is preserved after the linear combination.

The key challenge is to prove the *low-norm* property of witness openings. This is crucial for commitment's binding property and the folding scheme's knowledge soundness. Our design goal is to incur minimal overhead beyond the standard bare-bone folding mechanism above. This prevents us from using techniques such as witness decomposition [BC25a; NS25; GLLO26; NS26] and double-commitments [BC25b].

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 $\mathbf{Jw} \bmod q$, 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. The key question now is how to efficiently fold *multiple* norm-check statements of the projected vectors into a *single* linear statement.

Monomial embedding lookup. We fold the norm-check statements via the monomial lookup technique [BC25b]. 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 lookup polynomial $t(X) := \sum_{i \in [1, d/2)} i \cdot (X^{-i} + X^i) \in R_q$. On input a commitment to a vector $\mathbf{f} \in (-d/2, d/2)^n$, the prover sends a commitment to a monomial vector $\mathbf{g} \in R_q^n$, where $\mathbf{g}_i \approx X^{\mathbf{f}_i}$ embeds $\mathbf{f}_i$ onto exponent for all $i \in [n]$. To convince the verifier that $\mathbf{f} \in (-d/2, d/2)^n$, it suffices to show that the constant term of $\mathbf{g}_i \cdot t(X)$ equals $\mathbf{f}_i$ for all $i \in [n]$, which can be efficiently proved using a sumcheck-based protocol over a λ-*bit field*. The prover cost is $O(n)$ 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 is to send d monomial commitments, which incurs a large communication complexity. Instead, LatticeFold+ sends a *double commitment* c to the d monomial commitments and prove c's correctness. However, this technique requires the execution of two extra sumcheck protocols.

Our hybrid scheme. Our idea is to only check the norm of a *projected* vector $\mathbf{w}' = \mathbf{Jw} \bmod q \in R_q^{n/\ell_h}$, whose coefficient matrix $\mathbf{W}' \in ([-q/2, q/2) \cap \mathbb{Z})^{(n/\ell_h) \times d}$ has smaller dimensions.[4] By flattening $\mathbf{W}'$ into a vector, we directly apply the monomial lookup protocol *without double commitments*. The prover only computes $O(1)$ instead of d monomial commitments. By integrating our range proof with the bare-bone folding mechanism, we reduce the norm-checks of ℓ_{np} input witnesses into a *single* linear statement. Here, the exact norm check (via monomial lookup) is important to minimize

[4] As a tradeoff, we only achieve approximate range proofs. But this is sufficient when the folding depth is a small constant.

proving cost of the *reduced* statement. Otherwise the reduced statement needs to check the norms of $\ell_{\mathsf{np}} \gg 1$ projected vectors.

Our norm-check protocol is relatively simple, but the knowledge soundness proof requires some care. The key challenge lies in ensuring the consistency between the projected vector $\mathbf{w}' = \mathbf{J}\mathbf{w} \bmod q$ and the monomial vector $\mathbf{g}$. We carefully leverage sumcheck soundness and the algebraic property of tensor of rings (Section 2.3) to reduce the consistency check to evaluation claims of $\mathbf{w}$ and $\mathbf{g}$'s multilinear extensions. To formally apply sumcheck soundness, the extractor needs to first extract the committed vectors $\mathbf{w}'$ and $\mathbf{g}$ via rewinding. However, the monomial commitments and their underlying vectors $\mathbf{g}'$ sent by the prover is unlikely to be the same across different rewinding executions, because the projection matrix $\mathbf{J}$ is unpredictable and can lead to a different projected vector. Even worse, the rewinding procedure may take more than polynomial time if the prover's conditional success probability, after fixing public parameters and $\mathbf{J}$, is negligible. This makes it difficult to directly apply sumcheck soundness. We address this issue by carefully designing the extractor and corresponding hybrid mental experiments, see the proof of Theorem 3.1 for more details.

Prover complexity. As discussed in Remark 4.2, the folding prover requires space roughly the same as the witness size n of each input statement. As a tradeoff, the algorithm requires $O(\log \log(n))$ passes over the input data. Fortunately, the prover can start computing the witness commitments without loading the entire witness, and each subsequent pass takes a smaller amount of data and computation than the previous pass.

From high-arity folding to succinct proofs. For now, the verifier circuit complexity is dominated by the Fiat-Shamir heuristic, which is large with high folding arity ℓ_{np}. This makes the standard IVC/PCD compiler infeasible. We present a new *memory-efficient* paradigm for building folding-based SNARKs without recursively proving hash computations. Naively, given a folding scheme Π_{fold}, we can apply Fiat-Shamir to obtain a non-interactive folding scheme $\mathsf{FS}[\Pi_{\mathsf{fold}}]$ in the random oracle model. However, the folding verifier needs to read at least ℓ_{np} committed instances to generate a folding challenge. That already blows up the proof size for large ℓ_{np}. E.g., for $\ell_{\mathsf{np}} = 1000$, the folding proof π easily exceeds 30MB in size with lattice or hash-based schemes.

We address the proof-size inefficiency by letting the verifier run Fiat-Shamir on a *compressed* transcript. The prover first sends *short commitments* $\{c_{\mathsf{fs},i}\}_i$ to the prover round messages of Π_{fold}, the verifier then obtains the round challenges using $\{c_{\mathsf{fs},i}\}_i$ (rather than the original prover messages). Finally, the prover uses a Commit-and-Prove SNARK (CP-SNARK) to show that the openings of $\{c_{\mathsf{fs},i}\}_i$ gives a valid folding proof w.r.t. *public* verifier challenges. Importantly, since the folding challenges are generated by the *verifier*, the CP-SNARK statement does *not* need to prove hash computations.

In Appendix B, we reduce the knowledge soundness of the compiled *non-interactive* protocol (Section 7) to that of the original interactive protocol Π_{fold}, assuming that the "outer commitment" (for transcript compression) is straightline extractable in idealized models. Security proofs for *non-interactive* protocols in the random oracle model

are often more complex. For example, the analysis of the *non-interactive version* of LaBRADOR [Aar+24] relies on a sophisticated framework called predicate special soundness. Our approach enables a simpler analysis by exploiting the straightline extractability of the underlying commitment.

For better succinctness, we instantiate the "outer commitments" with Merkle hashes or KZG commitments. Compared to the warm-up, this scheme compresses $>$ 30MB-sized folding proofs into $\log(n)$ Merkle (or KZG) commitments, under 1KB for most applications. Finally, we also compress the reduced instance-witness pair $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_o$ of Π_{fold} into a tiny string using standard SNARKs.

Multi-layer folding 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.

In Section 8, we extend our scheme 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 avoid embedding Fiat-Shamir heuristics in circuits.

Extending to other folding schemes. While our compiler naturally extends for elliptic-curve-based folding schemes (e.g., [BC23; KS24b; KS24a]), it is more nuanced to compile hash-based folding schemes: although we can move the Fiat-Shamir transform outside the recursive SNARK circuit, the CP-SNARK statement still needs to check Merkle-path openings, which accounts for the dominant part of the recursive statement. That said, *efficiently* compiling a hash-based folding scheme into a succinct argument in the random oracle model remains an open problem.

1.3 Comparison with Monolithic SNARKs

Compared to folding-based IVC/PCDs, Symphony enables smaller recursion overhead and folding depth, and the security can be proven in the random oracle model. While memory-efficient SNARKs in the random oracle model do exist (e.g.[BCHO22; Baw+24]), we highlight two advantages of Symphony over monolithic SNARKs:

Modularity. Symphony is more *modular and generic*, which can combine different folding schemes and CP-SNARKs, compared to existing monolithic SNARKs that require careful and tailored memory and streaming optimizations. For example, the state-of-the-art lattice-based argument, LaBRADOR [BS23], requires the prover

to compute "cross terms" over all pairs of chunked witnesses, hardening memory and streaming optimizations when the number of chunks is large.

Scalability. High-arity folding combined with CP-SNARKs scales better for *large-scale* computation. While CP-SNARKs [Kil89; CLOS02; CFQ19] can already prove ℓ instance-witness pairs in a batch, their concrete cost is high when ℓ is large. For example, the state-of-the-art low-memory SNARK [Baw+24] requires $O(\ell n)$ expensive elliptic curve operations; certain hash-based and lattice-based monolithic SNARKs allow for fast proof generation on reasonable statement sizes (e.g., 2^{30}), but their performance deteriorate with limited memory and increasing statement sizes. For example, Gemini [BCHO22] incurs a quasilinear prover time complexity to achieve a small footprint; LaBRADOR needs to compute ℓ^2 inner products (i.e., $\Omega(n\ell^2)$ operations) for the cross terms, which is a big overhead for large ℓ. Our approach sidesteps these bottlenecks by shifting the main computational overhead to the more efficient folding phase, using CP-SNARKs only for a lightweight consistency check between input instances and the folded instance. This reduces memory consumption to $O(n)$ and time complexity down to $O_\lambda(\ell n)$.

1.4 Other Related Work

Lattice double-commitments. Lattice double-commitment schemes such as [BS23; BC25b] shrink the Fiat-Shamir transcript by decomposing and re-committing the lattice commitments sent by the prover in each round. Our commit-and-prove compiler shares a structural similarity but does not require the commitments to be linearly homomorphic. Moreover, unlike [Aar+24] that analyzes the *non-interactive version* of LaBRADOR using the sophisticated *predicate special soundness*, we provide a simpler security proof in the random oracle model. Furthermore, our two-layer folding scheme (Section 8) introduces a new splitting RoK that reduces a single committed linear statement into multiple "split" statements, which was not known before.

Sumcheck-based folding schemes. Hypernova [KS24b] initiated the use of Sumcheck protocols in folding schemes, which NeutronNova [KS24a] later optimized to support more inputs without significantly increasing verifier circuit costs. However, both schemes are based on elliptic curves and are vulnerable to quantum attacks. There are two types 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; BC25b; NS25; FKNP24; GLLO26; NS26] have smaller verifier circuit, but the range-proof and the Fiat-shamir circuit costs are still bottlenecks, which force them to use low folding arity under the standard IVC/PCD compiler [Bün+21; KST22].

Security of IVC/PCDs. 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 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].

1.5 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 5 and Section 6 present a generic compiler that converts the high-arity folding scheme into a *non-interactive* succinct argument. Section 7 gives a candidate instantiation. In Section 8, we extend our scheme to support constant folding depths.

2 Preliminaries

Section 2.1 provides necessary algebra background. Section 2.2 describes binding commitments and their instantiations over lattices. Section 2.3 presents the tensor-of-rings framework that enables interleaving between sumcheck and folding operations. Section 2.4 defines generalized reduction of knowledge. Section 2.5 interprets the sumcheck protocol in the language of 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 representatives 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

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 $i \in [n]$, $\mathbf{f}_i$ or $\mathbf{f}[i]$ denotes the ith element of $\mathbf{f}$. For vectors $\mathbf{f}, \mathbf{g}$ of the same length, their inner product is denoted as $\langle \mathbf{f}, \mathbf{g} \rangle$. $\mathbf{I}_n$ is the identity matrix of rank n. For vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k \in \bar{R}^n$ over a ring $\bar{R}$, 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 $\mathbf{M}$, and $\mathsf{flt}(\mathbf{M}) := (\mathbf{M}_{1,*}, \ldots, \mathbf{M}_{n,*}) \in \bar{R}^{nm}$ 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, $R_q := R/qR = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$ is 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 its coefficient matrix as $\mathsf{cf}(\mathbf{f}) := (\mathsf{cf}(\mathbf{f}_1)^\top, \ldots, \mathsf{cf}(\mathbf{f}_n)^\top) \in \mathbb{Z}_q^{n\times d}$, 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 $\mathsf{flt}(\mathbf{F}) \in R^{nm}$. We slightly abuse the notation and define **norms for matrices over R_q (and $\mathbb{Z}_q$)** as the norms of their canonical lifting[5] 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 $\|\mathcal{S}\|_{\mathsf{op}} := \max_{a\in\mathcal{S}} \|a\|_{\mathsf{op}}$. We can similarly define **the operator norm for set $\mathcal{S}$ over R_q**, where for $a \in \mathcal{S} \subseteq R_q$, $y \in R$, the multiplication $a \cdot y \in R$ is over R between y and the canonical embedding of a in R.

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

Gadget decomposition. Fix modulus $q \in \mathbb{N}$, integer base $2 \le b < q$, and set $k := 1 + \lfloor \log_b(q) \rfloor$. The base-$b$ decomposition $\mathsf{decomp}_{b,k}(\mathbf{f}) \in \mathbb{Z}_q^{nk}$ on input $\mathbf{f} \in \mathbb{Z}_q^n$ is defined 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 $\|\mathbf{g}_i\|_\infty \le b/2$ and $\mathbf{f}_i = \langle \mathbf{g}_i, (1, b, \ldots, b^{k-1})\rangle$. Then $\mathsf{decomp}_{b,k}(\mathbf{f}) := (\mathbf{g}_1, \ldots, \mathbf{g}_n) \in \mathbb{Z}_q^{nk}$.

[5] The canonical embedding maps elements from the quotient ring R_q (and $\mathbb{Z}_q$) to representatives in R (and $\mathbb{Z}$) that have minimal norms.

Monomial embedding. We review the monomial embedding framework from [BC25b], 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. We define the **table polynomial**

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

where the equality holds because $X^d = -1$ over R_q. Intuitively, $t(X)$ embeds the set $(-d/2, d/2)$ onto its coefficients.

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 every $a \in (-d/2, d/2)$. 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 [BC25b])**.** *For all $a \in (-d/2, d/2)$ and $b := \mathsf{Exp}(a)$, 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 $B \le q/125$ and vector $\mathbf{v} \in [-q/2, q/2)^n$ with $\|v\|_2 > B$,

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

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 binding commitment schemes and instantiate them with the module-variant of the Ajtai hash function [Ajt96; PR06; LM06].

Definition 2.1. *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:*

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

CM *satisfies the following properties:*

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

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

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

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

We instantiate the binding commitment below. Notably, we add a *strict* opening verification interface $\mathsf{VfyOpen}(\cdot)$ to distinguish *strict* and *relaxed* openings, where the former has a more restricted opening space. In our folding scheme, an honest prover always follows the strict opening procedure, relaxed openings are only used in knowledge soundness proofs.

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 $\kappa = \kappa(\lambda)$, (relaxed) 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 $\mathsf{CM} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{RVfyOpen}, \mathsf{VfyOpen})$ works as follows:

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

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

- $\mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{m}, o = (\mathbf{f}, s)) \to b$: 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 $\mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1$ in this case.

We focus on the binding property as completeness is straightforward. Let T be the operator norm of $\mathcal{S}$. As noted in [BS23], the (relaxed) binding property follows from the Module-SIS assumption $\mathsf{MSIS}_{q,\kappa,n,4TB_{\mathsf{rbnd}}}$ defined below.

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 adversaries $\mathcal{A}$,*

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

Fine-grained commitment opening relation. Sometimes we need a more fine-grained check on opening vectors. Set $\ell_h \in \mathbb{N}$ 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

$$\mathbf{A}\mathbf{f} = c \quad \wedge \quad \forall (i,j) \in [n/\ell_h] \times [d] \ : \ \|\mathbf{F}_{i,j}\|_2 \le B. \tag{12}$$

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,

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

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

2.3 Tensor of Rings

We describe a simplified **Tensor-of-Rings** framework inspired by [BC25b; DP24; NS25]. Let $\mathbb{K} = \mathbb{F}_{q^t}$ be an extension field over $\mathbb{F}_q = \mathbb{Z}_q$. We define the polynomial quotient ring

$$\mathbb{E} := \mathbb{K}[X]/\langle X^d + 1\rangle \cong \mathbb{K} \otimes_{\mathbb{F}_q} R_q\,. \tag{13}$$

Since $\mathbb{Z}_q \subseteq \mathbb{K}$, the base ring $R_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$ is a subring of $\mathbb{E}$. Hence, an element $e \in \mathbb{E}$ can be viewed as a polynomial of degree less than d with coefficients in $\mathbb{K}$, i.e., $e(X) = \sum_{i=1}^{d} e_i X^{i-1}$ for $(e_1, \ldots, e_d) \in \mathbb{K}^d$. $\mathbb{E}$ supports three algebraic operations below:

- Multiplication by $\mathbb{K}$ (used in sumcheck): For a scalar $a \in \mathbb{K}$ and $e \in \mathbb{E}$, $a \cdot e \in \mathbb{E}$ is a standard scalar multiplication applied to e's coefficients, that is, $a \cdot e$ has coefficients $(a \cdot e_1, \ldots, a \cdot e_d) \in \mathbb{K}^d$.
- Multiplication by R_q (used in folding with low-norm challenges): For a polynomial $b(X) \in R_q$ and $e \in \mathbb{E}$, $b \cdot e \in \mathbb{E}$ is simply a polynomial multiplication modulo $X^d + 1$ over $\mathbb{E}$ (by canonically lifting $b(X)$ to an $\mathbb{E}$-element).
- Multiplication between R_q and $\mathbb{K}$: For a scalar $a \in \mathbb{K}$ and $b(X) \in R_q$, $a \cdot b(X) \in \mathbb{E}$ is simply a special case of the above operations. We can view it as lifting $b(X)$ to $\mathbb{E}$ and applying a scalar multiplication, or lifting a to a constant in $\mathbb{E}$ and applying a polynomial multiplication over $\mathbb{E}$.

2.4 Generalized Reduction of Knowledge

We provide a generalized definition of reduction of knowledge (RoK) [KP23] that allows for both *imperfect* completeness and *relaxed* knowledge soundness. That is, an honest prover may fail with negligible probability, and extracted witnesses have a less stringent validity requirement. This definition remains useful for applications involving a small number of recursive folding steps.

Definition 2.3 (Generalized Reduction of Knowledge)**.** *Let $\mathcal{R}_1$, $\mathcal{R}'_1$, $\mathcal{R}_2$, $\mathcal{R}'_2$ be indexed relations where $\mathcal{R}_1 \subseteq \mathcal{R}'_1$ and $\mathcal{R}_2 \subseteq \mathcal{R}'_2$. A reduction of knowledge Π from $\mathcal{R}_1$ to $\mathcal{R}_2$, with relaxed input and output relations $\mathcal{R}'_1$, $\mathcal{R}'_2$, 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 adversaries $\mathcal{A}$,*

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

By default, we assume $\epsilon(\cdot)$ is negligible in λ.

Definition 2.5 (κ-Knowledge Soundness w.r.t. $\mathcal{R}'_1$, $\mathcal{R}'_2$)**.** *There exists a function $\kappa(\cdot)$ such that for all expected polynomial time adversaries P^*, there is an expected polynomial time extractor Ext, given $\mathbb{i} \leftarrow \mathcal{G}(1^\lambda)$, $(\mathbb{x}_1, \mathsf{st}) \leftarrow \mathsf{P}^*(\mathbb{i})$,*

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

When $\mathcal{R}'_2 = \mathcal{R}_2$, we say it satisfies κ-knowledge soundness with relaxed input relation $\mathcal{R}'_1$; when $\mathcal{R}'_1 = \mathcal{R}_1$ and $\mathcal{R}'_2 = \mathcal{R}_2$, we simply say it satisfies κ-knowledge soundness.

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

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

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

We note that generalized RoKs are sequentially composable, with an almost identical security proof for Thereom 5 of [KP23].

Proposition 2.1. *Given a generalized reduction of knowledge Π_1 from $\mathcal{R}_1$ to $\mathcal{R}_2$ with relaxed relations $\mathcal{R}'_1$, $\mathcal{R}'_2$, and a generalized reduction of knowledge Π_2 from $\mathcal{R}_2$ to $\mathcal{R}_3$ with relaxed relations $\mathcal{R}'_2$, $\mathcal{R}'_3$, the composed procotol $\Pi_2 \circ \Pi_1$ is a reduction of knowledge from $\mathcal{R}_1$ to $\mathcal{R}_3$ with relaxed relations $\mathcal{R}'_1$, $\mathcal{R}'_3$.*

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

$$\mathsf{ts}(\mathbf{r}) := (eq_b(\mathbf{r}))_{b \in \{0,1\}^k} \in \mathbb{K}^{2^k} \tag{14}$$

where $eq_b(\mathbf{r}) := \prod_{i \in [k]} \big[(1 - b_i)(1 - \mathbf{r}_i) + b_i \mathbf{r}_i\big]$. For a multilinear polynomial[6] $f(\mathbf{X}) \in \mathbb{K}^{\leq 1}[X_1, \ldots, X_{\log(n)}]$, we denote by its evaluation vector $\mathbf{f}$ as

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

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

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 $g(\mathbf{X}) \in \mathbb{K}^{\leq D}[X_1, \ldots, X_{\log(n)}]$ denote a multivariate polynomial over $\mathbb{K}$ of individual degree at most D. The classical sumcheck protocol [LFKN92] can be formulated as a special reduction of knowledge from

$$\mathcal{R}_{\mathsf{sum}} := \left\{ (\mathbb{x}, \mathbb{w}) \;:\; \begin{array}{l} \mathbb{x} = (g(\mathbf{X}),\, v \in \mathbb{K}), \mathbb{w} = \bot \text{ s.t.} \\ \sum_{b \in \{0,1\}^{\log(n)}} g(b) = v \end{array} \right\}. \tag{15}$$

to the relation

$$\mathcal{R}_{\mathsf{eval}} := \left\{ (\mathbb{x}, \mathbb{w}) \;:\; \begin{array}{l} \mathbb{x} = (g(\mathbf{X}),\, \mathbf{r} \in \mathbb{K}^{\log(n)}, v \in \mathbb{K}), \mathbb{w} = \bot \text{ s.t.} \\ g(\mathbf{r}) = v \end{array} \right\}. \tag{16}$$

The protocol has linear-time prover and polylogarithmic verifier. The soundness error is $\epsilon_{\mathsf{sum}} := D \log(n)/|\mathbb{K}|$ by the Schwartz-Zippel lemma.

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 $f_1, \ldots, f_k$. Let $(\mathbf{f}_i)_{i=1}^k$ denote the evaluation vectors of $f_1, \ldots, f_k$ respectively. The claim $g(\mathbf{r}) = v$ is equivalent to

$$\big(\forall i \in [k] \,:\, \langle \mathbf{f}_i, \mathsf{ts}(\mathbf{r}) \rangle = u_i\big) \;\wedge\; h(u_1, \ldots, u_k) = v \,. \tag{17}$$

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

$$\mathcal{R}'_{\mathsf{eval}} := \left\{ (\mathbb{x}, \mathbb{w}) \;:\; \begin{array}{l} \mathbb{x} = ((\mathbf{f}_i)_{i=1}^k,\, \mathbf{r} \in \mathbb{K}^{\log(n)}, (u_i \in \mathbb{K})_{i=1}^k), \mathbb{w} = \bot \text{ s.t.} \\ \forall i \in [k] \,:\, \langle \mathbf{f}_i, \mathsf{ts}(\mathbf{r}) \rangle = u_i \end{array} \right\}. \tag{18}$$

Sumcheck batching. Given k sumcheck claims over $\mathbb{K}$ w.r.t. polynomials $g_1, \ldots, g_k \in \mathbb{K}^{\leq D}[X_1, \ldots, X_{\log(n)}]$, we can reduce them to a single sumcheck claim for $\sum_{i=1}^k g_i \cdot \alpha^{i-1} \in \mathbb{K}^{\leq D}[X_1, \ldots, X_{\log(n)}]$, where $\alpha \xleftarrow{\mathrm{R}} \mathbb{K}$ 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* $\mathcal{A}$*,*

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

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 recall commit-and-prove SNARKs (CP-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. For PolyIOP-based SNARKs [Chi+20; BFS20] that rely on polynomial commitments [KZG10], their CP-SNARK versions have similar efficiency as standard SNARKs.

Definition 2.8 (CP-SNARKs [Kil89; CLOS02; CFQ19])**.** *Let* $\mathsf{GenR}(1^\lambda) \to (\mathcal{R}, \mathbb{i})$ *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* $\mathsf{pp}_{\mathsf{cm}} \leftarrow \mathsf{Setup}(1^\lambda)$*.*
- *Output* $\mathbb{i}' := (\mathsf{pp}_{\mathsf{cm}}, \mathbb{i})$ *and relation*

$$\mathcal{R}' := \left\{\begin{array}{l}\mathbb{i}' = (\mathsf{pp}_{\mathsf{cm}}, \mathbb{i}), \\ \mathbb{x}' := (\mathbb{x}, (c_i)_{i=1}^\ell), \\ \mathbb{w}' = ((\mathbb{w}_i, o_i)_{i=1}^\ell, \mathbb{w}^*)\end{array} : \begin{array}{l}(\mathbb{i}, \mathbb{x}, \mathbb{w} = (\mathbb{w}_1, \ldots, \mathbb{w}_\ell, \mathbb{w}^*)) \in \mathcal{R} \wedge \\ \forall i \in [\ell] : \\ \mathsf{CM.RVfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c_i, \mathbb{w}_i, o_i) = 1\end{array}\right\}. \quad (19)$$

A Commit-and-Prove SNARK (CP-SNARK) for CM *and* GenR *is a SNARK w.r.t.* GenR_c*.*

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:

$$\mathfrak{i} := (\mathsf{pp}_{\mathsf{cm}}, R_q, \mathbb{K}, \mathbb{E}) \tag{20}$$

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

3.1 Generic Committed Linear Relation

We introduce **generic committed linear relations**, which serve as the anchor between our folding scheme and a SNARK. Looking forward, the folding scheme reduces multiple statements, including the R1CS and range statements for the committed witnesses, into two efficiently provable claims in this linear relation.

The relation is parameterized by

$$\mathsf{aux} := (n_{\mathsf{in}} \in \mathbb{N}, (\mathbf{M}_i \in \mathbb{Z}_q^{m_i \times n})_{i=1}^{k_x}),\ M := \max_{i \in [k_x]}\{m_i\}$$

where n_{in} is the public input length, the matrices $(\mathbf{M}_i)_{i \in [k_x]}$ is a set of linear transformations to the witness, $(m_i)_{i=1}^{k_x}$ are powers of two and $n \geq n_{\mathsf{in}}$ is the witness length. The generic committed linear relation is defined as

$$\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = (c \in \mathbb{C},\ \mathbf{x} \in R_q^{n_{\mathsf{in}}},\ \mathbf{r} \in \mathbb{K}^{\log M},\ \mathbf{v} \in \mathbb{E}^{k_x}) \\ \mathbb{w} = \mathbf{f} \in R_q^n \quad \text{s.t.} \\ \mathbf{x} = \mathbf{f}[1..n_{\mathsf{in}}] \quad \wedge \quad \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1 \\ \wedge\ \forall i \in [k_x] \ : \ \langle \mathsf{ts}(\mathbf{r}[1..\log m_i]),\ \mathbf{M}_i \mathbf{f} \rangle = \mathbf{v}_i \end{array} \right\}. \tag{21}$$

In words, the witness $\mathbf{f}$ is a valid opening to the commitment c and matches the public input $\mathbf{x}$. For $i \in [k_x]$, $\mathbf{v}_i$ is the multilinear evaluation of $\mathbf{M}_i\mathbf{f}$ at the first $\log m_i$ coordinates of $\mathbf{r}$, using the tensor in Eq. (14).

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+[BC25b]. Fix parameter $\mathsf{aux} := (n_{\mathsf{in}} = 0, (\mathbf{M}_i \in \mathbb{Z}_q^{m \times n})_{i=1}^{3})$ where m is a power of two. The input relation is

$$\mathcal{R}_{\mathsf{had}}^{\mathsf{aux}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = c \in \mathbb{C}, \quad \mathbb{w} = \mathbf{F} \in \mathbb{Z}_q^{n \times d} \quad \text{s.t.} \\ (\mathbf{M}_1\mathbf{F}) \circ (\mathbf{M}_2\mathbf{F}) = \mathbf{M}_3\mathbf{F} \ \wedge \ \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathsf{cf}^{-1}(\mathbf{F})) = 1 \end{array} \right\}, \tag{22}$$

where $\circ$ denotes element-wise multiplication. The output relation is[7]

$$\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \begin{array}{l} \mathbb{x} = (c \in \mathbb{C},\, \mathbf{r} \in \mathbb{K}^{\log m}, \mathbf{v} \in \mathbb{E}^3), \\ \mathbb{w} = \mathbf{f} \in R_q^n \quad \text{s.t.} \\ \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1 \ \wedge \ \forall i \in [3] : \langle (\mathbf{M}_i \mathbf{f}),\, \mathsf{ts}(\mathbf{r}) \rangle = \mathbf{v}_i \end{array} \right\}. \quad (23)$$

The reduction protocol Π_{had} is described in Figure 1. It first translates the Hadamard product relation into a sumcheck claim via *structural* random linear combinations, and then runs a sumcheck protocol to reduce the claim into linear evaluation statements over $\mathbb{K}$. Finally, the verifier packs evaluation values into $\mathbb{E}$-elements to match the committed witness.

Proposition 3.1. Π_{had} *in Figure 1 is an RoK from* $\mathcal{R}_{\mathsf{had}}^{\mathsf{aux}}$ *(Eq. (22)) to* $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$ *(Eq. (23)).*

Proof. The proposition follows from the claims below.

Claim 1. Π_{had} *is perfectly complete.*

Proof. If the input $(\mathbb{x} = c, \mathbb{w} = \mathbf{F}) \in \mathcal{R}_{\mathsf{had}}^{\mathsf{aux}}$, the sumcheck claim in Eq. (24) holds. By the completeness of the sumcheck RoK, the verifier's check in Eq. (25) passes, and the output is in $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$. □

Claim 2. Π_{had} *has knowledge error* $(d + \log(m))/|\mathbb{K}| + \epsilon_{\mathsf{sum}} + \epsilon_{\mathsf{bind}}$*, where* ϵ_{sum} *is the soundness error of the sumcheck protocol and* ϵ_{bind} *is the binding error.*

Proof. The extractor Ext works as follows.

- Simulate Π_{had} once with P^*. Denote by S the event that the verifier accepts and the output pair is in $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$. Abort if S doesn't happen.
- Let $\mathbb{w}_o = \mathbf{f}$ be the output witness. Return $\mathbf{F} = \mathsf{cf}(\mathbf{f})$.

The extractor runs in expected polynomial time since it performs one simulation and checks the output relation in polynomial time. Write $\epsilon = \Pr[S]$, over setup and all coins.

We define two mental experiments:

$\underline{\mathsf{Exp}_1}$:

1. Run extractor Ext. Save the full state σ before $(\mathbf{s}, \alpha)$ is sampled.
2. Conditioned on S, replay from state σ with fresh verifier coins until the prover succeeds again. Let $\hat{\mathbf{f}}$ be the output witness.

$\underline{\mathsf{Exp}_2}$:

1. Replay from state σ with fresh verifier coins until the prover succeeds. Let $\hat{\mathbf{f}}$ be the output witness.

[7] 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

$$\sum_{b \in \{0,1\}^{\log m}} \left(\sum_{j=1}^{d} \alpha^{j-1} \cdot f_j(b) \right) = 0 \tag{24}$$

   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

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

   where $\mathbf{r} \leftarrow\!\$\ \mathbb{K}^{\log m}$ is the sumcheck challenge and $v_{\mathsf{ev}} \in \mathbb{K}$ is the evaluation value sent by the prover P
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

$$\sum_{j=1}^{d} \alpha^{j-1} \cdot eq(\mathbf{s}, \mathbf{r}) \cdot (\mathbf{U}_{1,j} \cdot \mathbf{U}_{2,j} - \mathbf{U}_{3,j}) \stackrel{?}{=} v_{\mathsf{ev}}\,. \tag{25}$$

   Abort if it fails, otherwise output $\mathbb{x}_o := (c, \mathbf{r}, \mathbf{v} \in \mathbb{E}^3)$ where[a] for $i \in [3]$, we define $\mathbf{v}_i \in \mathbb{E}$ as the polynomial whose coefficients are $\mathbf{U}_{i,*}$.
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 $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$.

2. Run extractor Ext.

Note that Exp_1 runs in expected polynomial time. Let $p \geq 0$ be the probability that S occurs from state σ. If $p = 0$, we never enter Step 2. Otherwise, in expectation, Step 2 calls the prover at most $p \cdot (1/p) = 1$ times.

Let $\mathsf{Bad}'_{\mathsf{bind}}$ be the event that S occurs in Exp_1 but $\hat{\mathbf{f}} \neq \mathbf{f}$. By the binding property, we have that $\Pr[\mathsf{Bad}'_{\mathsf{bind}}] \leq \epsilon_{\mathsf{bind}}$. We similarly define $\mathsf{Bad}_{\mathsf{bind}}$ for experiment Exp_2. Importantly, for each prefix state σ with $p > 0$, conditioned on S, the bad event happens with exactly the same probability in two experiments. Thus $\Pr[\mathsf{Bad}_{\mathsf{bind}}] \leq \epsilon_{\mathsf{bind}}$.

Let $\mathsf{Bad}_{\mathsf{sum}}$ be the event that in Exp_2, (i) S occurs, (ii) $\mathbf{f} = \hat{\mathbf{f}}$, and (iii) $\hat{\mathbf{f}}$ doesn't satisfy Eq. (24). By sumcheck soundness, $\Pr[\mathsf{Bad}_{\mathsf{sum}}] \leq \epsilon_{\mathsf{sum}}$.

Let $\mathsf{Bad}_{\mathsf{had}}$ be the event that in Exp_2, (i) S occurs, (ii) $\mathbf{f} = \hat{\mathbf{f}}$, (iii) $\hat{\mathbf{f}}$ satisfies Eq. (24), and (iv) $\hat{\mathbf{f}}$ does not satisfy the Hadamard product check in $\mathcal{R}^{\mathsf{aux}}_{\mathsf{had}}$. (iv) implies that there is a column $j \in [d]$ where the Hadamard check fails for $\mathbf{F}_{*,j}$. By the Schwartz-Zippel Lemma, with probability at least $1 - (\log(m)/|\mathbb{K}|)$ (over $\mathbf{s}$ sampled in Step 1),

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

Conditioned on this, similarly, with probability at least $1 - (d/|\mathbb{K}|)$ (over α sampled in Step 1), $\hat{\mathbf{f}}$ does not satisfy Eq. (24). Thus, $\Pr[\mathsf{Bad}_{\mathsf{had}}] \leq (d + \log(m))/|\mathbb{K}|$.

Therefore,

$$\Pr\left[S \wedge (c, \mathsf{cf}(\mathbf{f})) \notin \mathcal{R}^{\mathsf{aux}}_{\mathsf{had}}\right] \leq \Pr\left[\mathsf{Bad}_{\mathsf{bind}}\right] + \Pr\left[\mathsf{Bad}_{\mathsf{sum}}\right] + \Pr\left[\mathsf{Bad}_{\mathsf{had}}\right] \leq \epsilon_{\mathsf{bind}} + \epsilon_{\mathsf{sum}} + (d + \log(m))/|\mathbb{K}|\,,$$

finishing the proof. □

□

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*[8] $(\mathbf{M}_i\mathbf{F})_{i=1}^3$*.*
- ***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+[BC25b] introduces an RoK for monomials. The monomial statement 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)). The RoK then reduces this claim to a linear statement. It is also a building block for the range proof in Section 3.4.

[8] It takes $O((m+n)d)$ $\mathbb{Z}_q$-multiplications when $\mathbf{M}_i$'s are sparse R1CS matrices.

Lemma 3.1 (Lemma 4.2 of [BC25b]). *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*

$$\mathcal{R}_{\mathsf{mon},k_g} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = (c^{(i)} \in \mathbb{C})_{i=1}^{k_g}, \quad \mathbb{w} = (\mathbf{g}^{(i)} \in R_q^n)_{i=1}^{k_g} \quad \text{s.t.} \\ \forall i \in [k_g] \ : \ \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c^{(i)}, \mathbf{g}^{(i)}) = 1 \ \wedge \mathbf{g}^{(i)} \in \mathcal{M}^n \end{array} \right\}, \tag{26}$$

to relation

$$\mathcal{R}_{\mathsf{batchlin},k_g} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = (\mathbf{r} \in \mathbb{K}^{\log n}, [c^{(i)} \in \mathbb{C}, \mathbf{u}^{(i)} \in \mathbb{E}]_{i=1}^{k_g}), \\ \mathbb{w} = (\mathbf{g}^{(i)} \in R_q^n)_{i=1}^{k_g} \quad \text{s.t.} \\ \forall i \in [k_g] \ : \ \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c^{(i)}, \mathbf{g}^{(i)}) = 1 \ \wedge \ \langle \mathbf{g}^{(i)}, \mathsf{ts}(\mathbf{r}) \rangle = \mathbf{u}^{(i)} \end{array} \right\}. \tag{27}$$

Here the multiplication between $\mathbf{g}^{(i)}$ (over R_q) and $\mathsf{ts}(\mathbf{r})$ (over $\mathbb{K}$) is defined in Section 2.3.

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.

We present an RoK protocol in Appendix A, which simplifies and generalizes the construction from [BC25b]. We refer to [BC25b] for a detailed security proof.

3.4 Approximate Range Proofs for Ring Vectors

We show a reduction of knowledge from the norm-check of ring vectors to generic linear statements. Combined with the folding protocol in Section 4, it helps efficiently reduce multiple norm-check statements into a *single* linear statement. 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. This is sufficient when 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

$$B_{d,k_g} := (d'/2) \cdot (1 + d' + \cdots + d'^{k_g - 1}) \geq 9.5B \,. \tag{28}$$

Define input relation

$$\mathcal{R}_{\mathsf{rg}}^{\ell_h, B} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = c \in \mathbb{C} \quad \mathbb{w} = \mathbf{f} \in R_q^n \quad \text{s.t.} \\ \mathsf{VfyOpen}_{\ell_h, B}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1 \ (\text{Eq. (12)}) \end{array} \right\}. \tag{29}$$

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 $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_J} \times \mathcal{R}_{\mathsf{batchlin},k_g}$ where $\mathcal{R}_{\mathsf{batchlin},k_g}$ is defined in Eq. (27) and

$$\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_J} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = (c \in \mathbb{C}, \mathbf{r} \in \mathbb{K}^{\log m}, \mathbf{v} \in \mathbb{E}), \\ \mathbb{w} = \mathbf{f} \in R_q^n \quad \text{s.t.} \\ \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1 \ \wedge \ \langle (\mathbf{M}_J \mathbf{f}), \mathsf{ts}(\mathbf{r}) \rangle = \mathbf{v} \end{array} \right\}. \tag{30}$$

The reduction protocol Π_{rg} is described in Figure 2. We give a high-level overview below.

1. Step 1-3: The prover computes a projected matrix $\mathbf{H}$ from a small projection matrix $\mathbf{J}$, reducing the range-check task from $\mathsf{cf}(\mathbf{f})$ to $\mathbf{H}$. Towards this, the prover decomposes $\mathbf{H}$ into k_g parts $(\mathbf{H}^{(i)})_{i\in[k_g]}$ and encodes $\mathbf{H}^{(i)}$'s entries using monomials. Finally, the prover sends commitments $(c^{(i)})_i$ to these monomial encodings $(\mathbf{g}^{(i)})_i$.

2. Step 4-6: The parties run the monomial RoK from Lemma 3.1 to ensure that $(\mathbf{g}^{(i)})_i$ are monomials. To ensure every entry of $(\mathbf{H}^{(i)})_{i\in[k_g]}$ is in the range $(-d/2, d/2)$, the protocol checks that each entry matches the *constant term* of the product of the table polynomial $t(X)$ and the corresponding monomial. Instead of checking entries individually, the protocol reduces these equations into a single evaluation claim at a random point: at Step 6, the verifier checks that the random evaluations $(u^{(i)})_i$ for the monomials $(g^{(i)})_i$ are consistent with the value $\langle \mathsf{ts}(\mathbf{s}), \mathbf{H}^\top \mathsf{ts}(\mathbf{r})\rangle$.

3. Step 7-8: The prover outputs the witness vectors $\mathbf{f}$ and $(\mathbf{g}^{(i)})_i$; the verifier outputs the evaluation claims for $\mathbf{H}$ and $(\mathbf{g}^{(i)})_i$.

Remark 3.1. Our approximate range proofs require fewer monomial commitments than LatticeFold+[BC25b] and therefore avoid the added complexity of folding the so-called *double commitments*. Compared to LaBRADOR [BS23], we improve the verifier complexity from linear to polylogarithmic in n.

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

Proof. The theorem follows from the lemmas below. □

Lemma 3.2. Π_{rg} *is* ϵ*-complete.*

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

- P^* aborts in Step 2 with probability at most ϵ. This holds because the projected matrix $\|\mathbf{H}\|_\infty \le 9.5B \le B_{d,k_g}$ with probability $1-\epsilon$, by Eq. (28) and a union bound on Eq. (5).

- In Step 6, the verifier check $\sum_{i\in[k_g]} d'^{i-1} \cdot ut_1^{(i)} = \langle \mathsf{ts}(\mathbf{s}), \mathsf{cf}(v)\rangle$ passes because

  - for all $i \in [k_g]$, $u^{(i)} = \langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathsf{Exp}(\mathbf{h}^{(i)})\rangle \in \mathbb{E}$ by definition. Hence, By polynomial multiplication in $\mathbb{E}$, $u^{(i)} \cdot t(X)$ equals $\langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathsf{Exp}(\mathbf{h}^{(i)}) \cdot t(X)\rangle$. And the constant term of $u^{(i)} \cdot t(X)$ is $\langle \mathsf{ts}(\mathbf{r}||\mathbf{s}), \mathsf{ct}(\mathsf{Exp}(\mathbf{h}^{(i)}) \cdot t(X))\rangle$.
  - 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,

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

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. (28), $t(X) \in R_q$ in Eq. (3), and index $\mathbb{i}$ in Eq. (20)
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

$$\mathbb{x}_{\mathsf{bat}} = (\mathbf{r}' := (\mathbf{r}, \mathbf{s}) \in \mathbb{K}^{\log n}, [c^{(i)} \in \mathbb{C},\ u^{(i)} \in \mathbb{E}]_{i=1}^{k_g}) \,. \tag{31}$$

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
$$\mathbf{H} = \mathbf{H}^{(1)} + d'\mathbf{H}^{(2)} + \cdots + d'^{k_g-1}\mathbf{H}^{(k_g)} \tag{32}$$
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 $((c^{(i)})_{i=1}^{k_g}, (\mathbf{g}^{(i)})_{i=1}^{k_g})$
   - Let $\mathbf{r} \leftarrow\!\$\ \mathbb{K}^{\log(m)}$ denote the sumcheck challenges for the first $\log(m)$ rounds
   - Let $\mathbf{s} \leftarrow\!\$\ \mathbb{K}^{\log d}$ be the sumcheck challenges in the last $\log(d)$ rounds. 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
$$\left(\mathbb{x}_{\mathsf{bat}} = \left((\mathbf{r}, \mathbf{s}), [c^{(i)}, u^{(i)}]_{i=1}^{k_g}\right), (\mathbf{g}^{(i)})_{i=1}^{k_g}\right) \in \mathcal{R}_{\mathsf{batchlin},k_g} \,. \qquad \text{(Eq. (27))}$$
5. $\mathsf{P} \to \mathsf{V}$: Send $v \in \mathbb{E}$, defined as the polynomial with coefficient vector $\mathsf{cf}(v) = \mathbf{H}^\top \mathsf{ts}(\mathbf{r}) \in \mathbb{K}^d$.
6. V : For $i \in [k_g]$, extract $\left(ut_1^{(i)}, \ldots, ut_d^{(i)}\right) \in \mathbb{K}^d$ as the coefficients of $u^{(i)} \cdot t(X) \in \mathbb{E}$ over $\mathbb{K}$. Reject if $\sum_{i \in [k_g]} d'^{i-1} \cdot ut_1^{(i)} \neq \langle \mathsf{ts}(\mathbf{s}),\ \mathsf{cf}(v) \rangle$
7. V : Output $\mathbb{x}_o := (\mathbb{x}^*, \mathbb{x}_{\mathsf{bat}})$ where $\mathbb{x}^* := (c, \mathbf{r}, v \in \mathbb{E})$ and $\mathbb{x}_{\mathsf{bat}}$ is defined in Eq. (31)
8. 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 $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_J} \times \mathcal{R}_{\mathsf{batchlin},k_g}$.

– $\mathsf{cf}(v) = \mathbf{H}^\top \mathsf{ts}(\mathbf{r})$ by definition and $\mathbf{H} = \mathbf{H}^{(1)} + d'\mathbf{H}^{(2)} + \cdots + d'^{k_g-1}\mathbf{H}^{(k_g)}$ by Eq. (32).

In sum, the verifier check in Step 6 passes because

$$\sum_{i\in[k_g]} d'^{i-1} \cdot ut_1^{(i)} = \sum_{i\in[k_g]} d'^{i-1} \cdot \langle \mathsf{ts}(\mathbf{s}),\, \mathbf{H}^{(i)\top} \mathsf{ts}(\mathbf{r}) \rangle = \langle \mathsf{ts}(\mathbf{s}),\, \mathbf{H}^\top \mathsf{ts}(\mathbf{r}) \rangle = \langle \mathsf{ts}(\mathbf{s}),\, \mathsf{cf}(v) \rangle\,.$$

Next, we show that the output instance-witness pairs are valid:

- $((c, \mathbf{r}, v), \mathbf{f}) \in \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_J}$. Since c is a commitment to $\mathbf{f}$, it suffices to prove the linear statement $\langle (\mathbf{M}_J \mathbf{f}),\, \mathsf{ts}(\mathbf{r}) \rangle = v$, where $\mathbf{M}_J := \mathbf{I}_{n/\ell_h} \otimes \mathbf{J}$. By defining $\mathbf{H} := (\mathbf{I}_{n/\ell_h} \otimes \mathbf{J}) \times \mathsf{cf}(\mathbf{f})$, we simply need to check that $\mathbf{H}^\top \mathsf{ts}(\mathbf{r}) = \mathsf{cf}(v) \in \mathbb{K}^d$, which is the case by Step 5.
- $(\mathbb{x}_{\mathsf{bat}}, (\mathbf{g}^{(i)})_{i=1}^{k_g}) \in \mathcal{R}_{\mathsf{batchlin},k_g}$. This holds by assignments of $c^{(i)}$, $u^{(i)}$ in Step 3, 4.

□

Lemma 3.3. Π_{rg} *is knowledge sound w.r.t.* $\mathcal{R}_{\mathsf{rg}}^{\ell_h, B'}$.

Proof. The extractor Ext works as follows.

- Simulate Π_{rg} once with P^*. Denote by S the event that the verifier accepts and the output pair is in $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_J} \times \mathcal{R}_{\mathsf{batchlin},k_g}$. Abort if S doesn't happen.
- Parse $\mathbb{w}_o = (\mathbf{f}, (\mathbf{g}^{(i)})_{i=1}^{k_g})$ and return $\mathbf{f}$.

Ext runs in expected polynomial time, as the extractor performs one simulation and checks the output relation in polynomial time.

Claim 3. *Let* $\epsilon = \Pr[S]$. *With probability at least* $\epsilon - \kappa_{\mathsf{rg}}$, *the output satisfies* $\mathcal{R}_{\mathsf{rg}}^{\ell_h, B'}$, *where*

$$\kappa_{\mathsf{rg}} = \epsilon_{\mathsf{bind}} + \epsilon_{\mathsf{proj}} + \epsilon_{\mathsf{mon}} + (\log m + \log d)/|\mathbb{K}|\,.$$

Proof. Consider the following two mental experiments:
$\underline{\mathsf{Exp}_1}$:

1. Run extractor Ext. Save the state σ before the verifier coins are sampled. Save the state σ' right after Step 3, where helper commitments are fixed.
2. Conditioned on S, replay from state σ with fresh verifier coins until the prover succeeds. Let $\hat{\mathbb{w}}_o = (\hat{\mathbf{f}}, \cdot)$ be the output.
3. Conditioned on S, replay from state σ' with fresh verifier coins until the prover succeeds. Let $\bar{\mathbb{w}}_o = (\bar{\mathbf{f}}, (\bar{\mathbf{g}}^{(i)})_{i=1}^{k_g})$ be the output.

$\underline{\mathsf{Exp}_2}$:

1. Replay from state σ with fresh verifier coins until the prover succeeds. Let $\hat{\mathbb{w}}_o = (\hat{\mathbf{f}}, \cdot)$ be the output.
2. Partially run extractor Ext until finishing Step 3, where helper commitments are fixed. Let σ' be the state.
3. Replay from state σ' with fresh verifier coins until the prover succeeds. Let $\bar{\mathbb{w}}_o = (\bar{\mathbf{f}}, (\bar{\mathbf{g}}^{(i)})_{i=1}^{k_g})$ be the output.
4. Finish the execution of extractor Ext starting from σ'.

Exp_1 runs in expected polynomial time. Let $p \geq 0$ be the probability that S occurs from state σ. If $p = 0$, we never enter Step 2; otherwise Step 2 calls the prover at most $p \cdot (1/p) = 1$ times. Similarly, let $q \geq 0$ be the probability that S occurs from state σ'. With the same argument, Step 3 calls the prover at most $q \cdot (1/q) = 1$ times. Let $\mathsf{Bad}'_{\mathsf{bind}}$ be the event that Exp_1, obtains two different openings for a commitment. By the binding property,

$$\Pr[\mathsf{Bad}'_{\mathsf{bind}}] \leq \epsilon_{\mathsf{bind}} .$$

Similarly, we define $\mathsf{Bad}_{\mathsf{bind}}$ for Exp_2. Importantly, for σ, σ' with $p, q > 0$, conditioned on S, the bad event happens with the same probability in two experiments.

Let $\mathsf{Bad}_{\mathsf{proj}}$ be the event that in Exp_2, (i) S occurs, (ii) $\mathbf{f} = \hat{\mathbf{f}}$ and (iii)

$$\mathsf{VfyOpen}_{\ell_h, B'}(\mathsf{pp}_{\mathsf{cm}}, c, \hat{\mathbf{f}}) = 0 \text{ but } \left\| (\mathbf{I}_{n/\ell_h} \otimes \mathbf{J}) \times \mathsf{cf}(\hat{\mathbf{f}}) \right\|_\infty \leq B_{d,k_g} ,$$

where $\mathbf{J}$ is sampled in Step 1. By Eq. (6) of the random projection lemma (Lemma 2.2),

$$\Pr[\mathsf{Bad}_{\mathsf{proj}}] \leq \epsilon_{\mathsf{proj}} \approx m \cdot 2^{-128} .$$

Let $\mathsf{Bad}_{\mathsf{mon}}$ be the event that in Exp_2, (i) S occurs, (ii) $(\mathbf{g}^{(i)})_{i=1}^{k_g} = (\bar{\mathbf{g}}^{(i)})_{i=1}^{k_g}$, and (iii) $(\bar{\mathbf{g}}^{(i)})_{i=1}^{k_g}$ are not monomial vectors. By Lemma 3.1,

$$\Pr[\mathsf{Bad}_{\mathsf{mon}}] \leq \epsilon_{\mathsf{mon}} ,$$

where ϵ_{mon} is the soundness error of monomial check.

Let $\mathsf{Bad}_{\mathsf{rg}}$ be the event that in Exp_2, (i) S occurs, (ii) $\mathbf{f} = \hat{\mathbf{f}} = \bar{\mathbf{f}}$ and $(\mathbf{g}^{(i)})_{i=1}^{k_g} = (\bar{\mathbf{g}}^{(i)})_{i=1}^{k_g}$, (iii) $\neg\mathsf{Bad}_{\mathsf{proj}}$, (iv) $(\bar{\mathbf{g}}^{(i)})_{i=1}^{k_g}$ are monomial vectors, and (v) $(c, \bar{\mathbf{f}}) \notin \mathcal{R}_{\mathsf{rg}}^{\ell_h, B'}$ for the input commitment c. We show that

$$\Pr[\mathsf{Bad}_{\mathsf{rg}}] \leq (\log m + \log d)/|\mathbb{K}| .$$

Denote by $\mathbf{H} := (\mathbf{I}_{n/\ell_h} \otimes \mathbf{J}) \times \mathsf{cf}(\bar{\mathbf{f}})$. For $i \in [k_g]$, define $\mathbf{H}^{(i)} \in \mathbb{Z}_q^{m \times d}$ such that

$$\mathsf{flt}(\mathbf{H}^{(i)}) = \mathsf{ct}(\bar{\mathbf{g}}^{(i)} \cdot t(X)) \in [-d'/2, d'/2]^{md} . \tag{33}$$

Recall that if $\mathsf{Bad}_{\mathsf{rg}}$ occurs, then $\neg\mathsf{Bad}_{\mathsf{proj}}$ and $(c, \bar{\mathbf{f}}) \notin \mathcal{R}_{\mathsf{rg}}^{\ell_h, B'}$. So $\|\mathbf{H}\|_\infty > B_{d,k_g}$, thus $\mathbf{H}$ can't be the weighted sum of the small components, that is,

$$\mathbf{H} \neq \mathbf{H}^{(1)} + d'\mathbf{H}^{(2)} + \cdots + d'^{k_g - 1}\mathbf{H}^{(k_g)} . \tag{34}$$

On the other hand, if $\mathsf{Bad}_{\mathsf{rg}}$ occurs, the following conditions hold:

- $(\mathbb{x}_o, \bar{\mathbb{w}}_o)$ is a valid output, thus $u^{(i)} = \langle \mathsf{ts}(\mathbf{r}||\mathbf{s}),\, \bar{\mathbf{g}}^{(i)} \rangle$ for $i \in [k_g]$.
- Multiplying by $t(X)$ over $\mathbb{E}$, $u^{(i)} \cdot t(X)$ equals $\langle \mathsf{ts}(\mathbf{r}||\mathbf{s}),\, \bar{\mathbf{g}}^{(i)} \cdot t(X) \rangle$. Then taking the constant term of this polynomial over $\mathbb{K}$, we obtain $\langle \mathsf{ts}(\mathbf{r}||\mathbf{s}),\, \mathsf{ct}(\bar{\mathbf{g}}^{(i)} \cdot t(X)) \rangle$, 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 definition of $\mathsf{flt}(\mathbf{H}^{(i)})$ (Eq. (33)).
- $\langle \mathsf{ts}(\mathbf{s}),\, \mathsf{cf}(v) \rangle = \sum_{i \in [k_g]} d'^{i-1} \cdot ut_1^{(i)}$ by the check at Step 6.
- $(\mathbb{x}_o, \bar{\mathbb{w}}_o)$ is a valid output, thus $\mathsf{cf}(v) = \mathbf{H}^\top \mathsf{ts}(\mathbf{r})$.

From all above, it holds that

$$\begin{aligned}\langle \mathsf{ts}(\mathbf{s}),\, \mathbf{H}^\top \mathsf{ts}(\mathbf{r}) \rangle = \langle \mathsf{ts}(\mathbf{s}),\, \mathsf{cf}(v) \rangle &= \sum_{i=1}^{k_g} d'^{i-1} \cdot ut_1^{(i)} = \sum_{i=1}^{k_g} d'^{i-1} \cdot \langle \mathsf{ts}(\mathbf{s}),\, \mathbf{H}^{(i)\top} \mathsf{ts}(\mathbf{r}) \rangle \\ &= \langle \mathsf{ts}(\mathbf{s}),\, \left(\mathbf{H}^{(1)} + d'\mathbf{H}^{(2)} + \cdots d'^{k_g-1}\mathbf{H}^{(k_g)}\right)^\top \mathsf{ts}(\mathbf{r}) \rangle\,.\end{aligned}$$

However, since $\left(\bar{\mathbf{f}}, (\bar{\mathbf{g}}^{(i)})_{i=1}^{k_g}\right)$ are independent of randomness $\mathbf{r}$, $\mathbf{s}$ in Ext, by the Schwartz-Zippel lemma, this can happen only with probability $(\log m + \log d)/|\mathbb{K}|$.

Therefore,

$$\Pr[S \wedge (c, \mathbf{f}) \notin \mathcal{R}_{\mathsf{rg}}^{\ell_h, B'}] \leq \epsilon_{\mathsf{bind}} + \epsilon_{\mathsf{proj}} + \epsilon_{\mathsf{mon}} + (\log m + \log d)/|\mathbb{K}|\,,$$

finishing the proof. □

□

Proposition 3.3. Π_{rg} *in Figure 2 runs a single degree-3 sumcheck protocol over* $\mathbb{K} = \mathbb{F}_{q^t}$ *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 5), and* $T_p^{\mathsf{mon}}(k_g, n)$*.*
- ***Verifier cost*** $T_v^{\mathsf{rg}}(k_g, n)$***:*** $k_g t$ R_q*-ops,* $k_g + d$ $\mathbb{K}$*-ops (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 a generalized committed R1CS relation designed for real-world applications, particularly SIMD (Single Instruction Multiple Data) computations. Section 4.2 then describes an *interactive* folding protocol that compresses a large number (e.g. 1024) of generalized R1CS statements. Later in Section 5, we will provide a generic paradigm to transform these interactive folding protocols into *non-interactive* ones.

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] and is particularly suited for capturing SIMD computations.

Formally, setting auxiliary parameters

$$\mathsf{aux} := (n_{\mathsf{in}}, n_w, n := n_{\mathsf{in}} + n_w, \ell_h \in \mathbb{N}, B \in \mathbb{R}, (\mathbf{M}_i \in \mathbb{Z}_q^{m \times n})_{i=1}^3)\,. \tag{35}$$

The generalized R1CS relation is defined as follows:

$$\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}} := \left\{ (\mathbb{x}, \mathbb{w}) \;:\; \begin{array}{ll} \mathbb{x} = (c \in \mathbb{C}, \mathbf{X}_{\mathsf{in}} \in \mathbb{Z}_q^{n_{\mathsf{in}} \times d}), & \mathbb{w} = \mathbf{W} \in \mathbb{Z}_q^{n_w \times d} \quad \text{s.t.} \\ \mathbf{F}^\top := [\mathbf{X}_{\mathsf{in}}^\top, \mathbf{W}^\top] \in \mathbb{Z}_q^{d \times n} & \text{s.t.} \\ (\mathbf{M}_1 \times \mathbf{F}) \circ (\mathbf{M}_2 \times \mathbf{F}) = \mathbf{M}_3 \times \mathbf{F} & \\ \wedge\ \mathsf{VfyOpen}_{\ell_h, B}(\mathsf{pp}_{\mathsf{cm}}, c, \mathsf{cf}^{-1}(\mathbf{F})) = 1 & \end{array} \right\}. \tag{36}$$

We can view this as checking d R1CS statements over $\mathbb{Z}_q$ *in a 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

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

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. (36). It then proves the relation $\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}}$ where the parameter aux is defined as

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

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 $b/2$.

SIMD computations vs a single instance. The transformation above is particularly suited for capturing *SIMD* computations. To capture a *single* R1CS statement over $\mathbb{Z}_q$, we can adapt the transformation and embed the instance-witness pair into the first column of $\mathbf{X}_{\mathsf{in}}$ and $\mathbf{W}$. However, a more efficient approach is to use the encoding from Neo [NS25], which does not increase the number of rows in the witness matrix.

Define a $\mathbb{Z}_q$-module homomorphism $\phi : R_q \to \mathbb{Z}_q$ that maps a polynomial $v(X) = \sum_{i=0}^{d-1} v_i X^i \in R_q$ to $v(b) \in \mathbb{Z}_q$ for a small base $b \in \mathbb{Z}$. This allows us to transform an R1CS-like statement

$$\mathcal{R}_{\mathsf{r1cs}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = \bot, \quad \mathbb{w} \in \mathbb{Z}_q^n \quad \text{s.t.} \\ (\mathbf{M}_1 \times \mathbb{w}) \circ (\mathbf{M}_2 \times \mathbb{w}) = \mathbf{M}_3 \times \mathbb{w} \end{array} \right\} \tag{37}$$

into a committed relation where the witness $\mathbf{f} \in R_q^n$ is a *low-norm* vector satisfying $\phi(\mathbf{f}) := (\phi(\mathbf{f}_1), \ldots, \phi(\mathbf{f}_n)) = \mathbb{w}$ and $\|\mathbf{f}\|_\infty \leq b/2$:

$$\mathcal{R}_{\mathsf{r1cs}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = c \in \mathbb{C}, \quad \mathbb{w} = \mathbf{f} \in R_q^n \quad \text{s.t.} \\ (\mathbf{M}_1 \times \phi(\mathbf{f})) \circ (\mathbf{M}_2 \times \phi(\mathbf{f})) = \mathbf{M}_3 \times \phi(\mathbf{f}) \\ \wedge\ \mathsf{VfyOpen}(\mathsf{pp}_{\mathsf{cm}}, c, \mathbf{f}) = 1 \end{array} \right\}. \tag{38}$$

Importantly, $\mathbf{f}$ stays as a vector of length n rather than $\bar{n} = n \cdot k_{\mathsf{cs}}$. Neo [NS25] presents a sumcheck protocol to reduce this committed relation to the linear relation $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$ (as per Eq. (23)), with a logic similar to Π_{had} as in Figure 1. By substituting Π_{had} with Neo's sumcheck protocol, all folding protocols in our work can efficiently support single instance proofs without expanding witness matrices vertically.

4.2 The High-Arity Folding Scheme

We present a folding scheme that compresses $\ell_{\mathsf{np}} = \mathsf{poly}(\lambda)$ (generalized) R1CS statements into two committed linear 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. (35). We construct an RoK from $\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}}$ (Eq. (36)) to $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$ where $\mathcal{R}_{\mathsf{batchlin},k_g}$ is defined in Eq. (27). Intuitively, $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}}$ is a linear relation w.r.t. the input witness, and $\mathcal{R}_{\mathsf{batchlin},k_g}$ checks k_g linear statements w.r.t. the helper monomial vectors $(\mathbf{g}^{(i)})_i$ for range proof. 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).

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: $\mathbb{x} = (c \in \mathbb{C}, \mathbf{X}_{\mathsf{in}} \in \mathbb{Z}_q^{n_{\mathsf{in}}\times d})$, $\mathbb{w} = \mathbf{W} \in \mathbb{Z}_q^{n_w\times d}$
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. (24) 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

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

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 $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$.

Lemma 4.1. *Let* aux *be the R1CS parameter defined in Eq. (35). 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. (36)) to* $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$, *with the relaxed input relation* $\mathcal{R}^{\mathsf{aux}'}_{\mathsf{gr1cs}}$. *Here,* $\mathcal{R}_{\mathsf{batchlin},k_g}$ *is from Eq. (27) 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. $\mathcal{R}^{\mathsf{aux}'}_{\mathsf{gr1cs}}$.

Claim 4. Π_{gr1cs} *is* ϵ*-complete.*

Proof. 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. (23)) and $(((c, \bar{\mathbf{r}}, v), \mathbb{x}_{\mathsf{bat}}), \mathbb{w}_o)$ is in $\mathcal{R}^{\mathsf{aux}_J}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$ (Eq. (30) and Eq. (27)). This implies that the output is in $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$ as desired. □

Claim 5. Π_{gr1cs} *is knowledge sound w.r.t.* $\mathcal{R}^{\mathsf{aux}'}_{\mathsf{gr1cs}}$.

Proof. The extractor Ext works as follows.

- Simulate Π_{gr1cs} once with P^*. Denote by S the event that the verifier accepts and the output pair satisfies $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$. Abort if S doesn't occur.
- Read $\mathbf{f}$ from $\mathbb{w}_o$, parse $\mathsf{cf}(\mathbf{f})^\top = [\mathbf{X}^\top, \mathbf{W}^\top]$, and return $\mathbf{W}$.

The extractor runs in expected polynomial time since it performs one simulation and checks the output relation in polynomial time.

Conditioned on S, the output relation guarantees that $\mathbf{X} = \mathbf{X}_{\mathsf{in}}$. For each of the subprotocol, the extractor simulates the verifier messages as in Figure 3 while reusing the shared sumcheck challenges. If the prover succeeds, it provides the same $\mathbf{f}$ to both output relations. The proofs of Proposition 3.1 and Lemma 3.3 then bounds the probabilities $\Pr[S \wedge (c, \mathsf{cf}(\mathbf{f})) \notin \mathcal{R}^{\mathsf{aux}}_{\mathsf{had}}]$ and $\Pr[S \wedge (c, \mathbf{f}) \notin \mathcal{R}^{\ell_h, B'}_{\mathsf{rg}}]$ by κ_{had} and κ_{rg}, respectively. Thus, the knowledge error is at most $\kappa_{\mathsf{had}} + \kappa_{\mathsf{rg}}$. □

□

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. Naive implementation would run the sumcheck protocol $O(\ell_{\mathsf{np}})$ times. To improve efficiency, we merge the $2\ell_{\mathsf{np}}$ sumcheck instances into two via a random linear combination, and Π_{fold} only requires two parallel sumcheck executions.

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 (Output compression). Since the output monomial vectors $(\mathbf{g}^{*(i)})_{i=1}^{k_g}$ share the same linear relation $\mathcal{R}_{\mathsf{batchlin},1}$ (Eq. (27)), they can be folded into a single vector. This gives a reduction from $(\mathcal{R}^{\mathsf{aux}}_{\mathsf{gr1cs}})^{\ell_{\mathsf{np}}}$ to $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin},1}$. We omit this optimization for simplicity.

Remark 4.2 (Memory Efficiency). We describe a prover algorithm for Π_{fold} with roughly the same memory cost as running a single Π_{gr1cs} instance with witness size n. As a tradeoff, the prover takes $O(\log\log(n))$ 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 and ρ, the prover executes the sumcheck using the algorithm from Section 4 of [Baw+25]. For each of the $O(\log\log(n))$ 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. We refer to [Baw+25, Section 4.3] for more details.
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.* (35) *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* $(\mathcal{R}^{\mathsf{aux}}_{\mathsf{gr1cs}})^{\ell_{\mathsf{np}}}$ *(Eq.* (36)*) to* $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$ *(Eq.* (27)*), with the relaxed input relation* $(\mathcal{R}^{\mathsf{aux}'}_{\mathsf{gr1cs}})^{\ell_{\mathsf{np}}}$. *The completeness error is* $\ell_{\mathsf{np}} \cdot \epsilon$ *with* ϵ *being the completeness error of* Π_{gr1cs} *(Lemma 4.1).*

Remark 4.3. Theorem 4.1 extends to support relaxed output relation $\mathcal{R}^{\mathsf{aux}'_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}'_{\mathsf{batchlin},k_g}$, where the norm bounds in $\mathsf{aux}'_{\mathsf{cs}}$ and in $\mathcal{R}'_{\mathsf{batchlin},k_g}$ are larger than that in $\mathsf{aux}_{\mathsf{cs}}$ and $\mathcal{R}_{\mathsf{batchlin},k_g}$. The norm bounds B_{rbnd}, B_{bnd} for the input relation then have to be set following Eq. (50) using B from $\mathsf{aux}'_{\mathsf{cs}}$ rather than $\mathsf{aux}_{\mathsf{cs}}$.

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

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

$$\sum_{b\in\{0,1\}^{\log m}} \left(\sum_{\ell=1}^{\ell_{\mathsf{np}}} \rho^\ell \sum_{j=1}^{d} \alpha^{j-1} \cdot f^{\ell,j}(b)\right) = 0 \tag{45}$$

       where $f^{\ell,j}(\mathbf{X}) \in \mathbb{K}[X_1, \ldots, X_{\log m}]$ $(1 \le j \le d)$ 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 ρ being the random combiners
     - We emphasize that the random scalar ρ is sampled after the prover has sent the monomial helper commitments.

     The 1st sumcheck has $\log(m)$ rounds, the 2nd has $\log(n)$ rounds.

     The shared sumcheck challenge is

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

   - 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 $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux_{cs}}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$: 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 $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$: Part II.

Remark 4.4. 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. $(\mathcal{R}^{\mathsf{aux}'}_{\mathsf{gr1cs}})^{\ell_{\mathsf{np}}}$.

Completeness. By the ϵ-completeness of Π_{gr1cs} (Lemma 4.1) and the perfect completeness 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 $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$.
Next, we show that the folded statement is also in $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$:

- After linearly-combining the instances and the witnesses using $\boldsymbol{\beta}$, the linear relations under $\mathcal{R}^{\mathsf{aux}_{\mathsf{cs}}}_{\mathsf{lin}}$ and $\mathcal{R}_{\mathsf{batchlin},k_g}$ 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) $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$. 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

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

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 $\mathbb{w}_{o,\ell} := (\mathbf{f}^\ell, (\mathbf{h}^{i,\ell})_{i=1}^{k_g})$. Define $\hat{\mathcal{R}}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}}$, $\hat{\mathcal{R}}_{\mathsf{batchlin}}$ as the relaxed relations of $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}}$, $\mathcal{R}_{\mathsf{batchlin},k_g}$ 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 $\epsilon_{\mathsf{P}^*} - (\ell_{\mathsf{np}}/|\mathcal{S}|)$ (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 $(\hat{\mathcal{R}}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \hat{\mathcal{R}}_{\mathsf{batchlin}})_{\mathsf{np}}^{\ell}$ w.r.t. $(\mathbb{x}_{o,\ell})_{\ell=1}^{\ell_{\mathsf{np}}}$. By the same

[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.

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 $(\mathcal{R}^{\mathsf{aux}'}_{\mathsf{gr1cs}})^{\ell_{\mathsf{np}}}$ as required. □

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

In this section, we introduce a two-step compiler to make Π_{fold} non-interactive via the Fiat-Shamir transform. The compiler is a generic extension to the ideas behind Protostar [BC23] and lattice double-commitments [BS23; BC25b]. Notably, our scheme removes the need for the "outer commitments" to be linearly homomorphic. Moreover, unlike [Aar+24] that analyzes the *non-interactive version* of LaBRADOR using the sophisticated *predicate special soundness*, we provide a simpler security proof in the random oracle model.

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.

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

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.

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. (36)) to $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$ (Eq. (27)), 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 B.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]. The proof of Lemma B.2 is non-trivial and crucially uses the straightline extractability of Π_{cm}. We defer the full proof to Appendix B. □

Remark 5.1. Similar to Remark 4.3, Theorem 5.1 extends to support relaxed output relations whose norm bounds are slightly larger than that in $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$.

Remark 5.2. 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, we apply the Fiat-Shamir transform (Section 5) to our high-arity folding scheme Section 4 and combine this with a Commit-and-Prove-SNARK (Definition 2.8) to construct a succinct proof system in the random oracle model for batch-proving many R1CS statements.

[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).

Unlike folding-based IVCs and recursive 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 using hash-based (or pairing-based) SNARKs, the proof size is independent of the number of R1CS statements and grows poly-logarithmically with the *per-statement* witness size. Compared to monolithic CP-SNARKs, our approach leverages the folding scheme to handle the majority of the computational load, and the CP-SNARK is only used to prove a small recursive verifier 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.

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 $\mathcal{R}'$ and relaxed output relation $\mathcal{R}'_o$. 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 $\mathcal{R}_o$ (with relaxed relation $\mathcal{R}'_o$), 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 sub-linear size since commitments are succinct and $\sum_{i=1}^{\mathsf{rnd}+1} |r_i|$ is polylogarithmic in witness

size $|\mathbb{w}|$ by assumption. (ii) By succinctness of Π_{cp}, Π_{snark}, verifying π_{cp}, π takes time sublinear in the witness size (and linear in the public instance size).

Completeness. By the SNARK completeness of Π_{snark} and the completeness of Π_{rok}, with overwhelming probability, we have $\Pi_{\mathsf{snark}}.\mathsf{Vf}(\mathsf{vk}, \mathcal{R}_o, \mathbb{x}_o, \pi) = 1$ and $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_o$. 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 $(\mathbb{w}_{\mathsf{cp}}, \mathbb{w}_e)$. Denote the CP-SNARK instance as $\mathbb{x}_{\mathsf{cp}} := (\mathbb{x}, (r_i)_{i=1}^{\mathsf{rnd}+1}, (c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}, \mathbb{x}_o)$, 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 $\mathsf{Ext}^{\mathcal{A}}_{\mathsf{rok}}$ 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 $\mathsf{Ext}^{\mathcal{A}}_{\mathsf{rok}}$. Since $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}}, \Pi_{\mathsf{rok}}]$ is knowledge sound with relaxed input relation $\mathcal{R}'$ and relaxed output relation $\mathcal{R}'_o$, 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* $(\mathcal{R}^{\mathsf{aux}}_{\mathsf{gr1cs}})^{\ell_{\mathsf{np}}}$ *with relaxed relation* $(\mathcal{R}^{\mathsf{aux}'}_{\mathsf{gr1cs}})^{\ell_{\mathsf{np}}}$.

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

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. (28))
B	input witness norm bound, $0.5b\sqrt{\ell_h} \le B \le B_{d,k_g}/9.5$	2^{10}
B'	relaxed input norm bound, $16B_{d,k_g}/\sqrt{30} \le B' \le \beta_{\mathsf{SIS}}/(\sqrt{nd/\ell_h})$	353806

Table 1: Parameters and candidate instantiations

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. 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

[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.

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 $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$, is dominated by k_g inner products between $\mathcal{M}^n$ and $\mathbb{K}^n$ (Eq. (27)), 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 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 generation 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. However, the CP-SNARK verifier also takes this randomness as a public input, so its time complexity could be linear in the size of the random bits. We leave it as an open problem for further reducing this verification cost.

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 $n = n' \cdot \ell$ to 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 $\mathcal{R}_o := \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux_{cs}}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$) 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. (20). 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. (21). 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.)

For parameter aux, we write $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$ as

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

and we define relation

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

A statement $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}}$ can be reduced to ℓ statements in $\mathcal{R}_{\mathsf{split}}^{\mathsf{aux}}$, with some relaxed input relation $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}'}$, 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 $\mathbb{w}_i = \mathbf{f}_i$

Completeness is straightforward. For knowledge soundness, given a pair $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{split}}^{\mathsf{aux}}$ output by a prover who passes the verification, we can transform $(\mathbb{x}, \mathbb{w})$ into a pair in $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}'}$ where the norm bound B_{bnd} in aux' is $\sqrt{\ell}$ larger than that in aux.

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. (36)).

- 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 $\mathcal{R}_o := \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$, and define $\mathcal{R}_{\mathsf{split},o}$ 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 $\mathcal{R}_{\mathsf{split},o}$ using the splitting RoK above, with a relaxed input relation $\mathcal{R}'_o$.

- 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; BC25b] 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 a relation $\mathcal{R}_{\mathsf{dec}}$, which is almost identical to $\mathcal{R}_{\mathsf{split},o}$ 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 $\mathcal{R}_{\mathsf{dec}}$, 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.

Remark 8.1. During knowledge extraction, when we call the sub-extractor of the first-layer folding, the sub-extractor calls an adversary who only provides tuples in the relaxed relation $\mathcal{R}'_o$, whose parameter $\mathsf{aux}'_{\mathsf{cs}}$ has a larger norm bound. Hence, to guarantee knowledge soundness, the lattice parameters and performance of the 1st layer folding will be slightly worse compared to the framework that does high-arity folding only once.

Acknowledgments. The author acknowledges the use of AI tools for language revision. The simplification of earlier knowledge soundness proofs is also inspired by a conversation with AI. We thank Prof. Dan Boneh and anonymous reviewers for their valuable feedback and Dr. Srinath Setty for discussions about streaming-friendly sum-checks. We thank Brian Lawrence, Daniel Gulotta, Brian Liu, Jagdeep Sidhu, Alexander Frolov, Xiang Wang for discussions and for pointing out typos and minor mistakes. Thank Daniel Gulotta for noting a proof size optimization of Π_{rg}; thank an anonymous reviewer for noting a simplification of the tensor-of-ring framework; and thank Xiang Wang for noting a subtlety in an earlier version of knowledge soundness proofs. 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

[0xP26] 0xPARC. link. 2026.

[Aar+24] Marius A. Aardal, Diego F. Aranha, Katharina Boudgoust, Sebastian Kolby, and Akira Takahashi. "Aggregating Falcon Signatures with LaBRADOR". In: *CRYPTO 2024, Part I*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14920. LNCS. Springer, Cham, Aug. 2024, pp. 71–106. DOI: 10.1007/978-3-031-68376-3_3.

[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.

[BC25a] Dan Boneh and Binyi Chen. "LatticeFold: A Lattice-Based Folding Scheme and Its Applications to Succinct Proof Systems". In: *ASIACRYPT 2025, Part III*. Ed. by Goichiro Hanaoka and Bo-Yin Yang. Vol. 16247. LNCS. Springer, Singapore, Dec. 2025, pp. 330–362. DOI: 10.1007/978-981-95-5099-9_11.

[BC25b] Dan Boneh and Binyi Chen. "LatticeFold+: Faster, Simpler, Shorter Lattice-Based Folding for Succinct Proof Systems". In: *CRYPTO 2025, Part VII*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16006. LNCS. Springer, Cham, Aug. 2025, pp. 327–361. DOI: 10.1007/978-3-032-01907-3_11.

[BCFW25] Benedikt Bünz, Alessandro Chiesa, Giacomo Fenzi, and William Wang. *Linear-Time Accumulation Schemes*. Cryptology ePrint Archive, Report 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.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. "Transparent SNARKs from DARK Compilers". In: *EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Cham, May 2020, pp. 677–706. DOI: 10.1007/978-3-030-45721-1_24.

[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: *CRYPTO 2025, Part VII*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16006. LNCS. Springer, Cham, Aug. 2025, pp. 128–160. DOI: 10.1007/978-3-032-01907-3_5.

[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.

[Chi+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Cham, May 2020, pp. 738–768. DOI: 10.1007/978-3-030-45721-1_26.

[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.

[GLLO26] Albert Garreta, Helger Lipmaa, Urmas Luhaäär, and Michal Osadnik. "Cyclo: Lightweight Lattice-Based Folding via Partial Range Checks". In: *EUROCRYPT 2026, Part VII*. Ed. by Joan Daemen and Emmanuel Thomé. Vol. 16547. LNCS. Springer, Cham, May 2026, pp. 33–62. DOI: 10.1007/978-3-032-25336-1_2.

[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, Report 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: *CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. LNCS. Springer, Cham, Aug. 2025, pp. 3–26. DOI: 10.1007/978-3-032-01887-8_1.

[KS24a] Abhiram Kothapalli and Srinath Setty. *NeutronNova: Folding everything that reduces to zero-check*. Cryptology ePrint Archive, Report 2024/1606. 2024. URL: https://eprint.iacr.org/2024/1606.

[KS24b] Abhiram Kothapalli and Srinath T. V. Setty. "HyperNova: Recursive Arguments for Customizable Constraint Systems". In: *CRYPTO 2024, Part X*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14929. LNCS. Springer, Cham, Aug. 2024, pp. 345–379. DOI: 10.1007/978-3-031-68403-6_11.

[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.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. "Constant-Size Commitments to Polynomials and Their Applications". In: *ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. LNCS. Springer, Berlin, Heidelberg, Dec. 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11.

[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.

[NS26] Wilson Nguyen and Srinath Setty. *Neo and SuperNeo: Post-quantum folding with pay-per-bit costs over small fields*. Cryptology ePrint Archive, Paper 2026/242. 2026. URL: https://eprint.iacr.org/2026/242.

[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.

[Pro] ProveKit. link.

[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.

[Sys26] Syscoin. *Syscoin's Implementation and Applications of Symphony*. link. 2026.

[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 The Monomial RoK from LatticeFold+

We present a reduction of knowledge from the monomial relation $\mathcal{R}_{\mathsf{mon},k_g}$ (Eq. (26)) to the linear relation $\mathcal{R}_{\mathsf{batchlin},k_g}$ (Eq. (27)). It simplifies [BC25b, Construction 4.2] by removing the double commitments. We set the batching parameter k_g to 1 for brevity, but the protocol naturally extends to $k_g > 1$.

We first recall some useful notation: For $a = \sum_{i=1}^{d} a_i X^{i-1} \in R_q$, and a value $\beta \in \mathbb{K} = \mathbb{F}_{q^t}$, let $\hat{a}(X) \in \mathbb{Z}_q[X] \subseteq \mathbb{K}[X]$ be the canonical mapping from R_q to $\mathbb{K}[X]$. We define $\mathbf{ev}_a(\beta) := \sum_{i=1}^{d} a_i \beta^{i-1} \in \mathbb{K}$ as the evaluation of $\hat{a}(X)$ at $\beta \in \mathbb{K}$. More generally, for $\beta \in \mathbb{K}$ and $\mathbf{e}(X) \in \mathbb{E}$ with $\mathbb{K}$-coefficients $(\mathbf{e}_1, \ldots, \mathbf{e}_d)$, we define $\mathbf{ev}_{\mathbf{e}}(\beta) := \sum_{i=1}^{d} \mathbf{e}_i \cdot \beta^{i-1} \in \mathbb{K}$.

Before presenting the protocol, we recall a useful lemma from [BC25b], which states that a random test is sufficient to check whether a polynomial is a monomial.

Lemma A.1 (Corollary 4.1 [BC25b]). *Fix $\mathbb{K} = \mathbb{F}_{q^t}$ and a prime $q > 2$. For all $a \in \mathcal{M} \subseteq R_q$ and $\beta \in \mathbb{K}$, we have $\mathbf{ev}_a(\beta)^2 = \mathbf{ev}_a(\beta^2)$. And for every $a \in R_q$ not in $\mathcal{M}$,*

$$\Pr_{\beta \xleftarrow{R} \mathbb{K}} \left[\mathbf{ev}_a(\beta)^2 = \mathbf{ev}_a(\beta^2)\right] < 2d/|\mathbb{K}|\,.$$

From the above lemma, verifying that a committed vector $\mathbf{g} \in R_q^n$ contains only monomials reduces to checking the identity $\mathbf{ev}_{\mathbf{g}_i}(\beta)^2 = \mathbf{ev}_{\mathbf{g}_i}(\beta^2)$ for all $i \in [n]$ at a random point $\beta \in \mathbb{K}$. To do this efficiently, we convert these n equations into a single

sumcheck claim via a random linear combination. Running the sumcheck protocol then reduce this claim to a linear evaluation statement. We present the formal protocol below:

Input: $\mathbb{x} = c \in \mathbb{C}$ and $\mathbb{w} = \mathbf{g} \in R_q^n$ such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{mon},1}$
Output: $\mathbb{x}_o = (\mathbf{r} \in \mathbb{K}^{\log n}, c \in \mathbb{C}, \mathbf{u} \in \mathbb{E})$ and $\mathbb{w}_o = \mathbf{g} \in R_q^n$ such that $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_{\mathsf{batchlin},1}$

1. $\mathsf{V} \to \mathsf{P}$: Send challenges $\mathbf{c} \xleftarrow{R} \mathbb{K}^{\log n}$ and $\beta \xleftarrow{R} \mathbb{K}$.

2. $\mathsf{P} \leftrightarrow \mathsf{V}$: Define vectors

$$\mathbf{v}_{\mathbf{g},\beta} := (\mathbf{ev}_{\mathbf{g}_1}(\beta), \ldots, \mathbf{ev}_{\mathbf{g}_n}(\beta)) \in \mathbb{K}^n\,,$$
$$\mathbf{v}_{\mathbf{g},\beta^2} := (\mathbf{ev}_{\mathbf{g}_1}(\beta^2), \ldots, \mathbf{ev}_{\mathbf{g}_n}(\beta^2)) \in \mathbb{K}^n$$

   and let $\widetilde{\mathbf{v}_{\mathbf{g},\beta}}$, $\widetilde{\mathbf{v}_{\mathbf{g},\beta^2}}$ be their multilinear extensions. Run a sumcheck protocol for the claim

$$\sum_{\mathbf{b}\in\{0,1\}^{\log n}} eq(\mathbf{c},\mathbf{b}) \cdot \left(\widetilde{\mathbf{v}_{\mathbf{g},\beta}}(\mathbf{b})^2 - \widetilde{\mathbf{v}_{\mathbf{g},\beta^2}}(\mathbf{b})\right) = 0\,. \tag{59}$$

   Let $\mathbf{r} \xleftarrow{R} \mathbb{K}^{\log n}$ be the sumcheck challenge and $v \in \mathbb{K}$ be the claimed evaluation. The protocol reduces the sumcheck claim to an evaluation statement:

$$eq(\mathbf{c},\mathbf{r}) \cdot \left(\widetilde{\mathbf{v}_{\mathbf{g},\beta}}(\mathbf{r})^2 - \widetilde{\mathbf{v}_{\mathbf{g},\beta^2}}(\mathbf{r})\right) \stackrel{?}{=} v\,.$$

3. $\mathsf{P} \to \mathsf{V}$: Send $\mathbf{u} = \widetilde{\mathbf{g}}(\mathbf{r}) = \langle \mathbf{g}, \mathsf{ts}(\mathbf{r}) \rangle \in \mathbb{E}$

4. V : Check that:

$$eq(\mathbf{c},\mathbf{r}) \cdot \left(\mathbf{ev}_{\mathbf{u}}(\beta)^2 - \mathbf{ev}_{\mathbf{u}}(\beta^2)\right) \stackrel{?}{=} v\,, \tag{60}$$

   Abort and return $\perp$ if the check fails.

5. Return the reduced statement $(\mathbb{x}_o = (\mathbf{r}, c, \mathbf{u}), \mathbb{w} = \mathbf{g}) \in \mathcal{R}_{\mathsf{batchlin},1}$.

B 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 B.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*

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

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*

$$\epsilon_\Psi(\mathcal{A}) - (Q+1)\cdot \ell/|\mathcal{S}|, \tag{61}$$

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.

Next, we show that $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}},\Pi_{\mathsf{gr1cs}}]$ is a non-interactive RoK from the R1CS relation to the target linear relation. Here, it suffices to prove adaptive soundness as the input witness can be derived from the output witness. The key idea is to reduce the soundness of the non-interactive scheme to the round-by-round soundness of the interactive protocol. Specifically, we construct an adversary $\mathcal{A}'$ that simulates the interactive protocol by extracting prover messages from the random oracle queries made by adversary $\mathcal{A}$ for $\mathsf{FS}^{\mathsf{H}}[\Pi_{\mathsf{cm}},\Pi_{\mathsf{gr1cs}}]$.

Lemma B.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* $\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}}$ *(Eq. (36)) to* $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$*, with the relaxed input relation* $\mathcal{R}_{\mathsf{gr1cs}}^{\mathsf{aux}'}$*. Here,* $\mathcal{R}_{\mathsf{batchlin},k_g}$ *is from Eq. (27) 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) $\mathsf{tr} = [\mathbb{x}, (c_{\mathsf{fs},i})_{i=1}^{\mathsf{rnd}}, (r_i)_{i=1}^{\mathsf{rnd}+1}, (m_i)_{i=1}^{\mathsf{rnd}}]$ 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 $\mathcal{A}'$ further satisfies that

$$\mathsf{State}((\mathbb{x}, \mathbb{w}), \mathsf{tr}_i) = \mathsf{reject} \qquad \text{and} \qquad \mathsf{State}((\mathbb{x}, \mathbb{w}), \mathsf{tr}_{i+1}) = \mathsf{accept}\,, \tag{62}$$

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

Claim 6. $\epsilon' \leq Q \cdot (\epsilon_{\mathsf{rbr}} + \mathsf{negl}(\lambda))$.

Proof. WLOG, assume that $\mathcal{A}$ has queried H on input $(\mathbb{x}|r_1|c_{\mathsf{fs},1}|\ldots|r_i|c_{\mathsf{fs},i})$. Given index i, and the Q-query adversary $\mathcal{A}$, consider a mental experiment as follows:
$\underline{\mathsf{Exp}_1}$:

1. Sample a query index $q \leftarrow\!\!\$\ [Q]$.
2. Simulate the execution of $\mathcal{A}$ until the qth oracle query. Let σ be $\mathcal{A}$'s current state and the oracle table.
3. Continue the simulation of $\mathcal{A}$. Let S_q denote the event that $\mathcal{A}$ succeeds and the accepting transcript uses the q-th query input s for r_{i+1}. If S_q occurs, let $\mathbf{f}, (\mathbf{g}^{(i)})_{i=1}^{k_g}$ and $(m_j)_{j\in[i]}$ be the extracted vectors and prover messages.

4. Conditioned on S_q, keep replaying from the saved state σ with earlier answers fixed and sample fresh answers for query q onward, until S_q occurs again. Let $\bar{\mathbf{f}}, (\bar{\mathbf{g}}^{(i)})_{i=1}^{k_g}$ and and $(\bar{m}_j)_{j\in[i]}$ be the extracted vectors and prover messages.

5. Outputs 1 if and only if Eq. (62) holds w.r.t. $\mathsf{tr}_i := (\mathbb{x}|r_1|m_1'|r_2|m_2|\ldots|m_{i-1}|r_i)$ and $\mathsf{tr}_{i+1} := (\mathbb{x}|r_1|m_1'|r_2|m_2|\ldots|r_i|m_i|r_{i+1})$.

Note that Exp_1 runs in expected polynomial time. Let $p \geq 0$ be the probability that S_q occurs from state σ. If $p = 0$, we never enter Step 4; otherwise, Step 4 calls $\mathcal{A}$ for at most $p \cdot (1/p) = 1$ times in expectation. Thus, by the binding property,

$$\Pr\left[\mathbf{f} \neq \bar{\mathbf{f}} \vee (\mathbf{g}^{(i)})_{i=1}^{k_g} \neq (\bar{\mathbf{g}}^{(i)})_{i=1}^{k_g} \vee (m_j)_{j\in[i]} \neq (\bar{m}_j)_{j\in[i]}\right] \leq \epsilon_{\mathsf{bind}}\,.$$

Similar as in proofs of Claim 2 and Lemma 3.3, we define a similar experiment Exp_2 that alternates the order between Step 4 and Step 2-3, where Step 4 replays anyway regardless of S_q. Conditioned on S_q, the above probability is identical in two experiments.

We now consider experiment Exp_2. By definition of round-by-round soundness,

$$\Pr\left[\mathsf{Exp}_2 = 1\right] \leq \epsilon_{\mathsf{rbr}} + \epsilon_{\mathsf{bind}} = \epsilon_{\mathsf{rbr}} + \mathsf{negl}(\lambda)\,.$$

On the other hand, recall that ϵ' is is the probability that $\mathcal{A}'$'s output is valid and further satisfies Eq. (62). By the binding argument above, and because q is uniformly sampled,

$$(\epsilon' - \mathsf{negl}(\lambda))/Q - \mathsf{negl}(\lambda) \leq \Pr\left[\mathsf{Exp}_2 = 1\right].$$

Thus, $\epsilon_{\mathsf{rbr}} \geq \epsilon'/Q - \mathsf{negl}(\lambda)$, which implies that $\epsilon' \leq Q \cdot (\epsilon_{\mathsf{rbr}} + \mathsf{negl}(\lambda))$ 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 B.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 B.1 and Lemma B.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) $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}} \times \mathcal{R}_{\mathsf{batchlin},k_g}$. 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 B.1.

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

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

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

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

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

$$\mathbf{f}^\ell := (\mathbf{f}^{*,\ell} - \mathbf{f}^{*,0})/(\mathbf{u}_\ell[\ell] - \mathbf{u}_0[\ell]) \in R_q^n, \tag{63}$$

$$\forall i \in [k_g] \ :\ \mathbf{h}^{i,\ell} := (\mathbf{g}^{(i,\ell)} - \mathbf{g}^{(i,0)})/(\mathbf{u}_\ell[\ell] - \mathbf{u}_0[\ell]) \in R_q^n \tag{64}$$

   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 B.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 $\mathcal{R}_{\mathsf{lin}}^{\mathsf{aux}_{\mathsf{cs}}}$, $\mathcal{R}_{\mathsf{batchlin},k_g}$ 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 B.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.