LatticeFold+: Faster, Simpler, Shorter Lattice-Based Folding for Succinct Proof Systems

Dan Boneh and Binyi Chen

Stanford University

February 17, 2025

Abstract

Folding is a technique for building efficient succinct proof systems. Many existing folding protocols rely on the discrete-log based Pedersen commitment scheme, and are therefore not post-quantum secure and require a large (256-bit) field. Recently, Boneh and Chen constructed LatticeFold, a folding protocol using lattice-based commitments which is plausibly post-quantum secure and can operate with small (64-bit) fields. For knowledge soundness, LatticeFold requires the prover to provide a range proof on all the input witnesses using bit-decomposition, and this slows down the prover. In this work we present LatticeFold+, a very different lattice-based folding protocol that improves on LatticeFold in every respect: the prover is five to ten times faster, the verification circuit is simpler, and the folding proofs are shorter. To do so we develop two novel lattice techniques. First, we develop a new purely algebraic range proof which is much more efficient than the one in LatticeFold, and may be of independent interest. We further shrink the proof using double commitments (commitments of commitments). Second, we show how to fold statements about double commitments using a new sumcheck-based transformation.

Contents

1 Introduction **3**
1.1 Additional related work . . . 7

2 Preliminaries **7**
2.1 Cyclotomic rings . . . 8
2.2 Multilinear extensions and sumchecks over rings . . . 10
2.3 Module-based Ajtai commitments . . . 12
2.4 Reduction of Knowledge . . . 12

3 Generalized Committed Linear Relations **13**

4 A Toolbox of Reduction of Knowledge **14**
4.1 Linear commitments and double commitments . . . 15
4.2 Monomial set check . . . 16
4.3 Range check . . . 19
4.4 Commitment transformation . . . 24

5 Folding Generalized Committed Linear Relations **34**
5.1 Folding . . . 35
5.2 Decomposition . . . 37
5.3 Efficiency estimate . . . 38

6 Conclusion and future work **40**

A Reducing R1CS to the general linear relation **48**

1 Introduction

In recent years succinct non-interactive arguments of knowledge (SNARKs) have found many real-world applications: scaling and bridging blockchains [Whi18; Xie+22], authenticating media [NT16; DB22; KHSS22], machine learning [CWSK24; YCBC24], verifiable delay functions [BBBF18], and many others. To avoid the high memory requirements of some SNARK proof systems, provers in practice break the task of constructing a proof into small steps and prove each step separately. This approach is called incrementally verifiable computation (IVC) [Val08] or proof carrying data (PCD) [CT10]. It also provides additional opportunities for parallelizing the prover [Ngu+24].

The original IVC/PCD schemes were built using SNARK recursion [Val08; BCTV14], where the SNARK verifier is embedded in every computation step being proved. A more efficient approach, called *accumulation* or *folding*, was introduced in Halo [BGH19] and further developed in [BCMS20; Bün+21; BDFG21] and Nova [KST22]. Many elegant ideas have since further optimized and extended the folding paradigm [KS24b; BC23; EG23; BMNW24; Moh23; NBS23; FKNP24; KS24a; AS24].

Folding is best explained using the language of *reductions of knowledge* [KP23] (see Section 2.4 for details). Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two instance-witness relations. A reduction of knowledge from $\mathcal{R}_1$ to $\mathcal{R}_2$ is a protocol Π between a prover and verifier. The verifier takes as input an instance $\mathbb{x}_1$ for $\mathcal{R}_1$, interacts with the prover, and outputs an instance $\mathbb{x}_2$ for $\mathcal{R}_2$ at the end of the protocol. The key requirement is that if the prover can present a witness $\mathbb{w}_2$ for $\mathbb{x}_2$, then it is possible to extract from the prover a witnesses $\mathbb{w}_1$ for $\mathbb{x}_1$. Hence, knowledge of a valid witness for $\mathbb{x}_2$ proves knowledge of a valid witness for $\mathbb{x}_1$.

A folding scheme is a reduction of knowledge from some product relation $\mathcal{R}_{\mathsf{acc}} \times \mathcal{R}_{\mathsf{comp}}$ to $\mathcal{R}_{\mathsf{acc}}$. That is, the folding scheme can fold a pair of instances $(\mathbb{x}_{\mathsf{acc}}, \mathbb{x}_{\mathsf{comp}})$ into a single new instance $\mathbb{x}'_{\mathsf{acc}}$ of $\mathcal{R}_{\mathsf{acc}}$. By repeatedly folding in this way, the prover can accumulate many steps of a computation expressed by $\mathcal{R}_{\mathsf{comp}}$ into a single instance of the accumulation relation $\mathcal{R}_{\mathsf{acc}}$. Eventually, the prover proves knowledge of a witness for the final $\mathcal{R}_{\mathsf{acc}}$ instance, and this proves knowledge of a valid witness for every step of the computation. Alternatively, folding can take place along a d-ary tree, with d-to-1 foldings, as in [Ngu+24; RZ22]. The benefit of folding schemes is that now the statement being proved at every step only needs to ensure that folding was performed correctly at the previous step, which is far simpler than a full SNARK verifier. Folding is also much faster for the prover because folding two witnesses into one is primarily a random linear combination of the two witnesses.

Folding schemes typically extend the input relation to include a short commitment to the witness. The commitment scheme is linearly homomorphic, so that the verifier can compute the folded instance as a a random linear combination of these input commitments[1]. Often one uses Pedersen as the homomorphic commitment scheme. This poses three dif-

[1]The ARC folding scheme [BMNW24] is one example that does not use a linearly homomorphic commitment.

ficulties: first, the resulting schemes are unsound in a post-quantum setting; second, the Pedersen hash requires the prover to do a compute-intensive multiscalar multiplication over a large (256-bit) field; and third, the domain and range of the Pedersen hash are different groups, requiring arithmetic over two distinct fields to verify a folding proof. This can be expensive because arithmetic circuits natively support only one field.

Lattice-based folding. Boneh and Chen [BC24] recently proposed a lattice-based folding scheme, called LatticeFold, where the Pedersen commitment is replaced with an Ajtai commitment over modules. This resolves all three difficulties mentioned in the previous paragraph[2]. However, using Ajtai commitments in folding introduces significant challenges. The difficulty is that Ajtai commitments are binding only when the input vector is low norm. This is incompatible with the random linear combination approach used in folding. Repeatedly taking a random linear combination of witnesses quickly increases the norm of the accumulated witness to the point where Ajtai commitments are no longer binding. This breaks soundness of the resulting folding scheme.

To address this issue, folding in LatticeFold is done in two primary steps. First, a norm reduction step decomposes each of the two input witnesses into d lower norm witnesses. Second, the resulting $2d$ instances are folded into a single low norm instance. This ensures that after repeated folding, the norm of the accumulated witness remains small. However, this by itself is insufficient. One must also exhibit an extractor that extracts from the prover low norm witnesses for the two input instances. LatticeFold ensures that the extracted witnesses are low norm by using a novel sumcheck-based range proof. Forcing the prover to output this range proof for the two input witnesses is sufficient to ensure that the extracted witnesses are valid and low norm. However, the range proof in LatticeFold uses bit decomposition and is therefore expensive to construct because it requires the prover to commit to many decomposed witnesses.

Experiments by Nethermind [Net24] show that this additional complexity in folding using Ajtai commitments results in a LatticeFold prover that runs in approximately the same time as the Pedersen-based HyperNova folding prover [KS24b]. While LatticeFold has the benefit of post-quantum security, one would further expect that Ajtai-based folding would be faster than Pedersen-based folding.

Our results. In this work we design a new lattice-based folding scheme called LatticeFold+ that greatly improves the performance of LatticeFold using a number of new lattice-ring techniques.

As in HyperNova, LatticeFold+ represents the computation relation $\mathcal{R}_{\mathsf{comp}}$ as either a rank-1 constraint system (R1CS), using quadratic constraints, or a customizable constraint system (CCS) [STW23] using higher degree constraints. The first step in LatticeFold+ is

[2]Experiments by Nethermind [Net24] show that the Ajtai hash is about thirty times faster than a Pedersen hash for committing to $\approx 2^{19}$ field elements.

a sumcheck-based linearization process that is a reduction of knowledge from an R1CS or CCS relation $\mathcal{R}_{\mathsf{comp}}$ to a general linear relation we call $\mathcal{R}_{\mathsf{lin},B}$, as explained in Section 3. This step is similar to HyperNova's linearization step, but adapted to operate over rings as was done in LatticeFold [BC24]. This linearization step shows that it suffices to design a folding scheme that folds $L > 2$ instances of the linear relation $\mathcal{R}_{\mathsf{lin},B}$ into two instances.

An instance of $\mathcal{R}_{\mathsf{lin},B}$ is a triple $(\mathsf{cm}, \mathbf{r}, \mathbf{v})$ where cm is an Ajtai commitment to a witness vector $\mathbf{f} \in R_q^n$, where the ℓ_∞ norm of $\mathbf{f}$ satisfies $\|\mathbf{f}\|_\infty < B$ (as defined in the next section). The elements $\mathbf{r}$ and $\mathbf{v}$ will be explained in Section 3. Here the witness $\mathbf{f}$ is an n-vector of elements in the ring $R_q := \mathbb{Z}_q[X]/\langle X^d + 1\rangle$ for some prime q and degree $d > 1$.

The core of LatticeFold+ takes as input $L > 2$ instances of $\mathcal{R}_{\mathsf{lin},B}$ and folds them into two instances of $\mathcal{R}_{\mathsf{lin},B}$. It does so in two steps. In the first step, it folds the given L instances of $\mathcal{R}_{\mathsf{lin},B}$ into one instance of $\mathcal{R}_{\mathsf{lin},B^2}$, namely an instance with a norm bound B^2 on the witness. The new range proof in this step is purely algebraic — it does not require commitments to bit-decomposed vectors. The increase in norm from B to B^2 has to be corrected, and LatticeFold+ does this in the second step. It uses witness decomposition to reduce a linear instance with norm bound B^2 to two linear instances with norm bound B by splitting every scalar in the witness into its high and low order bits. This second step is approximately the same as decomposition in LatticeFold. The key new insights are in the first folding step, which we discuss next.

A new range proof. To ensure that an extractor is able to extract L low norm witnesses from a folding prover, the prover must provide a range proof that all the input witnesses are low norm. We develop a new range proof, presented in Section 4.3, that avoids bit decomposition and is far more efficient than the range proof in LatticeFold. This range proof method may be of independent interest. The key idea is as follows. Say we want to prove that a committed vector $\mathbf{f} = (f_1, \ldots, f_n) \in \mathbb{Z}_q^n$ satisfies $f_i \in (-d/2, d/2)$ for all $i \in [n]$. The prover will commit to a vector of ring elements $\mathbf{m} = (m_1, \ldots, m_n) \in R_q^n$ where each $m_i \in R_q$ is the monomial $m_i := X^{f_i}$. We denote this commitment by $[\![\mathbf{m}]\!]$ and note that computing the Ajtai commitment to $\mathbf{m}$ is fast because it is a vector of monomials (see Remark 4.3). Next we show in Lemma 2.2 that if m_i is a monomial and m_i and f_i satisfy a simple algebraic relation then $f_i \in (-d/2, d/2)$. Hence, our range proof needs to (i) prove that $[\![\mathbf{m}]\!]$ is a commitment to a vector of monomials, which we do in Section 4.2, and (ii) prove that $\mathbf{m}$ and $\mathbf{f}$ satisfy a certain algebraic relation, which we do in Section 4.3. Both steps are presented as reductions of knowledge that reduce these checks to checking a simple evaluation relation. The resulting range proof requires no commitments to bit-decomposed vectors, and is significantly faster than the LatticeFold range proof.

Double commitments. When applying this range proof to a vector of ring elements $\mathbf{f} \in R_q^n$, we could repeat the range proof above d times: once for the vector of all constant terms in $\mathbf{f}$, once for the vector of all linear terms in $\mathbf{f}$, and so on. This would require the prover to commit to d vectors $\mathbf{m}_0, \ldots, \mathbf{m}_{d-1} \in R_q^n$ which would result in a large transcript of d commitments $\mathbf{c} := ([\![\mathbf{m}_0]\!], \ldots, [\![\mathbf{m}_{d-1}]\!]) \in R_q^{\kappa \times d}$ that must be sent to the verifier. Instead, we introduce in Section 4.1 the concept of a *double commitment*, where the prover sends to the verifier a commitment to the vector of commitments $\mathbf{c}$. This vector $\mathbf{c}$ is not low norm, so we must first decompose it to reduce its norm and then commit to the decomposition using an Ajtai commitment. We use $(\!|\mathbf{M}|\!)$ to denote this double commitment to the matrix $\mathbf{M} := [\mathbf{m}_0, \ldots, \mathbf{m}_{d-1}] \in R_q^{n \times d}$. Note that $(\!|\mathbf{M}|\!) \in R_q^{\kappa}$ is short as required. These double commitments are crucial for making our folding proofs shorter and simpler to verify compared to LatticeFold. In Section 4.3 we present a range proof that uses double commitments to prove that a vector $\mathbf{f} \in R_q^n$ has low ℓ_∞ norm.

Commitment transformation. Double commitments are not linearly homomorphic, and as a result statements that use double commitments are hard to fold using random linear combinations. In Section 4.4 we develop a technique that transforms such statements to ones involving only *linear* commitments to smaller witnesses. We use sumchecks to ensure consistency between the double commitment of a witness $\mathbf{M}$ and a linear commitment to a transformed version of $\mathbf{M}$. Again, this technique may be of independent interest.

Evaluation. LatticeFold+ improves over LatticeFold in every respect: the folding prover is faster, the folding verifier is simpler, and the folding proof is shorter. We give a detailed evaluation analysis in Section 5.3. Here we present a brief summary when folding L witnesses each of dimension n and norm less than B:

- Prover complexity: In LatticeFold, the prover's complexity is dominated by a degree-4 sumcheck over R_q of size n and the computation of $L \log_2(B)$ commitments to bit-decomposed vectors. LatticeFold+ eliminates all these commitments and is consequently $\Omega(\log(B))$-times faster. Concretely, we expect the LatticeFold+ prover to be five to ten times faster than the LatticeFold prover.

- Verifier circuit complexity: The verifier circuit size is dominated by hashing needed for the Fiat-Shamir transform. In LatticeFold, all the $L \log_2(B)$ decomposed commitments must be hashed. LatticeFold+ eliminates all these commitments thereby shrinking the size of the verification circuit.

- Proof size: The LatticeFold proof size is $O_\lambda(\kappa d \log B + d \log n)$ bits. In LatticeFold+ the proof size is only $O_\lambda(\kappa d + \log n)$ bits. Concretely, the LatticeFold+ proof size is about 95KB, which is close to the proof size of LaBRADOR [BS23], the shortest known lattice-based NARK, which has a proof size of 60KB for a similarly sized statement.

Finally, since the prover running time in LatticeFold is comparable to HyperNova [Net24], we expect the prover time in LatticeFold+ to be significantly faster than HyperNova. Hence, LatticeFold+ provides both a fast prover and post-quantum security.

1.1 Additional related work

Several post-quantum monolithic SNARKs (i.e., not using folding) are built from hash-based Merkle commitments: Stark [BBHR18b], Ligero [AHIV17], Aurora [Ben+19], Brakedown [Gol+23], BaseFold [ZCF24], Blaze [Bre+24], and Fractal [COS20]. Their proof sizes scale sublinearly with the witness size. In practice they require a significant amount of memory when proving a large statement. Several elegant post-quantum lattice-based proof systems also offer sublinear proof size [Bau+18; BLNS20; ACK21; Alb+22; BCS23], however the resulting proofs are larger than the hash-based schemes. One exception is LaBRADOR [BS23] that produces relatively short proofs, but has a linear time verifier. We note that LaBRADOR also uses commitments to commitments to shrink the proof size. A polynomial commitment scheme derived from LaBRADOR, called Greyhound [NS24], has a square root time verifier. Other lattice-based proof systems, such as [ENS20; LNP22], perform well for small statements, but their proof size is linear in the size of the witness. Cini et al. [CMNW24] recently presented a lattice-based polynomial commitment scheme (PCS) from Bulletproof/FRI-like techniques [BBHR18a; Bün+18]. Their approach achieves better control over witness norm/slack blowup than previous works. However, their scheme incurs norm blowup/slack at each step, limiting it to supporting only a logarithmic number of folding steps.

Following $\mathsf{LatticeFold}$ [BC24], which uses module-based Ajtai commitments and the ℓ_∞-norm, Fenzi et al. proposed Lova [FKNP24] which operates similarly, but using integer-based Ajtai commitments and the ℓ_2-norm.

2 Preliminaries

Notation. $\lambda \in \mathbb{N}$ is the security parameter. For $l, r \in \mathbb{Z}$, $l < r$, we denote by (l, r) the set $\{l+1, l+2, \dots, r-1\}$ and $[l, r)$ the set $\{l, l+1, \dots, r-1\}$, and we define $[n] := [0, n)$. For a set S, we denote by $P(S)$ the power set of S that consists of all the subsets of S. If S supports element subtraction, we denote by $S - S$ the set of differences between any two distinct elements in S. 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**.

By default, a vector is a column vector and a ring is always commutative. For a vector $\mathbf{f}$ of length n, we use $\mathbf{f}_i$ or $\mathbf{f}[i]$ $(0 \le i < n)$ to denote its ith element. For two vectors $\mathbf{f}, \mathbf{g}$ of the same length we let $\langle \mathbf{f}, \mathbf{g} \rangle$ denote their inner product.

Let $\bar{R}$ be an arbitrary ring. Given column vectors $\mathbf{u}_1, \dots, \mathbf{u}_k \in \bar{R}^n$, we use $[\mathbf{u}_1, \dots, \mathbf{u}_k] \in \bar{R}^{n \times k}$ to denote horizontal concatenation and $(\mathbf{u}_1, \dots, \mathbf{u}_k) \in \bar{R}^{nk}$ to denote vertical concatenation. Concatenations of row vectors are similarly defined. For a matrix $\mathbf{M} \in \bar{R}^{n \times m}$, $\{\mathbf{M}_{i,*} \in \bar{R}^m\}_{i \in [n]}$ and $\{\mathbf{M}_{*,j} \in \bar{R}^n\}_{j \in [m]}$ denote the rows and columns of $\mathbf{M}$, respectively. We denote by $\mathsf{flat}(\mathbf{M}) := (\mathbf{M}_{0,*}, \dots, \mathbf{M}_{n-1,*}) \in \bar{R}^{nm}$ the vertical concatenation of its rows. $\bar{R}[X_1, \dots, X_k]$ denotes the set of k-variate polynomials over $\bar{R}$, and we use $\bar{R}^{\le d}[X_1, \dots, X_k]$

to denote the set of polynomials with the degree of each variable at most d.

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. In what follows, we omit the index $\mathbb{i}$ when it is clear in the context.

2.1 Cyclotomic rings

Let $d \in \mathbb{N}$ be a power of two. We denote by $R := \mathbb{Z}[X]/\langle X^d + 1\rangle$ the anti-cyclotomic ring of dimension d. Let $q > 2$ be a prime and denote by $R_q := R/qR = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$ where we represent $\mathbb{Z}_q := \{-\lfloor q/2\rfloor, \ldots, \lfloor q/2\rfloor\}$. If $q \equiv 1 + 2e \pmod{4e}$ for some $e \mid d$, it is well-known that $R_q \cong \mathbb{F}^e_{q^{d/e}}$ via the Number Theoretic Transform (NTT). For $f = \sum_{i\in[d]} f_i X^i \in R_q$, we use $\mathsf{cf}(f) := (f_0, \ldots, f_{d-1}) \in \mathbb{Z}_q^d$ to denote the coefficient vector of f, and $\mathsf{ct}(f) := f_0$ is the constant term of f. For a vector $\mathbf{f} \in R_q^n$, we denote by $\mathsf{cf}(\mathbf{f}) := (\mathsf{cf}(\mathbf{f}_0)^\top, \ldots, \mathsf{cf}(\mathbf{f}_{n-1})^\top) \in \mathbb{Z}_q^{n\times d}$ the concatenation of the transposed coefficient vectors, and $\mathsf{ct}(\mathbf{f}) = (\mathsf{ct}(\mathbf{f}_0), \ldots, \mathsf{ct}(\mathbf{f}_{n-1})) \in \mathbb{Z}_q^n$ is the first column of $\mathsf{cf}(\mathbf{f})$.

Monomial sets. Define **the $\mathbb{Z}_q[X]$-monomial set**

$$\mathcal{M}' := \{0, 1, X, X^2, X^3, \ldots\} \subseteq \mathbb{Z}_q[X]\,. \tag{1}$$

We will need the following lemma.

Lemma 2.1. *Let $q > 2$ be a prime. For every $a \in \mathbb{Z}_q[X]$ we have that $a(X^2) = a(X)^2$ if and only if $a \in \mathcal{M}'$.*

Proof. When $a \in \mathcal{M}'$ then indeed $a(X^2) = a(X)^2$. Conversely, we prove that $a(X^2) = a(X)^2$ implies that $a \in \mathcal{M}'$. Let $a(X) = \sum_{i=1}^n a_i X^{d_i}$ for some $d_n > d_{n-1} > \ldots > d_1 \geq 0$. If $n = 1$ and $a(X^2) = a(X)^2$ then $a_1 X^{2d_1} = a_1^2 X^{2d_1}$, which implies that $a_1^2 = a_1$. Therefore $a_1 \in \{0, 1\}$ and either way $a \in \mathcal{M}'$, as required. Next, suppose towards a contradiction that $a(X^2) = a(X)^2$, but $n > 1$ and all $a_1, \ldots, a_n \in \mathbb{Z}_q$ are non-zero. The highest degree term in $a(X)^2$ has degree $2d_n$. The second highest degree term in $a(X)^2$ has degree is $d_n + d_{n-1}$ and its coefficient is $2a_n a_{n-1}$. Since $2d_n > d_n + d_{n-1} > 2d_{n-1}$, the polynomial $a(X^2)$ has no term of degree $d_n + d_{n-1}$. Therefore, because $a(X^2) = a(X)^2$, we can conclude that $2a_n a_{n-1} = 0$. But since $q > 2$ this implies that one of a_n or a_{n-1} must be zero, which contradicts the assumption that $a_1, \ldots, a_n \in \mathbb{Z}_q$ are non-zero. □

Define another **monomial set**

$$\mathcal{M} := \left\{0, 1, X, \ldots, X^{d-1}\right\} \subseteq \mathcal{M}'\,. \tag{2}$$

In what follows we say that an element $a \in R_q$ is in $\mathcal{M}$ if and only if the natural embedding of a into $\mathbb{Z}_q[X]$ is in the set $\mathcal{M} := \left\{0, 1, X, \ldots, X^{d-1}\right\}$.

Remark 2.1. Lemma 2.1 does not hold in R_q. For example, in $\mathbb{Z}_q[X]/\langle X^4+1\rangle$ with $q \equiv 1 \pmod 4$, define

$$a(X) := X^3/2 + (i/2)X^2 + (i/2)X + (1/2) \in R_q$$

where $i \in \mathbb{Z}_q$ satisfies $i^2 = -1$. Then $a(X)$ satisfies $a(X)^2 = a(X^2)$, but a is not in $\mathcal{M}$.

For $a \in (-d, d) \subseteq \mathbb{Z}_q$, we denote by $\mathsf{sgn}(a) \in \{-1, 0, 1\}$ the sign of a and $\mathsf{sgn}(0) := 0$. Define $\mathsf{exp}(a) \in R_q$ as $\mathsf{exp}(a) := \mathsf{sgn}(a)X^a$. Note that $\mathsf{exp}(a)$ is in $\mathcal{M}$. E.g., if $a < 0$, then $\mathsf{exp}(a) = -X^a = -(-X^{a+d}) = X^{a+d} \in \mathcal{M}$. Similarly, we define set $\mathsf{EXP}(a) \subseteq \mathcal{M}$ to be

$$\mathsf{EXP}(a) := \begin{cases} \{\mathsf{exp}(a)\} & \text{if } a \neq 0 \\ \{0, 1, X^{d/2}\} & \text{if } a = 0 \end{cases} \tag{3}$$

For a matrix $\mathbf{M} \in (-d, d)^{m\times n}$, $\mathsf{exp}(\mathbf{M}) \in \mathcal{M}^{m\times n}$ denotes the matrix that replaces each entry $\mathbf{M}_{i,j}$ with $\mathsf{exp}(\mathbf{M}_{i,j})$, and $\mathsf{EXP}(\mathbf{M}) : [m] \times [n] \to P(\mathcal{M})$ denotes the product of sets that replaces each entry $\mathbf{M}_{i,j}$ with $\mathsf{EXP}(\mathbf{M}_{i,j})$.

We will need the following lemma.

Lemma 2.2. *Let $d' := d/2$ and $\psi := \sum_{i\in[1,d')} i \cdot (X^{-i} + X^i) \in R_q$. For every $a \in \mathbb{Z}_q$, if $a \in (-d', d')$, then for all $b \in \mathsf{EXP}(a)$ in $\mathcal{M}$, we have $\mathsf{ct}(b \cdot \psi) = a$. Conversely, if there exists $b \in \mathcal{M}$ such that $\mathsf{ct}(b \cdot \psi) = a$, then $a \in (-d', d')$ and $b \in \mathsf{EXP}(a)$.*

Proof. If $a \in (-d', d')$, then either $a = 0$ and $\mathsf{ct}(b \cdot \psi) = 0$ for all $b \in \mathsf{EXP}(0)$, or $a \neq 0$, and the only element $b := \mathsf{exp}(a)$ in $\mathsf{EXP}(a)$ satisfies that

$$\mathsf{ct}(b \cdot \psi) = \mathsf{ct}\left(\mathsf{sgn}(a)X^a \cdot \sum_{i\in[1,d')} i \cdot (X^{-i} + X^i)\right) = a\,.$$

Conversely, suppose $\mathsf{ct}(b^* \cdot \psi) = a$ for some $b^* \in \mathcal{M}$, we show that $a \in (-d', d')$. If $b^* = 0$, then $a = 0 \in (-d', d')$. Otherwise, $b^* \cdot \psi$ rotates and flips the signs of the coefficients of ψ, and the constant term $\mathsf{ct}(b^* \cdot \psi)$ stays in $(-d', d')$. Therefore, $a \in (-d', d')$ as required. Finally, given that $b^* \in \mathcal{M}$ and $\mathsf{ct}(b^* \cdot \psi) = a$ for $a \in (-d', d')$, by inspection, b^* is in $\mathsf{EXP}(a)$ and the claim holds. $\square$

Norms. Given $f = \sum_{i\in[d]} f_i X^i \in R$, the ℓ_∞-norm of f is defined as $\|f\|_\infty := \max_{i=0}^{d-1}(|f_i|)$. For a matrix $\mathbf{F} \in R^{n\times m}$ with entries $\{\mathbf{F}_{i,j}\}_{i\in[n],j\in[m]}$, its ℓ_∞-norm is defined as $\|\mathbf{F}\|_\infty := \max_{i\in[n],j\in[m]}\{\|\mathbf{F}_{i,j}\|_\infty\}$. We similarly define the ℓ_∞-norms for R_q-elements by lifting elements in R_q to R via the natural embedding.

For an element $a \in R$, we define its operator norm as

$$\|a\|_{\mathsf{op}} := \sup_{y\in R} \frac{\|a \cdot y\|_\infty}{\|y\|_\infty}\,. \tag{4}$$

For a finite set $\mathcal{S} \subseteq R$, the operator norm $\|\mathcal{S}\|_{\mathsf{op}}$ is defined as $\|\mathcal{S}\|_{\mathsf{op}} := \max_{a\in\mathcal{S}}\|a\|_{\mathsf{op}}$. Similarly, we can define operator norms for sets over R_q by lifting the elements to R. When we write $a \cdot b$ for $a \in R_q$ and $b \in R$, we meant the product over R between b and the embedding of a to R. We will use the following useful facts.

Lemma 2.3. *For every $a \in \mathcal{M}$ and $b \in R$, we have $\|a \cdot b\|_\infty \le \|b\|_\infty$.*

Proof. If $a \in \mathcal{M}$, then either $a = 0$ and the claim holds; or $a \cdot b$ rotates the coefficients of b and flips the sign of some coefficients of b. Hence the ℓ_∞-norm won't change. □

Lemma 2.4 (Corollary 1.2 of [LS18]). *Let $d, e \in \mathbb{N}$ be power-of-twos and $e \mid d$, and let $q \equiv 1 + 2e \pmod{4e}$ be a prime. Every non-zero $y \in R_q$ with $\|y\|_\infty < \frac{q^{1/e}}{\sqrt{e}}$ is invertible.*

Strong sampling sets. A **strong sampling set** $\mathcal{S} \subseteq R_q$ satisfies that the difference of any two distinct elements in $\mathcal{S}$ is invertible. E.g., if q is a prime, then $\mathbb{Z}_q \subseteq R_q$ is a strong sampling set as the difference of any two distinct elements in $\mathbb{Z}_q$ is invertible in R_q. By Lemma 2.4, the set of R_q-elements with small coefficients is strongly samplable, because the differences of small coefficients are still small and thus invertible. Moreover, such a set has small operator norm:

Lemma 2.5 (Prop. 2 of [AL21]). *For all $u \in R$, $\|u\|_{\mathsf{op}} \le d \cdot \|u\|_\infty$.*

Gadget matrix decomposition. Let $b > 1$ and $\hat{b} = b^k$ for some $k \in \mathbb{N}$. Denote by $\mathbf{g}_{b,k} = (1, b, \ldots, b^{k-1}) \in \mathbb{Z}^k$. Given a matrix $\mathbf{M} \in R^{n\times m}$ where $\|\mathbf{M}\|_\infty < \hat{b}$, we can deterministically decompose[3] $\mathbf{M}$ into a matrix $\mathbf{M}' \in R^{n\times mk}$ such that $\|\mathbf{M}'\|_\infty < b$ and $\mathbf{M} = \mathbf{M}'\mathbf{G}_{b,k}$, where $\mathbf{G}_{b,k} \in \mathbb{Z}^{mk\times m}$ is the gadget matrix $\mathbf{G}_{b,k} := I_m \otimes \mathbf{g}_{b,k}$. We denote by $\mathbf{G}^{-1}_{b,k} : R^{n\times m} \to R^{n\times mk}$ the deterministic function that maps $\mathbf{M}$ to $\mathbf{G}^{-1}_{b,k}(\mathbf{M}) = \mathbf{M}'$, that is, $\mathbf{G}^{-1}_{b,k}(\mathbf{M})\mathbf{G}_{b,k} = \mathbf{M}$. We simply write $\mathbf{G}$ and $\mathbf{G}^{-1}$ when m, b, k are all clear in the context.

2.2 Multilinear extensions and sumchecks over rings

For a binary vector $a \in \{0,1\}^k$, denote by $[a]_k := \sum_{i\in[k]} a_i 2^i \in [2^k]$; for integer $b \in [2^k]$, denote by $\langle b\rangle_k \in \{0,1\}^k$ the binary representation of b. We write $\langle b\rangle$ in short when there is no ambiguity in context. We review the notion of multilinear extensions over rings.

Definition 2.1 (Multilinear Extensions over Rings). *Let $\bar{R}$ be a ring with zero 0 and identity 1. The multilinear extension $\widetilde{f} \in \bar{R}^{\le 1}[X_1, \ldots, X_k]$ of a function $f : \{0,1\}^k \to \bar{R}$ is*

$$\widetilde{f}(\mathbf{x}) := \sum_{\mathbf{b}\in\{0,1\}^k} f(\mathbf{b}) \cdot eq(\mathbf{b}, \mathbf{x})$$

where $eq(\mathbf{b}, \mathbf{x}) := \prod_{i\in[k]} \left[(1 - \mathbf{b}_i)(1 - \mathbf{x}_i) + \mathbf{b}_i\mathbf{x}_i\right]$.

[3]For every entry $x \in (-\hat{b}, \hat{b})$, we first compute the base-b decomposition $(x_0, \ldots, x_{k-1})$ of $|x|$, if $x < 0$, we further flip the sign of x_i for all $i \in [k]$.

For $\mathbf{r} \in \bar{R}^k$ we define $\mathsf{tensor}(\mathbf{r}) := \bigotimes_{i\in[k]}(\mathbf{r}_i, 1-\mathbf{r}_i) \in \bar{R}^{2^k}$ as the tensor product of $\mathbf{r}$. More generally, we can define tensor product over "$\bar{R}$-vector spaces". Let M be an $\bar{R}$-module. For $\mathbf{m} \in \mathsf{M}^k$, we define $\mathsf{tensor}(\mathbf{m}) := \bigotimes_{i\in[k]}(\mathbf{m}_i, 1-\mathbf{m}_i) \in \mathsf{M}^{2^k}$. E.g., if $\mathsf{M} = \bar{R} \times \bar{R}$ and $\mathbf{m} = (\mathbf{r}^{(0)}, \mathbf{r}^{(1)}) \in \mathsf{M}^k$ where $\mathbf{r}^{(0)}, \mathbf{r}^{(1)} \in \bar{R}^k$, then $\mathsf{tensor}(\mathbf{m}) = (\mathsf{tensor}(\mathbf{r}^{(0)}), \mathsf{tensor}(\mathbf{r}^{(1)}))$ can be understood as a pair of tensor products of $\mathbf{r}^{(0)}, \mathbf{r}^{(1)} \in \bar{R}^k$, respectively.

Remark 2.2. For a function $f : \{0,1\}^k \to \bar{R}$ we define its table as

$$\mathbf{f} := \big(f(\langle 0\rangle_k), f(\langle 1\rangle_k), \ldots, f(\langle 2^k-1\rangle_k)\big) \in \bar{R}^{2^k} .$$

Observe that the evaluation of $\widetilde{f}$ at the point $\mathbf{r} \in \bar{R}^k$ is exactly $\widetilde{f}(\mathbf{r}) = \langle \mathbf{f}, \mathsf{tensor}(\mathbf{r})\rangle$. More generally, for a point $\mathbf{m} \in \mathsf{M}^k$, we define $\widetilde{f}(\mathbf{m}) := \langle \mathbf{f}, \mathsf{tensor}(\mathbf{m})\rangle \in \mathsf{M}$. E.g., if $\mathsf{M} = \bar{R} \times \bar{R}$, we can understand $\widetilde{f}(\mathbf{m})$ as the evaluations of f at a pair of points $\mathbf{m} = (\mathbf{r}^{(0)}, \mathbf{r}^{(1)})$ in $\bar{R}^k$.

Lemma 2.6 (Generalized Schwartz-Zippel [BCPS18])**.** *For a nonzero polynomial $f \in \bar{R}^{\leq d}[X_1, \ldots, X_k]$ and a strong sampling set $\mathcal{C}$, we have $\Pr_{\mathbf{r} \xleftarrow{R} \mathcal{C}^k}[f(\mathbf{r}) = 0] \leq \frac{dk}{|\mathcal{C}|}$.*

As noted by [CCKP19], we can do sumcheck over rings.

Lemma 2.7 (Generalized Sum-Check [CCKP19])**.** *For a polynomial $f \in \bar{R}^{\leq \ell}[X_1, \ldots, X_k]$ with individual degree at most ℓ, a strong sampling set $\mathcal{C} \subseteq \bar{R}$, and a value $s \in \bar{R}$, there is a public-coin interactive protocol that reduces the checking of*

$$s \overset{?}{=} \sum_{\mathbf{b}\in\{0,1\}^k} f(\mathbf{b})$$

to the checking of an evaluation claim $\widetilde{f}(\mathbf{r}) \overset{?}{=} v$ for some $\mathbf{r} \xleftarrow{R} \mathcal{C}^k$ and $v \in \bar{R}$. The protocol is perfectly complete, has prover time[4] *$\tilde{O}(2^k \ell)$, verifier time and proof size $O(k\ell)$, and soundness error $\frac{k\ell}{|\mathcal{C}|}$.*

Remark 2.3 (Boosting soundness). We can decrease the soundness error from $k\ell/|\mathcal{C}|$ to $(k\ell/|\mathcal{C}|)^r$ with $r > 1$ parallel repetitions. E.g., for $r = 2$, set $\mathsf{M}_{\mathcal{C}} := \mathcal{C} \times \mathcal{C}$, we can understand the 2-way parallel execution as a sumcheck with challenge set $\mathsf{M}_{\mathcal{C}}$.

Remark 2.4 (Batching multiple claims). As noted in [BCS21, Sect. 2.1] and in Hyperplonk [CBBZ23], when checking $s > 1$ sumcheck claims over the same domain, we can batch them into a *single* sumcheck over a larger domain, with no additional randomness needed outside the sumcheck itself. For example, we can view d sumcheck claims over $\mathbb{Z}_q$ as a single sumcheck claim over $R_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$. We can also view $s > 1$ sumcheck claims over R_q as a single sumcheck claim over $R_q[Y]/\langle Y^s + 1\rangle$.

[4] f needs to be a sum of product of multilinear polynomials to achieve prover time $\tilde{O}(2^k\ell)$.

2.3 Module-based Ajtai commitments

We recall the Module Short Integer Solution (MSIS) assumption.

Definition 2.2 (Module SIS with ℓ_∞-Norms [LS15; PR06; LM06; ACK21]). *Let $q = q(\lambda)$, $\kappa = \kappa(\lambda)$, $m = m(\lambda)$ and $\beta_{\mathsf{SIS}} = \beta_{\mathsf{SIS}}(\lambda)$. The module SIS assumption $\mathsf{MSIS}^{\infty}_{q,\kappa,m,\beta_{\mathsf{SIS}}}$ holds if for all expected polynomial-time adversary $\mathcal{A}$,*

$$\Pr\left[(\mathbf{A}\mathbf{x} = \mathbf{0} \bmod q) \wedge 0 < \|\mathbf{x}\|_\infty < \beta_{\mathsf{SIS}} \;\middle|\; \begin{array}{c} \mathbf{A} \xleftarrow{\text{R}} R_q^{\kappa\times m} \\ \mathbf{x} \in R^m \leftarrow \mathcal{A}(\mathbf{A}) \end{array} \right] = \mathsf{negl}(\lambda)\,.$$

Looking ahead, to argue knowledge soundness of the folding schemes, we need the commitments to satisfy a relaxed binding property defined below.

Definition 2.3 (Relaxed Binding Commitment [ALS20; ACK21; Ajt96; PR06; LM06]). *Fix $q = q(\lambda)$,$\kappa = \kappa(\lambda)$, $m = m(\lambda)$, bound $b \in \mathbb{N}$ and a set $\mathcal{S} \subseteq R_q^*$ with invertible elements. We say that a randomly sampled linear function $\mathbf{A} \xleftarrow{\text{R}} R_q^{\kappa\times m}$ is $(b, \mathcal{S})$-relaxed binding if for all expected polynomial-time adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} 0 < \|\mathbf{z}_1\|_\infty, \|\mathbf{z}_2\|_\infty < b \wedge s_1, s_2 \in \mathcal{S} \wedge \\ \mathbf{A}\mathbf{z}_1 s_1^{-1} = \mathbf{A}\mathbf{z}_2 s_2^{-1} \wedge \\ \mathbf{z}_1 s_1^{-1} \neq \mathbf{z}_2 s_2^{-1} \end{array} \;\middle|\; \begin{array}{c} \mathbf{A} \xleftarrow{\text{R}} R_q^{\kappa\times m} \\ (\mathbf{z}_1, \mathbf{z}_2 \in R_q^m, s_1, s_2) \leftarrow \mathcal{A}(\mathbf{A}) \end{array} \right] = \mathsf{negl}(\lambda)\,.$$

It is clear that if the $(b, \mathcal{S})$-relaxed binding property doesn't hold, then we can find $\mathbf{x} := s_2\mathbf{z}_1 - s_1\mathbf{z}_2 \neq \mathbf{0} \in R^m$ such that $\mathbf{A}\mathbf{x} = 0 \bmod q$. Here $s_2\mathbf{z}_1 - s_1\mathbf{z}_2$ is computed over R by first lifting $s_1, s_2, \mathbf{z}_1, \mathbf{z}_2$ to R. Moreover, $\|\mathbf{x}\|_\infty < B := 2b\|\mathcal{S}\|_{\mathsf{op}}$, thus we can reduce the $(b, \mathcal{S})$-relaxed binding property to the MSIS assumption $\mathsf{MSIS}^{\infty}_{q,\kappa,m,B}$.

2.4 Reduction of Knowledge

We review the notion of reduction of knowledge (RoK) from [KP23]. Informally, it converts the checking of a statement in relation $\mathcal{R}_1$ to that of a reduced statement in $\mathcal{R}_2$. Note that it also captures the notion of folding schemes.

Definition 2.4 (Reduction of Knowledge [KP23]). *Let $\mathcal{R}_1$, $\mathcal{R}_2$ be indexed relations. A reduction of knowledge Π from relation $\mathcal{R}_1$ to $\mathcal{R}_2$ consists of the following PPT algorithms:*

- $\mathcal{G}(1^\lambda) \to \mathbb{i}$: *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 $(\mathbb{x}_1, \mathbb{w}_1) \in \mathcal{R}_1$, interacts 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.*

For brevity, 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* $(\mathbb{i}, \mathbb{x}_1)$*, and assume without loss of generality that (i) the reduced instances output by the prover and verifier are the same, (ii)* $\bot \notin \mathcal{L}(\mathcal{R}_2)$ *by default.*

A reduction of knowledge with the above syntax satisfy the properties below.

Definition 2.5 (Perfect Completeness)**.** *For every PPT adversary* $\mathcal{A}$*,*

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

Definition 2.6 (Knowledge Soundness)**.** *There exists a knowledge-error function* $\kappa(\cdot)$ *and an expected polynomial time extractor* Ext *such that for all expected polynomial time adversaries* $(\mathcal{A}, \mathsf{P}^*)$ *that can output* $(\mathbb{i}, \mathbb{x}_2, \mathbb{w}_2) \in \mathcal{R}_2$ *with non-negligible probability after the interaction with* V*, we have that given* $\mathbb{i} \leftarrow \mathcal{G}(1^\lambda)$*,* $(\mathbb{x}_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathbb{i})$*,*

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

Definition 2.7 (Public reducibility.)**.** *There is a deterministic polynomial time algorithm* f *such that for all PPT adversary* $\mathcal{A}$ *and expected polynomial time adversary* P^**:*

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

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

We recall the knowledge composition theorem from [KP23].

Theorem 2.1 (Sequential Composition, Thm. 5 [KP23])**.** *Let* $\mathcal{R}_1$*,* $\mathcal{R}_2$*,* $\mathcal{R}_3$ *be indexed relations. Let* Π_1 *(and* Π_2*) be reduction of knowledge from* $\mathcal{R}_1$ *to* $\mathcal{R}_2$ *(and from* $\mathcal{R}_2$ *to* $\mathcal{R}_3$*), respectively. Then* $\Pi_2 \circ \Pi_1$ *is a reduction of knowledge from* $\mathcal{R}_1$ *to* $\mathcal{R}_3$*.*

3 Generalized Committed Linear Relations

Our goal is to build a folding scheme for R1CS relations — relations expressed as a set of quadratic constraints — and more generally, for customizable constraint systems (CCS) [STW23] that are expressed as a set of constraints of higher degree. LatticeFold [BC24, Sec. 4] gives a reduction of knowledge from CCS over rings to a linear relation over rings. This reduction of knowledge is inspired by a technique used over fields in HyperNova [KS24b].

Here we define a general linear relation over rings that is the target of this reduction of knowledge. The relation uses a commitment scheme $[\![\cdot]\!] : R_q^n \to R_q^\kappa$ for vectors $\mathbf{f} \in R_q^n$ where $\|\mathbf{f}\|_\infty < B$, for some norm bound B. Usually, this is simply the Ajtai commitment scheme from Section 2.3. The commitment to $\mathbf{f}$ is denoted by $\mathsf{cm} = [\![\mathbf{f}]\!]$.

Definition 3.1. *Let $[\![\cdot]\!] : R_q^n \to R_q^\kappa$ be a commitment scheme. A* ***generalized committed linear relation*** *parameterized by $B \in \mathbb{N}$, denoted $\mathcal{R}_{\mathsf{lin},B}$, is a set of triples $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ where*

$$\mathbb{i} = \left([\![\cdot]\!],\ (\mathbf{M}^{(i)} \in R_q^{n\times n})_{i\in[n_{\mathsf{lin}}]}\right), \quad \mathbb{x} = (\mathsf{cm}_{\mathbf{f}},\ \mathbf{r} \in \mathsf{M}_{\mathcal{C}}^{\log n},\ \mathbf{v} \in \mathsf{M}_q^{n_{\mathsf{lin}}}), \quad \mathbb{w} = \mathbf{f} \in R_q^n.$$

Here $\mathsf{M}_{\mathcal{C}} := \mathcal{C} \times \mathcal{C}$ where $\mathcal{C}$ is a strong sampling set[5]*, and $\mathsf{M}_q := R_q \times R_q$. A triple $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ is in the relation $\mathcal{R}_{\mathsf{lin},B}$ if*

$$(\|\mathbf{f}\|_\infty < B)\ \wedge\ (\mathsf{cm}_{\mathbf{f}} = [\![\mathbf{f}]\!])\ \wedge\ \left(\forall i \in [n_{\mathsf{lin}}] :\ \langle \mathbf{M}^{(i)} \cdot \mathbf{f}, \mathsf{tensor}(\mathbf{r})\rangle = \mathbf{v}_i\right). \tag{5}$$

In words, an instance $(\mathsf{cm}, \mathbf{r}, \mathbf{v})$ is in the language of the relation $\mathcal{R}_{\mathsf{lin},B}$ if cm is a commitment to a low-norm vector $\mathbf{f} \in R_q^n$, and for all $i \in [n_{\mathsf{lin}}]$, the multilinear extension of the vector $\mathbf{M}^{(i)}\mathbf{f} \in R_q^n$ evaluates to the pair $\mathbf{v}_i \in \mathsf{M}_q$ at the pair of points $\mathbf{r} \in \mathsf{M}_{\mathcal{C}}^{\log n}$.

In Appendix A we present a reduction of knowledge from R1CS to $\mathcal{R}_{\mathsf{lin},B}$ where $n_{\mathsf{lin}} = 4$. This protocol shows that a folding scheme for $\mathcal{R}_{\mathsf{lin},B}$ is sufficient for folding R1CS relations. At the heart of the protocol is a sumcheck to reduce the quadratic R1CS relation to four multilinear evaluation of degree 1, which is an instance of $\mathcal{R}_{\mathsf{lin},B}$. When the R1CS relation is defined by matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in R_q^{n\times m}$, the derived matrices $\mathbf{M}^{(1)}, \mathbf{M}^{(2)}, \mathbf{M}^{(3)}, \mathbf{M}^{(4)} \in R_q^{n\times n}$ in $\mathcal{R}_{\mathsf{lin},B}$ are

$$\mathbf{M}^{(1)} := \mathbf{I}_n, \quad \mathbf{M}^{(2)} := \mathbf{A} \cdot \mathbf{G}_{B,\hat{\ell}}^\top, \quad \mathbf{M}^{(3)} := \mathbf{B} \cdot \mathbf{G}_{B,\hat{\ell}}^\top, \quad \mathbf{M}^{(4)} := \mathbf{C} \cdot \mathbf{G}_{B,\hat{\ell}}^\top$$

where $\hat{\ell} := \lceil \log_B(q) \rceil$ and $\mathbf{G}_{B,\hat{\ell}} \in \mathbb{Z}_q^{m\hat{\ell}\times m} = \mathbb{Z}_q^{n\times m}$ is the gadget matrix from Section 2.1. A similar reduction of knowledge applies to CCS with higher degree constraints.

4 A Toolbox of Reduction of Knowledge

In this section, we introduce several reduction of knowledge protocols, which serve as the building blocks of our folding scheme for general linear relations. In Section 4.1, we introduce double commitments, which commits matrices in $R_q^{n\times m}$ to short vectors in R_q^κ. Section 4.2 introduces Π_{mon}, an RoK that checks if every entry of a committed matrix belongs to the monomial set from (2). In Section 4.3, we leverage Π_{mon} to construct a range-check protocol Π_{rgchk} for committed R_q-vectors. In Section 4.4, using Π_{rgchk}, we present an RoK Π_{cm} which transforms a committed relation associated with a double commitment, into a simpler relation associated with regular Ajtai commitments.

Recall that the power-of-two $d \in \mathbb{N}$ is the dimension of the ring R_q, and $d' := d/2$. Define $\ell := \lceil \log_{d'}(q) \rceil$. We will work over two types of challenge sets.

[5]Or more generally, a product of strong sampling sets.

Challenge sets. A **folding challenge set** $\bar{\mathcal{S}} \subseteq R_q$ is a strong sampling set with small operator norm. It is used to fold multiple witness vectors into a single low-norm vector. A **sumcheck challenge set** $\mathcal{C}$ is used to sample challenges for sumcheck protocols. For simplicity, we assume that $|\bar{\mathcal{S}}| = |\mathcal{C}| \geq 2^\lambda$. In particular, when q is a 128-bit prime, we set $\mathcal{C} := \mathbb{Z}_q$. More generally, for a smaller modulus q, we set $\mathcal{C}$ as a $\mathbb{Z}_q$-vector space that is large enough. E.g., if q is 64-bit, we can set $\mathcal{C} := \mathbb{F}_{q^2}$ or $\mathcal{C} := \mathbb{Z}_q \times \mathbb{Z}_q$, which is a $\mathbb{Z}_q$-vector space of dimension 2. To prove a sumcheck claim over R_q (or $\mathbb{Z}_q$), we run a sumcheck protocol over the challenge space $\mathcal{C}$, which can be understood as two parallel sumcheck executions over the challenge space $\mathbb{Z}_q$. For notation convenience, in what follows, we assume q is large enough and $\mathcal{C} = \mathbb{Z}_q$, but all protocols easily generalize to smaller modulus. Finally, we define the R_q-module $\mathsf{M}_q := R_q \times R_q$ and set $\mathsf{M}_{\mathcal{C}} := \mathcal{C} \times \mathcal{C}$. They are useful when we want to enforce even smaller sumcheck soundness error.

4.1 Linear commitments and double commitments

General linear commitments. For $\mathbf{a} \in R_q^n$ and $\mathbf{M} \in R_q^{n\times m}$, we denote by $[\![\mathbf{a}]\!] \in R_q^\kappa$ and $[\![\mathbf{M}]\!] \in R_q^{\kappa\times m}$ the "general linear commitments" to $\mathbf{a}$ and $\mathbf{M}$ respectively. In our paper, $[\![\mathbf{a}]\!]$ is computed as $\mathbf{Aa}$ given the SIS matrix $\mathbf{A} \in R_q^{\kappa\times n}$.

Fix $b \in \mathbb{N}$ and set $\mathcal{S} \subseteq R_q^*$, we say that $\mathbf{a} \in R_q^n$ is a $(b, \mathcal{S})$-**valid opening** of $\mathsf{cm}_{\mathbf{a}}$, if $\mathsf{cm}_{\mathbf{a}} = [\![\mathbf{a}]\!]$ and $\mathbf{a} = \frac{\mathbf{a}'}{s}$ for some $\mathbf{a}' \in R_q^n$, $s \in \mathcal{S}$ such that $\|\mathbf{a}'\|_\infty < b$. We simply say that $\mathbf{a}$ is a valid opening if $(b, \mathcal{S})$ is clear in context. Similarly, $\mathbf{M} \in R_q^{n\times m}$ is a valid opening of $C_{\mathbf{M}} \in R_q^{\kappa\times m}$ if for all $i \in [m]$, the i-th column of $\mathbf{M}$ is a valid opening of the i-th column of $C_{\mathbf{M}}$. We define the commitment opening relation (for R_q-vectors)

$$\mathcal{R}_{\mathsf{open}} := \left\{(\mathbb{x} = \mathsf{cm}_{\mathbf{f}} \in R_q^\kappa, \mathbb{w} = \mathbf{f} \in R_q^n) \,:\, \mathbf{f} \text{ is valid opening of } \mathsf{cm}_{\mathbf{f}}\right\}. \tag{6}$$

A general linear commitment is $(b, \mathcal{S})$-**binding** if it is infeasible to find two different $(b, \mathcal{S})$-valid openings $\mathbf{a}, \mathbf{a}'$ to the same commitment cm. In our work, binding is implied by the $(b, \mathcal{S} = \bar{\mathcal{S}} - \bar{\mathcal{S}})$-relaxed binding property (Definition 2.3). We simply say that the commitment is binding if $(b, \mathcal{S})$ is clear in context.

Double commitments. For a vector $\mathbf{m} \in R_q^n$, the double commitment of $\mathbf{m}$ is defined as $(\!|\mathbf{m}|\!) := [\![\mathbf{m}]\!] \in R_q^\kappa$. For a matrix $\mathbf{M} \in R_q^{n\times m}$ where $m > 1$, we compute the linear commitment $[\![\mathbf{M}]\!] \in R_q^{\kappa\times m}$, and decompose $[\![\mathbf{M}]\!]$ to a low norm vector $\tau \in (-d', d')^n \subseteq \mathbb{Z}_q^n$, and finally set $[\![\tau]\!] \in R_q^\kappa$ as the double commitment to $\mathbf{M}$, which is more compact. Specifically, assume that $\kappa m d \ell \leq n$, we denote by

$$(\!|\mathbf{M}|\!) := [\![\Phi^{-1}([\![\mathbf{M}]\!])]\!] \in R_q^\kappa \tag{7}$$

the **double commitment** to $\mathbf{M}$, which is much shorter than $[\![\mathbf{M}]\!]$. Here the map Φ^{-1} : $R_q^{\kappa\times m} \to (-d', d')^n$ is an injective function computed as follows:

Construction 4.1. On input $[\![\mathbf{M}]\!]$, compute $\Phi^{-1}([\![\mathbf{M}]\!]) \in (-d', d')^n$ as follows:

1. compute $\mathbf{M}' := \mathbf{G}_{d',\ell}^{-1}([\![\mathbf{M}]\!]) \in R_q^{\kappa \times m\ell}$—the gadget decomposition of $[\![\mathbf{M}]\!] \in R_q^{\kappa \times m}$, and flatten it to $\mathbf{M}'' := \mathsf{flat}(\mathbf{M}') \in R_q^{\kappa m \ell}$ where $\|\mathbf{M}''\|_\infty < d'$

2. $\tau'_M := \mathsf{flat}(\mathsf{cf}(\mathbf{M}'')) \in (-d', d')^{\kappa m \ell d}$ is set as the flattening of the coefficient matrix of $\mathbf{M}''$, it is clear that $\|\tau'_M\|_\infty < d'$

3. pad τ'_M with zeros to obtain $\tau_M \in (-d', d')^n$, and output $\Phi^{-1}([\![\mathbf{M}]\!]) := \tau_M$.

Φ^{-1} is injective given the injectivity of gadget decomposition and flattening. Moreover, by the property of gadget matrices, there is a function $\Phi : (-d', d')^n \to R_q^{\kappa \times m}$ such that $\Phi(\Phi^{-1}(\mathbf{D})) = \mathbf{D}$ for all $\mathbf{D} \in R_q^{\kappa \times m}$. (However, Φ is not injective.)

We say that $(\tau \in (-d', d')^n, \mathbf{M} \in R_q^{n \times m})$ is a **valid opening** of $C \in R_q^\kappa$ if

1. $\mathbf{M}$ is a valid opening of $\Phi(\tau) = [\![\mathbf{M}]\!]$, and

2. τ is a valid opening of C when treating C as a linear commitment.

Remark 4.1. τ is not necessarily $\Phi^{-1}([\![\mathbf{M}]\!])$ because Φ is not injective.

Define the double commitment opening relation

$$\mathcal{R}_{\mathsf{dopen},m} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = C_{\mathbf{M}} \in R_q^\kappa \\ \mathbb{w} = (\tau \in (-d', d')^n, \mathbf{M} \in R_q^{n \times m}) \\ \text{where } (\tau, \mathbf{M}) \text{ is valid opening of } C_{\mathbf{M}} \end{array} \right\}. \tag{8}$$

Lemma 4.1. *If $[\![\cdot]\!]$ is binding, then $(\!|\cdot|\!)$ is also binding, that is, it's infeasible to find $C, (\tau, \mathbf{M}), (\tau', \mathbf{M}')$ where $\mathbf{M} \neq \mathbf{M}'$ and $(\tau, \mathbf{M})$, $(\tau', \mathbf{M}')$ are both valid openings of C.*

Proof. We can find a collision for $[\![\cdot]\!]$ given such openings: if $[\![\mathbf{M}]\!] = [\![\mathbf{M}']\!]$, then we find a collision $\mathbf{M} \neq \mathbf{M}'$; if $\tau \neq \tau'$, we find a collision (τ, τ') given that $[\![\tau]\!] = [\![\tau']\!] = C$. Otherwise, suppose $\tau = \tau'$ and $[\![\mathbf{M}]\!] \neq [\![\mathbf{M}']\!]$. Since $(\tau, \mathbf{M})$, $(\tau', \mathbf{M}')$ are both valid openings, we reach a contradiction that $[\![\mathbf{M}]\!] = \Phi(\tau) = \Phi(\tau') = [\![\mathbf{M}']\!]$. □

4.2 Monomial set check

In this section, we present a protocol for checking that every element of a committed matrix $\mathbf{M}$ is in the monomial set $\mathcal{M}$ from (2). This will be useful later in our range proofs. We present the protocol as a reduction of knowledge from an input relation $\mathcal{R}_{\mathsf{m,in}}$ to an output relation $\mathcal{R}_{\mathsf{m,out}}$.

The input relation $\mathcal{R}_{\mathsf{m,in}}$ holds when the double commitment $C_{\mathbf{M}}$ is a commitment to a witness matrix $\mathbf{M}$ of monomials:

$$\mathcal{R}_{\mathsf{m,in}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = C_{\mathbf{M}} \in R_q^\kappa \, , \mathbb{w} = \mathbf{M} \in R_q^{n \times m} \text{ s.t.} \\ \mathbf{M}_{i,j} \in \mathcal{M} \text{ for all } (i,j) \in [n] \times [m] \\ (C_{\mathbf{M}}, (\Phi^{-1}([\![\mathbf{M}]\!]), \mathbf{M})) \in \mathcal{R}_{\mathsf{dopen},m} \end{array} \right\}. \tag{9}$$

The output relation $\mathcal{R}_{\mathsf{m,out}}$ is defined as

$$\mathcal{R}_{\mathsf{m,out}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = (C_{\mathbf{M}} \in R_q^{\kappa}, \mathbf{r} \in \mathcal{C}^{\log n}, \mathbf{e} \in R_q^{m}), \mathbb{w} = \mathbf{M} \in R_q^{n \times m} \text{ s.t.} \\ \mathbf{M}^{\top} \mathsf{tensor}(\mathbf{r}) = \mathbf{e} \ \wedge \ (C_{\mathbf{M}}, (\Phi^{-1}([\![\mathbf{M}]\!]), \mathbf{M})) \in \mathcal{R}_{\mathsf{dopen},m} \end{array} \right\}. \tag{10}$$

The output relation holds for an instance $\mathbb{x} = (C_{\mathbf{M}}, \mathbf{r}, \mathbf{e})$ and a witness matrix $\mathbb{w} = \mathbf{M}$, if $\mathbf{M}$ is an opening of $C_{\mathbf{M}}$ and satisfies $\mathbf{M}^{\top}\mathsf{tensor}(\mathbf{r}) = \mathbf{e}$.

Before the construction, we introduce some notation and a useful corollary.

Notation. For every $i \in [n]$, we write $\langle i \rangle \in \{0,1\}^{\log n}$ in short for $\langle i \rangle_{\log n}$. For a ring element $a = \sum_{i \in [d]} a_i X^i \in R_q$, and a value $\beta \in \mathbb{F}_{q^u}$ (where $u \geq 1$), we denote by $a[\beta] := \sum_{i \in [d]} a_i \beta^i \in \mathbb{F}_{q^u}$.

Corollary 4.1. *Let $q > 2$ be a prime and $\mathbb{F}_{q^u}$ an extension field of $\mathbb{Z}_q$ with $u \geq 1$. For all $a \in \mathcal{M}$ and $\beta \in \mathbb{F}_{q^u}$, we have $a[\beta]^2 = a[\beta^2]$ (where the arithmetic is over $\mathbb{F}_{q^u}$). And for every $a \in R_q$ not in $\mathcal{M}$,*

$$\Pr_{\beta \xleftarrow{R} \mathbb{F}_{q^u}} \left[a[\beta]^2 = a[\beta^2] \right] < 2d / |\mathbb{F}_{q^u}| \, .$$

Proof. The completeness is trivial. We focus on proving soundness. For $a \notin \mathcal{M}$, consider the natural embedding $a(X)$ of a in $\mathbb{Z}_q[X]$ as a polynomial of degree less than d. By Lemma 2.1, $a(X)^2 \neq a(X^2)$ where $a(X)^2$ and $a(X^2)$ both have degree less than $2d$. Moreover, since $\mathbb{Z}_q$ is a subfield of $\mathbb{F}_{q^u}$, we can lift $a(X)^2$ and $a(X^2)$ from $\mathbb{Z}_q[X]$ to $\mathbb{F}_{q^u}[X]$. Then by Lemma 2.6, the claim holds. $\square$

Corollary 4.1 suggests a natural way to do monomial set check, by testing $\mathbf{M}_{i,j}[\beta]^2 = \mathbf{M}_{i,j}[\beta^2]$ for every entry (i,j). However, this is too expensive for the verifier, thus we use sumcheck to improve verifier complexity. We formally describe the construction below.

Construction 4.2. On input $\mathbb{x} = C_{\mathbf{M}} \in R_q^{\kappa}$ and $\mathbb{w} = \mathbf{M} \in \mathcal{M}^{n \times m} \subseteq R_q^{n \times m}$ where $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{m,in}}$, the protocol Π_{mon} proceeds as follows.

1. $\mathsf{V} \rightarrow \mathsf{P}$: Send challenges $\mathbf{c} \xleftarrow{R} \mathcal{C}^{\log n}$ and $\beta \xleftarrow{R} \mathcal{C}$.
2. $\mathsf{P} \leftrightarrow \mathsf{V}$: Run a (batched) degree-3 sumcheck for the following claims: for all $j \in [m]$, denote by

$$\mathbf{m}^{(j)} := (\mathbf{M}_{0,j}[\beta], \ldots, \mathbf{M}_{n-1,j}[\beta]) \, , \qquad \mathbf{m'}^{(j)} := (\mathbf{M}_{0,j}[\beta^2], \ldots, \mathbf{M}_{n-1,j}[\beta^2])$$

and check that

$$\sum_{i \in [n]} eq(\mathbf{c}, \langle i \rangle) \cdot \left(\widetilde{\mathbf{m}^{(j)}}(\langle i \rangle)^2 - \widetilde{\mathbf{m'}^{(j)}}(\langle i \rangle) \right) = 0 \, . \tag{11}$$

Let $\mathbf{r} \xleftarrow{\mathrm{R}} \mathcal{C}^{\log n}$ be the sumcheck challenge, $\alpha \xleftarrow{\mathrm{R}} \mathcal{C}$ be the random combiner and $v \in \mathcal{C}$ be the claimed evaluation. The sumcheck reduces to the claim:

$$eq(\mathbf{c}, \mathbf{r}) \cdot \left[\sum_{j \in [m]} \alpha^j \cdot \left(\widetilde{\mathbf{m}^{(j)}}(\mathbf{r})^2 - \widetilde{\mathbf{m}'^{(j)}}(\mathbf{r}) \right) \right] \stackrel{?}{=} v \, .$$

3. $\mathsf{P} \to \mathsf{V}$: Send $\left\{ \mathbf{e}_j = \widetilde{M_{*,j}}(\mathbf{r}) \in R_q \right\}_{j \in [m]}$

4. V : Check that:

$$eq(\mathbf{c}, \mathbf{r}) \cdot \left[\sum_{j \in [m]} \alpha^j \cdot \left(\mathbf{e}_j[\beta]^2 - \mathbf{e}_j[\beta^2] \right) \right] \stackrel{?}{=} v \, , \tag{12}$$

   Abort and return $\perp$ if the checks fail.

5. Return the reduced statement $(\mathbb{x} = (C_{\mathbf{M}}, \mathbf{r}, \mathbf{e}), \mathbb{w} = \mathbf{M}) \in \mathcal{R}_{\mathsf{m,out}}$.

Remark 4.2 (Batching). We can extend Π_{mon} to check multiple committed matrices by simply batching all the sumchecks.

Remark 4.3 (Efficiency). The protocol Π_{mon} is highly efficient for reasons below.

1. The degree-3 sumcheck operates over $\mathcal{C}$ rather than R_q. E.g., if $q \geq 2^\lambda$, $\mathcal{C} = \mathbb{Z}_q$ has much more efficient arithmetic than R_q.

2. The commitment cost of $C_{\mathbf{M}} = (\!|\mathbf{M}|\!)$ is mainly for computing $[\![\mathbf{M}]\!]$. Recall that the entries of $\mathbf{M}$ are in the monomial set $\mathcal{M}$. Thus, for each column $\mathbf{M}_{*,j}$ ($j \in [m]$), $[\![\mathbf{M}_{*,j}]\!] = \mathbf{A}\mathbf{M}_{*,j}$ is essentially the sum of $\mathbf{A}$'s columns (after rotation and sign flipping). This requires only $n\kappa$ R_q-**additions**, much more efficient than R_q-multiplications. Overall, committing $[\![\mathbf{M}]\!]$ takes $\approx n\kappa m$ R_q-additions, or equivalently, $n\kappa dm$ (parallelizable) $\mathbb{Z}_q$-*additions*. E.g., for $m \approx d = 64$, and $q \approx 2^{128}$, the concrete cost is comparable to computing an Ajtai commitment for an arbitrary $\mathbf{a} \in R_q^n$, which takes $n\kappa$ R_q-multiplications, or equivalently, $\Omega(n\kappa d \log d)$ $\mathbb{Z}_q$-*multiplications*.

3. The multilinear evaluations $\{\mathbf{e}_j\}_{j \in [m]}$ can be computed in only $O(n)$ $\mathbb{Z}_q$-multiplications plus $O(nm)$ $\mathbb{Z}_q$-additions. We first compute the vector $\mathsf{tensor}(\mathbf{r})$ using $O(n)$ $\mathbb{Z}_q$-multiplications. Then for each $j \in [m]$, $\widetilde{\mathbf{M}_{*,j}}(\mathbf{r}) = \langle \mathbf{M}_{*,j}, \mathsf{tensor}(\mathbf{r}) \rangle$ is computed as follows: initialize $v := \sum_{i \in [d]} v_i X^i \in R_q$ as zero. For every $i \in [n]$, do nothing if $\mathbf{M}_{i,j} = 0$; otherwise, denote by $\mathbf{M}_{i,j} = X^{m_{i,j}} \in \mathcal{M}$ where $m_{i,j} \in [d]$, and update $v_{m_{i,j}} \leftarrow v_{m_{i,j}} + \mathsf{tensor}(\mathbf{r})_i$. The resulting v is exactly $\widetilde{\mathbf{M}_{*,j}}(\mathbf{r})$ and it takes only $O(n)$ $\mathbb{Z}_q$-additions.

Lemma 4.2. *The protocol* Π_{mon} *is a reduction of knowledge from* $\mathcal{R}_{\mathsf{m,in}}$ *to* $\mathcal{R}_{\mathsf{m,out}}$.

Proof. Public reducibility is trivial. The claim follows from the lemmas below. □

Lemma 4.3. Π_{mon} *is perfectly complete.*

Proof. For every input statement in $\mathcal{R}_{\mathsf{m,in}}$, the verifier checks pass and the prover's output is in $\mathcal{R}_{\mathsf{m,out}}$: Since $\mathbf{M}$'s entries are in the monomial set $\mathcal{M}$, by Corollary 4.1, the sumcheck claim in (11) holds. Thus, the sumcheck verifier accepts, and the evaluation claim holds. Since the prover sends correct evaluations $\{\mathbf{e}_j\}$, the check in (12) holds and the output statement is in $\mathcal{R}_{\mathsf{m,out}}$. □

Lemma 4.4. *If* $(\!|\cdot|\!)$ *is binding with binding error* ϵ_{bind}*, then* Π_{mon} *is knowledge sound with knowledge error* $\epsilon_{\mathsf{mon},m} := (2d + m + 4\log n)/|\mathcal{C}| + \epsilon_{\mathsf{bind}}$.

Proof. The extractor Ext simulates the execution with the malicious prover $(\mathcal{A}, \mathsf{P}^*)$ and returns P^*'s output if P^* passes the verification. Otherwise, Ext returns $\perp$. It is clear that Ext has polynomial time complexity.

Let $\mathbb{w} = \mathbf{M}$ denote Ext's output. Let B_0 be the event that $\mathbb{w}$ is not valid for $\mathcal{R}_{\mathsf{m,in}}$ but valid for $\mathcal{R}_{\mathsf{m,out}}$. That is, there exists an entry (i, j) where $\mathbf{M}_{i,j} \notin \mathcal{M}$. Consider the following bad events:

- Event B_1: B_0 occurs and $\mathbf{M}_{i,j}[\beta]^2 = \mathbf{M}_{i,j}[\beta^2]$. (Note that for a *fixed* $\mathbf{M}$, by Corollary 4.1, this happens with probability at most $2d/|\mathcal{C}|$.)
- Event B_2: B_0 occurs and $\mathbf{M}_{i,j}[\beta]^2 \neq \mathbf{M}_{i,j}[\beta^2]$ but the batched sumcheck claim (at Step 2) holds. (Note that for a *fixed* $\mathbf{M}$, by Lemma 2.6, this happens with probability at most $m/|\mathcal{C}| + \log n/|\mathcal{C}|$.)
- Event B_3: B_0 occurs, the sumcheck claim (at Step 2) does not hold, but the evaluations $\mathbf{e}$ of $\mathbf{M}$ are correct. (Note that for a *fixed* $\mathbf{M}$, by sumcheck soundness, this happens with probability at most $3\log n/|\mathcal{C}|$.)

Suppose for contradiction that $\Pr[B_0] \geq \Pr[B_1 \vee B_2 \vee B_3] > 2d/|\mathcal{C}| + m/|\mathcal{C}| + 4\log n/|\mathcal{C}| + \epsilon_{\mathsf{bind}}$. Then there is an adversary that breaks the binding property of $[\![\cdot]\!]$ with probability more than ϵ_{bind}, contradiction. Thus the claim holds. □

4.3 Range check

In this section, we show how to check that the $\mathbb{Z}_q$-coefficients of a committed R_q-vector $\mathbf{f}$ are within a fixed range $(-B, B)$, where $B = (d')^k$ for some $k \in \mathbb{N}$. Recall that $d' := d/2$.

Warm-up. As a warm-up, we first set the range to $(-d', d')$ and range-check a vector $\tau \in \mathbb{Z}_q^n$. Our goal is to reduce the relation

$$\mathcal{R}'_{\mathsf{rg}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = \left(\mathsf{cm}_\tau, \mathsf{cm}_{\mathbf{m}_\tau} \in R_q^\kappa\right), \mathbb{w} = (\tau, \mathbf{m}_\tau \in R_q^n) \text{ s.t.} \\ \tau \in (-d', d')^n \wedge (\mathsf{cm}_{\mathbf{m}_\tau}, \mathbf{m}_\tau) \in \mathcal{R}_{\mathsf{m,in}} \\ (\mathbf{m}_\tau \in \mathsf{EXP}(\tau)) \wedge (\mathsf{cm}_\tau, \tau) \in \mathcal{R}_{\mathsf{open}} \end{array} \right\}.$$

to

$$\mathcal{R}' := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{x} = \left(\mathsf{cm}_\tau, \mathsf{cm}_{\mathbf{m}_\tau} \in R_q^\kappa, \mathbf{r} \in \mathcal{C}^{\log n}, (a, b \in R_q)\right), \\ \mathbb{w} = (\tau, \mathbf{m}_\tau \in R_q^n) \text{ s.t.} \\ [\tau, \mathbf{m}_\tau]^\top \mathsf{tensor}(\mathbf{r}) = (a, b) \\ (\mathsf{cm}_\tau, \tau) \in \mathcal{R}_{\mathsf{open}} \wedge (\mathsf{cm}_{\mathbf{m}_\tau}, \mathbf{m}_\tau) \in \mathcal{R}_{\mathsf{open}} \end{array} \right\}.$$

In words, $\mathcal{R}'_{\mathsf{rg}}$ checks that $\tau, \mathbf{m}_\tau$ are openings to $\mathsf{cm}_\tau, \mathsf{cm}_{\mathbf{m}_\tau}$, and $\mathbf{m}_\tau$ is in $\mathsf{EXP}(\tau) \subseteq \mathcal{M}^n$ (where $\mathsf{EXP}(\cdot)$ is defined in (3)), and τ is in the range $(-d', d')^n$. $\mathcal{R}'$ checks that for instance $(\mathsf{cm}_\tau, \mathsf{cm}_{\mathbf{m}_\tau}, \mathbf{r}, a, b)$, the witnesses $\tau, \mathbf{m}_\tau$ are openings to $\mathsf{cm}_\tau, \mathsf{cm}_{\mathbf{m}_\tau}$ and the matrix-vector product $[\tau, \mathbf{m}_\tau]^\top \mathsf{tensor}(\mathbf{r}) = (a, b)$ holds. Here, $\mathsf{cm}_{\mathbf{m}_\tau}$ serves as a helper commitment for the range-check. One may understand it as a proof element in the reduction of knowledge. However, we put it directly into the instance to simplify the protocols and their analysis.

Construction 4.3. On input $\mathbb{x} = (\mathsf{cm}_\tau, \mathsf{cm}_{\mathbf{m}_\tau} \in R_q^\kappa)$, $\mathbb{w} = (\tau \in (-d', d')^n, \mathbf{m}_\tau \in \mathsf{EXP}(\tau) \subseteq \mathcal{M}^n)$ where[6] $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}'_{\mathsf{rg}}$, the protocol proceeds as follows:

1. Run monomial set check Π_{mon} for $\mathbf{m}_\tau$, which outputs a statement

   $$(\mathbb{x} = (\mathsf{cm}_{\mathbf{m}_\tau}, \mathbf{r} \in \mathcal{C}^{\log n}, b \in R_q), \mathbb{w} = \mathbf{m}_\tau) \in \mathcal{R}_{\mathsf{m,out}}$$

   where $b = \langle \mathbf{m}_\tau, \mathsf{tensor}(\mathbf{r}) \rangle \in R_q$.

2. $\mathsf{P} \to \mathsf{V}$: Send $a := \langle \tau, \mathsf{tensor}(\mathbf{r}) \rangle \in \mathcal{C}$.

3. V : Let $\psi := \sum_{i \in [1, d')} i \cdot (X^{-i} + X^i) \in R_q$. Check $\mathsf{ct}(\psi \cdot b) \overset{?}{=} a$, abort and return $\perp$ if the check fails.

4. Return the reduced statement $(\mathbb{x} = (\mathsf{cm}_\tau, \mathsf{cm}_{\mathbf{m}_\tau}, \mathbf{r}, (a, b)), \mathbb{w} = (\tau, \mathbf{m}_\tau)) \in \mathcal{R}'$.

Lemma 4.5. *If $[\![\cdot]\!]$ is binding with binding error ϵ_{bind}, then the above protocol is a reduction of knowledge from $\mathcal{R}'_{\mathsf{rg}}$ to $\mathcal{R}'$ with knowledge error $\epsilon'_{\mathsf{rg}} := \epsilon_{\mathsf{mon},1} + \epsilon_{\mathsf{bind}} + \log n / |\mathcal{C}|$.*

Proof. Completeness is straightforward, we prove knowledge soundness below.

The extractor simulates execution with the adversary $(\mathcal{A}, \mathsf{P}^*)$ and return P^*'s output. Conditioned on the verifier accepts, let $\mathbb{w} = (\tau, \mathbf{m}_\tau)$ be the extractor's output. Suppose $\mathbb{w}$ is not valid for $\mathcal{R}'_{\mathsf{rg}}$, then either (i) $\mathbf{m}_\tau$ is an invalid opening or $\mathbf{m}_\tau \notin \mathcal{M}^n$, which happens with probability at most $\epsilon_{\mathsf{mon},1}$; or (ii) $\mathbf{m}_\tau$ is a valid opening within $\mathcal{M}^n$, but $\tau \notin (-d', d')^n$ or

[6]In practice, a standard implementation will set $\mathbf{m}_\tau := \mathsf{exp}(\tau)$.

$\mathbf{m}_\tau \notin \mathsf{EXP}(\tau)$. By Lemma 2.2, we have $\mathsf{ct}(\psi \cdot \mathbf{m}_\tau - \tau) \neq \mathbf{0}^n$. Note that verifier accepts and $\mathbb{w}$ is valid for $\mathcal{R}'$, so $\mathsf{ct}(\psi \cdot b) = a$ where $a = \langle \tau, \mathsf{tensor}(\mathbf{r})\rangle$, $\mathsf{ct}(\psi \cdot b) = \langle \mathsf{ct}(\psi \cdot \mathbf{m}_\tau), \mathsf{tensor}(\mathbf{r})\rangle$. By the binding property and Lemma 2.6 (over $\mathcal{C}$), this happens with probability at most $\epsilon_{\mathsf{bind}} + \log n/|\mathcal{C}|$. In sum, with probability at most ϵ'_{rg}, the adversary succeeds while the extractor's output is invalid for $\mathcal{R}'_{\mathsf{rg}}$. Thus the claim holds. □

Next, we are ready to prove that a committed *ring* vector $\mathbf{f} \in R_q^n$ lies within a specific range. Naively, we can commit to the coefficient vectors $\mathsf{cf}(\mathbf{f}) \in \mathbb{Z}_q^{n\times d}$ and run the above protocol in parallel d times. However, this incurs high communication costs, as the input includes d commitments of total size $R_q^{\kappa\times d}$, which affects verifier circuit complexity for the Fiat-Shamir transform. To address this, we optimize communication via double commitments.

Notation. Given $\mathbf{f} \in R_q^n$ where $\|\mathbf{f}\|_\infty < B = (d')^k$, denote by

$$\mathbf{D_f} = [\mathbf{D}_{\mathbf{f},0}, \ldots, \mathbf{D}_{\mathbf{f},k-1}] = \mathbf{G}^{-1}_{d',k}(\mathsf{cf}(\mathbf{f})) \in \mathbb{Z}_q^{n\times dk} \tag{13}$$

the decomposition matrix of $\mathsf{cf}(\mathbf{f})$ so that $\|\mathbf{D_f}\|_\infty < d'$. Here $\mathbf{D}_{\mathbf{f},i} \in \mathbb{Z}_q^{n\times d}$ for all $i \in [k]$. Let $\mathbf{M_f} \in \mathsf{EXP}(\mathbf{D_f}) \subseteq \mathcal{M}^{n\times dk}$. Consider the double commitment $(\!|\mathbf{M_f}|\!) = [\![\tau_\mathbf{D}]\!]$ where $\tau_\mathbf{D} := \Phi^{-1}([\![\mathbf{M_f}]\!]) \in (-d', d')^n$ is the decomposed version of $[\![\mathbf{M_f}]\!]$. Let $\mathbf{m}_\tau \in \mathsf{EXP}(\tau_\mathbf{D}) \subseteq \mathcal{M}^n$. We consider a range-check relation

$$\mathcal{R}_{\mathsf{rg},B} := \left\{ (\mathbb{x}, \mathbb{w}) \;:\; \begin{array}{l} \mathbb{x} = (\mathsf{cm}_\mathbf{f}, C_{\mathbf{M_f}}, \mathsf{cm}_{\mathbf{m}_\tau} \in R_q^\kappa), \\ \mathbb{w} = [\tau_\mathbf{D}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{M_f}] \in \mathbb{Z}_q^n \times R_q^{n\times(2+dk)} \quad \text{s.t.} \\ \mathsf{cf}(\mathbf{f}) \in (-B, B)^{n\times d} \;\wedge\; (C_{\mathbf{M_f}}, \mathbf{M_f}) \in \mathcal{R}_{\mathsf{m,in}} \\ (\mathbf{M_f} \in \mathsf{EXP}(\mathbf{D_f})) \;\wedge\; ((C_{\mathbf{M_f}}, \mathsf{cm}_{\mathbf{m}_\tau}), (\tau_\mathbf{D}, \mathbf{m}_\tau)) \in \mathcal{R}'_{\mathsf{rg}} \\ (C_{\mathbf{M_f}}, (\tau_\mathbf{D}, \mathbf{M_f})) \in \mathcal{R}_{\mathsf{dopen},dk} \end{array} \right\} \tag{14}$$

The reduction of knowledge is from $\mathcal{R}_{\mathsf{rg},B}$ to the output relation

$$\mathcal{R}_{\mathsf{dcom}} := \left\{ (\mathbb{x}, \mathbb{w}) \;:\; \begin{array}{l} \mathbb{x} = \left(\mathsf{cm}_\mathbf{f}, C_{\mathbf{M_f}}, \mathsf{cm}_{\mathbf{m}_\tau} \in R_q^\kappa, \mathbf{r} \in \mathcal{C}^{\log n}, \mathbf{e} \in R_q^{3+dk}\right), \\ \mathbb{w} = [\tau_\mathbf{D}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{M_f}] \in \mathbb{Z}_q^n \times R_q^{n\times(2+dk)} \quad \text{s.t.} \\ [\tau_\mathbf{D}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{M_f}]^\top \mathsf{tensor}(\mathbf{r}) = \mathbf{e} \\ (\mathsf{cm}_\mathbf{f}, \mathbf{f}) \in \mathcal{R}_{\mathsf{open}} \;\wedge\; (\mathsf{cm}_{\mathbf{m}_\tau}, \mathbf{m}_\tau) \in \mathcal{R}_{\mathsf{open}} \\ (C_{\mathbf{M_f}}, (\tau_\mathbf{D}, \mathbf{M_f})) \in \mathcal{R}_{\mathsf{dopen},dk} \end{array} \right\} \tag{15}$$

In words, given instance the commitment $\mathsf{cm}_\mathbf{f}$, the helper commitments $C_{\mathbf{M_f}}, \mathsf{cm}_{\mathbf{m}_\tau}$, the witness $\mathbf{f}$, and the helper witness vectors $\tau_\mathbf{D}, \mathbf{m}_\tau$ and matrix $\mathbf{M_f}$, the relation $\mathcal{R}_{\mathsf{rg},B}$ checks that (i) $\mathbf{f}$ is an opening to $\mathsf{cm}_\mathbf{f}$; (ii) $\mathbf{m}_\tau$ is an opening to $\mathsf{cm}_{\mathbf{m}_\tau}$ and $(\tau_\mathbf{D}, \mathbf{M_f})$ is an opening to the double commitment $C_{\mathbf{M_f}}$; (iii) $\mathbf{M_f}$, $\mathbf{m}_\tau$ are in $\mathsf{EXP}(\mathbf{D_f}) \subseteq \mathcal{M}^{n\times dk}$ and $\mathsf{EXP}(\tau_\mathbf{D}) \subseteq \mathcal{M}^n$ respectively ($\mathsf{EXP}(\cdot)$ is defined in (3)); (iv) $\tau_\mathbf{D}$ is a $\mathbb{Z}_q$-vector in $(-d', d')^n$; and (v) the coefficients of $\mathbf{f}$ lie within $(-B, B)$. This is equivalent to a simpler relation that only checks (i) and (v) for instance $\mathsf{cm}_{\mathbf{f}}$ and witness $\mathbf{f}$, but we put all helper information into the statement to simplify the protocol and their analysis.

Similarly, the output relation $\mathcal{R}_{\mathsf{dcom}}$ checks that, for instance $(\mathsf{cm}_{\mathbf{f}}, C_{\mathbf{M}_{\mathbf{f}}}, \mathsf{cm}_{\mathbf{m}_\tau}, \mathbf{r}, \mathbf{e})$, the witnesses $\mathbf{f}, \mathbf{m}_\tau$ are openings to $\mathsf{cm}_{\mathbf{f}}, \mathsf{cm}_{\mathbf{m}_\tau}$ and $(\tau_{\mathbf{D}}, \mathbf{M}_{\mathbf{f}})$ is an opening to $C_{\mathbf{M}_{\mathbf{f}}}$, and the matrix-vector product $[\tau_{\mathbf{D}}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{M}_{\mathbf{f}}]^\top \mathsf{tensor}(\mathbf{r}) = \mathbf{e}$ holds.

The protocol. Intuitively, the protocol Π_{rgchk} does the following:

1. check that $\mathbf{M}_{\mathbf{f}} \in \mathsf{EXP}(\mathbf{D}_{\mathbf{f}})$ and $\mathbf{m}_\tau \in \mathsf{EXP}(\tau_{\mathbf{D}})$ are well-formed via the monomial set tests;
2. verify that each entry of $\mathsf{cf}(\mathbf{f})$, $\tau_{\mathbf{D}}$ matches the constant term of the corresponding entry in $\sum_{i\in[k]} d'^i \cdot (\psi \cdot \mathbf{M}_{\mathbf{f},i})$ and $\psi \cdot \mathbf{m}_\tau$; by Lemma 2.2, this implies that $\mathsf{cf}(\mathbf{f})$, $\tau_{\mathbf{D}}$ are within the ranges.

Construction 4.4. On input $\mathbb{x} = (\mathsf{cm}_{\mathbf{f}}, C_{\mathbf{M}_{\mathbf{f}}}, \mathsf{cm}_{\mathbf{m}_\tau}) \in R_q^{\kappa\times 3}$, $\mathbb{w} = [\tau_{\mathbf{D}}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{M}_{\mathbf{f}}] \in \mathbb{Z}_q^n \times R_q^{n\times(2+dk)}$ where $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{rg},B}$, the protocol Π_{rgchk} proceeds as follows:

1. Run a batched protocol Π_{mon} (Construction 4.2) w.r.t. $\mathbf{M}_{\mathbf{f}} = [\mathbf{M}_{\mathbf{f},0} \in R_q^{n\times d}, \dots, \mathbf{M}_{\mathbf{f},k-1}] \in \mathsf{EXP}(\mathbf{D}_{\mathbf{f}}) \subseteq \mathcal{M}^{n\times dk}$ and $\mathbf{m}_\tau \in \mathsf{EXP}(\tau_{\mathbf{D}}) \subseteq \mathcal{M}^n$. The output statements are
$$(\mathbb{x} = (\mathsf{cm}_{\mathbf{m}_\tau}, \mathbf{r}, b \in R_q), \mathbb{w} = \mathbf{m}_\tau) \in \mathcal{R}_{\mathsf{m,out}}$$
$$(\mathbb{x} = (C_{\mathbf{M}_{\mathbf{f}}}, \mathbf{r}, \mathbf{e} \in R_q^{dk}), \mathbb{w} = \mathbf{M}_{\mathbf{f}}) \in \mathcal{R}_{\mathsf{m,out}}$$
where $b = \langle \mathbf{m}_\tau, \mathsf{tensor}(\mathbf{r})\rangle$. For every $i \in [k]$, denote by
$$\mathbf{u}_i := (\mathbf{M}_{\mathbf{f},i})^\top \mathsf{tensor}(\mathbf{r}) = \mathbf{e}[di, d(i+1)) \in R_q^d\,.$$
2. $\mathsf{P} \to \mathsf{V}$: Send $\mathbf{v} := \mathsf{cf}(\mathbf{f})^\top \mathsf{tensor}(\mathbf{r}) \in \mathcal{C}^d$ and $a := \langle \tau_{\mathbf{D}}, \mathsf{tensor}(\mathbf{r})\rangle \in \mathcal{C}$
3. V : Let $\psi := \sum_{i\in[1,d')} i \cdot (X^{-i} + X^i) \in R_q$. Check that $\mathsf{ct}(\psi \cdot b) \overset{?}{=} a$ and
$$\mathsf{ct}\left(\psi \cdot (\mathbf{u}_0 + d'\mathbf{u}_1 + \cdots + d'^{k-1}\mathbf{u}_{k-1})\right) \overset{?}{=} \mathbf{v} \tag{16}$$
Abort and return $\perp$ if the checks fail.
4. Denote by $\hat{v} := \sum_{i\in[d]} \mathbf{v}_i X^i \in R_q$. Return the reduced statement
$$(\mathbb{x}_o = (\mathsf{cm}_{\mathbf{f}}, C_{\mathbf{M}_{\mathbf{f}}}, \mathsf{cm}_{\mathbf{m}_\tau}, \mathbf{r}, (a, b, \hat{v}, \mathbf{u}_0, \dots, \mathbf{u}_{k-1})), \mathbb{w}_o = [\tau_{\mathbf{D}}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{M}_{\mathbf{f}}]) \in \mathcal{R}_{\mathsf{dcom}}\,.$$

Remark 4.4. Π_{rgchk} can be extended to support multiple input instances by batching the execution of Π_{mon} with the same challenge $\mathbf{r} \in \mathcal{C}^{\log n}$.

Theorem 4.2. *Π_{rgchk} is a reduction of knowledge from $\mathcal{R}_{\mathsf{rg},B}$ to $\mathcal{R}_{\mathsf{dcom}}$.*

Proof. Public reducibility is trivial. The claim follows from the lemmas below. □

Lemma 4.6. *Π_{rgchk} is perfectly complete.*

Proof. For every input statement in $\mathcal{R}_{\mathsf{rg},B}$, we show that the verifier checks pass and the prover's output is in $\mathcal{R}_{\mathsf{dcom}}$.

First, the prover's output is in $\mathcal{R}_{\mathsf{dcom}}$: $\{\mathbf{u}_i\}_{i\in[k]}$, a, b sent by the honest prover are exactly the claimed values in $\mathcal{R}_{\mathsf{dcom}}$. And the sent $\mathbf{v}$ satisfies that $\mathbf{v} = \mathsf{cf}(\mathbf{f})^\top\mathsf{tensor}(\mathbf{r})$. Thus, $\hat{v} := \sum_{i\in[d]} \mathbf{v}_i X^i = \langle \mathbf{f}, \mathsf{tensor}(\mathbf{r})\rangle$ as required given that $\mathsf{tensor}(\mathbf{r})$ is in $\mathcal{C}^n = \mathbb{Z}_q^n$.

Next, we show that the verifier accepts: First, $\mathbf{m}_\tau$ is in $\mathsf{EXP}(\tau_{\mathbf{D}}) \subseteq \mathcal{M}^n$ and the matrix $\mathbf{M}_{\mathbf{f}}$ is in $\mathsf{EXP}(\mathbf{D}_{\mathbf{f}}) \subseteq \mathcal{M}^{n\times dk}$. By completeness of Π_{mon}, the monomial set checks pass. Also note that $\tau_{\mathbf{D}} \in (-d', d')^n$, by Lemma 2.2, we have $\mathsf{ct}(\psi\cdot\mathbf{m}_\tau) = \tau_{\mathbf{D}}$. Since $\mathsf{tensor}(\mathbf{r}) \in \mathcal{C}^n = \mathbb{Z}_q^n$, we have $\mathsf{ct}(\psi\cdot b) = a$. By definition of the decomposed matrix $\mathbf{D}_{\mathbf{f}}$, with a similar argument, Eq. (16) holds and thus the verifier accepts. □

Lemma 4.7. *If $[\![\cdot]\!]$ is binding with binding error ϵ_{bind}, then Π_{rgchk} is knowledge sound with knowledge error $\epsilon_{\mathsf{rg}} := \epsilon_{\mathsf{mon},dk+1} + \epsilon_{\mathsf{bind}} + \log n/|\mathcal{C}|$.*

Proof. The extractor Ext simulates the execution with adversary $(\mathcal{A}, \mathsf{P}^*)$ and returns P^*'s output witness if the verifier accepts. Otherwise Ext returns $\perp$. It is clear that Ext has polynomial time complexity.

Let $\mathbb{x}$, $\mathbb{x}_o$ denote the instances for the input and output relations. Conditioned on the verifier accepts, let $\mathbb{w} = (\mathbf{f}, \mathbf{M}_{\mathbf{f}}, \tau_{\mathbf{D}}, \mathbf{m}_\tau)$ be the extractor's output. Suppose $(\mathbb{x}, \mathbb{w})$ is not in $\mathcal{R}_{\mathsf{rg},B}$ but $(\mathbb{x}_o, \mathbb{w})$ is in $\mathcal{R}_{\mathsf{dcom}}$. Consider two cases.

Case 1: $\mathbf{M}_{\mathbf{f}} \in R_q^{n\times dk}$ or $\mathbf{m}_\tau \in R_q^n$ are invalid openings, or some of their entries are outside the monomial set. By knowledge soundness of Π_{mon}, this happens with probability at most $\epsilon_{\mathsf{mon},dk+1}$.

Case 2: $\mathbf{M}_{\mathbf{f}}$ is in $\mathcal{M}^{n\times dk}$ and $\mathbf{m}_\tau$ is in $\mathcal{M}^n$ and they are valid openings, but

$$\mathsf{cf}(\mathbf{f}) \notin (-B, B)^{n\times d} \vee \tau_{\mathbf{D}} \notin (-d', d')^n \vee \mathbf{M}_{\mathbf{f}} \notin \mathsf{EXP}(\mathbf{D}_{\mathbf{f}}) \vee \mathbf{m}_\tau \notin \mathsf{EXP}(\tau)\,.$$

For all $i \in [k]$, let $\mathbf{M}_{\mathbf{f},i} \in \mathcal{M}^{n\times d}$ be the $(di+1)$-th to $(di+d)$-th columns of $\mathbf{M}_{\mathbf{f}}$. By Lemma 2.2, we have $\mathsf{ct}(\psi\cdot\mathbf{m}_\tau - \tau_{\mathbf{D}}) \neq \mathbf{0}^n$ or

$$\mathsf{ct}\left(\psi\cdot(\mathbf{M}_{\mathbf{f},0} + d'\mathbf{M}_{\mathbf{f},1} + \cdot + d'^{k-1}\mathbf{M}_{\mathbf{f},k-1})\right) \neq \mathsf{cf}(\mathbf{f})\,.$$

Note that $(\mathbb{x}_o, \mathbb{w})$ is in $\mathcal{R}_{\mathsf{dcom}}$ and the verification passed, so $\mathsf{ct}(\psi\cdot b) = a$ and (16) holds, where $a = \langle\tau_{\mathbf{D}}, \mathsf{tensor}(\mathbf{r})\rangle$ and $\mathsf{ct}(\psi\cdot b) = \langle\mathsf{ct}(\psi\cdot\mathbf{m}_\tau), \mathsf{tensor}(\mathbf{r})\rangle$. By the binding property of $[\![\cdot]\!]$ and Lemma 2.6, this happens with probability at most $\epsilon_{\mathsf{bind}} + \log n/|\mathcal{C}|$. In sum, the knowledge error is $\epsilon_{\mathsf{rg}} = \epsilon_{\mathsf{mon},dk+1} + \epsilon_{\mathsf{bind}} + \log n/|\mathcal{C}|$. □

4.4 Commitment transformation

Double commitments are not linearly homomorphic, thus statements involving double commitments are hard to fold using random linear combinations. This section introduces a technique that simplifies such statements by reducing them to statements involving only linear commitments to *smaller* witnesses. The core idea is to use sumchecks to ensure consistency between the double commitment of a large witness matrix $\mathbf{M}$ and a linear commitment to a folded version of $\mathbf{M}$.

Specifically, let $B := (d')^k$ where $k \in \mathbb{N}$. Fix $b \in \mathbb{N}$ and folding challenge set $\bar{\mathcal{S}} \subset R_q$ such that $[\![\cdot]\!]$ is $(b, \mathcal{S} := \bar{\mathcal{S}} - \bar{\mathcal{S}})$-binding and $b \geq 2\|\bar{\mathcal{S}}\|_{\mathsf{op}}(d' + 1 + B + dk)$. The protocol Π_{cm} reduces relation $\mathcal{R}_{\mathsf{rg},B}$ from (14) to

$$\mathcal{R}_{\mathsf{com}} := \left\{ (\mathbb{x}, \mathbb{w}) \ : \ \begin{array}{c} \mathbb{x} = \left(\mathsf{cm}_{\mathbf{g}} \in R_q^{\kappa}, \mathbf{r}_o \in \mathsf{M}_{\mathcal{C}}^{\log n}, v_o \in \mathsf{M}_q\right), \quad \mathbb{w} = \mathbf{g} \in R_q^n \ \text{ s.t.} \\ \langle \mathbf{g}, \mathsf{tensor}(\mathbf{r}_o)\rangle = v_o \ \wedge \ \|\mathbf{g}\|_\infty < b/2 \ \wedge \ (\mathsf{cm}_{\mathbf{g}}, \mathbf{g}) \in \mathcal{R}_{\mathsf{open}} \end{array} \right\} \tag{17}$$

where $\mathsf{M}_{\mathcal{C}} := \mathcal{C} \times \mathcal{C}$ and $\mathsf{M}_q := R_q \times R_q$. We can understand $\mathcal{R}_{\mathsf{com}}$ as a special case of the general linear relation $\mathcal{R}_{\mathsf{lin},b/2}$, where the only linear check is the evaluation statement $\langle \mathbf{g}, \mathsf{tensor}(\mathbf{r}_o)\rangle = v_o$.

The protocol Π_{cm}, which we describe next, is the central protocol that we use in the next section to construct our folding scheme. It uses the range check protocol Π_{rgchk} from the previous section as a subroutine.

The protocol Π_{cm}. For simplicity, assume that $n = \kappa d^2 k \ell$ and κ, k, ℓ, d are powers of two (in Remark 4.8 we explain how to support smaller n). Recall that $\Phi^{-1} : R_q^{\kappa \times dk} \to (-d', d')^n$ is the injective map defined in Section 4.1 that decomposes a matrix in $R_q^{\kappa \times dk}$ to a low-norm vector in $(-d, d')^n$; and $\Phi : (-d', d')^n \to R_q^{\kappa \times dk}$ is the map such that $\Phi(\Phi^{-1}(\mathbf{D})) = \mathbf{D}$ for all $\mathbf{D} \in R_q^{\kappa \times dk}$.

Construction 4.5. The protocol takes as input an instance-witness pair $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{rg},B}$ where

$$\mathbb{x} = (\mathsf{cm}_{\mathbf{f}}, C_{\mathbf{M_f}} = [\![\tau_{\mathbf{D}}]\!], \mathsf{cm}_{\mathbf{m}_\tau}) \in R_q^{\kappa \times 3}, \quad \mathbb{w} = [\tau_{\mathbf{D}}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{M_f}] \in \mathbb{Z}_q^n \times (R_q^n)^2 \times R_q^{n \times dk}$$

and $\mathbf{m}_\tau, \mathbf{M_f}$ satisfy[7] $\mathbf{m}_\tau \in \mathsf{EXP}(\tau_{\mathbf{D}}) \subseteq \mathcal{M}^n$ and $\mathbf{M_f} \in \mathsf{EXP}(\mathbf{D_f}) \subseteq \mathcal{M}^{n \times dk}$. Here $\mathbf{D_f}$ is a decomposition of $\mathbf{f}$ as defined as in (13). The protocol Π_{cm} proceeds as follows:

1. $\mathsf{P} \leftrightarrow \mathsf{V}$: Run Π_{rgchk} to reduce the input statement in $\mathcal{R}_{\mathsf{rg},B}$ to that in $\mathcal{R}_{\mathsf{dcom}}$ from (15). Let $\mathbf{r} \in \mathcal{C}^{\log n}$ and $\mathbf{e} \in R_q^{3+dk}$ be the challenge and evaluations.

2. $\mathsf{V} \to \mathsf{P}$: Send folding challenges $\mathbf{s} \xleftarrow{\mathrm{R}} \bar{\mathcal{S}}^3$ and $\mathbf{s}' \xleftarrow{\mathrm{R}} \bar{\mathcal{S}}^{dk}$.

3. $\mathsf{P} \to \mathsf{V}$: Send $[\![\mathbf{h}]\!] := [\![\mathbf{M_f}]\!]\mathbf{s}' = [\![\mathbf{M_f}\mathbf{s}']\!] \in R_q^{\kappa}$.

[7]In practice, a standard implementation will set $\mathbf{m}_\tau := \mathsf{exp}(\tau)$ and $\mathbf{M_f} := \mathsf{exp}(\mathbf{D_f})$.

4. $\mathsf{V} \to \mathsf{P}$: Send challenges $(\mathbf{c}^{(0)}, \mathbf{c}^{(1)}) \xleftarrow{\mathrm{R}} \mathcal{C}^{\log \kappa} \times \mathcal{C}^{\log \kappa}$.

5. $\mathsf{P} \leftrightarrow \mathsf{V}$: Run two sumcheck protocols in parallel, where each of them checks the *same* sumcheck claims below in a batch[8]:

   - Set $u := \langle \mathbf{e}[3, 3+dk), \mathbf{s}' \rangle \in R_q$; the relation to be checked is

     $$[\tau_{\mathbf{D}}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{h}]^\top \cdot \mathsf{tensor}(\mathbf{r}) \overset{?}{=} (\mathbf{e}[0,2], u). \tag{18}$$

     This left hand side represents four sums each over n elements. Each sum can be compared to the corresponding value on the right hand side using a degree-2 sumcheck claim that uses the multilinear extensions of $\tau_{\mathbf{D}}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{h}$ and $\mathsf{tensor}(\mathbf{r})$. In total, (18) represents four sumcheck claims.

   - For every $z \in [2]$, define a vector $\mathbf{t}^{(z)} \in R_q^n$

     $$\mathbf{t}^{(z)} = \mathsf{tensor}(\mathbf{c}^{(z)}) \otimes \mathbf{s}' \otimes (1, d', \ldots, d'^{\ell-1}) \otimes (1, X, \ldots, X^{d-1})\,. \tag{19}$$

     Let $\widetilde{\mathbf{t}^{(z)}}$ denote $\mathbf{t}^{(z)}$'s multilinear extension which can be evaluated[9] by the verifier in time $O(\log \kappa + dk + \ell)$. Check the two degree-2 sumcheck claims:[10]

     $$\sum_{i \in [n]} \widetilde{\tau_{\mathbf{D}}}(\langle i \rangle) \cdot \widetilde{\mathbf{t}^{(z)}}(\langle i \rangle) \overset{?}{=} \langle \mathsf{tensor}(\mathbf{c}^{(z)}), [\![\mathbf{h}]\!] \rangle \in R_q \quad \text{for } z = 0, 1. \tag{20}$$

     If $\tau_{\mathbf{D}} \in (-d', d')^n$, this is equivalent to checking that (See Remark 4.5)

     $$\langle \mathsf{tensor}(\mathbf{c}^{(z)}), \Phi(\tau_{\mathbf{D}})\mathbf{s}' \rangle \overset{?}{=} \langle \mathsf{tensor}(\mathbf{c}^{(z)}), [\![\mathbf{h}]\!] \rangle\,. \tag{21}$$

     In total, (20) represents two sumcheck claims.

   The verifier can verify all six sumcheck claims from (18) and (20) as a batch using the sumcheck batching technique from Remark 2.4. In Remark 4.6 we explain that these six sumchecks can be treated as $6d$ sumcheck claims over $\mathbb{Z}_q$ that can all be batched into a single sumcheck over $\mathbb{Z}_q$ (or an extension field of $\mathbb{Z}_q$).

   P and V run two parallel sumcheck protocols for this single batch claim. The need for two parallel sumchecks in explained in Remark 4.9. Let $\mathbf{r}_o \xleftarrow{\mathrm{R}} (\mathcal{C} \times \mathcal{C})^{\log n}$ denote the corresponding sumcheck challenges. Then, since V can evaluate $eq(\mathbf{r}, \cdot)$, $\widetilde{\mathbf{t}^{(0)}}$, $\widetilde{\mathbf{t}^{(1)}}$ at both $\mathcal{C}^{\log n}$ points in $\mathbf{r}_o$ by itself, the two sumcheck executions reduce to the evaluation claims

   $$[\tau_{\mathbf{D}}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{h}]^\top \cdot \mathsf{tensor}(\mathbf{r}_o) \overset{?}{=} \mathbf{e}_o \tag{22}$$

[8] The two parallel sumcheck executions use independent verifier challenges.

[9] $\mathbf{t}^{(z)}$ is a tensor product defined in (19), so $\widetilde{\mathbf{t}^{(z)}}(\mathbf{r}) = \langle \mathbf{t}^{(z)}, \mathsf{tensor}(\mathbf{r}) \rangle$ can be efficiently evaluated due to the mixed-product property of tensors.

[10] $\langle \mathsf{tensor}(\mathbf{c}^{(z)}), [\![\mathbf{h}]\!] \rangle$ is directly computed by the verifier in time $O(\kappa)$.

for some $\mathbf{e}_o \in (R_q \times R_q)^4$, where $\tau_{\mathbf{D}}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{h} \in R_q^n$ are the committed witnesses. These are eight evaluation claims: a claim for each of $\tau_{\mathbf{D}}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{h}$ for each of the two points in $\mathbf{r}_o$. The next two steps aggregate these eight claims into two claims.

6. V : Compute $\mathsf{cm}_{\mathbf{g}} \in R_q^\kappa$ and $v_o \in R_q^2$ where

$$(\mathsf{cm}_{\mathbf{g}}, v_o) := \mathbf{s}_0 \cdot (C_{\mathbf{M_f}}, \mathbf{e}_{o,0}) + \mathbf{s}_1 \cdot (\mathsf{cm}_{\mathbf{m}_\tau}, \mathbf{e}_{o,1}) + \mathbf{s}_2 \cdot (\mathsf{cm}_{\mathbf{f}}, \mathbf{e}_{o,2}) + (\llbracket \mathbf{h} \rrbracket, \mathbf{e}_{o,3}) \in R_q^{\kappa+2}$$

   Here we treat $\mathbf{e}_o$ as a matrix in $R_q^{2\times 4}$ so that $\mathbf{e}_{o,i} \in R_q^2$ is its i'th column, for $i \in [4]$.

7. P : Compute $\mathbf{g} := \mathbf{s}_0 \cdot \tau_{\mathbf{D}} + \mathbf{s}_1 \cdot \mathbf{m}_\tau + \mathbf{s}_2 \cdot \mathbf{f} + \mathbf{h} \in R_q^n$.

8. Output the reduced statement $(\mathbb{x} = (\mathsf{cm}_{\mathbf{g}}, \mathbf{r}_o, v_o), \mathbb{w} = \mathbf{g}) \in \mathcal{R}_{\mathsf{com}}$.

Remark 4.5. To see why Eq. (20) and Eq. (21) are equivalent, it suffices to check $\langle \tau_{\mathbf{D}}, \mathbf{t}^{(z)} \rangle = \langle \mathsf{tensor}(\mathbf{c}^{(z)}), \Phi(\tau_{\mathbf{D}})\mathbf{s}' \rangle$. We can understand $\tau_{\mathbf{D}} \in (-d', d')^{\kappa d^2 k \ell}$ as a four-dimensional array $T \in (-d', d')^{\kappa \times dk \times d \times \ell}$ such that $T_{i,j,o,p} = \tau_{\mathbf{D}}[i \cdot (d^2 k \ell) + j \cdot (d\ell) + o \cdot \ell + p]$. Then we have

$$\begin{aligned}
\langle \tau_{\mathbf{D}}, \mathbf{t}^{(z)} \rangle &= \langle \tau_{\mathbf{D}}, \mathsf{tensor}(\mathbf{c}^{(z)}) \otimes \mathbf{s}' \otimes (1, d', \ldots, d'^{\ell-1}) \otimes (1, X, \ldots, X^{d-1}) \rangle \\
&= \sum_{i \in [\kappa], j \in [dk], o \in [d], p \in [\ell]} T_{i,j,o,p} \cdot \mathsf{tensor}(\mathbf{c}^{(z)})_i \cdot \mathbf{s}'_j \cdot d'^o \cdot X^p \\
&= \langle \mathsf{tensor}(\mathbf{c}^{(z)}), \sum_{j \in [dk]} \left(\sum_{o \in [d], p \in [\ell]} T_{*,j,o,p} \cdot d'^o \cdot X^p \right) \cdot \mathbf{s}'_j \rangle \\
&= \langle \mathsf{tensor}(\mathbf{c}^{(z)}), \sum_{j \in [dk]} \Phi(\tau_{\mathbf{D}})_{*,j} \cdot \mathbf{s}'_j \rangle = \langle \mathsf{tensor}(\mathbf{c}^{(z)}), \Phi(\tau_{\mathbf{D}})\mathbf{s}' \rangle \,.
\end{aligned}$$

Remark 4.6 (Efficient Instantiations of Sumchecks). Step 5 verifies six sumcheck claims over R_q in a batch. Observe that each sumcheck claim over R_q can be understood as d sumcheck claims over $\mathbb{Z}_q$: In (18), the elements of $\mathsf{tensor}(\mathbf{r})$ are in $\mathbb{Z}_q$; similarly, in (20), the elements of $\tau_{\mathbf{D}}$ are in $\mathbb{Z}_q$. Thus there are only scalar multiplications rather than R_q-multiplications in the degree-2 sumchecks. Hence the sumcheck claim over R_q can be understood as d coefficient-wise sumcheck claims over $\mathbb{Z}_q$.

In sum, the six sumcheck claims over R_q can then be understood as $6d$ sumcheck claims over $\mathbb{Z}_q$. We can batch the $6d$ sumcheck claims over $\mathbb{Z}_q$ to a single sumcheck claim over $\mathbb{Z}_q$ (or its field extension) via random linear combination. Thus Π_{cm} only needs to run two parallel sumcheck protocols over $\mathbb{Z}_q$ (or an extension field of $\mathbb{Z}_q$).

Remark 4.7 (Optimizing communication). The communication is dominated by $\mathbf{e}' := \mathbf{e}[3, 3+dk) \in R_q^{dk}$, which consists of dk "full" R_q-elements. In applications such as IVC, this affects the recursive verifier circuit size as the Fiat-Shamir transform is typically the bottleneck.[11] Fortunately, we can use the same compression trick for $\llbracket \mathbf{M}_f \rrbracket$ to compress $\mathbf{e}'$.

[11]E.g., a 2-to-1 hash over R_q takes more than 100 R1CS constraints over R_q.

Specifically, we decompose $\mathbf{e}'$ into $\tau_{\mathbf{e}} \in (-d', d')^{n'}$ using an injective map similar to Φ^{-1}, where $n' = nk/\kappa$. For simplicity, we assume that $n' = n$ and reuse the notation Φ^{-1} and Φ, but it easily generalizes to any $n' \leq n$. Note that $\Phi(\tau_{\mathbf{e}}) = \mathbf{e}'$.

In the protocol Π_{cm}, the prover replaces $\mathbf{e}'$ with $[\![\tau_{\mathbf{e}}]\!]$ in the execution of Π_{rgchk}. Specifically, when running Π_{mon} (Construction 4.2) to check that $\mathbf{M_f}$ is in $\mathcal{M}^{n\times dk}$, instead of sending $\mathbf{e}'$ in the clear, the prover sends the commitment $[\![\tau_{\mathbf{e}}]\!]$ and the claimed values $(\mathbf{v}_e, \mathbf{v}'_e) := (\mathbf{e}'[\beta], \mathbf{e}'[\beta^2]) \in \mathcal{C}^{dk}$ (used in the check (12)). The rest of the protocol is the same as Π_{cm} except for the following changes:

- The prover proves that the committed $\tau_{\mathbf{e}}$ is in range $(-d', d')^{n'}$. This can be achieved by sending $[\![\mathsf{exp}(\tau_{\mathbf{e}})]\!]$ and running Construction 4.3, which is really efficient.
- In Step 5, the prover sends the claimed value of u and batches a sumcheck claim to prove that $\langle \Phi(\tau_{\mathbf{e}}), \mathbf{s}' \rangle = u$. (Recall that $\mathbf{e}' = \Phi(\tau_{\mathbf{e}})$.)
- In Step 5, the prover batches two sumcheck claims to prove that $\Phi(\tau_{\mathbf{e}})[\beta] = \mathbf{v}_e$ and $\Phi(\tau_{\mathbf{e}})[\beta^2] = \mathbf{v}'_e$. The idea is almost identical to that explained in Eq. (19), Eq. (20), except that the term $(1, \ldots, X^{d-1})$ in Eq. (19) is replaced with $(1, \beta, \ldots, \beta^{d-1})$ and $(1, \beta^2, \ldots, (\beta^2)^{d-1})$ respectively.
- In Step 5, the prover batches a sumcheck claim for the constant term check in Eq. (16). Specifically, let $\mathbf{v} \in \mathcal{C}^d$ be the claimed values in Eq. (16). The verifier sends $\mathbf{c}' \xleftarrow{R} \mathcal{C}^{\log d}$ and the prover sends $v' = \langle \mathbf{u}_0 + \cdots + d'^{k-1}\mathbf{u}_{k-1}, \mathsf{tensor}(\mathbf{c}') \rangle \in R_q$. Denote by $\mathbf{z} := (1, d', \ldots, d'^{k-1}) \otimes \mathsf{tensor}(\mathbf{c}') \in \mathcal{C}^{dk}$. It suffices to check that (i) $\mathsf{ct}(\psi \cdot v') = \langle \mathbf{v}, \mathsf{tensor}(\mathbf{c}') \rangle$, and (ii)
$$\langle \Phi(\tau_{\mathbf{e}}), \mathbf{z} \rangle = v' . \tag{23}$$
The inner product (23) can be represented as a degree-2 sumcheck using the same trick in Eq. (19), Eq. (20).
- At Step 2, V samples two more challenges $r_1, r_2 \in \bar{\mathcal{S}}$, and at Step 6 and 7, V and P fold $([\![\tau_{\mathbf{e}}]\!], [\![\mathsf{exp}(\tau_{\mathbf{e}})]\!]$ and $(\tau_{\mathbf{e}}, \mathsf{exp}(\tau_{\mathbf{e}}))$ into $\mathsf{cm}_{\mathbf{g}}$ and $\mathbf{g}$ respectively, using scalars r_1, r_2.

This optimization removes $\mathbf{e}'$ that has dk R_q-elements, at the cost of adding two commitments $[\![\tau_{\mathbf{e}}]\!], [\![\mathsf{exp}(\tau_{\mathbf{e}})]\!] \in R_q^{\kappa}$, one R_q-element b and $\mathcal{C}$-element a in the execution of Construction 4.3, $\mathbf{v}_e, \mathbf{v}'_e$ which has $2dk$ $\mathcal{C}$-elements, and $v' \in R_q, \mathbf{c}' \in \mathcal{C}^{\log d}$. In practice, $d \gg \kappa > k$, so this gives us a $\approx dk/(2\kappa)$-factor saving on communication.

Remark 4.8 (Supporting $n < \kappa d^2 k\ell$). Construction 4.5 requires the witness length n to be large enough, i.e., $n \geq \kappa d^2 k\ell$. In this remark, we explain how to support smaller[12] n. To illustrate the idea, consider an example where $d = (d^*)^2$ and $n = \kappa dk\ell d^* \ll \kappa d^2 k\ell$. We need to make two changes to the protocol Π_{cm}:

[12]Specifically, we only need n to be slightly larger than $\kappa dk\ell$.

First, we need to slightly change how we compute $\tau_{\mathbf{D}} = \Phi^{-1}([\![\mathbf{M_f}]\!])$. In Step 2 of Construction 4.1, instead of flattening the entire coefficient matrix $\mathsf{cf}(\mathbf{M}''_{\mathbf{f}}) \in \mathbb{Z}_q^{\kappa dk\ell \times d}$ into a $\mathbb{Z}_q$-vector, we split $\mathbf{C} := \mathsf{cf}(\mathbf{M}''_{\mathbf{f}})$ into d^* parts, i.e.,

$$\mathbf{C} = [\mathbf{C}_0, \mathbf{C}_1, \dots, \mathbf{C}_{d^*-1}]$$

where for every $i \in [d^*]$, $\mathbf{C}_i \in \mathbb{Z}_q^{\kappa dk\ell \times d^*}$ and we denote by $\tau_i := \mathsf{flat}(\mathbf{C}_i) \in \mathbb{Z}_q^{n=\kappa dk\ell d^*}$. Finally, we set $\Phi^{-1}([\![\mathbf{M_f}]\!]) := \tau_{\mathbf{D}} = \sum_{i\in[d^*]} \tau_i X^i \in R_q^n$.

Recall that $C_{\mathbf{M_f}} = [\![\tau_{\mathbf{D}}]\!]$ is the double commitment in Π_{cm}. But now, the committed vector $\tau_{\mathbf{D}}$ is no longer in $(-d', d')^n$ but an R_q-vector whose elements have degree $d^* = \sqrt{d}$ and norm d'. This leads to the second change: Recall that we need to range-check the committed vector $\tau_{\mathbf{D}} \in R_q^n$. A naive approach is to send the linear commitment to a matrix $\mathbf{M}_\tau \in \mathcal{M}^{n \times d^*}$ and run monomial set checks. However, d^* may still be large. Fortunately, the commitment $[\![\mathbf{M}_\tau]\!] \in R_q^{\kappa \times d^*}$ is much shorter than $[\![\mathbf{M_f}]\!] \in R_q^{\kappa \times dk}$, i.e., $d^* \ll dk$ and $n = \kappa dk\ell d^* \geq \kappa d^* \ell d$. So the prover can compute $\tau' := \Phi^{-1}([\![\mathbf{M}_\tau]\!]) \in (-d', d)^n$ (following Construction 4.1 with no modification) and send the commitment $[\![\tau']\!]$, and then range-check τ' by running Construction 4.3. Additionally, the protocol checks the consistency between $[\![\tau']\!]$, $[\![\mathbf{h}']\!] := [\![\mathbf{M}_\tau]\!]\mathbf{s}''$ (where $\mathbf{s}'' \xleftarrow{R} \bar{\mathcal{S}}^{d^*}$) and $[\![\mathbf{M}_\tau]\!]$, using a sumcheck claim as in Eq. (20). Note that the sumcheck claim can be batched in Step 5.

There is a small tradeoff: instead of sending two helper commitments $C_{\mathbf{M_f}} = [\![\tau_{\mathbf{D}}]\!], \mathsf{cm}_{\mathbf{m}_\tau} = [\![\mathsf{exp}(\tau_{\mathbf{D}})]\!]$, we send three commitments $[\![\tau_{\mathbf{D}}]\!]$, $[\![\tau' = \Phi^{-1}([\![\mathbf{M}_\tau]\!])]\!]$, $[\![\mathsf{exp}(\tau')]\!]$ and an $\mathbf{h}$-commitment $[\![\mathbf{h}']\!]$ (that folds $[\![\mathbf{M}_\tau]\!]$).

Security analysis of Construction 4.5. Note that $\mathsf{cm}_{\mathbf{g}}$ is a random linear combination of the commitments $[C_{\mathbf{M_f}} = [\![\tau_{\mathbf{D}}]\!], \mathsf{cm}_{\mathbf{m}_\tau}, \mathsf{cm}_{\mathbf{f}}]$ and $[\![\mathbf{M_f}]\!]$. However, the prover never sends $[\![\mathbf{M_f}]\!]$, which is too large. Instead, the prover adaptively sends a folded commitment $[\![\mathbf{h}]\!] = [\![\mathbf{M_f}]\!]\mathbf{s}'$ after learning the challenge $\mathbf{s}'$. At a first glance, it is unclear how to enforce the prover to fix the witness $\mathbf{M_f}$ before learning the challenge. A key observation is that $C_{\mathbf{M_f}}$ was sent initially, which is claimed to be the double commitment to $\mathbf{M_f}$. By running a sumcheck to enforce the consistency between the instances $[\![\mathbf{h}]\!]$, $C_{\mathbf{M_f}}$ and the witness $[\![\mathbf{M_f}]\!]$, we can recover the binding of $\mathbf{M_f}$. Next, we formally prove the security of the scheme.

Theorem 4.3. *Let $k \in \mathbb{N}$ and $B := (d')^k$. Fix $b \in \mathbb{N}$ and $\bar{\mathcal{S}} \subseteq R_q$ such that $[\![\cdot]\!]$ is $(b, \mathcal{S} = \bar{\mathcal{S}} - \bar{\mathcal{S}})$-binding (defined in Section 4.1) and $b \geq 2\|\bar{\mathcal{S}}\|_{\mathsf{op}}(d' + 1 + B + dk)$. Then Π_{cm} is a reduction of knowledge from $\mathcal{R}_{\mathsf{rg},B}$ to $\mathcal{R}_{\mathsf{com}}$.*

Proof. Public reducibility is trivial. The theorem follows from the lemmas below. □

Lemma 4.8. *If $b \geq B' = 2\|\bar{\mathcal{S}}\|_{\mathsf{op}} \cdot (d' + 1 + B + dk)$, then Π_{cm} is perfectly complete.*

Proof. For every input instance $\mathbb{x} = (\mathsf{cm}_{\mathbf{f}} = [\![\mathbf{f}]\!], C_{\mathbf{M_f}} = [\![\tau_{\mathbf{D}}]\!], \mathsf{cm}_{\mathbf{m}_\tau} = [\![\mathbf{m}_\tau]\!])$ and witness $\mathbb{w} = [\tau_{\mathbf{D}}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{M_f}]$ such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{rg},B}$, we prove that the honest execution of Π_{cm} succeeds and the output statement is in $\mathcal{R}_{\mathsf{com}}$.

We first show that the verifier accepts and the evaluation claim in $\mathcal{R}_{\mathsf{com}}$ (Eq. (17)) holds. By completeness of Π_{rgchk}, Step 1 executes correctly and the output statement is in $\mathcal{R}_{\mathsf{dcom}}$ from (15). This implies that the first 4 sumcheck claims at Step 5 hold. Moreover, since $(C_{\mathbf{M_f}}, (\tau_{\mathbf{D}}, \mathbf{M_f})) \in \mathcal{R}_{\mathsf{dopen},dk}$ (defined in (8)), we have $[\![\tau_{\mathbf{D}}]\!] = C_{\mathbf{M_f}}$ and $\Phi(\tau_{\mathbf{D}}) = [\![\mathbf{M_f}]\!]$, thus

$$\Phi(\tau_{\mathbf{D}})\mathbf{s}' = [\![\mathbf{M_f}]\!]\mathbf{s}' = [\![\mathbf{h}]\!]$$

and the last two sumcheck claims (at Step 5) hold. Therefore, the batched sumcheck executes correctly and the evaluation claims in (22) holds, which implies that the evaluation claim in $\mathcal{R}_{\mathsf{com}}$ also holds.

Next, we show that the norm constraint in $\mathcal{R}_{\mathsf{com}}$ holds (Eq. (17)), which also implies that $\mathbf{g}$ is a valid opening of $\mathsf{cm}_{\mathbf{g}}$. Note that $\|\tau_{\mathbf{D}}\|_\infty < d'$, the entries of $\mathbf{m}_\tau$, $\mathbf{M_f}$ are in the monomial set from (2), and $\|\mathbf{f}\|_\infty < B$. By definition of operator norms from (4) and Lemma 2.3, we have that

$$\|\mathbf{g}\|_\infty < d'\|\bar{\mathcal{S}}\|_{\mathsf{op}} + \|\bar{\mathcal{S}}\|_{\mathsf{op}} + B\|\bar{\mathcal{S}}\|_{\mathsf{op}} + dk\|\bar{\mathcal{S}}\|_{\mathsf{op}} = B'/2 \leq b/2\,.$$

Thus, the output statement is in $\mathcal{R}_{\mathsf{com}}$, which completes the proof. □

Before proving knowledge soundness, we review an adapted version of the coordinate-wise special soundnesss lemma from [FMN24]. Fix $\mu \in \mathbb{N}$ and let $\bar{\mathcal{S}}$ be a finite set. For two vectors $\mathbf{a}, \mathbf{b} \in \bar{\mathcal{S}}^\mu$, we say that $\mathbf{a} \equiv_i \mathbf{b}$ for $i \in [\mu]$ if $\mathbf{a}_i \neq \mathbf{b}_i$ and $\mathbf{a}_j = \mathbf{b}_j$ for all $j \in [\mu] \setminus \{i\}$.

Lemma 4.9 ([FMN24], Lemma 7.1)**.** *Fix $\mu \in \mathbb{N}$ and let $\bar{\mathcal{S}}$ be a finite set. Define $U := \bar{\mathcal{S}}^\mu$ and let $\Psi : U \times T \to \{0,1\}$ be any predicate. Let $\mathcal{A}$ be a probabilistic algorithm $\mathcal{A} : U \to T$ and let*

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

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

$$\epsilon_\Psi(\mathcal{A}) - \mu/|\bar{\mathcal{S}}| \tag{24}$$

outputs $\mu+1$ pairs $(\mathbf{u}_0, y_0)$, $(\mathbf{u}_1, y_1 \leftarrow \mathcal{A}(\mathbf{u}_1))$, ..., $(\mathbf{u}_\mu, y_\mu \leftarrow \mathcal{A}(\mathbf{u}_\mu))$ such that $\Psi(\mathbf{u}_i, y_i) = 1$ for all $i \in [\mu+1]$ and $\mathbf{u}_{i+1} \equiv_i \mathbf{u}_0$ for all $i \in [\mu]$. $\mathcal{E}$ calls $\mathcal{A}$ for $1+\mu$ times in expectation, and $1 + \mu(|\bar{\mathcal{S}}| - 1) < \mu|\bar{\mathcal{S}}|$ times in the worst case.

Next, we are ready to prove knowledge soundness.

Lemma 4.10. *Fix $k \in \mathbb{N}$ and $B = (d')^k$. Let $\bar{\mathcal{S}}$, $\mathcal{C}$ be challenge sets where $|\bar{\mathcal{S}}| = |\mathcal{C}| \geq 2^\lambda$. Define*

$$\epsilon_{\mathsf{sum}} := (2\log(n)/|\mathcal{C}|)^2\,.$$

and $\mu = 3 + dk$. Fix any $b \in \mathbb{N}$ such that $[\![\cdot]\!]$ is $(b, \mathcal{S} = \bar{\mathcal{S}} - \bar{\mathcal{S}})$-binding with binding error ϵ_{bind}, the protocol Π_{cm} has knowledge error

$$\epsilon_{\mathsf{cm},k} = \frac{\mu + dk}{|\bar{\mathcal{S}}|} + 3\epsilon_{\mathsf{bind}} + \epsilon'_{\mathsf{rg}} + \epsilon_{\mathsf{rg}} + 2\epsilon_{\mathsf{sum}} + \frac{\mu(\log^2(\kappa) + (2\log n)^2)}{|\mathcal{C}|}$$

where $\epsilon'_{\mathsf{rg}}, \epsilon_{\mathsf{rg}}$ are defined in Lemma 4.5 and Lemma 4.7.

Proof. Let $\mathcal{A}^* := (\mathcal{A}, \mathsf{P}^*)$ be any malicious prover with winning probability $\epsilon_{\mathcal{A}^*}$. Without loss of generality, we assume that $\mathcal{A}^*$ is deterministic. We derive an adversary $\mathcal{A}'$ below that will be fed to the extractor in Lemma 4.9. Note that $\mathcal{A}'$ only calls $\mathcal{A}^*$ as an oracle. Recall that $\mathsf{M}_{\mathcal{C}} := \mathcal{C} \times \mathcal{C}$.

Let $\mu := 3 + dk$ and $U := \bar{\mathcal{S}}^{\mu}$. The adversary $\mathcal{A}'$, with a hardcoded $\mathbf{r}_o \in \mathsf{M}_{\mathcal{C}}^{\log n}$, on input $\mathbf{u} = (\mathbf{s} \in \bar{\mathcal{S}}^3, \mathbf{s}' \in \bar{\mathcal{S}}^{dk})$, simulates execution with $\mathcal{A}^*$ by using $\mathbf{u}$ as the folding challenge and $\mathbf{r}_o$ as the challenge of the last sumcheck, and internally sampling other randomness of the verifier. If $\mathcal{A}^*$ fails or $\mathcal{A}^*$'s output is not a valid witness for $\mathcal{R}_{\mathsf{com}}$, $\mathcal{A}'$ returns $\perp$; otherwise $\mathcal{A}'$ outputs $y := (\mathsf{tr}, \mathbb{w}_o)$, where $\mathbb{w}_o$ is $\mathcal{A}^*$'s output witness and tr is the execution transcript, which includes the input and output instances, the verifier randomness, and the prover messages.

Define the predicate $\Psi : U \times \{0,1\}^* \to \{0,1\}$ such that $\Psi(\mathbf{u}, y = (\mathsf{tr}, \mathbb{w}_o)) = 1$ if and only if the following two conditions both hold:

1. The execution transcript tr is accepted and uses $\mathbf{u}$ as the folding challenges.
2. $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}_{\mathsf{com}}$ where $\mathbb{x}_o$ is the output instance in the transcript.

Next, we describe the extractor Ext for Π_{cm}, which calls the extractor $\mathcal{E}$ in Lemma 4.9 as a subroutine:

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

1. Sample $\mathbf{r}_o \xleftarrow{\text{R}} \mathsf{M}_{\mathcal{C}}^{\log n}$ to be used by $\mathcal{A}'$. Each call of $\mathcal{A}'$ will reuse the same $\mathbf{r}_o$ but freshly sample other randomness for simulating execution with $\mathcal{A}^*$.
2. Sample $\mathbf{u}^{(0)} \xleftarrow{\text{R}} U$ and $y_0 \leftarrow \mathcal{A}'(\mathbf{u}^{(0)})$.
3. Call $\mathcal{E}^{\mathcal{A}'}(\mathbf{u}^{(0)}, y_0)$. If the execution fails, abort and return $\perp$; otherwise, parse the output
$$(\mathbf{u}^{(0)}, y_0), \quad (\mathbf{u}^{(1)}, y_1 \leftarrow \mathcal{A}(\mathbf{u}_1)), \quad \ldots, \quad (\mathbf{u}^{(\mu)}, y_\mu \leftarrow \mathcal{A}'(\mathbf{u}^{(\mu)}))\,,$$
where $y_i = (\mathsf{tr}_i = (\mathbb{x}_o^{(i)}, *), \mathbb{w}_o^{(i)})$ for all $i \in [\mu+1]$ and $\mathbf{u}^{(i+1)} \equiv_i \mathbf{u}^{(0)}$ for all $i \in [\mu]$.
4. For every $i \in [\mu]$, compute[13]
$$\mathbb{w}_i := \frac{\mathbb{w}_o^{(i+1)} - \mathbb{w}_o^{(0)}}{\mathbf{u}_i^{(i+1)} - \mathbf{u}_i^{(0)}} \in R_q^n\,. \tag{25}$$
5. Output $\mathbb{w} := [\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{m}}_\tau, \hat{\mathbf{f}}, \hat{\mathbf{M}}_{\mathbf{f}}] := [\mathbb{w}_0, \ldots, \mathbb{w}_{\mu-1}] \in R_q^{n \times \mu}$.

By Lemma 4.9, it is clear that Ext runs in expected polynomial time. Next, we focus on analyzing the success probability of Ext.

[13] $\mathbb{w}_i$ is well-defined as $\mathbf{u}_i^{(i+1)} - \mathbf{u}_i^{(0)} \in \bar{\mathcal{S}} - \bar{\mathcal{S}}$ is invertible.

Success probability. Define $\epsilon_\Psi(\mathcal{A}') := \Pr_{\mathbf{u} \xleftarrow{\mathrm{R}} U, \mathbf{r}_o \xleftarrow{\mathrm{R}} \mathsf{M}_{\mathcal{C}}^{\log n}}[\Psi(\mathbf{u}, \mathcal{A}'(\mathbf{u})) = 1]$, which is equal to $\epsilon_{\mathcal{A}^*}$. By Lemma 4.9, with non-negligible[14] probability $\epsilon_{\mathcal{A}^*} - \mu/|\bar{\mathcal{S}}|$, the extractor Ext does not abort. Conditioned on this, denote by $\mathbb{x} = (\mathsf{cm}_{\mathbf{f}}, C_{\mathbf{M}_{\mathbf{f}}}, \mathsf{cm}_{\mathbf{m}_\tau})$ the input instance in the execution, it suffices to upper-bound the probability that the extracted witness $\mathbb{w} = [\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{m}}_\tau, \hat{\mathbf{f}}, \hat{\mathbf{M}}_{\mathbf{f}}]$ is invalid, i.e., $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}_{\mathsf{rg},B}$.

We first review some properties of the extracted witness. Recall that $[\![\cdot]\!]$ and multilinear evaluations are linear functions; $[\![\cdot]\!]$ is $(b, \mathcal{S} = \bar{\mathcal{S}} - \bar{\mathcal{S}})$-binding and $(\mathbb{x}_o^{(i)}, \mathbb{w}_o^{(i)}) \in \mathcal{R}_{\mathsf{com}}$ for all $i \in [\mu+1]$. Thus, if Ext succeeds, $\mathbb{w} = [\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{m}}_\tau, \hat{\mathbf{f}}, \hat{\mathbf{M}}_{\mathbf{f}}]$ has the following properties:

1. $\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{m}}_\tau, \hat{\mathbf{f}}$ are $(b, \mathcal{S})$-valid openings of $C_{\mathbf{M}_{\mathbf{f}}} = [\![\hat{\tau_{\mathbf{D}}}]\!], \mathsf{cm}_{\mathbf{m}_\tau}, \mathsf{cm}_{\mathbf{f}}$ (Eq. (6)).

2. $\hat{\mathbf{M}}_{\mathbf{f}}$ is a valid opening of $[\![\hat{\mathbf{M}}_{\mathbf{f}}]\!]$. (However, $(\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{M}}_{\mathbf{f}})$ may not be a valid opening of the double commitment $C_{\mathbf{M}_{\mathbf{f}}}$ if $[\![\hat{\mathbf{M}}_{\mathbf{f}}]\!] \neq \Phi(\hat{\tau_{\mathbf{D}}})$.)

3. For every $i \in [\mu+1]$, parse $\mathbf{u}^{(i)} = (\mathbf{s}^{(i)}, \mathbf{s}'^{(i)} \in \bar{\mathcal{S}}^{dk})$ and denote by $[\![\mathbf{h}^{(i)}]\!]$ (included in tr_i) the $\mathbf{h}$-commitment sent by the prover at Step 3. By linearity of $[\![\cdot]\!]$ and the verifier computation at Step 6, we have that
$$[\![\mathbf{h}^{(i)}]\!] = [\![\hat{\mathbf{M}}_{\mathbf{f}}]\!]\mathbf{s}'^{(i)}\,. \tag{26}$$
We note that $\hat{\mathbf{M}}_{\mathbf{f}}\mathbf{s}'^{(i)}$, however, may not be a $(b, \mathcal{S})$-valid opening of $[\![\mathbf{h}^{(i)}]\!]$, because we do not have sufficient norm guarantee for $\hat{\mathbf{M}}_{\mathbf{f}}\mathbf{s}'^{(i)}$.

4. $[\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{m}}_\tau, \hat{\mathbf{f}}]^\top \mathsf{tensor}(\mathbf{r}_o) = \mathbf{e}_o^{(0)}[0,3)$ where $\mathbf{e}_o^{(0)}$ (in the first transcript tr_0) is the claimed evaluation in Eq. (22).

5. $\langle \hat{\mathbf{M}}_{\mathbf{f}}\mathbf{s}'^{(0)}, \mathsf{tensor}(\mathbf{r}_o)\rangle = \mathbf{e}_o^{(0)}[3]$.

Next, we analyze the probability of bad events below.

Define BAD_1 as the event that Ext does not abort, but $\hat{\tau_{\mathbf{D}}}$ (in the extracted witness $\mathbb{w}$) is not in $(-d', d')^n$.

Claim 1. $\Pr[\mathsf{BAD}_1] \leq \epsilon_1 := \epsilon_{\mathsf{bind}} + \epsilon'_{\mathsf{rg}} + \epsilon_{\mathsf{sum}}$ *where* ϵ'_{rg} *is defined in Lemma 4.5.*

Proof. If BAD_1 occurs, by Property (1) and Property (4), we have $(\bar{\mathbb{x}} = C_{\mathbf{M}_{\mathbf{f}}}, \bar{\mathbb{w}} = \hat{\tau_{\mathbf{D}}}) \in \mathcal{R}_{\mathsf{open}}$ and $\langle \hat{\tau_{\mathbf{D}}}, \mathsf{tensor}(\mathbf{r}_o)\rangle = \mathbf{e}_o^{(0)}[0]$.

Conditioned on Ext does not abort, consider a mental experiment where we re-run $\mathsf{Ext}^{\mathcal{A}^*}$ with fresh randomness. If the re-execution does not abort, let $\hat{\tau_{\mathbf{D}}}^*$ denote part of the extracted witness that is a claimed opening to $C_{\mathbf{M}_{\mathbf{f}}}$; otherwise $\hat{\tau_{\mathbf{D}}}^* := \perp$. Consider the *first* transcript generated by the re-execution. Let $\mathbf{r}, \mathbf{r}_o^*, \mathbf{e}[0], \mathbf{e}_o[0]$ be the random variables corresponding to the random challenges and evaluations at Step 1 and Eq. (22). Note that

[14] $\epsilon_{\mathcal{A}^*}$ is non-negligible and $\mu/|\bar{\mathcal{S}}|$ is negligible.

$\mathbf{r}, \mathbf{r}_o^*$ are uniformly random, because the first transcript is for the first call to $\mathcal{A}^*$, and $\mathsf{Ext}^{\mathcal{A}^*}$ cannot do rejection sampling to bias the distribution.

Let E^* be the event that at least one event below occurs in the mental experiment:

- Event B_1: $(\hat{\tau_{\mathbf{D}}}^* \neq \perp) \wedge (\hat{\tau_{\mathbf{D}}}^* \neq \hat{\tau_{\mathbf{D}}})$;
- Event B_2: $(\hat{\tau_{\mathbf{D}}}^* = \hat{\tau_{\mathbf{D}}})$ and $(\hat{\tau_{\mathbf{D}}} \notin (-d', d')^n \wedge \langle \hat{\tau_{\mathbf{D}}}, \mathsf{tensor}(\mathbf{r})\rangle = \mathbf{e}[0])$;
- Event B_3: $(\hat{\tau_{\mathbf{D}}}^* = \hat{\tau_{\mathbf{D}}})$ and $(\langle \hat{\tau_{\mathbf{D}}}, \mathsf{tensor}(\mathbf{r})\rangle \neq \mathbf{e}[0] \wedge \langle \hat{\tau_{\mathbf{D}}}, \mathsf{tensor}(\mathbf{r}_o^*)\rangle = \mathbf{e}_o[0])$.

By the binding property of $[\![\cdot]\!]$, Lemma 4.5, and Lemma 2.7, we have that

$$\Pr[E^*] \leq \Pr[B_1] + \Pr[B_2] + \Pr[B_3] \leq \epsilon_{\mathsf{bind}} + \epsilon'_{\mathsf{rg}} + \epsilon_{\mathsf{sum}}\,. \tag{27}$$

Note that if BAD_1 occurs in the mental experiment, then E^* must also occur. Moreover, BAD_1 occurs with the same probability in the real execution and the mental experiment. Thus, $\Pr[\mathsf{BAD}_1] \leq \Pr[E^*]$ and the claim holds. □

Define BAD_2 as the event that (i) Ext does not abort and BAD_1 does not occur, but (ii) $(\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{M}}_{\mathbf{f}})$ (in the extracted witness $\mathbb{w}$) is not a valid opening of the double commitment $C_{\mathbf{M}_{\mathbf{f}}}$, i.e., $(C_{\mathbf{M}_{\mathbf{f}}}, (\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{M}}_{\mathbf{f}})) \notin \mathcal{R}_{\mathsf{dopen},dk}$ (Eq. (8)).

Claim 2. *If* $|\mathcal{C}| = |\bar{\mathcal{S}}|$, *then* $\Pr[\mathsf{BAD}_2] \leq \epsilon_2 := \epsilon_{\mathsf{bind}} + \frac{\mu(\log^2(\kappa)+(2\log n)^2)}{|\mathcal{C}|}$.

Proof. If BAD_2 occurs, then the following conditions all hold for the extracted witness $\mathbb{w} = [\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{m}}_\tau, \hat{\mathbf{f}}, \hat{\mathbf{M}}_{\mathbf{f}}]$:

- $(C_{\mathbf{M}_{\mathbf{f}}}, \hat{\tau_{\mathbf{D}}}) \in \mathcal{R}_{\mathsf{open}}$ (Property (1)) and $\hat{\mathbf{M}}_{\mathbf{f}}$ is a valid opening of $[\![\hat{\mathbf{M}}_{\mathbf{f}}]\!]$ (Property (2));
- The sumcheck evaluation claims for $\hat{\tau_{\mathbf{D}}}$ in the $\mu + 1$ transcripts are correct;
- $(C_{\mathbf{M}_{\mathbf{f}}}, (\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{M}}_{\mathbf{f}})) \notin \mathcal{R}_{\mathsf{dopen},dk}$, which implies that $\Phi(\hat{\tau_{\mathbf{D}}}) \neq [\![\hat{\mathbf{M}}_{\mathbf{f}}]\!]$.

Conditioned on Ext does not abort, consider a mental experiment where we re-run $\mathsf{Ext}^{\mathcal{A}^*}$ with fresh randomness. If it does not abort, let $\mathbb{w}^* = (\hat{\tau_{\mathbf{D}}}^*, *, *, \hat{\mathbf{M}}_{\mathbf{f}}^*)$ denote the extracted output, otherwise $\mathbb{w}^* := \perp$. Let E^* be the event that at least one event below occurs in the mental experiment:

- Event B_1: $(\mathbb{w}^* \neq \perp) \wedge (\hat{\tau_{\mathbf{D}}}^* \neq \hat{\tau_{\mathbf{D}}})$;
- Event B_2: $(\hat{\tau_{\mathbf{D}}}^* = \hat{\tau_{\mathbf{D}}})$ and $\Phi(\hat{\tau_{\mathbf{D}}}) \neq [\![\hat{\mathbf{M}}_{\mathbf{f}}^*]\!]$.

Note that if BAD_2 occurs in the mental experiment, then E^* also occurs. Moreover, BAD_2 occurs with the same probability in the real execution and the mental experiment. Thus, $\Pr[\mathsf{BAD}_2] \leq \Pr[E^*] \leq \Pr[B_1] + \Pr[B_2]$. Suppose for contradiction that $\Pr[B_1] > \epsilon_{\mathsf{bind}}$, then we can find an adversary[15] that breaks the binding property of $[\![\cdot]\!]$ with probability $\Pr[B_1] > \epsilon_{\mathsf{bind}}$, contradiction. Thus $\Pr[B_1] \leq \epsilon_{\mathsf{bind}}$ and it suffices to analyze $\Pr[B_2]$.

For every $i \in [\mu+1]$, denote by $\mathbf{u}'_i = (*, \mathbf{s}'^{(i)} \in \bar{\mathcal{S}}^{dk})$ the folding challenge in the i-th transcript output by Ext in the mental experiment. Observe that event B_2 implies that there exists $i^* \in [\mu+1]$ where $\Phi(\hat{\tau_{\mathbf{D}}})\mathbf{s}'^{(i^*)} \neq [\![\hat{\mathbf{M}}_{\mathbf{f}}{}^*]\!]\mathbf{s}'^{(i^*)}$; moreover, $[\![\hat{\mathbf{M}}_{\mathbf{f}}{}^*]\!]\mathbf{s}'^{(i^*)} = [\![\mathbf{h}^{(i^*)}]\!]$ by Property (3), where $[\![\mathbf{h}^{(i^*)}]\!]$ is in the transcript tr_{i^*}. Thus, $\Phi(\hat{\tau_{\mathbf{D}}})\mathbf{s}'^{(i^*)} \neq [\![\mathbf{h}^{(i^*)}]\!]$, but recall that the evaluation claim for $\hat{\tau_{\mathbf{D}}}$ holds.

By Lemma 4.9, Ext calls the adversary for at most $\mu|\bar{\mathcal{S}}|$ times. For each fixed call, conditioned on that $\Phi(\hat{\tau_{\mathbf{D}}})\mathbf{s}' \neq [\![\mathbf{h}]\!]$ (where $\mathbf{s}'$ is the folding challenge and $[\![\mathbf{h}]\!]$ is the $\mathbf{h}$-commitment sent by the prover in this call), by Lemma 2.6 and Lemma 2.7, the sumcheck evaluation claim holds w.r.t. $\widetilde{\hat{\tau_{\mathbf{D}}}}$ with probability at most $(\log\kappa/|\mathcal{C}|)^2 + \epsilon_{\mathsf{sum}}$. By union bound over the $\mu|\bar{\mathcal{S}}|$ calls, we have that

$$\Pr[B_2] \leq \mu|\bar{\mathcal{S}}| \cdot \left((\log\kappa/|\mathcal{C}|)^2 + \epsilon_{\mathsf{sum}}\right) \leq \frac{\mu(\log^2(\kappa) + (2\log n)^2)}{|\mathcal{C}|} .$$

We note that this is where we rely on running two independent sumchecks to ensure a small soundness error despite the large union bound. We will come back to this in Remark 4.9.

In sum, $\Pr[\mathsf{BAD}_2] \leq \Pr[E^*] \leq \Pr[B_1] + \Pr[B_2] \leq \epsilon_2$ as claimed. □

Define BAD_3 as the event that (i) Ext does not abort and $(\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{M}}_{\mathbf{f}})$ in the extracted witness $\mathbb{w}$ is a valid opening of $C_{\mathbf{M}_{\mathbf{f}}}$; but (ii) $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}_{\mathsf{rg},B}$, where $\mathbb{x}$ is the input instance.

Claim 3. $\Pr[\mathsf{BAD}_3] \leq \epsilon_3 := \epsilon_{\mathsf{bind}} + \epsilon_{\mathsf{rg}} + (dk)/|\bar{\mathcal{S}}| + \epsilon_{\mathsf{sum}}$.

Proof. If BAD_3 occurs, by Property (4), (5), we have that

$$[\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{m}}_\tau, \hat{\mathbf{f}}, \hat{\mathbf{M}}_{\mathbf{f}}\mathbf{s}'^{(0)}]^\top \mathsf{tensor}(\mathbf{r}_o) = \mathbf{e}_o^{(0)} .$$

And by Property (1) and the premise of BAD_3, $\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{m}}_\tau, \hat{\mathbf{f}}$ are valid openings of $C_{\mathbf{M}_{\mathbf{f}}} = [\![\hat{\tau_{\mathbf{D}}}]\!], \mathsf{cm}_{\mathbf{m}_\tau}, \mathsf{cm}_{\mathbf{f}}$; and $(C_{\mathbf{M}_{\mathbf{f}}}, (\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{M}}_{\mathbf{f}})) \in \mathcal{R}_{\mathsf{dopen},dk}$.

Conditioned on Ext does not abort, define $\mathbb{w}' = [\hat{\tau_{\mathbf{D}}}, \hat{\mathbf{m}}_\tau, \hat{\mathbf{f}}]$ and consider a mental experiment where we re-run $\mathsf{Ext}^{\mathcal{A}^*}$ with fresh randomness. If it does not abort, let $\mathbb{w}^*$ denote the extracted output, otherwise $\mathbb{w}^* := \bot$. Consider the *first* transcript generated by the re-execution. Let $\mathbf{r}, \mathbf{r}_o^*$ and $\mathbf{e}, \mathbf{e}_o, u$ be the random variables corresponding to the verifier randomness and claimed evaluations at Step 1, Eq. (18) and Eq. (22). Let $\mathbf{s}^{*\prime} \in \bar{\mathcal{S}}^{dk}$ be the folding challenge. Note that $\mathbf{r}, \mathbf{r}_o^*, \mathbf{s}^{*\prime}$ are uniformly random, because the first transcript corresponds to the first call to $\mathcal{A}^*$, and thus $\mathsf{Ext}^{\mathcal{A}^*}$ cannot bias the distribution. Let E^* be the event that at least one event below occurs in the mental experiment:

- B_1: $(\mathbb{w}^* \neq \bot \wedge \mathbb{w}^* \neq \mathbb{w})$;

[15]The collision attacker keeps re-running $\mathsf{Ext}^{\mathcal{A}^*}$ until Ext does not abort, and then re-executes $\mathsf{Ext}^{\mathcal{A}^*}$ one more time with fresh randomness.

- B_2: $(\mathbb{w}^* = \mathbb{w})$ and $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}_{\mathsf{rg},B}$ and $\mathbb{w}^\top \mathsf{tensor}(\mathbf{r}) = \mathbf{e}$;
- B_3: $(\mathbb{w}^* = \mathbb{w})$ and $\mathbb{w}^\top \mathsf{tensor}(\mathbf{r}) \neq \mathbf{e}$ and $[\mathbb{w}', \hat{\mathbf{M}}_{\mathbf{f}}\mathbf{s}^{*'}]^\top \mathsf{tensor}(\mathbf{r}) = (\mathbf{e}[0,2], u)$;
- B_4: $(\mathbb{w}^* = \mathbb{w})$ and $[\mathbb{w}', \hat{\mathbf{M}}_{\mathbf{f}}\mathbf{s}^{*'}]^\top \mathsf{tensor}(\mathbf{r}) \neq (\mathbf{e}[0,2], u)$ and $[\mathbb{w}', \hat{\mathbf{M}}_{\mathbf{f}}\mathbf{s}^{*'}]^\top \mathsf{tensor}(\mathbf{r}_o^*) = \mathbf{e}_o$.

By the binding property of $[\![\cdot]\!]$ and $(\!|\cdot|\!)$ (Lemma 4.1), knowledge soundness of Π_{rgchk} (Lemma 4.7), Lemma 2.6, and Lemma 2.7, we have that

$$\begin{aligned}\Pr[E^*] &\leq \Pr[B_1] + \Pr[B_2] + \Pr[B_3] + \Pr[B_4] \\ &\leq \epsilon_{\mathsf{bind}} + \epsilon_{\mathsf{rg}} + dk/|\bar{\mathcal{S}}| + \epsilon_{\mathsf{sum}}\,.\end{aligned}$$

Note that if BAD_3 occurs in the mental experiment, then E^* must also occur. Moreover, BAD_3 occurs with the same probability in the real execution and the mental experiment. Thus $\Pr[\mathsf{BAD}_3] \leq \Pr[E^*]$ and the claim holds. □

Conditioned on that Ext does not abort and BAD_1, BAD_2, BAD_3 do not occur, the extracted witness $\mathbb{w}$ is valid for the input $\mathbb{x}$, i.e., $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{rg},B}$. Thus, Π_{cm} is knowledge sound with knowledge error $\epsilon_{\mathsf{cm},k} := (3 + dk)/|\bar{\mathcal{S}}| + \epsilon_1 + \epsilon_2 + \epsilon_3$. This completes the proof of Lemma 4.10. □

Remark 4.9. We need to run two parallel sumchecks at Step 5 because Claim 2 relies on a union bound over up to $\mu|\bar{\mathcal{S}}|$ adversary calls and the bound is meaningless if we only run one sumcheck. It is unclear whether this is merely a proof artifact or if a concrete attack exists. We leave it as an interesting future work.

5 Folding Generalized Committed Linear Relations

In this section, we present a folding scheme for generalized committed linear relations (Definition 3.1). Fix norm bound $B = (d')^k$ where $k \in \mathbb{N}$. Let L be the number of input instances and $\bar{\mathcal{S}}$ the folding challenge set. Let $[\![\cdot]\!]$ be a general linear commitment defined in Section 4.1. For simplicity, we assume that $[\![\cdot]\!]$ is $(2B^2, \bar{\mathcal{S}} - \bar{\mathcal{S}})$-binding and[16]

$$\|\bar{\mathcal{S}}\|_{\mathsf{op}} L(d' + 1 + B + dk) \leq B^2\,. \tag{28}$$

For an indexed relation $\mathcal{R}$ and $m \in \mathbb{N}$, we use $\mathcal{R}^{(m)}$ to denote the set of tuples $(\mathbb{i}, (\mathbb{x}_i, \mathbb{w}_i)_{i\in[m]})$ such that $(\mathbb{i}, \mathbb{x}_i, \mathbb{w}_i) \in \mathcal{R}$ for all $i \in [m]$. Define online relation $\mathcal{R}_{\mathsf{comp}} := \mathcal{R}_{\mathsf{lin},B}^{(L-2)}$ and accumulated relation $\mathcal{R}_{\mathsf{acc}} := \mathcal{R}_{\mathsf{lin},B}^{(2)}$. Our goal is to build a reduction of knowledge from $\mathcal{R}_{\mathsf{comp}} \times \mathcal{R}_{\mathsf{acc}}$ to $\mathcal{R}_{\mathsf{acc}}$. Our construction consists of two phases.

1. In Section 5.1, we reduce $\mathcal{R}_{\mathsf{comp}} \times \mathcal{R}_{\mathsf{acc}} = \mathcal{R}_{\mathsf{lin},B}^{(L)}$ to $\mathcal{R}_{\mathsf{lin},B^2}$ by leveraging and adapting the commitment-transformation protocol Π_{cm} from Section 4.4.

[16] Our scheme naturally extends to more flexible choices of B and L.

2. To avoid norm blowup and enable unbounded folding, Section 5.2 reduces $\mathcal{R}_{\mathsf{lin},B^2}$ to $\mathcal{R}_{\mathsf{acc}} = \mathcal{R}^{(2)}_{\mathsf{lin},B}$ by adapting a decomposition technique from [BC24].

Remark 5.1. One could also use the "decompose-then-fold" strategy from [BC24], where we first decompose $\mathcal{R}^{(L)}_{\mathsf{lin},B}$ to $\mathcal{R}^{(2L)}_{\mathsf{lin},\sqrt{B}}$, and then reduce back to $\mathcal{R}_{\mathsf{lin},B}$ via folding. This strategy has the advantage that the norm bound never exceeds B and thus we can choose a larger B if needed. However, it requires computing $2L$ decomposed commitments, which leads to a relatively less efficient prover.

We state our main result below.

Theorem 5.1. *Fix $\bar{\mathcal{S}}, L, B = (d')^k$ such that $|\bar{\mathcal{S}}| \geq 2^\lambda$, (28) holds and assume that $[\![\cdot]\!]$ is $(2B^2, \bar{\mathcal{S}} - \bar{\mathcal{S}})$-binding (Section 4.1). There exists a reduction of knowledge from $\mathcal{R}_{\mathsf{comp}} \times \mathcal{R}_{\mathsf{acc}}$ to $\mathcal{R}_{\mathsf{acc}}$, where $\mathcal{R}_{\mathsf{comp}} := \mathcal{R}^{(L-2)}_{\mathsf{lin},B}$ and $\mathcal{R}_{\mathsf{acc}} := \mathcal{R}^{(2)}_{\mathsf{lin},B}$.*

Proof. The claim follows from Theorem 5.2, Lemma 5.1 and the sequential composition theorem (Theorem 2.1). □

5.1 Folding

In this section, we show how to reduce $\mathcal{R}^{(L)}_{\mathsf{lin},B}$ to $\mathcal{R}_{\mathsf{lin},B^2}$. Let us first recall the definition of $\mathcal{R}_{\mathsf{lin},B}$ from Section 3. Given a triple $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ defined by

$$\mathbb{i} = \left([\![\cdot]\!], (\mathbf{M}^{(i)} \in R_q^{n\times n})_{i\in[n_{\mathsf{lin}}]}\right), \quad \mathbb{x} = \left(\mathsf{cm}_{\mathbf{f}} \in R_q^{\kappa}, \mathbf{r} \in \mathsf{M}_{\mathcal{C}}^{\log n}, \mathbf{v} \in \mathsf{M}_q^{n_{\mathsf{lin}}}\right), \quad \mathbb{w} = \mathbf{f} \in R_q^n,$$

where $\mathsf{M}_{\mathcal{C}} := \mathcal{C} \times \mathcal{C}$ and $\mathsf{M}_q := R_q \times R_q$, $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ is in the relation $\mathcal{R}_{\mathsf{lin},B}$ if

$$(\|\mathbf{f}\|_\infty < B) \wedge (\mathsf{cm}_{\mathbf{f}}, \mathbf{f}) \in \mathcal{R}_{\mathsf{open}} \wedge \left(\langle \mathbf{M}^{(i)} \cdot \mathbf{f}, \mathsf{tensor}(\mathbf{r})\rangle = \mathbf{v}_i \forall i \in [n_{\mathsf{lin}}]\right)$$

where the commitment opening relation $\mathcal{R}_{\mathsf{open}}$ is defined in (6). Next, we illustrate the key idea by showing how to reduce from $\mathcal{R}_{\mathsf{lin},B}$ to $\mathcal{R}_{\mathsf{lin},B^2/L}$; then via batching, we extend the idea and show the reduction from $\mathcal{R}^{(L)}_{\mathsf{lin},B}$ to $\mathcal{R}_{\mathsf{lin},B^2}$.

Warm-up. We first show how to reduce a single statement in $\mathcal{R}_{\mathsf{lin},B}$ to $\mathcal{R}_{\mathsf{lin},B^2/L}$. While this basic scheme is primarily for illustration, we later extend it to reduce $L > 1$ statements in $\mathcal{R}_{\mathsf{lin},B}$ to a single statement in $\mathcal{R}_{\mathsf{lin},B^2}$.

Our scheme, $\Pi_{\mathsf{lin},B}$, is a simple adaptation to Π_{cm} in Construction 4.5. The only difference is that at Step 5, the parties further represent each linear check in $\mathcal{R}_{\mathsf{lin},B}$ as a sumcheck claim, and run a (batched) sumcheck protocol to reduce the claim to a fresh linear check in the output statement. More formally, for each $\ell \in [n_{\mathsf{lin}}]$, let $\langle \mathbf{M}^{(\ell)}\mathbf{f}, \mathsf{tensor}(\mathbf{r})\rangle = \mathbf{v}_\ell$ be the ℓ-th input linear check, we can rewrite it as a sumcheck claim

$$\sum_{i\in[n]} \mathbf{m}(\langle i\rangle)\, eq(\mathbf{r}, \langle i\rangle) \stackrel{?}{=} \mathbf{v}_\ell \tag{29}$$

where $\mathbf{m}(\langle i \rangle)$ is defined as $\mathbf{m}(\langle i \rangle) := \sum_{j \in [n]} \mathbf{M}^{(\ell)}_{i,j} \mathbf{f}_j = \mathbf{M}^{(\ell)}_{i,*} \mathbf{f}$.

By running a sumcheck protocol, we can reduce the claim in (29) to an evaluation claim $\widetilde{\mathbf{m}}(\mathbf{r}_o) = \mathbf{v}_{o,\ell}$. By Remark 2.2, this is equivalent to

$$\widetilde{\mathbf{m}}(\mathbf{r}_o) = \sum_{i \in [n]} (\mathbf{M}^{(\ell)}_{i,*} \mathbf{f}) \cdot \mathsf{tensor}(\mathbf{r}_o)_i = \langle \mathbf{M}^{(\ell)} \mathbf{f}, \mathsf{tensor}(\mathbf{r}_o) \rangle \stackrel{?}{=} \mathbf{v}_{o,\ell} \,.$$

In sum, we reduce $\langle \mathbf{M}^{(\ell)} \mathbf{f}, \mathsf{tensor}(\mathbf{r}) \rangle \stackrel{?}{=} \mathbf{v}_\ell$ to $\langle \mathbf{M}^{(\ell)} \mathbf{f}, \mathsf{tensor}(\mathbf{r}_o) \rangle \stackrel{?}{=} \mathbf{v}_{o,\ell}$.

Denote by Π'_{cm} the adapted version of Π_{cm} that further checks the above sumcheck claims. By definition of $\mathcal{R}_{\mathsf{com}}$ (17), the output relation of Π'_{cm} is exactly $\mathcal{R}_{\mathsf{lin},B^2/L}$. We describe the protocol $\Pi_{\mathsf{lin},B}$ below:

Construction 5.1. On input the index $\mathbb{i} = ((\mathbf{M}^{(i)} \in R_q^{n \times n})_{i \in [n_{\mathsf{lin}}]}, [\![\cdot]\!])$ and

$$\mathbb{x} = (\mathsf{cm}_{\mathbf{f}} \in R_q^{\kappa}, \mathbf{r} \in \mathsf{M}_{\mathcal{C}}^{\log n}, \mathbf{v} \in \mathsf{M}_q^{n_{\mathsf{lin}}}) \,, \quad \mathbb{w} = \mathbf{f} \in R_q^n$$

where $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{lin},B}$, define the decomposed matrix $\mathbf{D}_{\mathbf{f}}$ for $\mathbf{f}$ as in (13). Set $\mathbf{M}_{\mathbf{f}} \in \mathsf{EXP}(\mathbf{D}_{\mathbf{f}})$, $\tau_{\mathbf{D}} = \Phi^{-1}([\![\mathbf{M}_{\mathbf{f}}]\!]) \in (-d', d')^n$ where Φ^{-1} is defined in Section 4.1, and $\mathbf{m}_\tau \in \mathsf{EXP}(\tau_{\mathbf{D}})$. The protocol $\Pi_{\mathsf{lin},B}$ proceeds as follows:

1. $\mathsf{P} \to \mathsf{V}$: Send commitments $C_{\mathbf{M}_{\mathbf{f}}} = (\!|\mathbf{M}_{\mathbf{f}}|\!) = [\![\tau_{\mathbf{D}}]\!] \in R_q^{\kappa}$, $\mathsf{cm}_{\mathbf{m}_\tau} = [\![\mathbf{m}_\tau]\!] \in R_q^{\kappa}$.

2. $\mathsf{P} \leftrightarrow \mathsf{V}$: Run protocol Π'_{cm}, checking that $\langle \mathbf{M}^{(i)} \mathbf{f}, \mathsf{tensor}(\mathbf{r}) \rangle = \mathbf{v}_i \, \forall i \in [n_{\mathsf{lin}}]$, and

$$\left(\mathbb{x}' = (\mathsf{cm}_{\mathbf{f}}, C_{\mathbf{M}_{\mathbf{f}}}, \mathsf{cm}_{\mathbf{m}_\tau}), \mathbb{w}' = (\tau_{\mathbf{D}}, \mathbf{m}_\tau, \mathbf{f}, \mathbf{M}_{\mathbf{f}})\right) \in \mathcal{R}_{\mathsf{rg},B}$$

where $\mathcal{R}_{\mathsf{rg},B}$ is defined in (14).

3. Return the reduced statement $\left(\mathbb{i}, \mathbb{x}_o = (\mathsf{cm}_{\mathbf{g}}, \mathbf{r}_o, \mathbf{v}_o), \mathbb{w}_o = \mathbf{g} \in R_q^n\right) \in \mathcal{R}_{\mathsf{lin},B^2/L}$.

By Eq. (28), Theorem 4.3, and the previous argument for reducing linear checks, $\Pi_{\mathsf{lin},B}$ is a reduction of knowledge from $\mathcal{R}_{\mathsf{lin},B}$ to $\mathcal{R}_{\mathsf{lin},B^2/L}$. The knowledge error $\epsilon_{\mathsf{lin},k}$ is identical to $\epsilon_{\mathsf{cm},k}$ (in Theorem 4.3).[17]

Folding multiple inputs. Next, we extend $\Pi_{\mathsf{lin},B}$ to reduce multiple statements in the committed linear relation to a single statement. Our goal is to reduce from $\mathcal{R}^{(L)}_{\mathsf{lin},B}$ to $\mathcal{R}_{\mathsf{lin},B^2}$. The construction is a simple adaptation of Construction 5.1 via batching. We can understand it as running L parallel instances of $\Pi_{\mathsf{lin},B}$, but with two modifications: First, all of the concurrent sumchecks are batched. Second, the output witness is a (randomly) folded sum of the L input (and intermediate) witnesses. This is equivalent to summing up the output witnesses from the L individual executions of $\Pi_{\mathsf{lin},B}$.

[17]The only difference between $\Pi_{\mathsf{lin},B}$ and Π_{cm} is that we have $6 + n_{\mathsf{lin}}$ (rather than 6) sumcheck claims to verify at Step 5, which can be batched using the technique from Remark 2.4.

Construction 5.2. On input the index $\mathbb{i} = ((\mathbf{M}^{(i)} \in R_q^{n\times n})_{i\in[n_{\mathsf{lin}}]}, [\![\cdot]\!])$ and

$$\mathbb{x} = \left\{\mathbb{x}_i = (\mathsf{cm}_{\mathbf{f}_i} \in R_q^{\kappa}, \mathbf{r}_i \in \mathsf{M}_{\mathcal{C}}^{\log n}, \mathbf{v}^{(i)} \in \mathsf{M}_q^{n_{\mathsf{lin}}})\right\}_{i\in[L]}, \quad \mathbb{w} = \left\{\mathbf{f}_i \in R_q^n\right\}_{i\in[L]}$$

where $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{lin},B}^{(L)}$, the protocol $\Pi_{\mathsf{mlin},L,B}$ proceeds as follows:

1. $\mathsf{P} \leftrightarrow \mathsf{V}$: Check $(\mathbb{i}, \mathbb{x}_i, \mathbf{f}_i) \in \mathcal{R}_{\mathsf{lin},B}\, \forall i \in [L]$ by batching L executions of $\Pi_{\mathsf{lin},B}$.
2. Denote by $\{(\mathsf{cm}_{\mathbf{g}_i}, \mathbf{r}_o, \mathbf{v}_o^{(i)}, \mathbf{g}_i) \in \mathcal{R}_{\mathsf{lin},B^2/L}\}_{i\in[L]}$ the reduced statements of the L (batched) executions of $\Pi_{\mathsf{lin},B}$. Return the reduced statement

$$\left(\mathbb{x} = \left(\sum_{i\in[L]} \mathsf{cm}_{\mathbf{g}_i}, \mathbf{r}_o, \sum_{i\in[L]} \mathbf{v}_o^{(i)}\right), \mathbb{w} = \sum_{i\in[L]} \mathbf{g}_i\right) \in \mathcal{R}_{\mathsf{lin},B^2}\,.$$

Theorem 5.2. *If $[\![\cdot]\!]$ is $(2B^2, \mathcal{S} = \bar{\mathcal{S}} - \bar{\mathcal{S}})$-binding and Eq. (28) holds, then $\Pi_{\mathsf{mlin},L,B}$ is a reduction of knowledge from $\mathcal{R}_{\mathsf{lin},B}^{(L)}$ to $\mathcal{R}_{\mathsf{lin},B^2}$ with knowledge error $\epsilon_{\mathsf{mlin},B,L} \le L \cdot \epsilon_{\mathsf{lin},k}$.*

Proof. Public reducibility and completeness follows from that of $\Pi_{\mathsf{lin},B}$. The proof of knowledge soundness is almost identical to that of Theorem 4.3, except that the number of folding challenges is multiplied by L. □

5.2 Decomposition

In this section, we show how to reduce $\mathcal{R}_{\mathsf{lin},B^2}$ to $\mathcal{R}_{\mathsf{lin},B}^{(2)}$. The scheme is a simple adaptation to the decomposition protocol from [BC24].

Construction 5.3. On input the index $\mathbb{i} = ((\mathbf{M}^{(i)} \in R_q^{n\times n})_{i\in[n_{\mathsf{lin}}]}, [\![\cdot]\!])$ and

$$\mathbb{x} = (\mathsf{cm}_{\mathbf{f}} \in R_q^{\kappa}, \mathbf{r} \in \mathsf{M}_{\mathcal{C}}^{\log n}, \mathbf{v} \in \mathsf{M}_q^{n_{\mathsf{lin}}}), \quad \mathbb{w} = (\mathbf{f} \in R_q^n)$$

such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{lin},B^2}$, the protocol $\Pi_{\mathsf{decomp},B}$ proceeds as follows:

1. $\mathsf{P} \to \mathsf{V}$: Decompose $\mathbf{f}$ to $\mathbf{F} = [\mathbf{F}^{(0)}, \mathbf{F}^{(1)}] \in R_q^{n\times 2}$ according to Footnote 3, such that $\|\mathbf{F}\|_\infty < B$ and $\mathbf{f} = \mathbf{F} \times [1, B]^\top$. Denote the commitment $\mathbf{C} = [\![\mathbf{F}]\!] \in R_q^{\kappa\times 2}$ and values $\mathbf{v}^{(0)}, \mathbf{v}^{(1)} \in \mathsf{M}_q^{n_{\mathsf{lin}}}$ such that for all $i \in [2]$,

$$(\mathbb{i}, \mathbb{x}_i = (\mathbf{C}_{*,i}, \mathbf{r}, \mathbf{v}^{(i)}), \mathbb{w}_i = \mathbf{F}^{(i)}) \in \mathcal{R}_{\mathsf{lin},B}\,.$$

   Send $\mathbf{C}$ and $(\mathbf{v}^{(i)})_{i\in[2]}$.

2. V : Check $\mathbf{C} \times [1, B]^\top \overset{?}{=} \mathsf{cm}_{\mathbf{f}}$ and $\mathbf{v}^{(0)} + B\mathbf{v}^{(1)} \overset{?}{=} \mathbf{v}$. Return $\perp$ if fail.

3. Return the reduced statement $\mathbb{x}_o = (\mathbf{C}_{*,i}, \mathbf{r}, \mathbf{v}^{(i)})_{i \in [2]}$ and $\mathbb{w}_o = \mathbf{F}$.

Remark 5.2. In applications like IVC/PCDs, L is typically kept small (e.g., < 10) to minimize the recursive verifier circuit size, and the norm $\bar{b}$ of the output witness in Construction 5.2 is much smaller than B^2. In this case, to improve efficiency, we can delay the decomposition step until more folding steps are completed, or replace the norm bound B in $\Pi_{\mathsf{decomp},B}$ with $\sqrt{\bar{b}}$.

Lemma 5.1. *$\Pi_{\mathsf{decomp},B}$ is RoK from $\mathcal{R}_{\mathsf{lin},B^2}$ to $\mathcal{R}^{(2)}_{\mathsf{lin},B}$ with no knowledge error.*

Proof. The proof is a simple adaptation to that in Lemma 3.3 of [BC24]. Public reducibility and completeness are trivial. Next, we prove knowledge soundness. Given malicious prover's output $\mathbb{x}_o = (\mathbf{C}_{*,i}, \mathbf{r}, \mathbf{v}^{(i)})_{i \in [2]}$ and $\mathbb{w}_o = \mathbf{F}$, the extractor simply outputs $\mathbf{f} := \mathbf{F} \times [1, B]^\top$. If the verifier checks at Step 2 pass and $(\mathbb{x}_o, \mathbb{w}_o) \in \mathcal{R}^{(2)}_{\mathsf{lin},B}$, then $\|\mathbf{f}\|_\infty < B^2$, and other checks in $\mathcal{R}_{\mathsf{lin},B^2}$ are satisfied w.r.t. $\mathbb{x} = (\mathsf{cm}_\mathbf{f}, \mathbf{r}, \mathbf{v})$. Thus, $(\mathbb{i}, \mathbb{x}, \mathbf{f}) \in \mathcal{R}_{\mathsf{lin},B^2}$ and the claim holds. $\square$

5.3 Efficiency estimate

Next, we analyze the complexity of the reduction of knowledge from $\mathcal{R}_{\mathsf{comp}} \times \mathcal{R}_{\mathsf{acc}}$ to $\mathcal{R}_{\mathsf{acc}}$. The scheme we consider is the version that applies the technique from Remark 4.6 and Remark 4.7.

Theorem 5.3. *Fix $\bar{\mathcal{S}}, L, B = (d')^k$ such that $|\bar{\mathcal{S}}| \geq 2^\lambda$ and (28) holds. Assume that $[\![\cdot]\!] : R_q^n \to R_q^\kappa$ is $(2B^2, \bar{\mathcal{S}} - \bar{\mathcal{S}})$-binding and $d \gg \max(\kappa, \log n)$. Let n_{lin} be the number of linear checks in $\mathcal{R}_{\mathsf{lin},B}$. The RoK from $\mathcal{R}_{\mathsf{comp}} \times \mathcal{R}_{\mathsf{acc}}$ to $\mathcal{R}_{\mathsf{acc}}$, combined with the optimization in Remark 4.6 and Remark 4.7, has the following complexity:*

Prover time: *Dominated by $Ln\kappa$ R_q-multiplications and $O(Ln\kappa dk)$ R_q-additions.*

Verifier time (excluding hashing): *Dominated by $O(Ldk)$ R_q-multiplications.*

Online instance size: *$(L-2) \cdot [(\kappa + 2n_{\mathsf{lin}}) \cdot \log |R_q| + 2 \log n \log |\mathcal{C}|]$ bits.*

Accumulated instance size: *$2 \cdot [(\kappa + 2n_{\mathsf{lin}}) \cdot \log |R_q| + 2 \log n \log |\mathcal{C}|]$ bits.*

Folding proof size: *Dominated by $L(5\kappa + 6) + 10$ R_q-elements, $L(dk + 5)$ $\bar{\mathcal{S}}$-elements, and $L(2dk + d + 1)$ $\mathcal{C}$-elements.*

Remark 5.3 (Further optimizing proof sizes). We can further shorten the proof size by replacing the $5L$ commitments with 5 commitments. The idea is to have a *single* batch of helper commitments (e.g., $(\!|\mathbf{M}_\mathbf{f}|\!)$, $[\![\mathbf{m}_\tau]\!]$, $[\![\mathbf{h}]\!]$, etc.) for all of the L inputs, rather than having separate helper commitments for each input.

Remark 5.4. An $\bar{\mathcal{S}}$-element is much smaller than an R_q-element. E.g., if we choose $d = 64$ and each coordinate of $\bar{\mathcal{S}}$ to be in $\{-1, 0, 1, 2\}$, it takes less than $2d$ bits to represent an element in $\bar{\mathcal{S}}$, compared to $d \log q$ bits for an R_q-element.

Proof of Theorem 5.3. We first analyze prover complexity. Since the sumchecks are all instantiated over fields (See Remark 4.6), even with more parallel repetitions (or running over extension fields) for small q, they remain much cheaper than computing commitments over R_q. Also, it is much more efficient to compute the commitments to $\tau_{\mathbf{D}} \in (-d', d')^n$ and $\mathbf{m}_\tau = \mathsf{exp}(\tau_{\mathbf{D}})$ (at Step 1) than standard Ajtai commitments: $[\![\tau_{\mathbf{D}}]\!]$ only requires scalar multiplications rather than R_q-multiplications; and because $\mathbf{m}_\tau \in \mathcal{M}^n$, by Remark 4.3, $[\![\mathbf{m}_\tau]\!]$ takes only $O(n\kappa)$ R_q-additions.

Therefore, for each online input $i \in [L-2]$, the dominant cost is for computing $[\![\mathbf{f}_i]\!]$ (which takes $n\kappa$ R_q-multiplications), plus the task of computing $[\![\mathbf{M}_{\mathbf{f}_i}]\!]$ where $\mathbf{M}_{\mathbf{f}_i} \in \mathcal{M}^{n \times dk}$. By Remark 4.3, the latter takes $O(n\kappa dk)$ R_q-additions. Additionally, the prover needs to compute two decomposed commitments in $\Pi_{\mathsf{decomp},}$. Thus the prover complexity is as claimed.

Next, we analyze verifier complexity. With the optimization from Remark 4.7, there are two main expensive steps: (i) the sumcheck verifier at Step 5 of Construction 4.5, which takes $O(dk)$ R_q-operations for each input $i \in [L]$; (ii) the folding operation at Step 6, which takes $O(\kappa)$ R_q-operation for each input $i \in [L]$. Since $d \gg \kappa$ in practice, the verifier complexity is as claimed.

The instance sizes are correct by inspection. Next, we analyze the folding proof size. Since the sumchecks are instantiated over fields and only have $\log n \ll d$ rounds, their communication complexities are much smaller compared to commitments and R_q-elements.

For each input $i \in [L]$, besides the 2 commitments, 2 R_q-elements, and $2dk$ $\mathcal{C}$-elements introduced from the optimization in Remark 4.7, the proof size is dominated by the 2 commitments $C_{\mathbf{M_f}}, \mathsf{cm}_{\mathbf{m}_\tau}$ sent at Step 1 of Construction 5.1, the $dk+5$ folding challenges $(\mathbf{s}, \mathbf{s}')$ in $\bar{\mathcal{S}}$ sent at Step 2, the $[\![\mathbf{h}]\!]$-commitment sent at Step 3, and the 4 R_q-elements $\mathbf{e}[0..2], u$ in Construction 4.5. Additionally, the verifier sends a vector $(\mathbf{v}, a) \in \mathcal{C}^{d+1}$ at Step 2 of Construction 4.4. In total, they are $(5\kappa + 6)$ R_q-elements, $dk + 5$ $\bar{\mathcal{S}}$-elements, and $2dk + d + 1$ $\mathcal{C}$-elements, all multiplied by L. Finally, we need to separately add the $\mathbf{e}_o$-values (consisting of $8 + 2$ R_q-elements) at Step 5 of Construction 4.5. Thus the claim holds. □

Comparison with LatticeFold. LatticeFold+ is a significant improvement over LatticeFold [BC24], both asymptotically and concretely.

In LatticeFold, the prover's complexity is dominated by an n-sized degree-4 sumcheck over R_q and the computation of $L \log_2(B)$ decomposed commitments. This requires at least $O(Ln\kappa \log_2(B)$ R_q-multiplication, which is $\Omega(\log_2(B))$-times worse than LatticeFold+.

The verifier circuit size of LatticeFold+, dominated by the Fiat-Shamir transform, is also much better. In LatticeFold, it takes more than $L \log_2(B)$ decomposed commitments and $\Omega(\log_2(n))$ R_q-elements (for sumcheck executions) into its transcript. This is at least $\Omega(\log_2(B))$-times worse than LatticeFold+.

Concrete prover complexity. By Theorem 5.3, Remark 4.3, Remark 4.6, for typical parameters (e.g., 128-bit or 64-bit q), we expect prover time to be comparable to the cost of only committing to input witnesses. Given the high efficiency of module-based Ajtai commitments (e.g., see Footnote 2), we anticipate promising performance and leave concrete implementation for future work.

Concrete proof sizes. As noted in Remark 4.7, proof size is crucial to minimize recursive verifier circuit complexity in applications like IVC/PCD. We provide a candidate set of parameters to show how small a folding proof can be. While not highly optimized, this choice serves as a baseline, and we expect further efficiency gains with better parameter tuning.

L	q	d	n	$\bar{S}$	B	k	κ	proof size	# of hashes
3	128-bit	64	2^{21}	$\{-1,0,1,2\}^d$	2^{10}	2	9	$\approx$ 95 KB	≈ 95

The parameters are chosen for 128-bit conjectured hardness against the best-known attacks on Module-SIS [APS15; Esg+19; ADPS16; BDGL16]. The prime q is selected so that R_q splits into 16 factors, and by Lemma 2.4, $\bar{S}$ is indeed a strong sampling set. We set $n = 2^{21}$ to allow proving a R1CS statements over[18] $\mathbb{F}_{q^4}$ of size $16 \cdot 2^{21} = 2^{25}$.

The proof size is $\approx$ 185KB without the optimization from Remark 5.3 and $\approx$ 95KB if we add the optimization, which translates to $\approx$ 95 hashes (over R_q) for the Fiat-Shamir transform. After the optimization, the proof size is approaching LaBRADOR [BS23] — the shortest known lattice-based NARK, which achieves $\approx$ 60KB proofs for a 2^{20}-sized statement. As noted in [BC24], LaBRADOR has a linear-time verifier and cannot be used to construct recursive SNARKs.

6 Conclusion and future work

We presented LatticeFold+, a lattice-based folding technique that leverages a new purely algebraic ring-based range check along with a commitment transformation protocol that converts a statement about double commitments into a linear statement that can be folded. The result is a lattice-based folding system that is faster than LatticeFold, produces shorter proofs, and has a simpler verification circuit.

This work raises two interesting directions for future work. First, throughout the paper we used the ℓ_∞ norm rather than ℓ_2 to simplify the analysis of our sub-protocols. However, the ℓ_∞ norm leads to slightly worse parameters in the definition of Module-SIS compared to ℓ_2. It would be interesting to develop an ℓ_2 variant of all of our protocols. Second, we defined all of our protocols over $\mathbb{Z}_q$ modules for a prime q. In practice, there may be interest in implementing these protocols modulo a power of two. It would be interesting to generalize all these results to operate over a non-prime modulus.

[18] By Remark 4.1 of [BC24], we can pack 16 R1CS field constraints to an R_q-constraint.

Acknowledgments. 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 necessarily reflect the views of DARPA.

References

[ACK21] Thomas Attema, Ronald Cramer, and Lisa Kohl. "A Compressed Σ-Protocol Theory for Lattices". In: *CRYPTO 2021, Part II*. Ed. by Tal Malkin and Chris Peikert. Vol. 12826. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 549–579. DOI: 10.1007/978-3-030-84245-1_19.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. "Post-quantum Key Exchange - A New Hope". In: *USENIX Security 2016*. Ed. by Thorsten Holz and Stefan Savage. USENIX Association, Aug. 2016, pp. 327–343.

[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.

[AL21] Martin R. Albrecht and Russell W. F. Lai. "Subtractive Sets over Cyclotomic Rings - Limits of Schnorr-Like Arguments over Lattices". In: *CRYPTO 2021, Part II*. Ed. by Tal Malkin and Chris Peikert. Vol. 12826. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 519–548. DOI: 10.1007/978-3-030-84245-1_18.

[Alb+22] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. "Lattice-Based SNARKs: Publicly Verifiable, Preprocessing, and Recursively Composable - (Extended Abstract)". In: *CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. LNCS. Springer, Cham, Aug. 2022, pp. 102–132. DOI: 10.1007/978-3-031-15979-4_4.

[ALS20] Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. "Practical Product Proofs for Lattice Commitments". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 470–499. DOI: 10.1007/978-3-030-56880-1_17.

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. “On the concrete hardness of learning with errors”. In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203.

[AS24] Arasu Arun and Srinath Setty. *Nebula: Efficient read-write memory and switchboard circuits for folding schemes*. Cryptology ePrint Archive, Paper 2024/1605. 2024. URL: https://eprint.iacr.org/2024/1605.

[Bau+18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. “Sub-linear Lattice-Based Zero-Knowledge Arguments for Arithmetic Circuits”. In: *CRYPTO 2018, Part II*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10992. LNCS. Springer, Cham, Aug. 2018, pp. 669–699. DOI: 10.1007/978-3-319-96881-0_23.

[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.

[BBHR18a] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. “Fast Reed-Solomon Interactive Oracle Proofs of Proximity”. In: *ICALP 2018*. Ed. by Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella. Vol. 107. LIPIcs. Schloss Dagstuhl, July 2018, 14:1–14:17. DOI: 10.4230/LIPIcs.ICALP.2018.14.

[BBHR18b] 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.

[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.

[BCPS18] Anurag Bishnoi, Pete L Clark, Aditya Potukuchi, and John R Schmitt. “On zeros of a polynomial in a finite grid”. In: *Combinatorics, Probability and Computing* 27.3 (2018), pp. 310–333.

[BCS21] Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. "Sumcheck Arguments and Their Applications". In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 742–773. DOI: 10.1007/978-3-030-84242-0_26.

[BCS23] Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. "Lattice-Based Succinct Arguments for NP with Polylogarithmic-Time Verification". In: *CRYPTO 2023, Part II*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14082. LNCS. Springer, Cham, Aug. 2023, pp. 227–251. DOI: 10.1007/978-3-031-38545-2_8.

[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.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. "New directions in nearest neighbor searching with applications to lattice sieving". In: *27th SODA*. Ed. by Robert Krauthgamer. ACM-SIAM, Jan. 2016, pp. 10–24. DOI: 10.1137/1.9781611974331.ch2.

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

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

[BLNS20] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. "A Non-PCP Approach to Succinct Quantum-Safe Zero-Knowledge". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 441–469. DOI: 10.1007/978-3-030-56880-1_16.

[BMNW24] Benedikt Bünz, Pratyush Mishra, Wilson Nguyen, and William Wang. *Arc: Accumulation for Reed–Solomon Codes*. Cryptology ePrint Archive, Paper 2024/1731. 2024. URL: https://eprint.iacr.org/2024/1731.

[Bre+24] Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D. Rothblum, and Hadas Zeilberger. *Blaze: Fast SNARKs from Interleaved RAA Codes.* Cryptology ePrint Archive, Paper 2024/1609. 2024. URL: https://eprint.iacr.org/2024/1609.

[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+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *2018 IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, May 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.

[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.

[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.

[CCKP19] Shuo Chen, Jung Hee Cheon, Dongwoo Kim, and Daejun Park. *Verifiable Computing for Approximate Computation.* Cryptology ePrint Archive, Report 2019/762. 2019. URL: https://eprint.iacr.org/2019/762.

[CMNW24] Valerio Cini, Giulio Malavolta, Ngoc Khanh Nguyen, and Hoeteck Wee. "Polynomial Commitments from Lattices: Post-quantum Security, Fast Verification and Transparent Setup". In: *CRYPTO 2024, Part X.* Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14929. LNCS. Springer, Cham, Aug. 2024, pp. 207–242. DOI: 10.1007/978-3-031-68403-6_7.

[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.

[CT10] Alessandro Chiesa and Eran Tromer. "Proof-Carrying Data and Hearsay Arguments from Signature Cards". In: *ICS 2010.* Ed. by Andrew Chi-Chih Yao. Tsinghua University Press, Jan. 2010, pp. 310–331.

[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.

[EG23] Liam Eagen and Ariel Gabizon. *ProtoGalaxy: Efficient ProtoStar-style folding of multiple instances*. Cryptology ePrint Archive, Report 2023/1106. 2023. URL: https://eprint.iacr.org/2023/1106.

[ENS20] Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. "Practical Exact Proofs from Lattices: New Techniques to Exploit Fully-Splitting Rings". In: *ASIACRYPT 2020, Part II*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. LNCS. Springer, Cham, Dec. 2020, pp. 259–288. DOI: 10.1007/978-3-030-64834-3_9.

[Esg+19] Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. "Short Lattice-Based One-out-of-Many Proofs and Applications to Ring Signatures". In: *ACNS 19International Conference on Applied Cryptography and Network Security*. Ed. by Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung. Vol. 11464. LNCS. Springer, Cham, June 2019, pp. 67–88. DOI: 10.1007/978-3-030-21568-2_4.

[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.

[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.

[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.

[KS24a] Abhiram Kothapalli and Srinath Setty. *NeutronNova: Folding everything that reduces to zero-check*. Cryptology ePrint Archive, Paper 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.

[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.

[LNP22] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. "Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General". In: *CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. LNCS. Springer, Cham, Aug. 2022, pp. 71–101. DOI: 10.1007/978-3-031-15979-4_3.

[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.

[Moh23] Nicolas Mohnblatt. *Sangria: a folding scheme for PLONK*. link. 2023.

[NBS23] Wilson D. Nguyen, Dan Boneh, and Srinath T. V. Setty. "Revisiting the Nova Proof System on a Cycle of Curves". In: *5th Conference on Advances in Financial Technologies, AFT 2023*. Vol. 282. LIPIcs. 2023, 18:1–18:22.

DOI: 10.4230/LIPICS.AFT.2023.18. URL: https://doi.org/10.4230/LIPIcs.AFT.2023.18.

[Net24] Nethermind. *Latticefold and lattice-based operations performance report.* 2024.

[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.

[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.

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

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

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

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

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

[Xie+22] Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, Fan Zhang, Yupeng Zhang, Yongzheng Jia, Dan Boneh, and Dawn Song. "zkBridge: Trustless Cross-chain Bridges Made Practical". In: *ACM CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM Press, Nov. 2022, pp. 3003–3017. DOI: 10.1145/3548606.3560652.

[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 Reducing R1CS to the general linear relation

In this appendix we describe a reduction of knowledge from a committed R1CS relation to the general linear relation from Section 3. First, let us define the committed R1CS relation.

Definition A.1 (Committed R1CS)**.** *A* ***committed R1CS relation*** *is an indexed relation parameterized by* $B \in \mathbb{N}$*:*

$$\mathcal{R}_{\mathsf{cR1CS},B} := \left\{ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{i} = (\mathbf{A}, \mathbf{B}, \mathbf{C} \in R_q^{n\times m}), \\ \mathbb{x} = \left(\mathsf{cm} \in R_q^{\kappa},\ \mathbf{x} \in R_q^{\ell_{\mathsf{in}}}\right), \quad \mathbb{w} = \mathbf{f} \in R_q^n \quad \text{where} \\ \mathsf{cm} = [\![\mathbf{f}]\!],\quad \|\mathbf{f}\|_\infty < B, \quad \text{and for} \ \ \mathbf{z} = \mathbf{G}_{B,\hat{\ell}}^\top \cdot \mathbf{f} \in R_q^m \ : \\ \quad (\mathbf{A}\mathbf{z}) \circ (\mathbf{B}\mathbf{z}) = (\mathbf{C}\mathbf{z}), \quad \mathbf{z}[0 \ldots \ell_{\mathsf{in}}] = (1, \mathbf{x}) \end{array} \right\} \tag{30}$$

where $\hat{\ell} := \lceil \log_B(q) \rceil$ *and* $\mathbf{G}_{B,\hat{\ell}} \in \mathbb{Z}_q^{m\hat{\ell}\times m} = \mathbb{Z}_q^{n\times m}$ *is the gadget matrix from Section 2.1.*

In this definition we assumed that the dimension of the R1CS matrices is $n \times m$ and that $n = m \times \hat{\ell}$, where $\hat{\ell}$ is the expansion factor of the gadget matrix $\mathbf{G}_{B,\hat{\ell}}$. One can just as easily describe a reduction of knowledge when the R1CS matrices are $n' \times m$, for an unconstrained n', however this complicates the tensor notation from Remark 2.2 since it requires tensors on varying number of variables. To keep things simple we assume $n = m\times\hat{\ell}$.

We first introduce a simplified linear relation,

$$\mathcal{R}'_{\mathsf{lin},B} := \left\{ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \ : \ \begin{array}{l} \mathbb{i} = ([\![\cdot]\!],\ (\mathbf{M}^{(i)} \in R_q^{n\times n})_{i\in[n_{\mathsf{lin}}]}), \\ \mathbb{x} = (\mathsf{cm}_{\mathbf{f}},\ \mathbf{r} \in \mathcal{C}^{\log n},\ \mathbf{v} \in R_q^{n_{\mathsf{lin}}}), \quad \mathbb{w} = \mathbf{f} \in R_q^n \quad \text{s.t.} \\ (\|\mathbf{f}\|_\infty < B) \ \wedge \ (\mathsf{cm}_{\mathbf{f}} = [\![\mathbf{f}]\!]) \ \wedge \\ \left(\forall i \in [n_{\mathsf{lin}}] :\ \langle \mathbf{M}^{(i)} \cdot \mathbf{f}, \mathsf{tensor}(\mathbf{r})\rangle = \mathbf{v}_i\right) \end{array} \right\} \tag{31}$$

which is almost identical to $\mathcal{R}_{\mathsf{lin},B}$ except that we replace $\mathsf{M}_q := R_q \times R_q$, $\mathsf{M}_{\mathcal{C}} := \mathcal{C} \times \mathcal{C}$ (from $\mathcal{R}_{\mathsf{lin},B}$) with R_q, $\mathcal{C}$ respectively.

The protocol to reduce $\mathcal{R}_{\mathsf{cR1CS},B}$ to $\mathcal{R}'_{\mathsf{lin},B}$ is shown in Figure 1. Its completeness, knowledge soundness, and public reducibility are analyzed in [BC24, Sec. 4]. Let $\mathbb{x}_o :=$

$\big(\mathsf{cm}, \mathbf{r}_o, (v, v_{\mathbf{A}}, v_{\mathbf{B}}, v_{\mathbf{C}})\big)$ denote the reduced instance in $\mathcal{R}'_{\mathsf{lin},B}$. To complete the reduction, we expand $\mathbb{x}_o$ to an instance $\mathbb{x}'_o$ in $\mathcal{R}_{\mathsf{lin},B}$ defined as follows:

$$\mathbb{x}'_o := \big(\mathsf{cm}, \mathbf{r}'_o := (\mathbf{r}_o, \mathbf{r}_o) \in \mathsf{M}_{\mathcal{C}}, ((v, v), (v_{\mathbf{A}}, v_{\mathbf{A}}), (v_{\mathbf{B}}, v_{\mathbf{B}}), (v_{\mathbf{C}}, v_{\mathbf{C}})) \in \mathsf{M}_q^4\big)\,.$$

Remark A.1. In Figure 1, the evaluation $v = \widetilde{\mathbf{f}}(\mathbf{r}_o)$ is never used. A reader may wonder why we need it in the reduced instance. This is because the accumulated instance in our folding scheme (Section 5.1) requires an evaluation check, which is the result of the sumcheck in Construction 5.2. So we add this evaluation value to align with the accumulated instance.

Parameters: A sumcheck challenge set $\mathcal{C}$ (Section 4)
Input: $\mathbb{i} = (\mathbf{A}, \mathbf{B}, \mathbf{C} \in R_q^{n \times m})$ and $\mathbb{x} := (\mathsf{cm}, \mathbf{x}) \in R_q^{\kappa} \times R_q^{\ell_{\mathsf{in}}}$ and $\mathbb{w} := \mathbf{f} \in R_q^n$
Output: $\mathbb{x}_o := \left(\mathsf{cm},\ \mathbf{r} \in \mathcal{C}^{\log n},\ (v, v_{\mathbf{A}}, v_{\mathbf{B}}, v_{\mathbf{C}}) \in R_q^4\right)$ and $\mathbb{w}_o := \mathbf{f}$
The protocol $\langle \mathsf{P}(\mathbb{x}; \mathbb{w}); \mathsf{V}(\mathbb{x}) \rangle$**:**

1. $\mathsf{V} \to \mathsf{P}$: V sends P a random vector $\mathbf{r} \leftarrow \mathcal{C}^{\log n}$.
2. $\mathsf{P} \leftrightarrow \mathsf{V}$: Define the polynomial $g \in R_q^{\leq 2}[X_1, \ldots, X_{\log n}]$ as

$$g(\mathbf{x}) := eq(\mathbf{r}, \mathbf{x}) \cdot [g_{\mathbf{A}}(\mathbf{x}) \cdot g_{\mathbf{B}}(\mathbf{x}) - g_{\mathbf{C}}(\mathbf{x})]$$

   where for every $\mathbf{M} \in \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$ set $\mathbf{M}' := \mathbf{M} \cdot \mathbf{G}_{B,\hat{\ell}}^{\top} \in R_q^{n \times n}$ and define

$$g_{\mathbf{M}}(\mathbf{x}) := \sum_{\mathbf{b} \in \{0,1\}^{\log n}} (\widetilde{\mathbf{M}'})(\mathbf{x}, \mathbf{b}) \cdot \widetilde{\mathbf{f}}(\mathbf{b}).$$

   P and V run a sum-check protocol for the claim $\sum_{\mathbf{b} \in \{0,1\}^{\log n}} g(\mathbf{b}) = 0$.
   Let $\mathbf{r}_o \leftarrow \mathcal{C}^{\log n}$ be the sum-check challenge vector. The protocol reduces to a random evaluation check $g(\mathbf{r}_o) \stackrel{?}{=} s$ for some $s \in R_q$.
3. $\mathsf{P} \to \mathsf{V}$: P sends V values $(v, v_{\mathbf{A}}, v_{\mathbf{B}}, v_{\mathbf{C}}) \in R_q^4$. When P is honest these values are $v := \widetilde{\mathbf{f}}(\mathbf{r}_o) = \langle \mathbf{f}, \mathsf{tensor}(\mathbf{r}_o) \rangle \in R_q$, and for every $\mathbf{M} \in \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$

$$v_{\mathbf{M}} := \sum_{\mathbf{b} \in \{0,1\}^{\log n}} \widetilde{\mathbf{M}'}(\mathbf{r}_o, \mathbf{b}) \cdot \widetilde{\mathbf{f}}(\mathbf{b}) \in R_q$$

   where $\mathbf{M}' := \mathbf{M} \cdot \mathbf{G}_{B,\hat{\ell}}^{\top}$. Applying Remark 2.2 to the columns of $\mathbf{M}'$ shows that

$$v_{\mathbf{M}} = \left((\mathbf{M}')^{\top} \cdot \mathsf{tensor}(\mathbf{r}_o)\right)^{\top} \cdot \mathbf{f} = (\mathbf{M}' \cdot \mathbf{f})^{\top} \cdot \mathsf{tensor}(\mathbf{r}_o) = \langle \mathbf{M}' \cdot \mathbf{f}, \mathsf{tensor}(\mathbf{r}_o) \rangle.$$

4. V computes $e := eq(\mathbf{r}, \mathbf{r}_o)$ and checks that $e \cdot (v_{\mathbf{A}} v_{\mathbf{B}} - v_{\mathbf{C}}) \stackrel{?}{=} s$.
5. The derived $\mathcal{R}'_{\mathsf{lin},B}$ triple $(\mathbb{i}_o, \mathbb{x}_o, \mathbb{w}_o)$ is defined by
   V outputs $\mathbb{i}_o := (\mathbf{I}_n, \mathbf{A}', \mathbf{B}', \mathbf{C}')$ and $\mathbb{x}_o := \left(\mathsf{cm}, \mathbf{r}_o, (v, v_{\mathbf{A}}, v_{\mathbf{B}}, v_{\mathbf{C}})\right)$,
   P outputs $\mathbb{w}_o := \mathbf{f}$

Figure 1: The protocol to reduce $\mathcal{R}_{\mathsf{cR1CS},B}$ to $\mathcal{R}'_{\mathsf{lin},B}$.