Neo: Lattice-based folding scheme for CCS over small fields and pay-per-bit commitments

Wilson Nguyen
Stanford University

Srinath Setty
Microsoft Research

Abstract. This paper introduces Neo, a new lattice-based folding scheme for CCS, an NP-complete relation that generalizes R1CS, Plonkish, and AIR. Neo's folding scheme can be viewed as adapting the folding scheme in HyperNova (CRYPTO'24), which assumes elliptic-curve based linearly homomorphic commitments, to the lattice setting. Unlike HyperNova, Neo can use "small" prime fields (e.g., over the Goldilocks prime). Additionally, Neo provides plausible post-quantum security.

Prior to Neo, folding schemes in the lattice setting, notably LatticeFold (ePrint 2024/257), worked with constraint systems defined over a cyclotomic polynomial ring. This structure allows packing a fixed batch of constraint systems over a small prime field into a single constraint system over a polynomial ring. However, it introduces significant overheads, both for committing to witnesses (e.g., the commitment scheme cannot take advantage of bit-width of values), and within the folding protocol itself (e.g., the sum-check protocol is run over cyclotomic polynomial rings). Additionally, the required ring structure places restrictions on the choice of primes (e.g., LatticeFold is not compatible with the Goldilocks field).

Neo addresses these problems, by drawing inspiration from both HyperNova and LatticeFold. A key contribution is a folding-friendly instantiation of Ajtai's commitments, with "pay-per-bit" commitment costs i.e., the commitment costs scale with the bit-width of the scalars (e.g., committing to a vector of bits is 32× cheaper than committing to a vector of 32-bit values). This scheme commits to vectors over a small prime field. It does so by transforming the provided vector into a matrix and committing to that matrix. We prove that this commitment scheme provides the desired linear homomorphism for building a folding scheme. Additionally, like HyperNova, Neo runs a single invocation of the sum-check protocol, where in HyperNova it is over the scalar field of an elliptic curve and in Neo it is over an extension of a small prime field.

Table of Contents

1 Introduction

A folding scheme [44] is a cryptographic primitive that reduces the task of checking that two instance-witness pairs are in some NP relation to the task of checking that a single instance-witness pair is in the same relation. As an example, for a circuit C and two public inputs (i.e., instances) x_1 and x_2, a folding scheme reduces the task of checking that there exists witnesses w_1 and w_2 such that $C(w_1, x_1) = 1$ and $C(w_2, x_2) = 1$ to the task of checking that there exists a single witness w for a specific public input x such that $C(w, x) = 1$. Furthermore, the verifier's work in a folding scheme is limited to roughly taking the weighted sum of the commitments to underlying witnesses. By using a folding scheme in a recursive manner, one can continually fold many instance-witness pairs into a single instance-witness pair, providing powerful primitives such as incrementally verifiable computation (IVC) [62] and proof-carrying data (PCD) [13].

Benefits of folding schemes. Folding schemes provide a more efficient approach to construct SNARKs [37, 49] that can scale to large computations.

A modern approach to construct SNARKs is to combine a polynomial interactive oracle proof (PIOP) [19, 25, 57] with a polynomial commitment scheme (PCS) [36], and then apply the Fiat-Shamir transformation [33]. In particular, the PIOP reduces the task of checking the validity of an instance-witness pair in some relation (e.g., R1CS) to a collection of tasks where each task is to check if a (committed) polynomial evaluates to a certain value at a certain point in their domain (i.e., a set of polynomial evaluation instance-witness pairs). In turn, the polynomial evaluation argument provided by the PCS is used by the prover to prove those polynomial evaluation instances. However, this approach only provides a "monolithic" SNARK, meaning that a prover must prove a fixed-sized computation at once. To scale to larger computations, one typically breaks the computation into smaller pieces and then uses SNARK recursion (a la IVC or PCD) to produce a succinct argument in a scalable manner [12].

Folding schemes provide a more direct and a more efficient approach. In particular, folding schemes allow recursion to operate at the "statement" level (i.e., prior to producing a PIOP or a PCS evaluation argument). This has two concrete benefits. First, the overheads from recursion are far lower than traditional SNARK recursion. For example, even with one of the earliest folding schemes, Nova [44], it only takes 10,000 R1CS gates to fold a proof. Whereas, traditional SNARK recursion takes millions of gates [24, 26]. Second, the prover incurs far less work from not having to produce a PIOP or a PCS evaluation argument. For instance, the prover in monolithic SNARKs such as Marlin [25] perform at least 20× higher work over simply committing to a witness. In contrast, with state-of-the-art folding schemes [17, 29, 42, 43], the prover's work is dominated by the cost to commit to a witness. This results in at least an order of magnitude speedup over monolithic SNARKs, and up to two orders of magnitude speedup when the witness contains values from a small subset of the entire field.[1]

[1] If the witness contains "small" field elements (e.g., they are from a small subset of the entire finite field say $\{0, 1, \ldots, 2^{32} - 1\}$), the state-of-the-art folding schemes perform

State-of-the-art folding schemes. Nova [44] formally introduced folding schemes. It also provides a folding scheme for R1CS, where the prover's work is dominated by two multi-scalar multiplications (MSMs) of size proportional to the circuit size, and the recursive verifier circuit is dominated by two group scalar multiplications and hashing a constant number of field elements.[2] Folding schemes further developed in HyperNova [42], Protostar [17], ProtoGalaxy [29], and NeutronNova [43]. These works provide folding schemes for more expressive relations such as CCS [59], a generalization of R1CS, Plonkish, and AIR, including lookup checks. With recent works, the prover's work is dominated by a single MSM that commits to a purported witness, and the verifier circuit size is dominated by 3 group scalar multiplications and hashing a constant number of field elements.

These state-of-the-art folding schemes [2, 17, 28, 29, 40–44, 52, 65] rely on cryptographic groups. Specifically, they leverage linearly homomorphic commitments for vectors over a finite field. In practice, these commitments are instantiated with Pedersen commitments over regular cycles of elliptic curves (e.g., Pallas/Vesta) or with KZG commitments over half pairing cycles of elliptic curves (e.g., BN254/Grumpkin).

The area of folding schemes has witnessed significant progress in just the last year. There is a lot of recent work to extend folding schemes with additional constructs such as a read-only and a read-write memory [8, 21, 30, 34], and some works prove non-uniform constraint systems (e.g., Plonkish) in a space-efficient manner [53] while others focus on providing an efficient on-chain verifier [64]. Due to their efficiency, some projects have adopted folding schemes as a foundation for proving virtual machine executions [3, 4]. In a similar vein, a recent work [43] provides a systematic framework to build efficient folding schemes for complex relations in a composable manner, which can then be used to prove virtual machine executions more efficiently.

---

up to 200× less work than a monolithic SNARK prover such as Marlin [25] because Marlin ends up committing to "random" field elements as part of its PIOP and PCS evaluation argument, which are at least an order of magnitude more expensive than committing to a witness with "small" field elements. Proof systems such as Spartan and variants [57, 59] incur lower overheads than Marlin, but they must still produce a PIOP and PCS evaluation argument.

[2] Prior to Nova [44], Halo [16] provides a recursive SNARK for achieving IVC/PCD where the verifier circuit does *not* verify a full SNARK proof. However, this approach still requires producing a PIOP and a PCS evaluation argument. Furthermore, Halo's verifier circuit is significantly larger than Nova's. Folding schemes not only aim to minimize work in a recursive verifier circuit, but also aim to avoid producing a SNARK in the first place. Following Halo and in concurrent with Nova, Bunz et al. [18] achieve IVC/PCD while only requiring NARKs in which the verifier provides a split-accumulation scheme. Folding schemes offer a cleaner abstraction and a more efficient instantiation. They also lead to IVC/PCD without even arguments of knowledge. Nova [44] provides additional context on works that inspired folding schemes.

1.1 Open problems and recent progress on those problems

Despite the above progress, there are two downsides associated with the aforementioned state-of-the-art folding schemes.

1. They rely on cryptographic groups based on elliptic curves. As a result, constraint systems that they work with are defined over the scalar field of an elliptic curve. For security (e.g., to ensure the hardness of DLOG), this field is a prime field where the modulus is approximately 256 bits. Even if the original computation is naturally represented with "small" numbers, the computation must be lifted to work with 256-bit fields. This incurs an "embedding" overhead. For example, computations on a real machine are naturally expressed with 32-bit or 64-bit field, but when using the aforementioned folding schemes, they must be expressed with 256-bit fields.
2. Due to their reliance on the hardness of the discrete logarithm problem, they are not post-quantum secure.

In the rest of this paper, when we say "small" prime fields, we refer to prime fields where the prime modulus q fits within a machine register. Examples of such prime modulus include M61 ($q = 2^{61} - 1$) and the Goldilocks prime ($q = 2^{64} - 2^{32} + 1$). Prime fields with such a modulus provide fast arithmetic; they also allow the use of vector instructions (SIMD) to perform multiple field operations at once.

Research question. Can we construct a folding scheme that provides better efficiency than the aforementioned state-of-the-art folding schemes while addressing the two problems noted above?

Recent progress and their downsides. Boneh and Chen [14] make significant progress toward addressing the above research question. They construct LatticeFold [14], a new folding scheme for CCS [59], where security holds under a structured lattice assumption. More recently, Lova [31] provides a Nova-like folding scheme using an unstructured lattice assumption. In another work, Arc [20] provides a folding scheme for general constraint systems including CCS while only relying on hash functions. However, compared to existing group-based folding schemes, all these works incur significant overheads.

Lova [31] constructs a folding scheme for the subset sum problem defined over the integers. In particular, it does not provide a folding scheme for CCS or even R1CS. So, if one were to use Lova for real world problems of interest, one must transform their CCS or R1CS instance-witness pairs to instance-witness pairs in the subset sum relation over the integers. This likely incurs significant overheads. Even ignoring these overheads, based on Lova's reported performance for the subset sum relation, Lova (for subset sum) is already multiple orders of magnitude slower than Nova (for R1CS). For a subset sum instance length of 2^{19}, Lova reports a prover time of ≈3,000 seconds [31, Table 2]. On a machine with the same number of vCPUs and lower speed, Nova reports 500 ms for an R1CS instance of size 2^{19}. Given this, Lova appears to be at least $6,000\times$ slower

than Nova. Representing R1CS with subset sum likely blows up the size of the corresponding subset sum instance substantially. If we include the overhead from R1CS to subset sum transformation (which could be at least $10\times$ and likely far higher), Lova is more than four orders of magnitude slower than Nova.

Arc [20] incurs high folding overheads.[3] To fold two instances, Arc requires $\lambda/\log(1/\rho)$ Merkle tree openings in the recursive verifier circuit, where λ is the security parameter and ρ is the Reed-Solomon code rate used. For $\lambda = 128$ and the widely used rate of $\rho = 1/2$ (which leads to the fastest encoding time for Reed-Solomon codes), Arc requires 128 Merkle tree openings. This translates to about $800,000$ constraints in R1CS when using a SNARK-friendly hash function such as Poseidon [35]. Note that this is a lower bound on the verifier circuit size as the verifier must perform other tasks besides verifying Merkle proofs. As a comparison point, Nova [1, 44] only needs ≈10,000 R1CS constraints in total for the entire verifier circuit.

LatticeFold [14] is arguably more "practical" than Lova (for the prover) and Arc (for the verifier circuit). However, LatticeFold has many significant downsides, especially when compared to existing state-of-the-art group-based folding schemes. For instance, LatticeFold is not a replacement for existing group-based schemes (e.g., HyperNova [42]) unless one is willing to accept a significant prover slow down. We now provide details of these downsides.

1. LatticeFold works with CCS defined over cyclotomic polynomial rings rather than over (small) prime fields. LatticeFold partially mitigates this issue with an approach to "pack" a batch of independent constraints defined over a "small" prime field into a single constraint over a cyclotomic polynomial ring [14, Remark 4.1]. However, this imposes a requirement that one must have a "data parallel" (or SIMD) constraint system. Additionally, "packing" introduces a significant performance problem: The prover's cost to commit to a vector of values is the same regardless of the bit-width of the values. For example, for a given vector length, it costs the same to commit to a vector of 64-bit values or a vector of 1-bit values.[4]
2. For security and to support packing of constraint systems over a small prime field (see [14, §3.3]), LatticeFold requires a particular type of cyclotomic polynomial ring that is not "fully splitting". This requires rephrasing LatticeFold's relations and modifying the overall protocol to accommodate the

[3] In an earlier work, Bunz et al. [22] provide a different hash-based folding scheme. However, the verifier circuit overheads of this construction is worse than those of Arc [20, Table 1]. Additionally, that construction only provides a "bounded depth" IVC (i.e., there exist concrete attacks if the depth of recursion exceeds a pre-determined bound).

[4] This is because CCS witness values are packed and embedded in the NTT form of cyclotomic ring elements. Additionally, LatticeFold's commitment scheme must decompose the corresponding ring elements in their coefficient form to ensure norm bounds. This entire process unfortunately prevents LatticeFold's commitment scheme from leveraging the bit-width of the original witness elements to provide a pay-per-bit commitment costs. We provide more details of this issue in Section 3.

smaller field, since the protocol descriptions assume a large prime field that causes the cyclotomic ring to fully split. Additionally, using small prime fields adds a τ (extension field degree) multiplicative factor loss to the efficiency of LatticeFold's protocols, due to the need to run a further τ instances in parallel. In LatticeFold's sample parameterization (see [14, §5]), despite the field being 64-bits in size, the extension field degree is required to be $\tau = 4$. This leads to a $4\times$ overhead in LatticeFold's protocols (due to the required repetition) and requires the use of a much larger (extension) field $\mathbb{F}_{q^4}$ (if the NTT representation is used).

3. As part of its folding scheme, LatticeFold runs the sum-check protocol over cyclotomic polynomial rings. Cyclotomic polynomial ring operations are 10–100$\times$ more expensive than (extension) field operations and even 256-bit fields, making the overall protocol significantly more expensive than desired.[5]
4. LatticeFold's use of cyclotomic polynomial rings imposes many requirements on the ring structure. This in turn limits the choice of the underlying "small" prime field. We find that LatticeFold's requirements do not allow one to use popular small field modulus such as Golidlocks' prime or M61.[6]

Beyond these, there is a less fundamental issue. LatticeFold runs two sequential invocations of the sum-check protocol, making the verifier perform twice the work. LatticeFold addresses this problem with an optimization that rearranges the protocol steps, allowing the batching of the two sum-check invocations [14, §4.3]. However, this protocol does not fit within LatticeFold's modular framework, so the security proofs provided in the paper do not cover this stand-alone protocol.

1.2 Our work in a nutshell: Neo

We address the aforementioned problems with our work, which we refer to as Neo. Our work draws inspiration from prior works including HyperNova [42] and LatticeFold [14], but introduces several new techniques. Before we introduce Neo's underlying techniques, we provide a brief overview.

Neo provides a folding scheme for CCS [59], an NP-complete relation that generalizes widely used arithmetizations such as R1CS, Plonkish, and AIR. Unlike HyperNova (where constraints are defined over the scalar field of an elliptic curve group) and LatticeFold (where the constraints are defined over a cyclotomic polynomial ring), the constraints that Neo folds are defined natively over a small

[5] Recent benchmarks report that a polynomial ring multiplication costs ≈213 ns [51]. Whereas, a field multiplication with M61 costs a fraction of a ns.

[6] As described, LatticeFold [14] works with cyclotomic polynomial rings where the modulus polynomial is of the form $X^d + 1$ and d is a power of 2. If the polynomial ring is instantiated with a popular small field such as a prime field over the Goldilocks' prime ($q = 2^{64} - 2^{32} + 1$, which is a 64 bit prime), the polynomial ring splits completely. This ruins the security of LatticeFold as this complete split makes the ring R_q isomorphic to F_q^d, so LatticeFold instantiated with this prime gives at most 64 bits bits of security. See Section 6 and [14, §3.3] for more details.

prime field. Specifically, our construction supports the use of popular fields such as M61 and the Goldilocks field. These fields feature extremely efficient field arithmetic implementations. Additionally, unlike LatticeFold, Neo does not have to pack multiple constraints over a prime field into a single constraint over a ring. So, Neo's folding scheme, like HyperNova, runs a single invocation of the sum-check protocol. In Neo, the sum-check protocol is run over an extension of a small prime field.[7]

Beyond this, Neo provides a new folding-friendly lattice-based commitment scheme. Unlike the commitment scheme in LatticeFold, our scheme provides a key performance property: the commitment costs scale linearly with the bit width of the values in the vectors committed. For example, for $n > 1$, it is $64\times$ cheaper to commit to a length-n vector of bits than it is to commit to a length-n vector of 64-bit values. This commitment scheme may be of independent interest.

Overall, Neo provides a folding scheme similar to HyperNova with the two added benefits: (1) Neo supports "small" prime fields (e.g., M61) and a light-weight commitment scheme, opening door for a faster prover; and (2) Neo is plausibly post-quantum secure.

1.3 A technical overview of Neo

We now provide an overview of various components in Neo.

(1) Folding-friendly lattice-based commitments with pay-per-bit commitment costs. A key contribution of our work is a new folding-friendly lattice-based commitment (Ajtai with a new embedding for elements). As discussed above, it works with vectors over a "small" prime field and provides a pay-per-bit commitment cost.

A starting point for Neo's commitment scheme is Ajtai commitments [5], which commits to a vector of cyclotomic polynomial ring elements, and the security holds under a structured lattice assumption: Module SIS. In particular, to commit to a vector of small field elements, we provide a more efficient mapping from a vector of elements from a small field F_q to a vector of cyclotomic polynomial ring elements, where the cyclotomic polynomial is defined over F_q. The vector of ring elements is then committed with Ajtai's commitment scheme.

Our mapping provides two key properties. First, it provides the aforementioned pay-per-bit commitment costs.[8] Discrete-log based commitment schemes such as Pedersen [54] and KZG [15, 36, 64], which can be used with HyperNova [42], also provide the property that it is cheaper to commit to a vector of bits than it is to commit to a vector of arbitrary field elements. But, as noted earlier, they do

[7] When using a 64-bit field, a degree-2 extension is sufficient for 128 bits of security.

[8] As noted earlier, pay-per-bit commitment cost property is not achieved by LatticeFold's commitment scheme because it uses a different map that must preserve more complex properties including the satisfiability of packed CCS constraints. Additionally, in our context, we are not restricted to the power-of-2 cyclotomic polynomial rings. This allows a wider choice of q.

not provide post-quantum security.[9] In other words, Neo's commitment scheme allows porting a performance property that is currently available within the discrete-log setting to the lattice setting while providing post-quantum security.

Second, we show that our commitment scheme provides the required linear homomorphism property. In particular, we treat a vector underneath a commitment as a multilinear polynomial represented in evaluation form over the Boolean hypercube. For example, a vector of size n uniquely determines a multilinear polynomial in $\ell = \log n$ variables. The commitment scheme then provides a folding scheme for evaluation claims. Informally, suppose that we have a collection of $\beta \geq 2$ commitments and claimed multilinear evaluations at an evaluation point over their entire domain: $\{(C_i, r, y_i)\}_{i\in[\beta]}$. Suppose we have the corresponding witnesses: $\{w_i\}_{i\in[\beta]}$. That is, witnesses are satisfying if and only if for $i \in [\beta]$, C_i is a commitment to w_i and that $\widetilde{w_i}(r) = y_i$. Neo's commitment scheme provides a reduction of knowledge (RoK) that outputs a new instance (C, r, y) and a witness w such that the new instance-witness pair is satisfying if and only if all the original instances are satisfying. With discrete-log-based commitments, it is quite easy to construct such a RoK. Whereas, in the lattice setting, many challenges arise. We provide details of these challenges and how we address them in Section 3. In a nutshell, the RoK must tame the norm growth when taking a random linear combination of commitments, which we address by adapting prior decomposition techniques along with a use of "small norm" challenges.

(2) A folding scheme for CCS. With the lattice-based commitment scheme in hand, devising a folding scheme for CCS is relatively straightforward. Our starting point here is the folding scheme in HyperNova [42],[10] which assumes a discrete-log-based linearly homomorphic commitments. Roughly speaking, we replace the commitment scheme in HyperNova with the aforementioned commitment scheme that provides the required linear homomorphism with respect to multilinear polynomial evaluation claims.

There are however some challenges. As mentioned above, our commitment scheme relies on decomposition techniques to tame norm growth of committed vectors. This decomposition process requires the prover to establish that the prover indeed decomposed its vectors correctly. We phrase these checks as a sum-check claim that is then proven with the sum-check protocol. At a high level, this idea is similar to how LatticeFold proves norm checks of decomposed vectors.

[9] Hash-based commitments such as FRI [11] and the commitment schemes used in Arc [20] do not provide a pay-per-bit cost. Some recent hash-based commitments, such as Binius [27], provide a pay-per-bit commitment costs, but as they are hash based, they incur much higher recursion overheads than group-based folding schemes. Binius also requires the use of binary fields and the constraint systems are defined over binary fields. Such constraint systems are not widely used in practice.

[10] An alternative is Protostar [17] or NeutronNova [43], both of which provide folding scheme verifier with a constant size whereas with HyperNova it is logarithmic-sized. Unfortunately, they require a truly linearly homomorphic commitment scheme, not just a scheme that can fold multilinear evaluation claims associated with committed vectors. Our commitment scheme cannot be used with those folding schemes.

However, due to a different mapping that we use to map CCS witness vectors to cyclotomic polynomial ring elements, we are able to avoid running the sum-check protocol over cyclotomic polynomial rings. In fact, we simply batch the norm check claim with the sum-check claim arising from CCS, and run the sum-check protocol over an extension of a small prime field. Section 4 provides details.

Note that this norm-check claim introduces additional work for the prover in the sum-check protocol as well as in the commitment scheme, relative to HyperNova. However, one can tame this with the following. Neo's folding scheme, like HyperNova's, can be naturally extended to a multi-folding scheme [42] i.e., it can fold multiple CCS instances at once. In particular, the additional work related to decomposition is performed only for the "running" instance-witness to which multiple CCS instance-witness pairs are folded. So, by folding multiple CCS instance-witness pairs at once, we can amortize the costs of decomposition.

(3) Security analysis. Proving the security of our commitment scheme as well as of the folding scheme for CCS is non-trivial. We structure our folding scheme for CCS with three reductions: (1) a reduction that allows decomposing an evaluation claim about a committed vector with a particular norm B into a batch of k instances with a lower norm $b < B^{1/k}$ (for some chosen value of k); (2) a reduction that folds a batch of $k+1$ instances with norm b into a single instance with norm at most B; and (3) a reduction that transforms a claim about a CCS instance-witness pair into a claim about linearized form of CCS instance-witness pair (this reduction is inherited from HyperNova). The first one is a RoK with standard completeness and knowledge soundness properties. Unfortunately, the second and the third reductions are not. To formally capture their properties, we introduce relaxed notions of knowledge soundness. Additionally, we prove that by sequentially composing these reductions with relaxed knowledge soundness guarantees, we get a RoK with standard knowledge soundness. These proof techniques may be of independent interest. Ultimately, we obtain Neo's folding scheme for CCS. Section 5 provides details.

(4) Concrete parameter choices. To instantiate Neo's folding scheme, we must choose a prime q as well as a suitable cyclotomic polynomial ring. The sum-check protocol will be run on an extension of a field F_q. For efficiency, it is imperative to choose a value of q that minimizes operations over these structures. We provide multiple different options for q including the very efficient Mersenne-61 i.e., $q = 2^{61} - 1$, the widely used Goldilocks field $q = 2^{64} - 2^{32} + 1$, and a third field that we refer to as "almost" Goldilocks field (AGL). The AGL field allows the use a power-of-2 cyclotomic polynomial ring. Section 6 provides details.

1.4 Lattice-based folding scheme for lookups and read-write memory

Neo extends easily to support folding lookup checks. Specifically, Shout [58] is a recent sum-check-based lookup argument. It can be viewed as a RoK from the lookup relation to the multilinear polynomial evaluation relation. Similar to how Neo transforms HyperNova [42] to a lattice-based folding scheme for

CCS, by leveraging our instantiation of Ajtai's commitments, we can use Shout's RoK to fold lookup relations alongside CCS, as both our folding scheme and Shout's RoK use the sum-check protocol [46]. The same approach works with Twist [58] to support a lattice-based folding scheme for read-write memory. For high performance, Shout and Twist rely on commitment schemes that can commit to sparse vectors efficiently. Prior to this work, the choice of commitment schemes includes group-based commitment schemes such as Pedersen [54] and HyperKZG [64], or hash-based commitment schemes over binary fields such as Binius [27]. Fortunately, Neo's commitment scheme provides the required performance property in the lattice setting.

Beyond this, supporting lookups alongside CCS opens up the possibility improving Neo further. In particular, within Neo's folding scheme, the prover establishes that every value in a committed vector are within a certain range. For a range of $[-b, b]$, our current approach, which is adapted from LatticeFold [14] to the prime field setting, is somewhat naive: the range check is performed by representing it with a degree-$2b$ sum-check instance. To minimize work in the sum-check protocol, b must be chosen to be small (e.g., $b = 2$). Instead of the naive approach, by using our adaptation of Shout, the degree of the multivariate polynomial in the sum-check can be made independent of b. We leave it to future work to incorporate these ideas into our folding scheme.

1.5 Constructing an IVC/PCD scheme, with proof compression

By applying prior compilers from folding schemes to IVC [42,44] and PCD [65] to Neo's folding scheme, we immediately obtain a lattice-based IVC/PCD scheme. Note that operations in the folding scheme verifier (e.g., extension field operations) can be natively represented in the finite field over which the constraint system is defined. Also, constructing an IVC/PCD scheme using Neo's folding schemes does not require any cycles of elliptic curves [41, 52].

With prior compilers, the IVC proof is a pair of instance-witness pairs, one in the CCS relation and another in a linearized version of the CCS relation (Section 4.1 provides details of the specific relations used in Neo). If IVC proof size is a concern, as in prior work [44], Neo's prover can instead provide a SNARK proof which proves the knowledge of a valid IVC proof. The SNARK proof can be exponentially smaller than the size of the underlying IVC proof. In particular, we can apply Neo's folding scheme to fold the two instance-witness pairs in an IVC proof into a single instance-witness pair in the linearized variant of CCS. We can then apply (Super)Spartan [57, 59] to reduce the task of proving the knowledge of a valid witness to a linearized CCS relation to a set of multilinear polynomial evaluation claims. Unfortunately, our lattice-based commitment scheme does not provide an efficient procedure to prove polynomial evaluations directly. Fortunately, we can use Spartan with a FRI-based polynomial commitment scheme [11] to prove the multilinear polynomial evaluations. Note that this preserves plausible post-quantum security. It also does *not* require any non-native arithmetic or "wrong" field emulation, since we natively support SNARK friendly fields like Goldilocks. In particular, the size of the circuit proven with

Spartan+FRI will be $O(n)$, where n is the size of the CCS witness polynomial. Note that the Spartan-based proof is an evaluation argument for our lattice-based commitment scheme, but a more direct approach would be preferable.

2 Preliminaries

In this section, we fix our notation and recall reductions of knowledge and the sum-check protocol. In Appendix A, we formally present multilinear polynomials and relevant properties. We adapt some preliminaries from a prior work [43].

2.1 Notation

We let λ to denote the security parameter. We let $\mathsf{negl}(\lambda)$ to denote a negligible function in λ. Throughout the paper, the depicted asymptotics depend on λ, but we elide this for brevity. We let PPT denote probabilistic polynomial time and let EPT denote expected probabilistic polynomial time. We let $[n]$ denote the set $\{1, \ldots, n\}$. We let $\{u_i\}_{i\in[n]}$ denote the set $\{u_1, \ldots, u_n\}$.

We let $\mathbb{F}$ denote a prime field of order q, and $\mathbb{K} \supseteq \mathbb{F}$ be the smallest degree extension field of $\mathbb{F}$ such that $1/|\mathbb{K}| = \mathsf{negl}(\lambda)$. Let $\mathbb{F}^n$ denote vectors of length n over elements in $\mathbb{F}$. For a scalar $s \in \mathbb{F}$, we define the scalar matrix $\overline{s} := s \cdot \mathrm{I}_d$, where I_d is the $d \times d$ identity matrix.

We write $\mathbb{F}^d[X_1, \ldots, X_n]$ to denote multivariate polynomials over field $\mathbb{F}$ in the variables $(X_1, \ldots, X_n)$ with degree bound $\leq d$ for each variable. We omit the superscript if there is no degree bound. We let $\mathsf{eq}(x, y) \in \mathbb{F}^1[X_1, \ldots, X_\ell, Y_1, \ldots, Y_\ell]$ denote the polynomial that outputs 1 if $x = y$ and 0 otherwise for $x, y \in \{0,1\}^\ell$. We define ZS_ℓ as the set of all multivariate polynomials $F \in \mathbb{F}[X_1, \ldots, X_\ell]$ such that for all $x \in \{0,1\}^\ell$, $F(x) = 0$ (i.e. vanish over the Boolean hypercube). For vector $v \in \mathbb{F}^n$ we let $\widetilde{v} \in \mathbb{F}^1[X_1, \ldots, X_{\log n}]$ denote the multilinear polynomial extension of v (i.e., $\widetilde{v}(i) = \sum_j \mathsf{eq}(i, j) \cdot v_j$). For matrix $M \in \mathbb{F}^{d\times n}$ we let $\widetilde{M} \in \mathbb{F}^1[X_1, \ldots, X_{\log(dn)}]$ denote the multilinear polynomial extension of M (i.e., $\widetilde{v}(i, j) = \sum_{u,v} \mathsf{eq}((i, j), (u, v)) \cdot M_{u,v}$). In other words, the multilinear extension of a matrix M is the multilinear extension of the vector $m := M^{(1)}||M^{(2)}|| \ldots ||M^{(d)}$ which is the concatenation of the rows of M. For a vector $r \in \mathbb{K}^{\log n}$ (for n a power of two), we denote the tensor as $\widehat{r} = \bigotimes_{i=1}^{\log n}(r_i, 1 - r_i)$. Notably, for a vector $f \in \mathbb{F}^n$, we have $\widetilde{f}(r) = \langle f,\ \widehat{r}\rangle$. $M^\intercal$ is the transpose of the matrix M.

Norm For an element $a \in \mathbb{F}$, we define $\|a\|_\infty$ as follows: Let $a' \in [0,\ q-1]$ denote the integer representation of $a \mod q$. If $a' \leq (q-1)/2$, then $\|a\|_\infty = a'$. Otherwise, if $a' > (q-1)/2$, then $\|a\|_\infty = a' - q$. This maps each $\mathbb{F}$ element into the interval $[-(q-1)/2,\ (q-1)/2] \subseteq \mathbb{Z}$. For a matrix $Z \in \mathbb{F}^{d\times m}$, we define the ℓ_∞-norm $\|Z\|_\infty$ to be the max infinity norm of its elements.

The sum-check protocol [46] The sum-check protocol is a classic interactive proof protocol between two PPT algorithms $(\mathcal{P}, \mathcal{V})$ that checks that the sum

of evaluations of a ℓ-variate polynomial $Q \in \mathbb{F}^{\leq d}[X_1, \ldots, X_\ell]$ on the Boolean hypercube results in some value T. The output of the sum-check protocol is a claim that $v \stackrel{?}{=} Q(r)$ for some random point $r \in \mathbb{F}^\ell$ and claimed evaluations v, which the verifier $\mathcal{V}$ can query Q to check. The protocol is public-coin, has a completeness error of 0, and has a soundness error of $\leq \ell d / |\mathbb{F}|$. More generally, the field can be chosen to be an extension field $\mathbb{K}$. In this case, the soundness error is $\leq \ell d / |\mathbb{K}|$. A self-contained description of the sum-check protocol can be found in this note [61].

2.2 Reductions of knowledge

We now recall the reductions of knowledge framework, introduced by Kothapalli and Parno [39]. Reductions of knowledge are a generalization of arguments of knowledge, in which a verifier interactively *reduces* checking a prover's knowledge of a witness in a relation $\mathcal{R}_1$ to checking the prover's knowledge of a witness in another (simpler) relation $\mathcal{R}_2$. In particular, both parties take as input a claimed instance u_1 to be checked, and the prover additionally takes as input a corresponding witness w_1 such that $(u_1, w_1) \in \mathcal{R}_1$. After interaction, the prover and verifier together output a new instance u_2 to be checked in place of the original instance, and the prover additionally outputs a corresponding witness w_2 such that $(u_2, w_2) \in \mathcal{R}_2$.

Definition 1 (Reduction of knowledge [38, 39]). *A **reduction** from $\mathcal{R}_1$ to $\mathcal{R}_2$ is defined by PPT algorithms $(\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V})$ called the generator, encoder (deterministic), prover, and verifier respectively with the following interface.*

- $\mathcal{G}(1^\lambda, \mathsf{sz}) \to \mathsf{pp}$*: Takes as input a security parameter 1^λ and size parameters sz. Outputs public parameters pp.*
- $\mathcal{K}(\mathsf{pp}, \mathsf{s}) \to (\mathsf{pk}, \mathsf{vk})$*: Takes as input public parameters pp and a structure s. Outputs a prover key pk and a verifier key vk.*
- $\mathcal{P}(\mathsf{pk}, u_1, w_1) \to (u_2, w_2)$*: Takes as input a proving key pk and an instance-witness pair (u_1, w_1). Interactively reduces the task of checking $(\mathsf{pp}, \mathsf{s}, u_1, w_1) \in \mathcal{R}_1$ to the task of checking $(\mathsf{pp}, \mathsf{s}, u_2, w_2) \in \mathcal{R}_2$.*
- $\mathcal{V}(\mathsf{vk}, u_1) \to u_2$*: Takes as input a verifier key vk and an instance u_1 in $\mathcal{R}_1$. Interactively reduces the task of checking the instance u_1 to the task of checking a new instance u_2 in $\mathcal{R}_2$.*

*Let $\langle \mathcal{P}, \mathcal{V} \rangle$ denote the interaction between $\mathcal{P}$ and $\mathcal{V}$. We treat $\langle \mathcal{P}, \mathcal{V} \rangle$ as a function that takes as input $((\mathsf{pk}, \mathsf{vk}), u_1, w_1)$ and runs the interaction on the prover's input (pk, u_1, w_1) and the verifier's input (vk, u_1). At the end of the interaction, $\langle \mathcal{P}, \mathcal{V} \rangle$ outputs the verifier's instance u_2 and the prover's witness w_2. A **reduction of knowledge** is a reduction, $(\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V})$, that satisfies the following properties:*

(i) ***Completeness**: For any EPT adversary $\mathcal{A}$, given $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz})$, $(\mathsf{s}, u_1, w_1) \leftarrow \mathcal{A}(\mathsf{pp})$ such that $(\mathsf{pp}, \mathsf{s}, u_1, w_1) \in \mathcal{R}_1$, we have that the prover's output instance is equal to the verifier's output instance u_2, and that*

$$(\mathsf{pp}, \mathsf{s}, \langle \mathcal{P}, \mathcal{V} \rangle((\mathsf{pk}, \mathsf{vk}), u_1, w_1)) \in \mathcal{R}_2.$$

(ii) ***Knowledge soundness****: For any EPT adversary $(\mathcal{A}, \mathcal{P}^*)$, there exists an EPT extractor $\mathcal{E}$ such that if the success probability of the adversary*

$$\epsilon(\mathcal{A}, \mathcal{P}^*) := \Pr\left[(\mathsf{pp}, \mathsf{s}, \langle \mathcal{P}^*, \mathcal{V} \rangle((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st})) \in \mathcal{R}_2 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ (\mathsf{s}, u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}) \end{array} \right]$$

$\geq 1/\mathsf{poly}(\lambda)$*, then we have that*

$$\Pr\left[(\mathsf{pp}, \mathsf{s}, u_1, w_1) \in \mathcal{R}_1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ (\mathsf{s}, u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}) \\ w_1 \leftarrow \mathcal{E}(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st}) \end{array} \right] \geq \epsilon(\mathcal{A}, \mathcal{P}^*) - \mathsf{negl}(\lambda).$$

(iii) ***Public Coin****: All of the verifier's messages are uniformly random strings of some prescribed length. Furthermore, the verifier's messages contain all of the random coins (randomness) used by the verifier.*[11]

Typically, we are interested in reducing several relations at once. We can interpret several relations as a single relation using the following product operator.

Definition 2 (Relation product). *For relations $\mathcal{R}_1$ and $\mathcal{R}_2$ over public parameter, structure, instance, and witness pairs we define the relation product as follows.*

$$\mathcal{R}_1 \times \mathcal{R}_2 = \left\{ (\mathsf{pp}, \mathsf{s}, (u_1, u_2), (w_1, w_2)) \;\middle|\; (\mathsf{pp}, \mathsf{s}, u_1, w_1) \in \mathcal{R}_1, (\mathsf{pp}, \mathsf{s}, u_2, w_2) \in \mathcal{R}_2 \right\}.$$

We let $\mathcal{R}^n$ denote $\mathcal{R} \times \ldots \times \mathcal{R}$ for n times.

A motivating property of reductions of knowledge is that they are composable, allowing us to build complex reductions by stitching together simpler ones. In particular, given reductions of knowledge $\Pi_1 : \mathcal{R}_1 \to \mathcal{R}_2$ and $\Pi_2 : \mathcal{R}_2 \to \mathcal{R}_3$ we have that $\Pi_2 \circ \Pi_1$ (i.e., running Π_1 first and then running Π_2 on the outputs) is a reduction of knowledge from $\mathcal{R}_1$ to $\mathcal{R}_3$. We define the formal semantics of the sequential composition operator $\circ$.

Lemma 1 (Sequential composition [38, 39]). *For reductions of knowledge $\Pi_1 = (\mathcal{G}, \mathcal{K}, \mathcal{P}_1, \mathcal{V}_1) : \mathcal{R}_1 \to \mathcal{R}_2$ and $\Pi_2 = (\mathcal{G}, \mathcal{K}, \mathcal{P}_2, \mathcal{V}_2) : \mathcal{R}_2 \to \mathcal{R}_3$, we have that $\Pi_2 \circ \Pi_1 = (\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V}) : \mathcal{R}_1 \to \mathcal{R}_3$ is a reduction of knowledge where $\mathcal{K}(\mathsf{pp}, \mathsf{s})$ computes $(\mathsf{pk}, \mathsf{vk})$ and where*

$$\mathcal{P}(\mathsf{pk}, u_1, w_1) = \mathcal{P}_2(\mathsf{pk}, \mathcal{P}_1(\mathsf{pk}, u_1, w_1))$$
$$\mathcal{V}(\mathsf{vk}, u_1) = \mathcal{V}_2(\mathsf{vk}, \mathcal{V}_1(\mathsf{vk}, u_1, w_1))$$

In this work, we are primarily interested in building folding schemes, a particular type of reduction of knowledge that reduces the task of checking instances in some relation $\mathcal{R}_2$ into a running instance in a relation $\mathcal{R}_1$.

Definition 3 (Folding scheme). *A folding scheme for $\mathcal{R}_1$ and $\mathcal{R}_2$ is a reduction of knowledge of type $\mathcal{R}_1 \times \mathcal{R}_2 \to \mathcal{R}_1$.*

[11] If a reduction of knowledge is public-coin, then it trivially satisfies the property of **public reducibility** described in [39] as the execution of the verifier $\mathcal{V}$ can be emulated using the randomness from the transcript.

2.3 Rings and Modules

Definition 4 (Modules). *Modules are a generalization of vector spaces for which the field of scalars is replaced by a ring R. Suppose R is a commutative ring with identity 1 and G is an abelian (commutative) group. The group G is an R-module if there is an operation $\cdot : R \times G \to G$ such that for all $r, s \in R$ and $x, y \in G$, $r \cdot (x + y) = r \cdot x + r \cdot y$, $(r + s) \cdot x = r \cdot x + s \cdot x$, $(rs) \cdot x = r \cdot (s \cdot x)$, $1 \cdot x = x$. Suppose G_1 and G_2 are R-modules. Similarly, an R-module homomorphism is a map $\mathcal{L} : G_1 \to G_2$ that is a generalization of a linear map of vector spaces. $\mathcal{L}$ is an R-module homomorphism if for all $x, y \in G_1$ and $r \in R$, $f(x + y) = f(x) + f(y), f(r \cdot x) = r \cdot f(x)$.*

Definition 5 (Cyclotomic ring). *Let $\eta \in \mathbb{N}$ be a prime power, and Φ_η is the η-th cyclotomic polynomial with degree d.*[12] *We define the cyclotomic ring that we operate over as the quotient ring $R_q := \mathbb{F}[X]/(\Phi_\eta)$, whose elements can be viewed as polynomials over $\mathbb{F}$ with degree less than d.*

Definition 6 (Coefficient maps). *We denote the **coefficient vector** of an element $a \in R_q$ as $a' = \mathsf{cf}(a) \in \mathbb{F}^d$ and $a = \mathsf{cf}^{-1}(a') \in R_q$ to be the inverse map which takes a vector of d coefficients and outputs the corresponding ring element. Given a vector $z \in R_q^m$, we denote $\mathsf{cf}(z)$ to be the matrix $Z = \left[\, \mathsf{cf}(z_1) \;\middle|\; \mathsf{cf}(z_2) \;\middle|\; \cdots \;\middle|\; \mathsf{cf}(z_m) \,\right] \in \mathbb{F}^{d \times m}$ and $\mathsf{cf}^{-1}(Z)$ to be the inverse map which takes a matrix in $\mathbb{F}^{d \times m}$ and outputs the corresponding vector of ring elements.*

Norm. For an element $a \in R_q$, we define $\|a\|_\infty$ to be the ℓ_∞-norm of the vector $\mathsf{cf}(a)$. Similarly, for a vector $z \in R_q^m$, we define $\|z\|_\infty$ to be the ℓ_∞-norm of the matrix $\mathsf{cf}(Z)$.

Definition 7 (Rotation matrices). *We define a **shift matrix** $\mathsf{F} \in \mathbb{F}^{d \times d}$ and **rotation matrix**, $\mathsf{rot}(a) \in \mathbb{F}^{d \times d}$ for an element $a \in R_q$ [50, pg. 11]:*

$$\mathsf{F} := \left[\begin{array}{c|c} \mathbf{0} & -c_0 \\ \hline & -c_1 \\ \mathrm{I}_{d-1} & \vdots \\ & -c_{d-1} \end{array}\right], \quad \mathsf{rot}(a) := \left[\, \mathsf{cf}(a) \;\middle|\; \mathsf{F} \cdot \mathsf{cf}(a) \;\middle|\; \cdots \;\middle|\; \mathsf{F}^{d-1} \cdot \mathsf{cf}(a) \,\right]$$

where the coefficients of the cyclotomic polynomial are $\Phi_\eta = x^d + c_{d-1}x^{d-1} + c_{d-2}x^{d-2} + \cdots + c_0$. For all $a \in R_q$, we have $\mathsf{cf}(X \cdot a) = \mathsf{F} \cdot \mathsf{cf}(a)$. Hence, for all $a, b \in R_q$, $\mathsf{rot}(a) \cdot \mathsf{cf}(b) = \mathsf{cf}(ab)$.

Remark 1 (Visualizing rotation matrices). The shift matrix F effectively rotates a vector and shifts by the coefficents of the cyclotomic polynomial scaled by the

[12] If d is a power of two, then $\Phi_\eta = X^d + 1$ for $\eta = 2d$. If $\eta = 3^4$, then $\Phi_\eta = X^{54} + X^{27} + 1$.

last entry of the vector. Below, we have an example for a rotation matrix for an element $a \in R_q$ where $\Phi_\eta = X^d + 1$.

$$\mathsf{F} \cdot \mathsf{cf}(a) = \begin{bmatrix} 0 \\ a_0 \\ \vdots \\ a_{d-2} \end{bmatrix} + a_{d-1} \begin{bmatrix} -c_0 \\ -c_1 \\ \vdots \\ -c_{d-1} \end{bmatrix}, \quad \mathsf{rot}(a) := \begin{bmatrix} a_0 & -a_{d-1} & \dots & -a_1 \\ a_1 & a_0 & \dots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{d-1} & a_{d-2} & \dots & a_0 \end{bmatrix}$$

Definition 8 (Module short integer solution [45, 47, 55]). *Define the ring $R := \mathbb{Z}[X]/(\Phi_\eta)$. The $\mathsf{MSIS}^{\infty,\kappa,q}_{m,B}$ problem is defined as follows: Given a matrix $M \xleftarrow{\$} R_q^{\kappa \times m}$ sampled uniformly at random, find a non-zero vector $z \in R$ such that $Mz = 0 \mod q$ and $\|z\|_\infty \leq B$.*

Definition 9 (Ajtai commitment scheme [5]). *Assume the $\mathsf{MSIS}^{\infty,\kappa,q}_{m,2B}$ problem is hard. Let message length $m \in \mathbb{N}$. The Ajtai commitment scheme $\mathsf{com} := (\mathsf{Setup}, \mathsf{Commit})$ consists of the following algorithms:*

- $\mathsf{Setup}(\kappa, m) \to \mathsf{pp}$*: Sample a random matrix $M \xleftarrow{\$} R_q^{\kappa \times m}$. Output $\mathsf{pp} \leftarrow M$.*
- $\mathsf{Commit}(\mathsf{pp}, z) \to c$*: Given public parameters pp and vector $z \in R_q^m$ such that $\|z\|_\infty \leq B$, output Mz.*

Theorem 1 (Low norm invertibility [48, Theorem 1.1]). *Let $z \in \mathbb{N}$ such that $z \mid \eta$, $q \equiv 1 \pmod z$, and $\mathrm{ord}_\eta(q) = \eta/z$. Define $\mathsf{b_{inv}} := 1/\sqrt{\tau(z)} \cdot q^{1/\phi(z)}$ where $\tau(z) := z$ if z is odd, otherwise $\tau(z) = z/2$. For an arbitrary $a \in R_q$, if $0 < \|\mathsf{cf}(a)\|_\infty < \mathsf{b_{inv}}$, then a is invertible in R_q.*

Definition 10 (Strong sampling sets [23]). *Define $\mathcal{C}_R \subseteq R_q$ to be any set of ring elements such that for any distinct elements $a, b \in \mathcal{C}_R$, $\|\mathsf{cf}(a - b)\|_\infty < \mathsf{b_{inv}}$ (Theorem 1).*

3 Neo's folding-friendly lattice-based commitments

This section describes Neo's lattice-based commitment scheme that can be used to construct folding schemes. We begin with requirements on this commitment scheme and highlight challenges that must be addressed.

3.1 Requirements

To construct our lattice-based folding scheme for CCS, we need a lattice-based commitment scheme for a vector of field elements $z \in \mathbb{F}^m$ (e.g., a witness vector for CCS). We require this commitment scheme to satisfy several properties.

(1) Linear homomorphism for folding multilinear evaluation claims. Given a collection of commitments $(c_i)_{i \in [k]}$ and the corresponding claimed evaluations $(r_i, y_i)_{i \in [k]}$, we want to be able to *reduce* the task of proving that the commitments indeed open to corresponding vectors, $(z_i)_{i \in [k]}$, such that their multilinear

extension polynomials evaluate to the claimed evaluations to the task of proving that a single commitment C opens to a vector, z, whose multilinear extension polynomial evaluates to some value y.[13]

Challenges. For discrete-log based commitments (e.g., Pedersen [54], HyperKZG [64]), this simply amounts to taking a random linear combination of the commitments to get a single commitment c, and proving that a vector z underneath c evaluates to some y at r, where y is the weighted sum of $\{y_i\}_{i\in[k]}$ (the weights used for combining evaluations is the same as the weights used for combining commitments). However, for lattice-based commitments such as Ajtai (Definition 9), the binding of the commitment scheme depends on the norm of the vector z. Thus, there are two main challenges when taking linear combinations. First, the norm of the vector z may be larger than allowed for the binding of the commitment scheme, and the extraction procedure of the folding scheme (for knowledge soundness) may not preserve the original norm. Second, how can we even commit to arbitrary vectors $z \in \mathbb{F}^m$, when the norm of z may be *large*?

(2) Pay-per-bit costs. We require that the cost to commit to a vector of values scales with the bit width of the values in the vector. This is not necessary for constructing a folding scheme for CCS, but it is a highly desirable property in practice as often times the witness values are in a small subset of the field.

Challenges. With discrete-log based commitments, the cost of committing to a vector z scales with the bit-width of its elements. When moving to Ajtai commitments, this is non-trivial to achieve. For example, prior works such as LatticeFold [14] do not provide such a cost profile. We now provide details.

LatticeFold does not provide pay-per-bit commitment costs because of the way it embeds a vector $z \in \mathbb{F}^m$ into the cyclotomic ring $z' \in R_q^m$ prior to committing with Ajtai's commitment scheme. In particular, for certain choices of cyclotomic rings, the ring $R_q \simeq (\mathbb{F}_{q^\tau})^t$ (for which $t \cdot \tau = d$ and d is the degree of Φ_η) is isomorphic to multiple copies of an extension field $\mathbb{F}_{q^\tau}$. This is often referred to as the NTT representation of an R_q element. When committing to a vector $z \in \mathbb{F}^m$ (e.g., the CCS witness), LatticeFold first embeds the vector z into the extension field $\mathbb{F}_{q^\tau}$. Together with $t-1$ other vectors, LatticeFold applies the so-called NTT *transformation* to obtain a ring vector $z' \in R_q^m$ whose NTT representation equals to those t vectors. The important point here is that the NTT transformation does *not* preserve the norm of the input vectors. If $z \in \mathbb{F}^m$ has a low norm, then that does not mean $z' \in R_q^m$ has a low norm. Hence, the final step to commit using Ajtai's commitment scheme is to decompose $z' \in R_q^m$ into a longer vector $z'' \in R_q^{m\cdot\ell}$ whose ring elements have low enough norm or into multiple vectors $(z'_i)_{i\in[k]}$, which further increases the cost to commit. Thus, regardless of the bit-width of the original vectors, the cost to commit is the same (i.e., LatticeFold does not achieve pay-per-bit commitment costs).

[13] We treat a vector $z \in \mathbb{F}^n$, where n is a power of 2, as evaluations of a multilinear polynomial over the Boolean hypercube $\{0,1\}^{\log n}$. Since a $\log n$-variate multilinear polynomial is uniquely determined by its evaluations over $\{0,1\}^{\log n}$, committing to z commits to the unique multilinear extension polynomial $\widetilde{v}$.

(3) Support for small fields and for folding linear transforms. To fold CCS instance-witness pairs, we need to fold evaluations of not only the multilinear extensions of CCS witness vector z, but also evaluations of $\widetilde{Mz}$ for some CCS constraint matrix M.

Challenges. In prior work [14], this is achieved by taking random linear combinations of the witness commitments and evaluations, where the witnesses are embedded in the NTT representation of ring elements. However, as noted [14, Section 3.3], when supporting "small" fields (i.e., fields whose order is not 2^λ), this strategy leads to a τ multiplicative factor blow-up in the cost of the protocols, where τ is the degree of the extension field in the NTT representation. For example, in LatticeFold's example parameterization [14, Sec. 5] for 64-bit fields, one would need to set $\tau = 4$. This is due to security requirements and a mapping between coefficient and NTT representation that loses a τ factor of packing. Furthermore, this transformation between coefficient and NTT representation incurs another factor of 2, because evaluations over both NTT representation and coefficient representation need to be accounted for. In LatticeFold's folding protocol, the prover and the verifier must execute the sum-check protocol over cyclotomic polynomial rings (instead of fields). When using the NTT representation $(\mathbb{F}_{q^\tau})^t$ for ring elements, this sum-check can be performed as operations over $\mathbb{F}_{q^\tau}$. Despite the field being 64-bits in size, they must work over a larger extension field $\left|\mathbb{F}_{q^4}\right| \approx 2^{256}$. As we describe later, Neo's folding scheme does not have to operate over such a large extension field, since we avoid this security issue entirely. In our parameterization Section 6, the sum-check protocol can draw challenges over $\mathbb{F}_{q^2}$ instead when q is about 64 bits.

3.2 Neo's solution, part-1: A matrix commitment scheme

Research question. Can we construct a simple and efficient lattice-based commitment scheme that can satisfy all of our requirements and that works natively over small prime fields (e.g., M61)?

Our starting point is Ajtai's commitment scheme (Definition 9). This scheme commits to a vector of low-norm ring elements $z \in R_q^m$ (i.e., the coefficients of ring elements have low-norm) by multiplying the vector with a matrix M sampled in the setup algorithm. Looking ahead, we can view the vector z of ring elements as a matrix of its coefficients.

The first question that we need to resolve is the following: given a vector $z \in \mathbb{F}^m$, how can we embed z into a low-norm vector $z' \in R_q^m$ such that z' can be committed directly with Ajtai's commitment scheme while preserving the required homomorphism properties?

We map each element $z_i \in \mathbb{F}$ to a single ring element $z'_i \in R_q$ by embedding the b-bit words of $z_i := \sum_{j=1}^{d} b^{j-1} z_{i,j}$ into the coefficients $z'_i := \sum_{j=1}^{d} z_{i,j} \cdot X^{j-1}$. This embedding guarantees that z'_i has low-norm. More formally, we define several decomposition mappings that we employ: (1) Decomp_b maps $z \in \mathbb{F}^m$ into a matrix $Z \in \mathbb{F}^{d \times m}$ with low-norm; and (2) Given a matrix $Z \in \mathbb{F}^{d \times m}$, split_b splits Z into multiple matrices with low-norm.

Definition 11 (Decomposition and splitting). *Let $b, m \in \mathbb{N}$. We define $\mathsf{Decomp}_b : \mathbb{F}^m \to \mathbb{F}^{* \times m}$ as the map which takes a vector z and performs the b-ary decomposition into a matrix $Z := \mathsf{Decomp}_b(z) \in \mathbb{F}^{* \times m}$. For example, if $z \in \mathbb{F}^m$ such that $\|z\|_\infty < b^d$, then we have*

$$\mathsf{Decomp}_b(z) := \left[\begin{array}{c} Z^{(1)} \\ \hline Z^{(2)} \\ \hline \cdots \\ \hline Z^{(d)} \end{array}\right] \text{ such that } z = \sum_{i=1}^{d} b^{i-1} \cdot Z^{(i)} \text{ and } \left\|Z^{(i)}\right\|_\infty < b$$

where $Z^{(i)}$ is the i-th row of Z.

We define $\mathsf{split}_b : \mathbb{F}^{d \times m} \to (\mathbb{F}^{d \times m})^*$ to be the b-ary decomposition map, which performs the b-ary decomposition of a matrix $Z \in \mathbb{F}^{d \times m}$ into matrices $Z_1, Z_2, \ldots, Z_*$. For example, if $Z \in \mathbb{F}^{d \times m}$ such that and $\|Z\|_\infty < b^k$, then we have

$$\mathsf{split}_b(Z) := (Z_1, Z_2, \ldots Z_k) \quad \text{such that} \quad Z = \sum_{i=1}^{k} b^{i-1} \cdot Z_i \text{ and } \|Z_i\|_\infty < b$$

We then apply Ajtai's commitment scheme to commit to low-norm matrices $Z \in \mathbb{F}^{d \times m}$ (Theorem 2). In particular, each column of a low-norm matrix gets mapped as the coefficients of a single ring element. Each ring element corresponds 1-to-1 with the decomposition of a single witness element. We refer to the resulting vector as $z' \leftarrow \mathsf{cf}^{-1}(Z)$. To compute the commitment to Z, we compute the Ajtai commitment Mz'. Since Z is low-norm, Ajtai's commitment scheme provides the desired binding property.

Bit-width scaling multiplication. We now discuss how the cost of Ajtai's commitment scheme, with our particular embedding of witnesses into ring elements, scales linearly with the bit-width of the witness z. Recall that Ajtai's commitment scheme is merely a ring matrix multiplication Mz'.

We show how the cost of a ring multiplication for which $a, b \in R_q$ where b only has binary coefficients scales with the number of non-zero coefficients of b. As a result, this shows that the cost of the matrix multiplication Mz' scales with the bit-width of z'. By the definition of rotation matrices (Definition 7),

$$\mathsf{cf}(a \cdot b) = \mathsf{rot}(a) \cdot \mathsf{cf}(b) = \left[\begin{array}{c|c|c|c} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \ldots & \boldsymbol{a}_d \end{array}\right] \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_d \end{bmatrix} = \sum_{i=1}^{d} b_i \cdot \boldsymbol{a}_i$$

where $\boldsymbol{a}_i$ denotes the i-th column of $\mathsf{rot}(a)$. Since coefficients $(b_i)_i$ are bits, this amounts to just adding the columns which correspond to the $b_i = 1$. For a choice of cyclotomic ring (Definition 5) (with a cyclotomic polynomial containing a constant number of coefficients), computing these columns $\boldsymbol{a}_i$ requires just a rotation of the prior column and adding a constant number of field elements (Remark 1).

Homomorphism. When Ajtai's commitment scheme is viewed as a commitment scheme for vectors $z \in R_q^m$, there is a natural understanding of taking linear combinations, namely the Ajtai commitment scheme is an R_q-module homomorphism $\mathcal{L} : R_q^m \to R_q^\kappa$ (i.e. $c_1 + r \cdot c_2 = \mathcal{L}(z_1 + r \cdot z_2)$ for $r \in R_q$). But, what is the analogue for matrices? We now provide details.

Definition 12 (Ring of rotation matrices). *Let R_q be a cyclotomic ring. We define $\mathcal{S} := \{ \mathsf{rot}(a) \mid a \in R_q \} \subseteq \mathbb{F}^{d \times d}$ to be the **ring of all rotation matrices** for elements in R_q. For our purposes, $\mathcal{S}$ can be thought of as a commutative subring of matrices $\mathbb{F}^{d \times d}$, which contains all scalar matrices (exactly the rotation matrices for the constant ring elements).*

It turns out that the ring R_q is isomorphic to the ring of rotation matrices $\mathcal{S} \subseteq \mathbb{F}^{d \times d}$ (Theorem 7). Hence, instead of an R_q-module homomorphism, the commitment scheme can be viewed as a $\mathcal{S}$-module homomorphism $\mathcal{L} : \mathbb{F}^{d \times m} \to \mathbb{F}^{d \times \kappa}$, where the matrices $Z \in \mathbb{F}^{d \times d}$ and commitments in $\mathbb{F}^{d \times \kappa}$ (i.e., the coefficient matrix of the ring elements) are left multiplied by elements $r \in \mathcal{S} \subseteq \mathbb{F}^{d \times d}$. We formalize this view with the following definition.

Definition 13 (Matrix commitment scheme). *Let $\mathbb{F}$ be a field. A matrix commitment scheme $\mathsf{com} := (\mathsf{Setup}, \mathsf{Commit})$ consists of two algorithms.*

- $\mathsf{Setup}(1^\lambda, d, m) \to \mathsf{pp}$*: Takes as input a security parameter 1^λ and matrix dimensions $d, m \in \mathbb{N}$, outputs public parameters pp.*
- $\mathsf{Commit}(\mathsf{pp}, Z) \to c$*: Takes as input public parameters pp and a matrix $Z \in \mathbb{F}^{d \times m}$, outputs a commitment $c \in \mathbb{C}$.*

These algorithms can satisfy the following properties.

***$\mathcal{S}$-homomorphic**: Let space $\mathcal{S} \subset \mathbb{F}^{d \times d}$ be a commutative sub-ring of $d \times d$ matrices. For all $m \in \mathbb{N}$ and public parameters pp output by $\mathsf{Setup}(1^\lambda, d, m)$, we have that for all matrices $Z_1, Z_2 \in \mathbb{F}^{d \times m}$ and $\rho_1, \rho_2 \in \mathcal{S}$ that*

$$\rho_1 \cdot \mathsf{Commit}(\mathsf{pp}, Z_1) + \rho_2 \cdot \mathsf{Commit}(\mathsf{pp}, Z_2) = \mathsf{Commit}(\mathsf{pp}, \rho_1 \cdot Z_1 + \rho_2 \cdot Z_2)$$

More formally, the commitment algorithm $\mathsf{Commit}(\mathsf{pp}, \cdot) : \mathbb{F}^{d \times m} \to \mathbb{C}$ is an $\mathcal{S}$-module homomorphism.

*(d, m, B)-**Binding**: For all expected polynomial time adversaries $\mathcal{A}$, we have*

$$\Pr\left[\begin{array}{l|l} \mathsf{Commit}(\mathsf{pp}, Z_1) = \mathsf{Commit}(\mathsf{pp}, Z_2) & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d, m) \\ \wedge\ Z_1 \neq Z_2 & Z_1, Z_2 \in \mathbb{F}^{d \times m} \leftarrow \mathcal{A}(\mathsf{pp}) \\ \wedge\ \|Z_1\|_\infty,\ \|Z_2\|_\infty < B & \end{array}\right]$$

$\leq \epsilon_{\mathsf{bind}}(\mathsf{com}, d, m, B) \leq \mathsf{negl}(\lambda)$*. We refer to the pair of messages (Z_1, Z_2) which satisfies the conditions in the probability as an (d, m, B)-**binding collision**.*

Theorem 2. *Let R_q be a cyclotomic ring (Definition 5) and $\mathcal{S}$ be the ring of all rotation matrices (Definition 7). If the $\mathsf{MSIS}^{\infty,\kappa,q}_{m,2B}$ problem is hard then Ajtai's commitment scheme (Definition 9) is a matrix commitment scheme (Definition 13) that is $\mathcal{S}$-homomorphic and (d, m, B)-binding.*

Proof. For brevity, we defer the proof to Appendix B.1. □

3.3 Neo's solution, part-2: linear homomorphism for folding multilinear evaluation claims

Based on the description thus far, we have a commitment scheme that commits to CCS witnesses $z := x \,||\, w \in \mathbb{F}^m$ as low-norm matrices $Z := X \,||\, W \in \mathbb{F}^{d \times m}$. As discussed earlier, we want to be able to fold evaluations of the multilinear extensions $\widetilde{Mz}$ for some matrix M (e.g., the CCS constraint matrices).

Our starting point is the following lemma, which observes that linear combinations of commitments respect right multiplication of their openings. Looking ahead, we carefully choose the matrix V to be $M^\intercal \widehat{r}$ which will allow us to fold the desired multilinear evaluations at a point r. In particular, consider a vector $Z \leftarrow \mathsf{Decomp}_b(z) \in \mathbb{F}^{d \times m}$ (Definition 11). For a matrix $M \in \mathbb{F}^{n \times m}$ and $r \in \mathbb{K}^{\log n}$ ($\widehat{r} \in \mathbb{K}^n$), observe that

$$\begin{aligned} ZM^\intercal \widehat{r} &= [Z^{(1)} M^\intercal \widehat{r},\, Z^{(2)} M^\intercal \widehat{r},\, \ldots,\, Z^{(d)} M^\intercal \widehat{r}] \\ &= [\widetilde{MZ^{(1)}}(r),\ \widetilde{MZ^{(2)}}(r),\ \ldots,\ \widetilde{MZ^{(d)}}(r)] \end{aligned}$$

where $Z^{(i)}$ is the i-th row of Z. To obtain the evaluation $\widetilde{Mz}(r)$, it suffices to compute the sum $\sum_{i=1}^{d} b^{i-1} Z^{(i)} M^\intercal \widehat{r} = (\sum_{i=1}^{d} b^{i-1} Z^{(i)}) M^\intercal \widehat{r} = zM^\intercal \widehat{r} = \widetilde{Mz}(r)$. Another way to view $ZM^\intercal \widehat{r}$ is as a partial evaluation $y \in \mathbb{F}^d$ such that $\widetilde{y}(X_1, \ldots, X_d) = \widetilde{ZM^\intercal}(X_1, \ldots, X_d, r)$.

Lemma 2 (Linear combination lemma). *Let $\mathcal{S} \subseteq \mathbb{F}^{d \times d}$ be any commutative sub-ring of $d \times d$ matrices and $m, n, k \in \mathbb{N}$. Let $\mathcal{L} : \mathbb{F}^{d \times m} \to \mathbb{C}$ be a $\mathcal{S}$-module homomorphism. Consider arbitrary $\rho_1, \ldots, \rho_k \in \mathcal{S}$, $Z_1, \ldots, Z_k \in \mathbb{F}^{d \times m}$, and $V \in \mathbb{K}^{m \times n}$. For all $i \in [k]$, define*

$$c_i := \mathcal{L}(Z_i) \in \mathbb{C} \qquad \text{and} \qquad v_i := Z_i V \in \mathbb{K}^{d \times n}$$

Define

$$c := \sum_{i \in [k]} \rho_i c_i \in \mathbb{C} \qquad Z := \sum_{i \in [k]} \rho_i Z_i \in \mathbb{F}^{d \times m} \qquad v := \sum_{i \in [k]} \rho_i v_i \in \mathbb{K}^{d \times n}$$

Then, $c = \mathcal{L}(Z)$ and $v = ZV$.

Proof. First, we will prove that $c = \mathcal{L}(Z)$. Since $\mathcal{L}$ is a $\mathcal{S}$-module homomorphism, the following holds

$$c = \sum_{i \in [k]} \rho_i c_i = \sum_{i \in [k]} \rho_i \mathcal{L}(Z_i) = \mathcal{L}\left(\textstyle\sum_{i \in [k]} \rho_i Z_i\right) = \mathcal{L}(Z).$$

Now, we will prove that $v = ZV$, as follows

$$v = \sum_{i \in [k]} \rho_i v_i = \sum_{i \in [k]} \rho_i Z_i V = \left(\textstyle\sum_{i \in [k]} \rho_i Z_i\right) V = ZV$$

This concludes our proof. □

The following corollary can be immediately obtained from Lemma 2 by defining $V := M_j^\intercal \widehat{r}$ for each $j \in [t]$ and choosing $\mathcal{L}_x$ as the trivial $\mathcal{S}$-homomorphism, which projects the first m_{in} columns of Z.

Corollary 1. *Let $\mathcal{S} \subseteq \mathbb{F}^{d\times d}$ be any commutative sub-ring of $d \times d$ matrices and $m, n, k \in \mathbb{N}$. Let $\mathcal{L} : \mathbb{F}^{d\times m} \to \mathbb{C}$ and $\mathcal{L}_x : \mathbb{F}^{d\times m} \to \mathbb{F}^{d\times m_{\mathsf{in}}}$ be $\mathcal{S}$-module homomorphisms. Consider arbitrary $\rho_1, \ldots, \rho_k \in \mathcal{S}$, $Z_1, \ldots, Z_k \in \mathbb{F}^{d\times m}$, $M_1, \ldots, M_t \in \mathbb{F}^{n\times m}$, $r \in \mathbb{K}^{\log n}$. Define*

$$\forall\, i \in [k],\ \ c_i := \mathcal{L}(Z_i) \in \mathbb{C},\ \ X_i := \mathcal{L}_x(Z_i) \in \mathbb{F}^{d\times m_{\mathsf{in}}}$$

$$\forall\, i \in [k],\, j \in [t],\ \ y_{(i,j)} := Z_i M_j^\intercal \widehat{r} \in \mathbb{K}^d$$

Define

$$c := \textstyle\sum_{i\in[k]} \rho_i c_i \in \mathbb{C} \quad X := \sum_{i\in[k]} \rho_i X_i \in \mathbb{C}$$
$$Z := \textstyle\sum_{i\in[k]} \rho_i Z_i \in \mathbb{F}^{d\times m} \quad y_j := \sum_{i\in[k]} \rho_i \cdot y_{(i,j)} \in \mathbb{K}^d$$

Then, $c = \mathcal{L}(Z)$, $X = \mathcal{L}_x(Z)$, and for all $j \in [t]$, $y_j = Z M_j^\intercal \widehat{r}$.

3.4 Challenge sets

As hinted earlier, when taking random linear combinations of lattice-based commitments, there are two concerns: (1) the random combination of the openings might be larger than the norm-bound required for binding; and (2) for extraction (i.e., knowledge soundness of the folding scheme), we need the differences between challenges to be invertible. We formalize these requirements by lifting strong sampling sets over cyclotomic rings R_q (Definition 10) to our setting.

Definition 14 (Strong sampling set). *Let $\mathcal{S} \subseteq \mathbb{F}^{d\times d}$ be any commutative sub-ring of $d \times d$ matrices. A **strong sampling set** for $\mathcal{S}$ is a subset $\mathcal{C} \subseteq \mathcal{S}$ such that for any two distinct elements $\rho, \rho' \in \mathcal{C}$, $(\rho - \rho')$ is invertible in the ring $\mathcal{S}$. Furthermore, we define the*

$$\textbf{\textit{expansion factor}} \text{ of } \mathcal{C} \ := \ \max_{\substack{v\in\mathbb{F}^d \\ \rho\in\mathcal{C}}} \frac{\|\rho v\|_\infty}{\|v\|_\infty}$$

Lattice instantiation. Consider a strong sampling set $\mathcal{C}_R$ over the cyclotomic ring R_q (Definition 10). By definition, for any distinct elements $a, b \in \mathcal{C}_R$, $\|\mathsf{cf}(a-b)\|_\infty < \mathsf{b_{inv}}$. Thus, by Theorem 1, $a - b$ is invertible in R_q. Since R_q and $\mathcal{S}$ (Definition 12) are isomorphic, we can define a strong sampling set $\mathcal{C} := \{\, \mathsf{rot}(c) \mid c \in \mathcal{C}_R \,\}$ to be the correspond set of rotation matrices. Consider arbitrary distinct elements $\mathsf{rot}(a)\,, \mathsf{rot}(b) \in \mathcal{C}$. By Theorem 7, we have $(\mathsf{rot}(a) - \mathsf{rot}(b))^{-1} = \mathsf{rot}\big((a-b)^{-1}\big)$.

Theorem 3 (Expansion factors). *Let $\mathcal{C}_R$ be a strong sampling set over the cyclotomic ring R_q (Definition 10), and $\mathcal{C} := \{\, \mathsf{rot}(c) \mid c \in \mathcal{C}_R \,\}$ be the corresponding set of rotation matrices. We denote the Euler totient function as ϕ. We*

must have that the expansion factor (Definition 14) of $\mathcal{C}$ is

$$\max_{\substack{v \in \mathbb{F}^d \\ \rho \in \mathcal{C}}} \frac{\|\rho v\|_\infty}{\|v\|_\infty} \leq 2 \cdot \phi(\eta) \cdot \max_{\rho' \in \mathcal{C}_R} \|\rho'\|_\infty$$

Proof. For brevity, we defer the proof to Appendix B.3. □

Looking ahead, our security analysis will require a different notion of binding called relaxed binding. Here, we lift the notion from prior works to the matrix setting, where $\mathcal{C}$ will be a challenge set with which we take linear combinations of commitments.

Definition 15 ($(d, m, B, \mathcal{C})$-relaxed binding [9, 10, 14]). *Let* $\mathsf{com} := (\mathsf{Setup}, \mathsf{Commit})$ *be an arbitrary matrix commitment scheme (Definition 13) that is $\mathcal{S}$-homomorphic. Let $\mathcal{C}$ be any subset of $\mathcal{S}$. The commitment scheme* com *satisfies $(d, m, B, \mathcal{C})$-**relaxed binding** if for all expected polynomial time adversaries $\mathcal{A}$, we have*

$$\Pr\left[\begin{array}{l} \Delta_1 \cdot c = \mathsf{Commit}(\mathsf{pp}, Z_1) \\ \wedge\ \Delta_2 \cdot c = \mathsf{Commit}(\mathsf{pp}, Z_2) \\ \wedge\ \|Z_1\|_\infty,\ \|Z_2\|_\infty < B, \\ \wedge\ \Delta_1 Z_2 \neq \Delta_2 Z_1 \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, m) \\ \left(\begin{array}{c} c \in \mathbb{C}, \\ \Delta_1, \Delta_2 \in (\mathcal{C} - \mathcal{C}), \\ Z_1, Z_2 \in \mathbb{F}^{d \times m} \end{array}\right) \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right]$$

$\leq \epsilon_{\mathsf{relax}}(\mathsf{com}, d, m, B, \mathcal{C}) \leq \mathsf{negl}(\lambda)$. *We refer to a tuple of elements $(c, \Delta_1, \Delta_2, Z_1, Z_2)$ which satisfies the conditions in the probability as an $(d, m, B, \mathcal{C})$-**relaxed binding collision**.*

It turns out that regular binding implies relaxed binding. Here, we lift the corresponding lemmas from prior work to our setting.

Lemma 3 (Binding implies relaxed binding [9,10,14]). *Let* $\mathsf{com} := (\mathsf{Setup}, \mathsf{Commit})$ *be an arbitrary matrix commitment scheme (Definition 13) that is $\mathcal{S}$-homomorphic and $\mathcal{C} \subseteq \mathcal{S}$ be a strong-sampling set with expansion factor T (Definition 14). If* com *is $(d, m, 2TB)$-binding (Definition 13), then* com *is $(d, m, B, \mathcal{C})$-relaxed binding (Definition 15).*

Proof. For brevity, we defer the proof to Appendix B.2. □

4 Neo's folding scheme for CCS

This section describes Neo's folding scheme for CCS. At a high level, we leverage the commitment scheme we provided in Section 3 in conjunction with HyperNova's folding scheme for CCS [42]. To make this sketch work, we first lift the CCS relation [59] to the matrix setting, and provide three reductions. These reductions can also be viewed as adapting the ideas in LatticeFold [14] to the prime field setting in a way that natively supports "small" primes. The security analysis of some of our reductions will require new security definitions and a new composition theorem for reductions, which we provide in Section 5. By composing the three reductions, we obtain a lattice-based folding scheme for CCS.

4.1 Relations

The CCS structure contains the CCS constraint matrices $\{M_j\}_{j\in[t]}$ and a constraint polynomial f.

Definition 16 (Structure). *Let $m, n, u, t \in \mathbb{N}$. We define a **structure** as a collection of elements*

$$\mathsf{s} := \left\{ \left\{ M_j \in \mathbb{F}^{n\times m} \right\}_{j\in[t]},\ f \in \mathbb{F}^{<u}[X_1, \ldots, X_t] \right\},$$

which consists of matrices and a degree-u polynomial.

Below, we provide an analogue of the CCS relation updated to match our low norm requirement (which is enforced by decomposition) and a matrix commitment scheme (more generally, any $\mathcal{S}$-module homomorphism).

Definition 17 (Matrix constraint system relation). *Let $\mathcal{L} : \mathbb{F}^{d\times m} \to \mathbb{C}$ be a $\mathcal{S}$-module homomorphism. Let s be a structured as defined in Definition 17. We define the **Matrix constraint system relation** as follows:*

$$\mathsf{MCS}(b, \mathcal{L}) := \left\{ \begin{array}{l|l} \left(\mathsf{s};\ (c \in \mathbb{C},\ x \in \mathbb{F}^{m_{\mathsf{in}}});\ w \in \mathbb{F}^{m-m_{\mathsf{in}}}\right) & \begin{array}{l} \text{For } z := x \,||\, w \text{ and} \\ Z := \mathsf{Decomp}_b(z), \\ c = \mathcal{L}(Z) \\ f\left(\widetilde{M_1 z}, \ldots, \widetilde{M_t z}\right) \in \mathsf{ZS}_n \end{array} \end{array} \right\}$$

HyperNova's folding scheme [42] can be viewed as providing a reduction from CCS to linearized CCS, a variant of CCS that they define. In a nutshell, linearized CCS amounts to the multilinear evaluations of $(M_1 z, \ldots, M_t z)$. Here, we define the analogue of linearized CCS in our setting, where we get the partial evaluation instead. Ultimately, we fold these partial evaluations using random linear combinations as hinted in our linear combination lemma (Corollary 1).

Definition 18 (Matrix evaluation relation). *Let $\mathcal{L} : \mathbb{F}^{d\times m} \to \mathbb{C}$ be a $\mathcal{S}$-module homomorphism. Let $\mathcal{L}_x : \mathbb{F}^{d\times m} \to \mathbb{F}^{d\times m_{\mathsf{in}}}$ be the trivial $\mathcal{S}$-module homomorphism which projects the first m_{in} columns. Let s be a structure as defined in Definition 17.*

$$\mathsf{ME}(b, \mathcal{L}) := \left\{ \left(\mathsf{s};\ \begin{pmatrix} c \in \mathbb{C}, \\ X \in \mathbb{F}^{d\times m_{\mathsf{in}}}, \\ r \in \mathbb{K}^{\log n}, \\ \{y_j \in \mathbb{K}^d\}_{j\in[t]} \end{pmatrix};\ Z \in \mathbb{F}^{d\times m} \right) \ :\ \begin{array}{l} c = \mathcal{L}(Z) \ \wedge\ X = \mathcal{L}_x(Z) \\ \|Z\|_\infty < b \\ \forall\, j \in [t],\ y_j = Z M_j^\intercal \widehat{r} \end{array} \right\}$$

The condition $y_j = Z M_j^\intercal \widehat{r}$ is equivalently expressed as $\widetilde{y}_j = \widetilde{Z M_j^\intercal}\left(X_{[1,\log d]}, r\right)$.

4.2 Constructing a folding scheme via reductions of knowledge

In this section, we provide an overview of three reductions. By sequentially composing the three, we obtain a lattice-based folding scheme for CCS.

(1) CCS reduction Π_{CCS}**.** This reduction takes as input $\mathsf{ME}(b, \mathcal{L})^k \times \mathsf{MCS}(b, \mathcal{L})$, which contains k b-norm partial evaluation claims and one CCS claim, and reduces to $k+1$ b-norm partial evaluation claims $\mathsf{ME}(b, \mathcal{L})^{k+1}$. This reduction can be viewed as the lattice analogue of the CCS to linearized CCS reduction in HyperNova [42], which reduces CCS claims to evaluation claims by invoking the sum-check protocol [46].

(2) Random linear combination reduction Π_{RLC}**.** This reduction takes as input $k+1$ b-norm partial evaluation claims $\mathsf{ME}(b, \mathcal{L})^{k+1}$, and reduces to a single $(B = b^k)$-norm partial evaluation claim $\mathsf{ME}(B, \mathcal{L})$. This reduction simply takes linear combinations of the input commitments and claims using challenges from a strong sampling set (Definition 14).

(3) Decomposition reduction Π_{DEC}**.** This reduction takes as input a single B-norm partial evaluation claim $\mathsf{ME}(B, \mathcal{L})$ and reduces to k b-norm partial evaluation claims $\mathsf{ME}(b, \mathcal{L})^k$. Informally, the prover effectively sends k commitments which represent the decomposition of the original B partial evaluation claim. This reduction does not use any randomness from the verifier, and is required to keep the norm of the commitments lower than the binding requirement for the commitment scheme.

In terms of security, the composition $\Pi_{\mathsf{RLC}} \circ \Pi_{\mathsf{CCS}}$ is a **reduction of knowledge** from $\mathsf{ME}(b, \mathcal{L})^k \times \mathsf{MCS}(b, \mathcal{L})$ to $\mathsf{ME}(B, \mathcal{L})$, and Π_{DEC} alone is a reduction of knowledge from $\mathsf{ME}(B, \mathcal{L})$ to $\mathsf{ME}(b, \mathcal{L})^k$. All together, the composition $\Pi := \Pi_{\mathsf{DEC}} \circ \Pi_{\mathsf{RLC}} \circ \Pi_{\mathsf{CCS}}$ is a reduction of knowledge from $\mathsf{ME}(b, \mathcal{L})^k \times \mathsf{MCS}(b, \mathcal{L})$ to $\mathsf{ME}(b, \mathcal{L})^k$. This allows for the continual folding of $\mathsf{MCS}(b, \mathcal{L})$ claims. Thus, we obtain a folding scheme for CCS.

4.3 Reduction parameters

Here, we define global parameters for all the three reductions.

- Let $\mathcal{S} \subseteq \mathbb{F}^{d \times d}$ be a commutative subring of $d \times d$ matrices (Definition 12) and $m, n, b, k, B = b^k < q/2 \in \mathbb{N}$.
- Let $\mathcal{C} \subseteq \mathcal{S}$ be a strong sampling set (Definition 14) with expansion factor T such that $(k+1)T(b-1) < B$ and $1/|\mathcal{C}| = \mathsf{negl}(\lambda)$.
- Let $\mathsf{com} := (\mathsf{Setup}, \mathsf{Commit})$ be an commitment scheme (Definition 13), which is $\mathcal{S}$-homomorphic and $(d, m, 2B, \mathcal{C})$-relaxed binding (Definition 15). For $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d, m)$, define $\mathcal{L} := \mathsf{Commit}(\mathsf{pp}, \cdot) : \mathbb{F}^{d \times m} \to \mathbb{C}$, which is a $\mathcal{S}$-module homomorphism by definition.
- Let $\mathcal{L}_x : \mathbb{F}^{d \times m} \to \mathbb{F}^{d \times m_{\mathsf{in}}}$ be the trivial $\mathcal{S}$-module homomorphism that projects the first m_{in} columns.
- Let s denote a structure as defined in Definition 17.

4.4 CCS Reduction – Π_{CCS}

For notational simplicity, we describe the protocol as a reduction from $\mathsf{MCS}(b, \mathcal{L}) \times \mathsf{ME}(b, \mathcal{L})^{k-1}$ to $\mathsf{ME}(b, \mathcal{L})^k$ instead of $\mathsf{MCS}(b, \mathcal{L}) \times \mathsf{ME}(b, \mathcal{L})^k$ to $\mathsf{ME}(b, \mathcal{L})^{k+1}$. This only affects indexing.

Overview. In this reduction, we first construct a multivariate polynomial $Q(X_1, \ldots, X_{\log(dn)})$ such that we can encode the CCS constraints (encoded as the polynomial F), norm constraints (encoded as polynomials NC_i for $i \in [k]$), and evaluation claims $Z_i M_j^\intercal \widehat{r}$ (encoded as polynomials $\mathsf{Eval}_{i,j}$ for $i \in [k]$, $j \in [t]$) to a claimed sum of a polynomial $Q(X_1, \ldots, X_{\log(dn)})$ over the Boolean hypercube $\{0,1\}^{\log(dn)}$. Then, we rely on the classic sum-check protocol [46] to reduce this sum claim to a single evaluation claim for $v \stackrel{?}{=} Q((\alpha', r'))$, which can be derived by the partial evaluations of each sub-component polynomial on r'. This leaves k new partial evaluation claims over r'.[14]

Without loss of generality, assume that $m = n$ and $n, d \cdot n$ are both powers of two and that $M_1 = I_n$ is the identity matrix. By choosing $M_1 = I_n$, we simplify our notation as folding $M_1 z$ evaluations is equivalent to folding z evaluations.

CCS reduction Π_{CCS}

Parameters: Refer to Section 4.3.

Input: $\big(\mathsf{s};\ (c_1 \in \mathbb{C},\ x_1 \in \mathbb{F}^{m_{\mathsf{in}}});\ w_1 \in \mathbb{F}^{m-m_{\mathsf{in}}}\big)$,

$$\Big(\mathsf{s};\ c_i \in \mathbb{C},\ X_i \in \mathbb{F}^{d\times m_{\mathsf{in}}},\ r \in \mathbb{K}^{\log n},\ \{y_{(i,j)} \in \mathbb{K}^d\}_{j\in[t]};\ Z_i \in \mathbb{F}^{d\times m}\Big)_{i=2}^{k}$$

$$\in \mathsf{MCS}(b,\mathcal{L}) \times \mathsf{ME}(b,\mathcal{L})^{k-1}$$

Output: $\big(\mathsf{s};\ c_i \in \mathbb{C},\ X_i \in \mathbb{F}^{d\times m_{\mathsf{in}}},\ r' \in \mathbb{K}^{\log n},\ \{y'_{(i,j)} \in \mathbb{K}^d\}_{j\in[t]};\ Z_i \in \mathbb{F}^{d\times m}\big)_{i\in[k]}$

$\in \mathsf{ME}(b,\mathcal{L})^k$

Setup $\mathcal{G}(1^\lambda, \mathsf{sz}) \to \mathsf{pp}$: Output $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{sz})$, which defines $\mathcal{L} := \mathsf{Commit}(\mathsf{pp}, \cdot) : \mathbb{F}^{d\times m} \to \mathbb{C}$.

Encoder $\mathcal{K}(\mathsf{pp}, \mathsf{s}) \to (\mathsf{pk}, \mathsf{vk})$: Output $\big((\mathsf{pp}, \mathsf{s}), \perp\big)$.

Reduction $\langle \mathcal{P}, \mathcal{V}\rangle((\mathsf{pk}, \mathsf{vk}), u_1, w_1) \to (u_2; w_2)$:

1. $\mathcal{V}$: Send challenges $\alpha \xleftarrow{\$} \mathbb{K}^{\log d}$, $\beta \xleftarrow{\$} \mathbb{K}^{\log(dn)}$, $\gamma \xleftarrow{\$} \mathbb{K}$ to $\mathcal{P}$.
2. $\mathcal{V} \leftrightarrow \mathcal{P}$: Define $z_1 := x_1 \,||\, w_1$, $X_1 := \mathsf{Decomp}_b(x_1)$, $Z_1 := \mathsf{Decomp}_b(z_1)$, for all $i \in [k], j \in [t]$, $M_{(i,j)} := Z_i M_j^\intercal$, and

$$\mathsf{F}\left(X_{[1,\log n]}\right) := f\left(\widetilde{M_1 z_1}, \ldots, \widetilde{M_t z_1}\right)$$

$$\mathsf{NC}_i\left(X_{[1,\log(dn)]}\right) := \prod_{j=-b-1}^{b-1}\Big(\widetilde{Z_i}(\boldsymbol{X}) - j\Big) \qquad \forall\, i \in [k]$$

$$\mathsf{Eval}_{(i,j)}\left(X_{[1,\log(dn)]}\right) := \mathsf{eq}\big(\boldsymbol{X},\ (\alpha, r)\big) \cdot \widetilde{M}_{(i,j)}(\boldsymbol{X}) \qquad \forall\, i \in [2,\, k],\ \forall\, j \in [t]$$

[14] Note that the polynomial Q to which the sum-check protocol is applied includes $\mathsf{Eval}_{i,j}$ because we rely on the sum-check protocol to rerandomize all evaluation claims, including those arising from F, to be over the same random point r'. This is important to be able to take a random linear combination of all evaluation claims. HyperNova [42] and LatticeFold [14] also perform a similar rerandomization.

$$Q\left(X_{[1,\,\log(dn)]}\right) := \mathsf{eq}(\boldsymbol{X},\beta)\big(\mathsf{F}(X_{[\log(d)+1,\ \log(dn)]}) + \textstyle\sum_{i\in[k]}\gamma^i\mathsf{NC}_i(\boldsymbol{X})\big) + \gamma^k\textstyle\sum_{j=1,i=2}^{t,k}\gamma^{i+(j-1)k-1}\cdot\mathsf{Eval}_{(i,j)}(\boldsymbol{X})$$

Define claimed sum of Q over $\{0,1\}^{\log(dn)}$ as

$$\mathrm{T} := \gamma^k\textstyle\sum_{j=1,i=2}^{t,k}\gamma^{i+(j-1)k-1}\cdot\widetilde{y}_{(i,j)}(\alpha)$$

Perform $\mathsf{SumCheck}\,(\mathrm{T};\ Q)$ which reduces to evaluation claim $v \stackrel{?}{=} Q(\alpha',r')$ for $(\alpha',r')\in\mathbb{F}^{\log d}\times\mathbb{F}^{\log n}$.

3. $\mathcal{P}$: For all $i\in[k]$ and $j\in[t]$, send $y'_{(i,j)} := Z_i M_j^{\intercal}\widehat{r}' \in \mathbb{K}^d$ (i.e. $\widetilde{y}'_{(i,j)} = \widetilde{M}_{(i,j)}\big(X_{[1,\log d]}, r'\big)$).
4. $\mathcal{V}$: Check the evaluation claim $v \stackrel{?}{=} Q\big(\alpha',r'\big)$ as follows,

$$\forall j\in[t],\ m_j := \textstyle\sum_{\ell\in[d]} b^{\ell-1}\cdot y'_{(1,j),\ell}$$

$$F := f(m_1,\ldots,m_t), \qquad \forall i\in[k],\ N_i := \textstyle\prod_{j=-b-1}^{b-1}\big(\widetilde{y}'_{(i,1)}(\alpha') - j\big),$$

$$\forall\, i\in[2,\,k],\ \forall\, j\in[t],\ E_{(i,j)} := \mathsf{eq}((\alpha',r'),\,(\alpha,r))\cdot\widetilde{y}'_{(i,j)}(\alpha')$$

$$v \stackrel{?}{=} \mathsf{eq}\big((\alpha',r'),\ \beta\big)\cdot\Big(F + \textstyle\sum_{i\in[k]}\gamma^i\cdot N_i\Big) + \gamma^k\textstyle\sum_{j=1,i=2}^{t,k}\gamma^{i+(j-1)k-1}\cdot E_{(i,j)}$$

5. Output $\big(\mathsf{s};\ c_i,\ X_i,\ r',\ \{y'_{(i,j)}\}_{j\in[t]};\ Z_i\big)_{i\in[k]}$

Lemma 4. *The CCS reduction* Π_{CCS} *is a* ***complete*** *and* ***public coin*** *reduction from* $\mathsf{MCS}(b,\mathcal{L})\times\mathsf{ME}(b,\mathcal{L})^{k-1}$ *to* $\mathsf{ME}(b,\mathcal{L})^k$.

Proof. For brevity, we defer the proof to Appendix B.4. □

Security of Π_{CCS}. We defer the formal security analysis of the reduction Π_{CCS} to Section 5, because the reduction by itself is not a **reduction of knowledge** (Definition 1). This is because Π_{CCS} is not directly knowledge sound (against arbitrary adversaries). Instead, in Section 5, we introduce new security properties which Π_{CCS} satisfies. In particular, Π_{CCS} is knowledge sound against restricted adversaries $(\mathcal{A},\mathcal{P}^*)$ which can only output the same output witnesses w_2 with all but negligible probability. A trivial property of Π_{CCS} is that it always outputs the same commitments from the instance regardless of what the potentially malicious prover $\mathcal{P}^*$ does. This property will help with our composition with Π_{RLC}. Ultimately, we prove that the composition $\Pi_{\mathsf{RLC}}\circ\Pi_{\mathsf{CCS}}$ is a ROK.

4.5 Random linear combination reduction – Π_{RLC}

Overview. This reduction takes a random linear combination of the inputs using challenges from a strong sampling set $\mathcal{C}$ (Definition 14).

Random linear combination reduction Π_{RLC}

Parameters: Refer to Section 4.3.

Input: $\big(\mathsf{s};\ c_i \in \mathbb{C},\ X_i \in \mathbb{F}^{d\times m_{\mathsf{in}}},\ r \in \mathbb{K}^{\log n},\ \{y_{(i,j)} \in \mathbb{K}^d\}_{j\in[t]};\ Z_i \in \mathbb{F}^{d\times m}\big)_{i\in[k+1]} \in \mathsf{ME}(b,\mathcal{L})^{k+1}$

Output: $\big(\mathsf{s};\ c \in \mathbb{C}, X \in \mathbb{F}^{d\times m_{\mathsf{in}}}, r \in \mathbb{K}^{\log n}, \{y_j \in \mathbb{K}^d\}_{j\in[t]};\ Z \in \mathbb{F}^{d\times m}\big) \in \mathsf{ME}(B,\mathcal{L})$

Setup $\mathcal{G}(1^\lambda, \mathsf{sz}) \to \mathsf{pp}$: Output $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{sz})$, which defines $\mathcal{L} := \mathsf{Commit}(\mathsf{pp}, \cdot) : \mathbb{F}^{d\times m} \to \mathbb{C}$.

Encoder $\mathcal{K}(\mathsf{pp}, \mathsf{s}) \to (\mathsf{pk}, \mathsf{vk})$: Output $\big((\mathsf{pp}, \mathsf{s}), \bot\big)$.

Reduction $\langle \mathcal{P}, \mathcal{V}\rangle((\mathsf{pk}, \mathsf{vk}), u_1, w_1) \to (u_2; w_2)$:

1. $\mathcal{V}$: Sample $\rho_1, \ldots, \rho_{k+1} \xleftarrow{\$} \mathcal{C}$ and compute:

$$c \leftarrow \sum_{i\in[k+1]} \rho_i c_i, \qquad X \leftarrow \sum_{i\in[k+1]} \rho_i X_i, \qquad \text{and} \qquad y_j \leftarrow \sum_{i\in[k+1]} \rho_i \cdot y_{(i,j)}$$

   Send $\rho_1, \ldots, \rho_{k+1}$ to $\mathcal{P}$.
2. $\mathcal{P}$: Compute $Z \leftarrow \sum_{i\in[k+1]} \rho_i Z_i$.
3. Output $\big(\mathsf{s};\ c,\ X,\ r,\ \{y_j\}_{j\in[t]};\ Z\big)$.

Lemma 5. *The random linear combination protocol Π_{RLC} is a **complete** and **public coin** reduction from $\mathsf{ME}(b,\mathcal{L})^{k+1}$ to $\mathsf{ME}(B,\mathcal{L})$.*

Proof. For brevity, we defer the proof to Appendix B.5. □

Security of Π_{RLC}. As with Π_{CCS}, the reduction Π_{RLC} is not knowledge-sound against arbitrary adversaries $(\mathcal{A}, \mathcal{P}^*)$. However, it satisfies a *relaxed* notion of knowledge-soundness for which the extractor is able to extract witnesses $(Z_i)_{i\in[k+1]}$ such that $(\mathsf{s}, c_i, X_i, \ldots; Z_i)_{i\in[k+1]}$ belongs to $\mathsf{ME}(q/2,\mathcal{L})^{k+1}$ (which has a trivial norm bound) instead of $\mathsf{ME}(b,\mathcal{L})^{k+1}$, which is the original input relation. Furthermore, this extractor can only output the same witnesses $(Z_i)_i$ with all but negligible probability, or else we could use the extractor to construct a $(d, m, 2B, \mathcal{C})$-relaxed binding (Definition 15) adversary which succeeds with non-negligible probability. These properties are formalized in Section 5. In particular, using this extractor, we will be able to construct a restricted adversary for Π_{CCS}.

4.6 Decomposition reduction – Π_{DEC}

Overview. Our final reduction aims to reduce the norm of claims from $B = b^k$ to b, which will allow us to continually fold CCS claims without increasing the norm of the openings $(Z_i)_i$ to the commitments. Otherwise, the norm would eventually exceed the norm-bound requirement of the commitment scheme.

This reduction works by having the prover do a bit-decomposition of the witness Z into k lower norm witnesses $(Z_i)_i$. Then, the prover sends over new commitments and partial evaluation claims for this new witnesses. The verifier checks these new commitments and evaluations with respect to the original input commitment and evaluation.

Decomposition reduction Π_{DEC}

Parameters: Refer to Section 4.3.

Input: $\big(\mathsf{s};\ c \in \mathbb{C},\ X \in \mathbb{F}^{d \times m_{\mathsf{in}}},\ r \in \mathbb{K}^{\log n},\ \{y_j \in \mathbb{K}^d\}_{j \in [t]};\ Z \in \mathbb{F}^{d \times m}\big) \in \mathsf{ME}(B, \mathcal{L})$

Output: $\big(\mathsf{s};\ c_i \in \mathbb{C},\ X_i \in \mathbb{F}^{d \times m_{\mathsf{in}}},\ r \in \mathbb{K}^{\log n},\ \{y_{(i,j)} \in \mathbb{K}^d\}_{j \in [t]};\ Z_i \in \mathbb{F}^{d \times m}\big)_{i \in [k]} \in \mathsf{ME}(b, \mathcal{L})^k$

Setup $\mathcal{G}(1^\lambda, \mathsf{sz}) \to \mathsf{pp}$: Output $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{sz})$, which defines $\mathcal{L} := \mathsf{Commit}(\mathsf{pp}, \cdot) : \mathbb{F}^{d \times m} \to \mathbb{C}$.

Encoder $\mathcal{K}(\mathsf{pp}, \mathsf{s}) \to (\mathsf{pk}, \mathsf{vk})$: Output $\big((\mathsf{pp}, \mathsf{s}), \perp\big)$.

Reduction $\langle \mathcal{P}, \mathcal{V} \rangle((\mathsf{pk}, \mathsf{vk}), u_1, w_1) \to (u_2; w_2)$:

1. $\mathcal{P}$: Compute $\big(c_i,\ \{y_{(i,j)}\}_{j \in [t]};\ Z_i\big)_{i \in [k]}$ as follows,

$$(Z_1, \ldots, Z_k) \leftarrow \mathsf{split}_b(Z), \qquad c_i \leftarrow \mathcal{L}(Z_i), \qquad y_{(i,j)} \leftarrow Z_i M_j^\intercal \widehat{r} \in \mathbb{K}^d$$

   Send $\big(c_i,\ \{y_{(i,j)}\}_{j \in [t]}\big)_{i \in [k]}$ to $\mathcal{V}$.

2. $\mathcal{V}$: Compute $(X_1, \ldots, X_k) \leftarrow \mathsf{split}_b(X)$. Check for all $j \in [t]$,

$$c \stackrel{?}{=} \sum_{i=1}^k \overline{b}^{\,i-1} \cdot c_i \quad \text{and} \quad y_j \stackrel{?}{=} \sum_{i=1}^k \overline{b}^{\,i-1} \cdot y_{(i,j)}$$

3. Output $\big(\mathsf{s};\ c_i,\ X_i,\ r,\ \{y_{(i,j)}\}_{j \in [t]};\ Z_i\big)_{i \in [k]}$

Unlike Π_{CCS} and Π_{RLC}, the reduction Π_{DEC} is directly a reduction of knowledge.

Theorem 4. *The reduction Π_{DEC} is a **reduction of knowledge** (Definition 1) from $\mathsf{ME}(B, \mathcal{L})$ to $\mathsf{ME}(b, \mathcal{L})^k$.*

Proof. For brevity, we defer the proof to Appendix B.6. □

5 Security analysis of reductions

This section provides a formal security analysis of the Π_{CCS} and Π_{RLC} reductions. In particular, we design new security properties that a reduction can satisfy, and prove that Π_{CCS} and Π_{RLC} satisfy these properties. Finally, we show that the composition $\Pi_{\mathsf{RLC}} \circ \Pi_{\mathsf{CCS}}$ is a reduction of knowledge by using a new general composition theorem for reductions (Theorem 5).

5.1 New properties

We begin with an informal description of each of our new properties, and then provide their formal definitions (Definition 19)

- ϕ-*restricted.* Regardless of the adversary, the evaluation of ϕ on the output instance u_2 remains the same (with all but negligible probability). For example, in Π_{CCS}, the reduction always preserves the instance commitments $(c_i)_i$.
- ϕ-*relaxed knowledge soundness.* Nearly identical to knowledge soundness, but with two major differences: (1) the extractor only succeeds in extracting witnesses that satisfy a relaxed relation (a superset of the original relation); and (2) if the adversary is restricted in its choice of input instances by ϕ, then the extractor can only output a single witness with all but negligible probability. For example, in Π_{RLC}, the extractor will only be able to extract witnesses with potentially large norm; furthermore, if the commitments in the instance remain the same across executions, then the extractor can only output at most one satisfying witness; otherwise, the extractor could break the relaxed binding of the commitment scheme.
- *restricted knowledge soundness.* Nearly identical to knowledge soundness, except that the extraction only succeeds against restricted adversaries. In particular, the potentially malicious prover $\mathcal{P}^*$ is restricted to outputing the exact same witness w_2 with all but negligible probability. For example, in Π_{CCS}, the prover $\mathcal{P}^*$ would be restricted to only one possible set of openings $(Z_i)_i$ for the output commitments.

Theorem 5 (Composition Theorem). *Let ϕ be an arbitrary function. Consider relations $\mathcal{R}_1$ and $\mathcal{R}_2 \subseteq \mathcal{R}_2'$ and $\mathcal{R}_3$. Given a reduction (Definition 1) $\Pi_1 := (\mathcal{G}, \mathcal{K}, \mathcal{P}_1, \mathcal{V}_1)$ from $\mathcal{R}_1$ to $\mathcal{R}_2$ ($\mathcal{R}_2'$) such that Π_1 is (1) complete and public-coin, (2) ϕ-restricted, (3) and has restricted knowledge soundness and a reduction $\Pi_2 := (\mathcal{G}, \mathcal{K}, \mathcal{P}_2, \mathcal{V}_2)$ from $\mathcal{R}_2$ ($\mathcal{R}_2'$) to $\mathcal{R}_3$ such that Π_2 is (1) complete and public-coin (2) and has ϕ-relaxed knowledge soundness then the composition $\Pi := \Pi_2 \circ \Pi_1$ is knowledge sound.*

Proof. For brevity, we defer the proof to Appendix B.10. □

Corollary 2. *The composition $\Pi := \Pi_{\mathsf{RLC}} \circ \Pi_{\mathsf{CCS}}$ is a **reduction of knowledge** (Definition 1) from $\mathsf{ME}(b, \mathcal{L})^k \times \mathsf{MCS}(b, \mathcal{L})$ to $\mathsf{ME}(B, \mathcal{L})$.*

Proof. Follows directly from Π_{CCS}'s completeness and public-coin (Lemma 4), Π_{RLC}'s completeness and public-coin (Lemma 5), the new composition theorem (Theorem 5), Π_{CCS}'s ϕ-restricted (Lemma 6), Π_{CCS}'s restricted knowledge soundness (Lemma 7), and Π_{RLC}'s ϕ-relaxed knowledge soundness (Lemma 8). □

Corollary 3. *The composition $\Pi := \Pi_{\mathsf{DEC}} \circ \Pi_{\mathsf{RLC}} \circ \Pi_{\mathsf{CCS}}$ is a **reduction of knowledge** (Definition 1) from $\mathsf{ME}(b, \mathcal{L})^k \times \mathsf{MCS}(b, \mathcal{L})$ to $\mathsf{ME}(b, \mathcal{L})^k$.*

Proof. Follows directly from Corollary 2, Π_{DEC} is a reduction of knowledge (Theorem 4), and the sequential composition of reductions (Lemma 1). □

Definition 19. *Consider relations $\mathcal{R}_1$ and $\mathcal{R}_2$ over public parameters, structure, instance, and witness tuples. Further, consider relations $\mathcal{R}'_1$ and $\mathcal{R}'_2$ such that $\mathcal{R}_1 \subseteq \mathcal{R}'_1$ and $\mathcal{R}_2 \subseteq \mathcal{R}'_2$. We will refer to these relations as the **relaxed** versions of $\mathcal{R}_1$ and $\mathcal{R}_2$, because they are supersets. Let ϕ be an arbitrary function. A reduction $(\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V})$ (Definition 1) from $\mathcal{R}_1$ ($\mathcal{R}'_1$) to $\mathcal{R}_2$ ($\mathcal{R}'_2$) can satisfy the following properties,*

(i) ***ϕ-restricted**: For any expected polynomial-time adversary $(\mathcal{A}, \mathcal{P}^*)$,*

$$\Pr\left[\begin{array}{c} u_2, u'_2 \neq \perp \\ \Downarrow \\ \phi(u_2) = \phi(u'_2) \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda) \\ (\mathsf{s}, u_1, \mathsf{st}_1) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}) \\ (u_2, w_2) \leftarrow \langle \mathcal{P}^*, \mathcal{V} \rangle ((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st}) \\ (u'_2, w'_2) \leftarrow \langle \mathcal{P}^*, \mathcal{V} \rangle ((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st}) \end{array}\right] = 1$$

(ii) ***ϕ-relaxed knowledge soundness**: For any expected polynomial-time adversary $(\mathcal{A}, \mathcal{P}^*)$, there exists an expected polynomial-time extractor $\mathcal{E}$ such that if the success probability of the adversary $\epsilon(\mathcal{A}, \mathcal{P}^*) \geq 1/\mathsf{poly}(\lambda)$, then*

$$\Pr\left[(\mathsf{pp}, \mathsf{s}, u_1, w_1) \in \mathcal{R}'_1 \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ (\mathsf{s}, u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}) \\ w_1 \leftarrow \mathcal{E}(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st}) \end{array}\right] \geq \epsilon(\mathcal{A}, \mathcal{P}^*) - \mathsf{negl}(\lambda).$$

and if $\mathcal{A} := (\mathcal{B}, \mathcal{B}')$ such that

$$\Pr\left[\begin{array}{c} u_1, u'_1 \neq \perp \\ \Downarrow \\ \phi(u_1) = \phi(u'_1) \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ (\mathsf{s}, \mathsf{st}^*) \leftarrow \mathcal{B}(\mathsf{pp}) \\ (u_1, \mathsf{st}) \leftarrow \mathcal{B}'(\mathsf{st}^*) \\ (u'_1, \mathsf{st}') \leftarrow \mathcal{B}'(\mathsf{st}^*) \end{array}\right] = 1,$$

then

$$\Pr\left[\begin{array}{c} \\ w_1, w'_1 \neq \perp \\ \wedge\ w_1 \neq w'_1 \\ \\ \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ (\mathsf{s}, \mathsf{st}^*) \leftarrow \mathcal{B}(\mathsf{pp}) \\ (u_1, \mathsf{st}) \leftarrow \mathcal{B}'(\mathsf{st}^*) \\ w_1 \leftarrow \mathcal{E}(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st}) \\ (u'_1, \mathsf{st}') \leftarrow \mathcal{B}'(\mathsf{st}^*) \\ w'_1 \leftarrow \mathcal{E}(\mathsf{pp}, \mathsf{s}, u'_1, \mathsf{st}') \end{array}\right] \leq \mathsf{negl}(\lambda)$$

(iii) ***Restricted Knowledge Soundness**: For any expected polynomial-time adversary $(\mathcal{A}, \mathcal{P}^*)$, there exists an expected polynomial-time extractor $\mathcal{E}$ such that if the **relaxed** success probability of the adversary*

$$\epsilon'(\mathcal{A}, \mathcal{P}^*) := \Pr\left[(\mathsf{pp}, \mathsf{s}, \langle \mathcal{P}^*, \mathcal{V} \rangle ((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st})) \in \mathcal{R}'_2 \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ (\mathsf{s}, u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}) \end{array}\right]$$

$\geq 1/\mathsf{poly}(\lambda)$, *and*

$$\Pr\left[\begin{array}{c} w_2, w_2' \neq \bot \\ \wedge \\ w_2 \neq w_2' \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda) \\ (\mathsf{s}, u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}) \\ (u_2, w_2) \leftarrow \langle \mathcal{P}^*, \mathcal{V} \rangle ((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st}) \\ (u_2', w_2') \leftarrow \langle \mathcal{P}^*, \mathcal{V} \rangle ((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

then we have that

$$\Pr\left[(\mathsf{pp}, \mathsf{s}, u_1, w_1) \in \mathcal{R}_1 \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ (\mathsf{s}, u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}) \\ w_1 \leftarrow \mathcal{E}(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st}) \end{array}\right] \geq \epsilon'(\mathcal{A}, \mathcal{P}^*) - \mathsf{negl}(\lambda).$$

Lemma 6. *Let ϕ be the function which projects commitments $(c_i)_{i\in[k]}$ from the instance. The reduction Π_{CCS} is ϕ-restricted.*

Proof. For brevity, we defer the proof to Appendix B.7. □

Lemma 7. *The reduction Π_{CCS} is a restricted knowledge sound reduction from $\mathsf{MCS}(b, \mathcal{L}) \times \mathsf{ME}(b, \mathcal{L})^{k-1}$ to $\mathsf{ME}(b, \mathcal{L})^k$ ($\mathsf{ME}(q/2, \mathcal{L})^k$).*

Proof. For brevity, we defer the proof to Appendix B.8. □

Lemma 8. *Let ϕ be the function which projects commitments $(c_i)_{i\in[k+1]}$ from the instance. The reduction Π_{RLC} is a ϕ-relaxed knowledge sound reduction from $\mathsf{ME}(b, \mathcal{L})^{k+1}$ ($\mathsf{ME}(q/2, \mathcal{L})^{k+1}$) to $\mathsf{ME}(B, \mathcal{L})$.*

Proof. For brevity, we defer the proof to Appendix B.9. □

6 Concrete parameters

This section provides three efficient parameterizations over ≤ 64-bit fields. Additionally, Appendix B.11 and Appendix B.12. provide the corresponding sage scripts that we used to determine valid parameterizations. In Section 4.3, we require the commitment scheme to be $(d, m, 2B, \mathcal{C})$-relaxed binding (Definition 15). Thus, by Lemma 3, we need the commitment scheme to be $(d, m, 4TB)$-binding (Definition 15). Finally, Ajtai's commitment scheme is $(d, m, 4TB)$-binding if $\mathsf{MSIS}^{\infty,\kappa,q}_{m,8TB}$ is hard. We estimate the hardness of Module-SIS using the lattice estimator library provided by [7] using our script (Appendix B.12).

6.1 Almost Goldilocks: $(2^{64} - 2^{32} + 1) - 32$

We provide a new field, which we refer to as *Almost Goldilocks*. This field's order is $q = (2^{64} - 2^{32} + 1) - 32$, which is close to the order of the Goldilocks field $2^{64} - 2^{32} + 1$. Because of this, the field admits an efficient implementation with a small change to the Solinas prime reduction algorithm (which is typically used for the Goldilocks field).

Parameterization. $\eta = 128$, $\Phi_\eta = X^{64} + 1$, $d = 64$, $R_q := \mathbb{F}_q[X]/(\Phi_\eta)$, $\kappa = 13$, $m = 2^{26}$, $b = 2$, $k = 11$, $B = 2^{11}$. Define $\mathcal{C}_R$ to be the set polynomials in R_q whose coefficients belong to $[-1, 0, 1, 2]$, and $\mathcal{C} = \{\mathsf{rot}(a) \mid a \in \mathcal{C}_R\}$. By Theorem 3, $T = 128$. By Theorem 1, $\mathsf{b}_{\mathsf{inv}} \approx 4$. $\mathbb{K} = \mathbb{F}_{q^2}$.

Security $|\mathcal{C}| = 2^{128}$, $|\mathbb{K}| \approx 2^{128}$, $\mathsf{MSIS}^{\infty,\kappa,q}_{m,8TB} \approx 127$ bits of security.

6.2 Goldilocks: $2^{64} - 2^{32} + 1$

This is a popular choice of field for SNARKs as the field admits an efficient implementation: field operations can be implemented with essentially only bit-shifts and the field has high 2-adicity ($2^{32} \mid (p-1)$), which is useful for compressing Neo's IVC proofs with SNARKs.

Parameterization. $\eta = 81$, $\Phi_\eta = X^{54} + X^{27} + 1$, $d = 54$, $R_q := \mathbb{F}_q[X]/(\Phi_\eta)$, $\kappa = 16$, $m = 2^{24}$, $b = 2$, $k = 12$, $B = 2^{12}$. Define $\mathcal{C}_R$ to be the set polynomials in R_q whose coefficients belong to $[-2, -1, 0, 1, 2]$, and $\mathcal{C} = \{\mathsf{rot}(a) \mid a \in \mathcal{C}_R\}$. By Theorem 3, $T = 216$. By Theorem 1, $\mathsf{b}_{\mathsf{inv}} \approx 2.5 \cdot 10^9$. $\mathbb{K} = \mathbb{F}_{q^2}$.

Security $|\mathcal{C}| \approx 2^{125}$, $|\mathbb{K}| \approx 2^{128}$, $\mathsf{MSIS}^{\infty,\kappa,q}_{m,8TB} \approx 128$ bits of security.

Remark 2 (Incompatibility with Latticefold [14]). In LatticeFold [14], the constructions and analysis are limited to power-of-two cyclotomic polynomials, namely of the form $X^d + 1$ with d being a power-of-two. Since the Goldilocks field has high 2-acidity, the cyclotomic polynomial completely factors into linear terms. This means that the ring R_q is isomorphic to $\mathbb{F}_q^d$ (the NTT representation). The security of LatticeFold's construction depends on the size of the field in the NTT representation [14, Sec 3.3], which here is only 64 bits.

6.3 Mersenne 61: $2^{61} - 1$

This field admits an incredibly efficient implementation as it is only one off from a power-of-two. Specifically, modular arithmetic over this field can be implemented with simple bit-shifts with an algorithm more efficient than Goldilocks.

Parameterization $\eta = 81$, $\Phi_\eta = X^{54} + X^{27} + 1$, $d = 54$, $R_q := \mathbb{F}_q[X]/(\Phi_\eta)$, $\kappa = 16$, $m = 2^{22}$, $b = 2$, $k = 12$, $B = 2^{12}$. Define $\mathcal{C}_R$ to be the set polynomials in R_q whose coefficients belong to $[-2, -1, 0, 1, 2]$, and $\mathcal{C} = \{\mathsf{rot}(a) \mid a \in \mathcal{C}_R\}$. By Theorem 3, $T = 216$. By Theorem 1, $\mathsf{b}_{\mathsf{inv}} \approx 383$. $\mathbb{K} = \mathbb{F}_{q^2}$.

Security $|\mathcal{C}| \approx 2^{125}$, $|\mathbb{K}| \approx 2^{122}$, $\mathsf{MSIS}^{\infty,\kappa,q}_{m,8TB} \approx 129$ bits of security.

Remark 3 (Incompatibility with Latticefold [14]). As stated earlier, LatticeFold's constructions and analysis are limited to power-of-two cyclotomic polynomials, namely of the form X^d+1 for d being a power-of-two. For Mersenne 61, there is no choice of power-of-two cyclotomic polynomials, which satisfies the requirements of Theorem 1. Hence, it cannot be determined whether a choice of parameters with $\Phi_\eta = X^d + 1$ leads to a secure construction.

References

[1] Nova: Recursive SNARKs without trusted setup. https://github.com/Microsoft/Nova

[2] Ova: A slightly better Nova. https://hackmd.io/V4838nnlRKal9ZiTHiGYzw

[3] Folding (Jolt Book). https://jolt.a16zcrypto.com/future/folding.html (2024)

[4] The Nexus zkVM. https://github.com/nexus-xyz/nexus-zkvm (2024)

[5] Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: 28th Annual ACM Symposium on Theory of Computing. pp. 99–108. ACM Press, Philadephia, PA, USA (May 22–24, 1996). https://doi.org/10.1145/237814.237838

[6] Albrecht, M.R., Lai, R.W.F.: Subtractive sets over cyclotomic rings - limits of Schnorr-like arguments over lattices. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021, Part II. Lecture Notes in Computer Science, vol. 12826, pp. 519–548. Springer, Cham, Switzerland, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/978-3-030-84245-1_18

[7] Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015)

[8] Arun, A., Setty, S.: Nebula: Efficient read-write memory and switchboard circuits for folding schemes. Cryptology ePrint Archive (2024)

[9] Attema, T., Cramer, R., Kohl, L.: A compressed Σ-protocol theory for lattices. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021, Part II. Lecture Notes in Computer Science, vol. 12826, pp. 549–579. Springer, Cham, Switzerland, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/978-3-030-84245-1_19

[10] Attema, T., Lyubashevsky, V., Seiler, G.: Practical product proofs for lattice commitments. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 470–499. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56880-1_17

[11] Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In: CRYPTO (2019)

[12] Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable zero knowledge via cycles of elliptic curves. In: CRYPTO (2014)

[13] Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKs and proof-carrying data. In: STOC (2013)

[14] Boneh, D., Chen, B.: LatticeFold: A lattice-based folding scheme and its applications to succinct proof systems. Cryptology ePrint Archive, Paper 2024/257 (2024)

[15] Bootle, J., Chiesa, A., Hu, Y., Orrú, M.: Gemini: Elastic snarks for diverse environments. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 427–457 (2022)

[16] Bowe, S., Grigg, J., Hopwood, D.: Halo: Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021 (2019)

[17] Bünz, B., Chen, B.: Protostar: Generic efficient accumulation/folding for special sound protocols. Cryptology ePrint Archive, Paper 2023/620 (2023)
[18] Bünz, B., Chiesa, A., Lin, W., Mishra, P., Spooner, N.: Proof-carrying data without succinct arguments. In: CRYPTO (2021)
[19] Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: EUROCRYPT (2020)
[20] Bunz, B., Mishra, P., Nguyen, W., Wang, W.: Arc: Accumulation for reed–solomon codes. Cryptology ePrint Archive, Paper 2024/1731 (2024)
[21] Bünz, B., Chen, J.: Proofs for deep thought: Accumulation for large memories and deterministic computations. Cryptology ePrint Archive, Paper 2024/325 (2024)
[22] Bünz, B., Mishra, P., Nguyen, W., Wang, W.: Accumulation without homomorphism. Cryptology ePrint Archive, Paper 2024/474 (2024)
[23] Chen, S., Cheon, J.H., Kim, D., Park, D.: Verifiable computing for approximate computation. Cryptology ePrint Archive, Report 2019/762 (2019), https://eprint.iacr.org/2019/762
[24] Chen, W., Chiesa, A., Dauterman, E., Ward, N.P.: Reducing participation costs via incremental verification for ledger systems. Cryptology ePrint Archive, Report 2020/1522 (2020)
[25] Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: EUROCRYPT (2020)
[26] Chiesa, A., Ojha, D., Spooner, N.: Fractal: Post-quantum and transparent recursive proofs from holography. In: EUROCRYPT (2020)
[27] Diamond, B.E., Posen, J.: Succinct arguments over towers of binary fields. Cryptology ePrint Archive, Paper 2023/1784 (2023)
[28] Dimitriou, N., Garreta, A., Manzur, I., Vlasov, I.: Mova: Nova folding without committing to error terms. Cryptology ePrint Archive, Paper 2024/1220 (2024)
[29] Eagen, L., Gabizon, A.: Protogalaxy: Efficient protostar-style folding of multiple instances. Cryptology ePrint Archive, Paper 2023/1106 (2023)
[30] Eagen, L., Gabizon, A., Sefranek, M., Towa, P., Williamson, Z.J.: Stackproofs: Private proofs of stack and contract execution using protogalaxy. Cryptology ePrint Archive, Paper 2024/1281 (2024)
[31] Fenzi, G., Knabenhans, C., Nguyen, N.K., Pham, D.T.: Lova: Lattice-based folding scheme from unstructured lattices. Cryptology ePrint Archive, Paper 2024/1964 (2024)
[32] Fenzi, G., Moghaddas, H., Nguyen, N.K.: Lattice-based polynomial commitments: Towards asymptotic and concrete efficiency. Journal of Cryptology **37**(3), 31 (Jul 2024). https://doi.org/10.1007/s00145-024-09511-8
[33] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO. pp. 186–194 (1986)
[34] Garreta, A., Manzur, I.: FLI: Folding lookup instances. Cryptology ePrint Archive, Paper 2024/1531 (2024), https://eprint.iacr.org/2024/1531
[35] Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for zero-knowledge proof systems. Cryptology ePrint Archive, Paper 2019/458 (2019)

[36] Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: ASIACRYPT. pp. 177–194 (2010)
[37] Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: STOC (1992)
[38] Kothapalli, A.: A Theory of Composition for Proofs of Knowledge. Ph.D. thesis, Carnegie Mellon University (2024)
[39] Kothapalli, A., Parno, B.: Algebraic reductions of knowledge. In: CRYPTO (2023)
[40] Kothapalli, A., Setty, S.: SuperNova: Proving universal machine executions without universal circuits. Cryptology ePrint Archive (2022)
[41] Kothapalli, A., Setty, S.: Cyclefold: Folding-scheme-based recursive arguments over a cycle of elliptic curves. Cryptology ePrint Archive, Paper 2023/1192 (2023), https://eprint.iacr.org/2023/1192, https://eprint.iacr.org/2023/1192
[42] Kothapalli, A., Setty, S.: HyperNova: Recursive arguments for customizable constraint systems. In: CRYPTO (2024)
[43] Kothapalli, A., Setty, S.: NeutronNova: Folding everything that reduces to zero-check. Cryptology ePrint Archive (2024)
[44] Kothapalli, A., Setty, S., Tzialla, I.: Nova: Recursive Zero-Knowledge Arguments from Folding Schemes. In: CRYPTO (2022)
[45] Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography **75**(3), 565–599 (2015). https://doi.org/10.1007/s10623-014-9938-4
[46] Lund, C., Fortnow, L., Karloff, H., Nisan, N.: Algebraic methods for interactive proof systems. In: FOCS (Oct 1990)
[47] Lyubashevsky, V., Micciancio, D.: Generalized compact Knapsacks are collision resistant. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006: 33rd International Colloquium on Automata, Languages and Programming, Part II. Lecture Notes in Computer Science, vol. 4052, pp. 144–155. Springer Berlin Heidelberg, Germany, Venice, Italy (Jul 10–14, 2006). https://doi.org/10.1007/11787006_13
[48] Lyubashevsky, V., Seiler, G.: Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018, Part I. Lecture Notes in Computer Science, vol. 10820, pp. 204–224. Springer, Cham, Switzerland, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78381-9_8
[49] Micali, S.: CS proofs. In: FOCS (1994)
[50] Micciancio, D., Regev, O.: Lattice-based cryptography. In: Post-quantum cryptography, pp. 147–191. Springer (2009)
[51] Nethermind Research: Lattice-based operations performance report. https://nethermind.notion.site/Latticefold-and-lattice-based-operations-performance-report-153360fc38d080ac930cdeeffed69559 (2025)
[52] Nguyen, W., Boneh, D., Setty, S.: Revisiting the Nova proof system on a cycle of curves. Cryptology ePrint Archive, Paper 2023/969 (2023)

[53] Nguyen, W., Datta, T., Chen, B., Tyagi, N., Boneh, D.: Mangrove: A scalable framework for folding-based SNARKs. In: CRYPTO (2024)
[54] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: CRYPTO (1991)
[55] Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006: 3rd Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 3876, pp. 145–166. Springer Berlin Heidelberg, Germany, New York, NY, USA (Mar 4–7, 2006). https://doi.org/10.1007/11681878_8
[56] Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. J. ACM **27**(4) (1980)
[57] Setty, S.: Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: CRYPTO (2020)
[58] Setty, S., Thaler, J.: Twist and Shout: Faster memory checking arguments via one-hot addressing and increments. Cryptology ePrint Archive, Paper 2025/105 (2025)
[59] Setty, S., Thaler, J., Wahby, R.: Customizable constraint systems for succinct arguments. Cryptology ePrint Archive (2023)
[60] Thaler, J.: Time-optimal interactive proofs for circuit evaluation. In: CRYPTO (2013)
[61] Thaler, J.: The sum-check protocol. https://people.cs.georgetown.edu/jthaler/sumcheck.pdf (Sep 2017)
[62] Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: TCC. pp. 552–576 (2008)
[63] Vu, V., Setty, S., Blumberg, A.J., Walfish, M.: A hybrid architecture for verifiable computation. In: S&P (2013)
[64] Zhao, J., Setty, S., Cui, W.: MicroNova: Folding-based arguments with efficient (on-chain) verification. Cryptology ePrint Archive, Paper 2024/2099 (2024), https://eprint.iacr.org/2024/2099
[65] Zhou, Z., Zhang, Z., Dong, J.: Proof-carrying data from multi-folding schemes. Cryptology ePrint Archive, Paper 2023/1282 (2023)

Supplementary Material

A Additional Background

A.1 Polynomials and multilinear extensions

We adapt this some of this subsection from prior work [57]. We recall several definitions and results regarding multivariate polynomials.

Definition 20 (Multilinear polynomial). *A multivariate polynomial is called a multilinear polynomial if the degree of the polynomial in each variable is at most one.*

Definition 21 (Multilinear polynomial extension). *Given a vector $v \in \mathbb{F}^n$ a* multilinear polynomial extension *of v is an $(\log n)$-variate multilinear polynomial, denoted $\widetilde{v}$, such that $\widetilde{v}(x) = v_x$ for all $x \in \{0,1\}^{\log n}$. Specifically, $\widetilde{v}$ can be computed as follows.*

$$\widetilde{v}(x) = \sum_{y \in \{0,1\}^\ell} v_y \cdot \mathsf{eq}(x, y)$$

where $\mathsf{eq}(x,y) = \prod_{i=1}^{\ell}(x_i \cdot y_i + (1-x_i)\cdot(1-y_i))$, outputs 1 if $x = y$ and 0 otherwise for $x, y \in \{0,1\}^{\log n}$.

For any $r \in \mathbb{F}^\ell$, $\widetilde{v}(r)$ can be computed in $O(2^\ell)$ operations in $\mathbb{F}$ [60, 63].

Lemma 9 (Schwartz-Zippel [56]). *let $g : \mathbb{F}^\ell \to \mathbb{F}$ be an ℓ-variate polynomial of total degree at most d. Then, on any finite set $S \subseteq \mathbb{F}$,*

$$\Pr_{x \leftarrow S^\ell}[g(x) = 0] \leq d/|S|.$$

Lemma 10. *Let $Q \in \mathbb{F}[X_1, \ldots, X_\ell]$ be an arbitrary multivariate polynomial. Define multivariate polynomial $Q'(\boldsymbol{X}, \boldsymbol{Z}) := \mathsf{eq}(\boldsymbol{X}, \boldsymbol{Z}) \cdot Q(\boldsymbol{X})$.*

$$0 = \sum_{\boldsymbol{x} \in \{0,1\}^{\log \ell}} Q'(\boldsymbol{x}, \boldsymbol{Z}) \quad \textit{if and only if} \quad Q(\boldsymbol{X}) \in \mathsf{ZS}_\ell$$

Definition 22 (Special sets [32]). *Let $\mathcal{C}$ be a set and $\ell \in \mathbb{N}$. Consider two vectors $x, y \in \mathcal{C}^\ell$. We define the relation $\equiv_i$ for $i \in [\ell]$ as follows:*

$$x \equiv_i y \iff x_i \neq y_i \ \wedge \ x_j = y_j \text{ for all } j \in [\ell] \setminus \{i\}.$$

A special set $\mathsf{SS}(\mathcal{C}, \ell)$ is as follows:

$$\mathsf{SS}(\mathcal{C}, \ell) = \left\{ (\boldsymbol{c}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_\ell) \in (\mathcal{C}^\ell)^{\ell+1} \ : \ \begin{matrix} \forall\, i \in [\ell], \\ \boldsymbol{c} \equiv_i \boldsymbol{c}_i \end{matrix} \right\},$$

Theorem 6 (Coordinate-wise extraction [32, Lemma 7.1]). *Let $\mathcal{C}$ be a finite set, $\ell \in \mathbb{N}$, and $\boldsymbol{\mathcal{C}} := \mathcal{C}^\ell$ be a challenge space. Let $A : \boldsymbol{\mathcal{C}} \to \{0,1\}^*$ be an arbitrary (probabilistic) expected polynomial-time algorithm (adversary), and $V : \boldsymbol{\mathcal{C}} \times \{0,1\}^* \to \{0,1\}$ be an arbitrary (probabilistic) polynomial-time function (verification). Define the success probability of adversary A as*

$$\epsilon^V(A) := \Pr_{\boldsymbol{c} \xleftarrow{\$} \boldsymbol{\mathcal{C}}}[V(\boldsymbol{c}, A(\boldsymbol{c})) = 1]$$

Then, there exists an expected polynomial-time oracle algorithm E_A (extractor) that makes at most $\ell + 1$ queries to A in expectation and with probability at least $\epsilon^V(A) - \frac{\ell}{|\mathcal{C}|}$ outputs $\ell + 1$ pairs $(\boldsymbol{c}, w), (\boldsymbol{c}_1, w_1), \ldots (\boldsymbol{c}_\ell, w_\ell)$ such that
- *$V(\boldsymbol{c}, w) = 1$,*
- *for all $i \in [\ell]$, $V(\boldsymbol{c}_i, w_i) = 1$,*
- *and $(\boldsymbol{c}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_\ell) \in \mathsf{SS}(\mathcal{C}, \ell)$.*

B Deferred theorems and proofs

Theorem 7. *The rings R_q (Definition 5) and $\mathcal{S}$ (Definition 7) are isomorphic. That is, the rotation matrix mapping $\mathsf{rot} : R_q \to \mathcal{S}$ is an isomorphism.*

Proof. We first show that rot is a homomorphism:

$$\begin{aligned}
\mathsf{rot}(a) + \mathsf{rot}(b) &= \big[\, \mathsf{cf}(a) + \mathsf{cf}(b) \;\big|\; \mathsf{F} \cdot \mathsf{cf}(a) + \mathsf{F} \cdot \mathsf{cf}(b) \;\big|\; \cdots \;\big|\; \mathsf{F}^{d-1} \cdot \mathsf{cf}(a) + \mathsf{F}^{d-1} \cdot \mathsf{cf}(b) \,\big] \\
&= \big[\, \mathsf{cf}(a+b) \;\big|\; \mathsf{F} \cdot \mathsf{cf}(a+b) \;\big|\; \cdots \;\big|\; \mathsf{F}^{d-1} \cdot \mathsf{cf}(a+b) \,\big] \\
&= \mathsf{rot}(a+b) \\
\mathsf{rot}(a) \cdot \mathsf{rot}(b) &= \mathsf{rot}(a) \cdot \big[\, \mathsf{cf}(b) \;\big|\; \mathsf{F} \cdot \mathsf{cf}(b) \;\big|\; \cdots \;\big|\; \mathsf{F}^{d-1} \cdot \mathsf{cf}(b) \,\big] \\
&= \big[\, \mathsf{rot}(a) \cdot \mathsf{cf}(b) \;\big|\; \mathsf{rot}(a) \cdot \mathsf{F} \cdot \mathsf{cf}(b) \;\big|\; \cdots \;\big|\; \mathsf{rot}(a) \cdot \mathsf{F}^{d-1} \cdot \mathsf{cf}(b) \,\big] \\
&= \big[\, \mathsf{rot}(a) \cdot \mathsf{cf}(b) \;\big|\; \mathsf{rot}(a) \cdot \mathsf{cf}(X \cdot b) \;\big|\; \cdots \;\big|\; \mathsf{rot}(a) \cdot \mathsf{cf}\big(X^{d-1} \cdot b\big) \,\big] \\
&= \big[\, \mathsf{cf}(ab) \;\big|\; \mathsf{cf}(X \cdot ab) \;\big|\; \cdots \;\big|\; \mathsf{cf}\big(X^{d-1} \cdot ab\big) \,\big] \\
&= \big[\, \mathsf{cf}(ab) \;\big|\; \mathsf{F} \cdot \mathsf{cf}(ab) \;\big|\; \cdots \;\big|\; \mathsf{F}^{d-1} \cdot \mathsf{cf}(ab) \,\big] \\
&= \mathsf{rot}(ab) \\
\mathsf{rot}(1) &= \big[\, \mathsf{cf}(1) \;\big|\; \mathsf{cf}(X) \;\big|\; \cdots \;\big|\; \mathsf{cf}\big(X^{d-1}\big) \,\big] \\
&= I_d
\end{aligned}$$

We can observe that rot is trivially a bijection because the first column contains the polynomial coefficients; hence rot is additionally a ring isomorphism. □

B.1 Proof of Theorem 2

Proof. We make a small notational modification to Ajtai's commitment scheme to adapt it to our matrix setting. In particular, the commitment scheme is functionally exactly identical.

Ajtai's commitment scheme. Let $m \in \mathbb{N}$ denote the message length. Ajtai's commitment scheme $\mathsf{com} := (\mathsf{Setup}, \mathsf{Commit})$ consists of the following algorithms:

- $\mathsf{Setup}(\kappa, m) \to \mathsf{pp}$: Sample a random matrix $M \xleftarrow{\$} R_q^{\kappa \times m}$. Output $\mathsf{pp} \leftarrow M$.
- $\mathsf{Commit}(\mathsf{pp}, Z \in \mathbb{F}^{d \times m}) \to c$: Parse $M \leftarrow \mathsf{pp}$. Assign $z := \mathsf{cf}^{-1}(Z) \in R_q^m$. Output $c \leftarrow \mathsf{cf}(Mz)$.

We will first show that the commitment scheme is **homomorphic**. We can trivially observe that the groups $\mathbb{F}^{d \times m}$ and $\mathbb{F}^{d \times \kappa}$ are $\mathcal{S}$-modules by left multiplication, and R_q^m and R_q^κ are isomorphic (as groups) to $\mathbb{F}^{d \times m}$ and $\mathbb{F}^{d \times \kappa}$, respectively. That is, the coefficient map $\mathsf{cf}(\cdot)$ is a trivial group isomorphism. Consider two arbitrary matrices $A, B \in \mathbb{F}^{d \times m}$ and $\mathsf{rot}(s) \in \mathcal{S}$. We must have

$$\begin{aligned}
\mathsf{Commit}(\mathsf{pp}, A) + \mathsf{Commit}(\mathsf{pp}, B) &= \mathsf{cf}\big(M\mathsf{cf}^{-1}(A)\big) + \mathsf{cf}\big(M\mathsf{cf}^{-1}(B)\big) && (1)\\
&= \mathsf{cf}\big(M\mathsf{cf}^{-1}(A) + M\mathsf{cf}^{-1}(B)\big) && (2)\\
&= \mathsf{cf}\big(M(\mathsf{cf}^{-1}(A) + \mathsf{cf}^{-1}(B))\big)\\
&= \mathsf{cf}\big(M\mathsf{cf}^{-1}(A + B)\big) && (3)\\
&= \mathsf{Commit}(\mathsf{pp}, A + B) && (4)
\end{aligned}$$

$$\begin{aligned}
\mathsf{rot}(s) \cdot \mathsf{Commit}(\mathsf{pp}, A) &= \mathsf{rot}(s) \cdot \mathsf{cf}\big(M\mathsf{cf}^{-1}(A)\big) && (5)\\
&= \mathsf{cf}\big(s \cdot M\mathsf{cf}^{-1}(A)\big) && (6)\\
&= \mathsf{cf}\big(M \cdot s \cdot \mathsf{cf}^{-1}(A)\big)\\
&= \mathsf{cf}\big(M \cdot \mathsf{cf}^{-1}(\mathsf{rot}(s) \cdot A)\big) && (7)\\
&= \mathsf{Commit}(\mathsf{pp}, \mathsf{rot}(s) \cdot A) && (8)
\end{aligned}$$

where (1), (4), and (5), (8) are by the definition of Ajtai's commitment scheme, (2) and (3) are by the homomorphism of the $\mathsf{cf}(\cdot)$ map, and (6) and (7) come from the fact that $\mathsf{rot}(a) \cdot \mathsf{cf}(b) = \mathsf{cf}(ab)$ for any $a, b \in R_q$.

Now, we will show that the commitment scheme is (d, m, B)**-binding**. For this, it suffices to show that a (d, m, B)-binding collision produces an $\mathsf{MSIS}^{\infty,\kappa,q}_{m,2B}$ solution with respect to the uniformly sampled M in the pubic parameters pp. Consider an arbitrary (d, m, B)-binding collision $Z_1, Z_2 \in \mathbb{F}^{d \times m}$. We will show $\mathsf{cf}^{-1}(Z_1 - Z_2)$ is an $\mathsf{MSIS}^{\infty,\kappa,q}_{m,2B}$ solution. By the definition of collision, we must have

$$\begin{aligned}
\mathsf{Commit}(\mathsf{pp}, Z_1) &= \mathsf{Commit}(\mathsf{pp}, Z_2)\\
\mathsf{Commit}(\mathsf{pp}, Z_1 - Z_2) &= 0 \in \mathbb{F}^{d \times \kappa} && (9)\\
\mathsf{cf}\big(M\mathsf{cf}^{-1}(Z_1 - Z_2)\big) &= 0 \in \mathbb{F}^{d \times \kappa} && (10)\\
M\mathsf{cf}^{-1}(Z_1 - Z_2) &= 0 \in R_q^\kappa && (11)
\end{aligned}$$

where (9) is by homomorphism, (10) is by the definition of Ajtai's commitment scheme, and (11) is by applying the map $\mathsf{cf}^{-1}(\cdot)$ to both sides. Since $\|Z_1\|_\infty, \|Z_2\|_\infty < B$ (from the definition of an (d, m, B)-binding collision), we must have $\|Z_1 - Z_2\|_\infty < 2B$, and therefore $\big\|\mathsf{cf}^{-1}(Z_1 - Z_2)\big\|_\infty = \|Z_1 - Z_2\|_\infty < 2B$. □

B.2 Proof of Lemma 3

Proof. It suffices to show that an arbitrary $(d, m, B, \mathcal{C})$-relaxed binding collision $(c, \Delta_1, \Delta_2, Z_1, Z_2)$ can be used to construct a $(d, m, 2TB)$-binding collision. Specifically, we will show the pair $(\Delta_1 Z_2, \Delta_2 Z_1)$ is an $(d, m, 2TB)$-binding collision. We must have

$$\Delta_1 Z_2 \neq \Delta_2 Z_1 \tag{12}$$

$$\Delta_1 \cdot c = \mathsf{Commit}(\mathsf{pp}, Z_1) \tag{13}$$

$$\Delta_2 \cdot \Delta_1 \cdot c = \Delta_2 \cdot \mathsf{Commit}(\mathsf{pp}, Z_1)$$

$$\Delta_1 \cdot \mathsf{Commit}(\mathsf{pp}, Z_2) = \Delta_2 \cdot \mathsf{Commit}(\mathsf{pp}, Z_1) \tag{14}$$

$$\mathsf{Commit}(\mathsf{pp}, \Delta_1 Z_2) = \mathsf{Commit}(\mathsf{pp}, \Delta_2 Z_1) \tag{15}$$

$$\left\|\Delta_1 Z_2\right\|_\infty, \left\|\Delta_2 Z_1\right\|_\infty \leq 2TB \tag{16}$$

where (12) and (13) follow from the definition of relaxed binding collision, (14) is by commutativity and substitution, and (15) follows from $\mathcal{C} \subseteq \mathcal{S}$ and homomorphism. Since expansion factor of $\mathcal{C}$ is T and $\Delta_1, \Delta_2 \in (\mathcal{C} - \mathcal{C})$, we must have that that $\left\|\Delta_1 Z_2\right\|_\infty \leq 2T \left\|Z_2\right\|_\infty$ and $\left\|\Delta_2 Z_1\right\|_\infty \leq 2T \left\|Z_1\right\|_\infty$. Since $\left\|Z_1\right\|_\infty, \left\|Z_2\right\|_\infty < B$ by definition, we have (16) holds. From (12), (15), and (16), we have the pair $(\Delta_1 Z_2, \Delta_2 Z_1)$ is an $(d, m, 2TB)$-binding collision. □

B.3 Proof of Theorem 3

Proof. By [6, Proposition 2], we have that

$$\max_{u,v \in R_q} \frac{\left\|uv\right\|_\infty}{\left\|u\right\|_\infty \cdot \left\|v\right\|_\infty} \leq 2 \cdot \phi(\eta) \tag{17}$$

For all $u, v \in R_q$, we must have that

$$\frac{\left\|uv\right\|_\infty}{\left\|v\right\|_\infty} \leq 2 \cdot \phi(\eta) \cdot \left\|u\right\|_\infty \tag{18}$$

$$\frac{\left\|\mathsf{rot}(u) \cdot \mathsf{cf}(v)\right\|_\infty}{\left\|\mathsf{cf}(v)\right\|_\infty} \leq 2 \cdot \phi(\eta) \cdot \left\|u\right\|_\infty \tag{19}$$

where (18) follows from (17), and (19) follows from $\mathsf{rot}(u) \cdot \mathsf{cf}(v) = \mathsf{cf}(uv)$ and the definition of norm $\left\|\cdot\right\|_\infty$ for ring elements. Therefore, we must have for all $u \in R_q, v \in \mathbb{F}^d$ that

$$\frac{\left\|\mathsf{rot}(u) \cdot v\right\|_\infty}{\left\|v\right\|_\infty} \leq 2 \cdot \phi(\eta) \cdot \left\|u\right\|_\infty \tag{20}$$

Since $\mathcal{C}_R \subseteq R_q$, we must also have the prior bound holds for all $u \in \mathcal{C}_R, v \in \mathbb{F}^d$. Since $\mathcal{C} := \{\, \mathsf{rot}(u) \mid u \in \mathcal{C}_R \,\}$, we must have for all $\rho := \mathsf{rot}(u) \in \mathcal{C}, v \in \mathbb{F}^d$,

$$\frac{\left\|\rho \cdot v\right\|_\infty}{\left\|v\right\|_\infty} \leq 2 \cdot \phi(\eta) \cdot \left\|u\right\|_\infty \leq 2 \cdot \phi(\eta) \cdot \max_{\rho' \in \mathcal{C}_R} \left\|\rho'\right\|_\infty \tag{21}$$

which is exactly what we wanted to show. □

B.4 Proof of Lemma 4

We first provide a lemma that will be helpful for both the security and completeness of the reduction.

Lemma 11. *Consider an arbitrary structure* s*, a vector* $z_1 \in \mathbb{F}^m$*, matrices* $\big(Z_i \in \mathbb{F}^{d\times m}\big)_{i=2}^k$*, a point* $r \in \mathbb{K}^{\log n}$*, and vectors* $(\{y_{(i,j)} \in \mathbb{K}^d\}_{j\in[t]})_{i=2}^k$*. Define* $Z_1 := \mathsf{Decomp}_b(z_1)$*. For all* $i \in [k], j \in [t]$*, define* $M_{(i,j)} := Z_i M_j^\intercal$*. Define* $\delta := \log d$ *and* $\zeta := \log(dn)$*. Define polynomials* $\mathsf{F}(X_{[1,n]})$ *and* $\mathsf{NC}_i(X_{[1,\zeta]})$ *for* $i \in [k]$ *as in* Π_{CCS}*. Define indeterminates* $\boldsymbol{X} := X_{[1,\zeta]}$*,* $\boldsymbol{A} := A_{[1,\delta]}$*, and* $\boldsymbol{B} := B_{[1,\zeta]}$*. Additionally, define polynomials*

$$\mathsf{Eval}_{i,j}(\boldsymbol{X},\boldsymbol{A}) := \mathsf{eq}\big(\boldsymbol{X},\boldsymbol{A},r\big)\cdot\widetilde{M}_{i,j}(\boldsymbol{X})\ \forall\, i\in[k+1,\,2k],\ \forall\, j\in[t]$$

$$\begin{aligned} Q(\boldsymbol{X},\boldsymbol{A},\boldsymbol{B},C) := \mathsf{eq}(\boldsymbol{X},\boldsymbol{B})\big(\mathsf{F}(X_{[\delta+1,\ \zeta]}) + \textstyle\sum_{i\in[k]} C^i\cdot\mathsf{NC}_i(\boldsymbol{X})\big) \\ + C^k\textstyle\sum_{j=1,i=2}^{t,k} C^{i+(j-1)k-1}\cdot\mathsf{Eval}_{(i,j)}(\boldsymbol{X}) \end{aligned}$$

Further, define quantity $\mathrm{T} := C^k\sum_{j=1,i=2}^{t,k} C^{i+(j-1)k-1}\cdot\widetilde{y}_{(i,j)}(\boldsymbol{A})$*. We must have* $T \neq \sum_{x\in\{0,1\}^\zeta} Q(x,\boldsymbol{A},\boldsymbol{B},C)$ *if and only if*

1. $\mathsf{F}(X_{[\delta+1,\zeta]}) \notin \mathsf{ZS}_n$ *(identically,* $\mathsf{F}(X_{[1,n]}) \notin \mathsf{ZS}_n$*),*
2. *OR there exists an* $i \in [k]$ *for which* $\|Z_i\|_\infty \geq b$*,*
3. *OR there exists an* $i \in [2,k]$ *for which* $\widetilde{y}_{(i,j)}(\boldsymbol{A}) \neq \widetilde{M}_{i,j}(\boldsymbol{A},r)$ *(identically,* $y_{(i,j)} = Z_i M_j^\intercal \widehat{r}$*)*

Proof. Since the powers of indeterminate C are linearly independent, $\mathrm{T} \neq \sum_{x\in\{0,1\}^\zeta} Q(x,\boldsymbol{A},\boldsymbol{B},C)$ if and only if

$$0 \neq \sum_{x\in\{0,1\}^\zeta} \mathsf{eq}(x,\boldsymbol{B})\cdot\mathsf{F}(x_{[\delta+1,\zeta]}), \tag{22}$$

$$\text{OR}\quad \exists\, i\in[k], \qquad 0 \neq \sum_{x\in\{0,1\}^\zeta} \mathsf{eq}(x,\boldsymbol{B})\cdot\mathsf{NC}_i(x), \tag{23}$$

$$\text{OR}\quad \exists\, i\in[2,k],\, j\in[t] \qquad \widetilde{y}_{(i,j)}(\boldsymbol{A}) \neq \sum_{x\in\{0,1\}^\zeta} \mathsf{Eval}_{i,j}(x,\boldsymbol{A}) \tag{24}$$

By Lemma 10, we must have (22) if and only if $\mathsf{F}(\boldsymbol{X}) \notin \mathsf{ZS}_{dn}$ (Item 1). Since, for all $i \in [k]$, $\|Z_i\|_\infty \geq b$ if and only if $\mathsf{NC}_i(\boldsymbol{X}) \notin \mathsf{ZS}_\zeta$, by Lemma 10, we have that (23) if and only if there exists an $i \in [k]$ for which $\|Z_i\|_\infty \geq b$ (Item 2). Observe for all i, j that

$$\begin{aligned} \sum_{x\in\{0,1\}^\zeta} \mathsf{Eval}_{i,j}(x,\boldsymbol{A}) &= \sum_{x\in\{0,1\}^\zeta} \mathsf{eq}\big(x,\ \boldsymbol{A},r\big)\cdot\widetilde{M}_{i,j}(x) \\ &= \sum_{x\in\{0,1\}^\zeta} \mathsf{eq}\big(x,\ \boldsymbol{A},r\big)\cdot\widetilde{M}_{i,j}(x) \end{aligned}$$

$$= \widetilde{M}_{i,j}(\boldsymbol{A}, r) \tag{25}$$

where (25) follows from the definition of a multilinear extension. Thus, by (25), we have (24) if and only if there exists an $i \in [2,k]$ for which $\widetilde{y}_{i,j}(\boldsymbol{A}) \neq \widetilde{M}_{i,j}(\boldsymbol{A}, r)$ (Item 3).

In conclusion, we have shown $\mathrm{T} \neq \sum_{x \in \{0,1\}^\zeta} Q(x, \boldsymbol{A}, \boldsymbol{B}, C)$ if and only if (22), (23), or (24), which holds if and only if (Item 1), (Item 2), or (Item 3). This concludes our proof. □

Now, we provide the proof of Lemma 4.

Proof.

Completeness: We will first argue that the sum-check verifier does not reject if the original tuples belong to the input relations. Consider the contrapositive of Lemma 11,

1. $\mathsf{F}(X_{[\delta+1,\zeta]}) \in \mathsf{ZS}_n$ (identically, $\mathsf{F}(X_{[1,n]}) \in \mathsf{ZS}_n$),
2. for all $i \in [k]$, $\|Z_i\|_\infty < b$, and
3. for all $i \in [2,k]$, $\widetilde{y}_{(i,j)}(\boldsymbol{A}) \neq \widetilde{M}_{i,j}(\boldsymbol{A}, r)$ (identically, $y_{(i,j)} = Z_i M_j^\intercal \widehat{r}$)

if and only if $\mathrm{T} = \sum_{x \in \{0,1\}^\zeta} Q(x, \boldsymbol{A}, \boldsymbol{B}, C)$. We will show each condition is guaranteed by the input tuples belonging to $\mathsf{MCS}(b, \mathcal{L}) \times \mathsf{ME}(b, \mathcal{L})^{k-1}$.

By the definition of $\mathsf{MCS}(b, \mathcal{L})$ (Definition 17), $(\mathsf{s};\ (c_1,\ x_1);\ w_1) \in \mathsf{MCS}(b, \mathcal{L})$ guarantees that $\mathsf{F}(X_{[\delta+1,\zeta]}) \in \mathsf{ZS}_n$ and $\|Z_1\|_\infty < b$. By the definition of $\mathsf{ME}(b, \mathcal{L})$ (Definition 18),

$$\left(\mathsf{s};\ c_i,\ X_i,\ r,\ \{y_{(i,j)}\}_{j \in [t]};\ Z_i\right)_{i=2}^{k} \in \mathsf{ME}(b, \mathcal{L})^{k-1}$$

guarantees that for all $i \in [2,k]$, we have both $\|Z_i\|_\infty < b$ and $y_{(i,j)} = Z_i M_j^\intercal \widehat{r}$. Thus, in total, we have by the contrapositive of Lemma 11 that $\mathrm{T} = \sum_{x \in \{0,1\}^\zeta} Q(x, \boldsymbol{A}, \boldsymbol{B}, C)$, where $\mathrm{T} := C^k \sum_{j=1,i=2}^{t,k} C^{i+(j-1)k-1} \cdot \widetilde{y}_{(i,j)}(\boldsymbol{A})$. Thus, since the sum holds for indeterminate variables $\boldsymbol{A}, \boldsymbol{B}, C$, they must hold for the verifier challenges α, β, γ. Therefore, the verifier will not reject during the honest execution of the sum-check protocol, since the Q does sum to T.

Now, we will argue that the verifier's evaluation check passes. Consider an arbitrary $j \in [t]$. By construction (Item 3), $\widetilde{y}'_{(1,j)} = Z_1 M_j^\intercal \widehat{r}'$, where $Z_1 = \mathsf{Decomp}_b(z_1)$. By the definition of Decomp_b, for all $\ell \in [d]$, we have $y'_{(1,j),\ell} = Z_1^{(\ell)} M_j^\intercal \widehat{r}'$, where $Z_1^{(\ell)}$ is the ℓ-th row of Z_1. Therefore, we have

$$\begin{aligned} m_j = \textstyle\sum_{\ell \in [d]} b^{\ell-1} y'_{(1,j),\ell} &= \textstyle\sum_{\ell \in [d]} b^{\ell-1} Z_1^{(\ell)} M_j \widehat{r}' \\ \textstyle\sum_{\ell \in [d]} b^{\ell-1} y'_{(1,j),\ell} &= \left(\textstyle\sum_{\ell \in [d]} b^{\ell-1} Z_1^{(\ell)}\right) M_j^\intercal \widehat{r}' \\ \textstyle\sum_{\ell \in [d]} b^{\ell-1} y'_{(1,j),\ell} &= z \cdot M_j^\intercal \widehat{r}' = \widetilde{M_j z}(r') \end{aligned}$$

Therefore, the value $F = \mathsf{F}(r') = f(m_1, \ldots, m_t)$, since we considered an arbitrary $j \in [t]$. Since we required $M_1 = I_n$, we must have for all $i \in [k]$, $\widetilde{y'}_{(i,1)}(\alpha') = \widetilde{Z_i M_1^\intercal}(\alpha', r') = \widetilde{Z_i}(\alpha', r')$. Therefore, the value $N_i = \mathsf{NC}(\alpha', \gamma')$ for all $i \in [k]$. By construction (Item 3), we can trivially observe that $E_{(i,j)} = \mathsf{Eval}_{(i,j)}(\alpha', r')$. All together, the verifier's check in Item 4 must pass by construction, since the values F, N_i for all $i \in [k]$, and $E_{(i,j)}$ for all $i \in [k]$, $j \in [t]$, exactly match their corresponding evaluations in the definition of $Q(\alpha', r')$.

Now, we will argue the output tuple does belong to $\mathsf{ME}(b, \mathcal{L})^k$. The condition that for all $i \in [2, k]$, $c_i = \mathcal{L}(Z_i)$ and $X_i = \mathcal{L}_x(Z_i)$ is trivially guaranteed by the input tuples belonging to $\mathsf{ME}(b, \mathcal{L})^{k-1}$ (Definition 18). By the construction of $X_1 \leftarrow \mathsf{Decomp}_b(x_1)$ and the first tuple belonging to MCS (Definition 17), we have $c_1 = \mathcal{L}(Z_1)$ and $X_1 = \mathcal{L}_x(Z_1)$. Finally, by construction in step Item 3, we exactly have that $y'_{(i,j)} = Z_i M_j^\intercal \widehat{r}'$ for all $i \in [k]$ and $j \in [t]$. All together, we have that all of the conditions are satisfied for the output tuples to belong to $\mathsf{ME}(b, \mathcal{L})^k$.

Public coin. The sum-check protocol itself is a public-coin protocol. The remaining randomness from the verifier are the challenges $\alpha \in \mathbb{K}^{\log d}$, $\beta \in \mathbb{K}^{\log(dn)}$, $\gamma \in \mathbb{K}$, which are sampled uniformly at random. □

B.5 Proof of Lemma 5

Completeness: Since $(\mathsf{s};\ c_i,\ X_i,\ r,\ \{y_{(i,j)}\}_{j\in[t]};\ Z_i)_{i\in[k+1]} \in \mathsf{ME}(b, \mathcal{L})^{k+1}$, by Corollary 1, we must have $c = \mathcal{L}(Z)$ and $\forall j \in [t]$, $y_j = Z M_j^\intercal \widehat{r}$. Furthermore, we must have

$$\|Z\|_\infty = \left\| \sum_{i=1}^{k+1} \rho_i \cdot Z_i \right\|_\infty \leq \sum_{i=1}^{k+1} \|\rho_i \cdot Z_i\|_\infty \leq \sum_{i=1}^{k+1} T \cdot \|Z_i\|_\infty \leq (k+1)T(b-1) < B,$$

where the second inequality is from the expansion factor of $\mathcal{C}$ being T, the third inequality is from the definition of $\mathsf{ME}(b, \mathcal{L})^{k+1}$, which enforces a norm bound of b, and the last inequality is by assumption (Section 4.3). Hence, we must have that the output tuple $\big(\mathsf{s};\ c,\ X,\ r,\ \{y_j\}_{j\in[t]};\ Z\big) \in \mathsf{ME}(B, \mathcal{L})$.

Public coin. The only randomness from the verifier is the challenges $\rho_1, \ldots, \rho_{k+1}$, which are sampled uniformly at random from $\mathcal{C}$.

B.6 Proof of Theorem 4

Completeness: First, we show that the verifier's checks pass. By the definition of $\mathsf{ME}(B, \mathcal{L})$, we must have that $\|Z\|_\infty < B$. Thus, by definition of split_b, we must have $Z = \sum_{i=1}^{k} b^{i-1} \cdot Z_i$. Therefore, we must have

$$\begin{aligned} Z &= \textstyle\sum_{i=1}^{k} b^{i-1} \cdot Z_i \\ Z &= \textstyle\sum_{i=1}^{k} \overline{b}^{\,i-1} \cdot Z_i \\ \mathcal{L}(Z) &= \mathcal{L}(\textstyle\sum_{i=1}^{k} \overline{b}^{\,i-1} \cdot Z_i), \end{aligned} \tag{26}$$

$$c = \sum_{i=1}^{k} \bar{b}^{i-1} \cdot \mathcal{L}(Z_i), \tag{27}$$

$$c = \sum_{i=1}^{k} \bar{b}^{i-1} \cdot c_i, \tag{28}$$

Since $Z_i \in \mathbb{F}^{d \times m}$, (26) follows from $\bar{b} \in \mathbb{F}^{d \times d}$ being a scalar matrix. Since all scalar matrices are contained in $\mathcal{S}$, (27) follows directly from $\mathcal{L}$ being a $\mathcal{S}$-module homomorphism. (28) follows by construction, in step 1, $c_i \leftarrow \mathcal{L}(Z_i)$. Starting from equation (26), we must have for all $j \in [t]$,

$$\begin{aligned} Z &= \textstyle\sum_{i=1}^{k} \bar{b}^{i-1} \cdot Z_i \\ Z \cdot M_j^{\intercal} \widehat{r} &= (\textstyle\sum_{i=1}^{k} \bar{b}^{i-1} \cdot Z_i) \cdot M_j^{\intercal} \widehat{r} \\ y_j &= \textstyle\sum_{i=1}^{k} \left(\bar{b}^{i-1} \cdot Z_i M_j^{\intercal} \widehat{r} \right) \qquad (29) \\ y_j &= \textstyle\sum_{i=1}^{k} \bar{b}^{i-1} \cdot y_{(i,j)} \qquad (30) \end{aligned}$$

(29) follows from the definition of $\mathsf{ME}(B, \mathcal{L})$ and distribution. (30) follows by construction, in step 1, $y_{(i,j)} \leftarrow Z_i M_j^{\intercal} \widehat{r}$. Thus, by (28) and (30), we have the verifier's checks must pass.

Next, we show that the output tuple, $(\mathsf{s};\ \{c_i,\ X_i,\ r,\ \{y_{(i,j)}\}_{j\in[t]}\}_{i\in[k]};\ \{Z_i\}_{i\in[k]})$, belongs to $\mathsf{ME}(b, \mathcal{L})^k$. By the definition of split_b, we must have that $\|Z_i\|_\infty < b$ for all $i \in [k]$. Since $\mathcal{L}_x$ is the trivial $\mathcal{S}$-module homomorphism which projects the first m_{in} columns, we must have that, by construction in step 2, that $X_i = \mathcal{L}_x(Z_i)$ for all $i \in [k]$. Thus, in total, we must have, along with the construction of $\big(c_i,\ \{y_{(i,j)}\}_{j\in[t]}\big)_{i\in[k]}$ in step 1, that the output tuple belongs to $\mathsf{ME}(b, \mathcal{L})^k$.

Knowledge soundness: Consider an arbitrary expected-polynomial time adversary $(\mathcal{A}, \mathcal{P}^*)$ for Π_{DEC} with success probability, $\epsilon(\mathcal{A}, \mathcal{P}^*) \geq 1/\mathsf{poly}(\lambda)$. We construct an extractor $\mathcal{E}$ for Π_{DEC} as follows,

$\mathcal{E}\big(\mathsf{pp},\ \mathsf{s},\ u_1 := \big\{c,\ X,\ r,\ \{y_j\}_{j\in[t]}\big\},\ \mathsf{st}\big)$:

1. Execute encoder $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s})$.
2. Simulate $(u_2, w_2) \leftarrow \langle \mathcal{P}^*(\mathsf{pk}, u_1, \mathsf{st}),\ \mathcal{V}(\mathsf{vk}, u_1)\rangle$.
3. If $u_2 = \perp$, output $\perp$.
4. Parse $(Z_1, \ldots, Z_k) \leftarrow w_2$.
5. Output $w_1 := \sum_{i=1}^{k} \bar{b}^{i-1} Z_i$.

Extractor runtime: The extractor runs in expected polynomial time, since it simulates only one execution between the adversary $\mathcal{P}^*$ and verifier $\mathcal{V}$, which both run in expected polynomial time.

Extractor success probability: Assume that the simulated adversary $(\mathcal{A}, \mathcal{P}^*)$ succeeds in convincing the verifier $\mathcal{V}$ and the parties jointly output $(\mathsf{s}, u_2, w_2) \in \mathsf{ME}(b, \mathcal{L})^k$; note that this occurs with probability $\epsilon(\mathcal{A}, \mathcal{P}^*)$. Define $(c_i,\ X_i,\ r,$

$\{y_{(i,j)}\}_{j\in[t]})_{i\in[k]} := u_2$ and $Z_1, \ldots, Z_k := w_2$. By the definition of $\mathsf{ME}(b,\mathcal{L})^k$, we have for all $i \in [k]$ and $j \in [t]$,

$$c_i := \mathcal{L}(Z_i), \qquad X_i := \mathcal{L}_x(Z_i), \qquad \|Z_i\|_\infty < b \qquad \text{and} \qquad y_{(i,j)} := Z_i M_j^\intercal \widehat{r} \tag{31}$$

Since the adversary succeeds in convincing the verifier, we must have that for all $j \in [t]$,

$$c = \textstyle\sum_{i=1}^k \bar{b}^{i-1} \cdot c_i \qquad \text{and} \qquad y_j = \textstyle\sum_{i=1}^k \bar{b}^{i-1} \cdot y_{(i,j)} \tag{32}$$

By construction in step 2 (i.e. definition of split_b), we also must have $X = \sum_{i=1}^k \bar{b}^{i-1} \cdot X_i$. By defining $Z := \sum_{i=1}^k \bar{b}^{i-1} Z_i$, observe that $X = \sum_{i=1}^k \bar{b}^{i-1} \cdot X_i$, (31), and (32) satisfy the remaining conditions stated in Corollary 1. Since $\mathcal{S}$ contains all scalar matrices, by Corollary 1, we must have $c = \mathcal{L}(Z)$, $X = \mathcal{L}_x(Z)$, and for all $j \in [t]$, $y_j = Z M_j^\intercal \widehat{r}$. Since in (31), we have for all $i \in [k]$, $\|Z_i\|_\infty < b$, we must also have $\|Z\|_\infty < B = b^k$. These are exactly the conditions for $(\mathsf{s};\ u_1 := \{c,\ X,\ r,\ \{y_j\}_{j\in[t]}\};\ w_1 := Z)$ to belong to $\mathsf{ME}(B,\mathcal{L})$. Therefore, since the adversary succeeds with probability $\epsilon(\mathcal{A},\mathcal{P}^*)$, we must have by construction, that $\mathcal{E}$ outputs a satisfying witness such that $(\mathsf{s}, u_1, w_1) \in \mathsf{ME}(B,\mathcal{L})$ with probability $\epsilon(\mathcal{A},\mathcal{P}^*)$.

Public coin: The verifier uses no randomness in this protocol. Thus, the protocol is trivially public coin.

B.7 Proof of Lemma 6

Proof. By construction, the verifier trivially outputs the same commitments $(c_i)_{i\in[k]}$ from the original instance u_1 to u_2. Hence, for repeated executions with respect to the same original instance u_1, the commitments in the output instances u_2 must be the same. □

B.8 Proof of Lemma 7

Proof. Consider an arbitrary expected polynomial-time adversary $(\mathcal{A},\mathcal{P}^*)$, such that the relaxed success probability of the adversary $\epsilon'(\mathcal{A},\mathcal{P}^*) \geq 1/\mathsf{poly}(\lambda)$ and

$$\Pr\left[\begin{array}{c} \\ w_2, w_2' \neq \bot \\ \wedge \\ w_2 \neq w_2' \\ \\ \end{array}\middle|\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda) \\ (\mathsf{s}, u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}) \\ (u_2, w_2) \leftarrow \langle \mathcal{P}^*, \mathcal{V}\rangle((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st}) \\ (u_2', w_2') \leftarrow \langle \mathcal{P}^*, \mathcal{V}\rangle((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st}) \end{array}\right] \leq \mathsf{negl}(\lambda) \tag{33}$$

then we will show that there exists an expected polynomial-time extractor $\mathcal{E}$ such that

$$\Pr\left[(\mathsf{pp}, \mathsf{s}, u_1, w_1) \in \mathcal{R}_1 \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ (\mathsf{s}, u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}) \\ w_1 \leftarrow \mathcal{E}(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st}) \end{array}\right] \geq \epsilon'(\mathcal{A},\mathcal{P}^*) - \mathsf{negl}(\lambda).$$

Namely, the following extractor $\mathcal{E}$,

$\mathcal{E}(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st}) \to w_1$:

1. $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s})$.
2. Assign $\mathsf{result} := \bot$.
3. While $\mathsf{result} = \bot$:
   - Simulate $\mathsf{result} \leftarrow \langle \mathcal{P}_1^*, \mathcal{V}_1 \rangle((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st}^*)$
   - If $\mathsf{result} \neq \bot$
     - Parse $(u_2, w_2) \leftarrow \mathsf{result}$.
     - If $(\mathsf{pp}, \mathsf{s}, u_2, w_2) \notin \mathcal{R}_2'$, then set $\mathsf{result} = \bot$.
4. Simulate $\mathsf{result}' \leftarrow \langle \mathcal{P}_1^*, \mathcal{V}_1 \rangle((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st}^*)$
5. If $\mathsf{result}' \neq \bot$:
   - Parse $(u_2', w_2') \leftarrow \mathsf{result}'$.
   - If $(\mathsf{pp}, \mathsf{s}, u_2', w_2') \notin \mathcal{R}_2'$, then set $\mathsf{result}' = \bot$.
6. If $\mathsf{result}' = \bot$, then output $\bot$.
7. Parse $(u_2, w_2) \leftarrow \mathsf{result}$ and $(u_2', w_2') \leftarrow \mathsf{result}'$.
8. If $w_2 \neq w_2'$, then output $\bot$.
9. Parse $(Z_1, \ldots, Z_k) \leftarrow w_2$.
10. Assign $z_1 \leftarrow \sum_{i=1}^{d} b^{i-1} \cdot Z_1^{(i)}$.
11. Output $w_1 := (z_1, Z_2, \ldots, Z_k)$.

Extractor runtime. We will show that the extractor $\mathcal{E}$ makes at most $1 + 1/\epsilon'(\mathcal{A}, \mathcal{P}^*)$ calls to $\mathcal{P}^*$ in expectation. Since $\epsilon'(\mathcal{A}, \mathcal{P}^*) \geq 1/\mathsf{poly}(\lambda)$, we have that the extractor makes at most a polynomial number of calls to $\mathcal{P}^*$ in expectation. Hence, since $\mathcal{K}$ and $\mathcal{V}_1$ run in $\mathsf{poly}(\lambda)$ time, we have that overall the extractor runs in expected polynomial-time.

By construction, the while loop (Item 3) terminates when the adversary $(\mathcal{A}, \mathcal{P}^*)$ succeeds. Since the relaxed success probability is $\epsilon'(\mathcal{A}, \mathcal{P}^*)$, the while loop executes $1/\epsilon'(\mathcal{A}, \mathcal{P}^*)$ times in expectation. This implies the while loop performs $1/\epsilon'(\mathcal{A}, \mathcal{P}^*)$ calls to $\mathcal{P}^*$ in expectation. Finally, Item 4 performs one call to $\mathcal{P}^*$.

Extractor success probability. First, we will show

$$\Pr\left[\begin{array}{c|l} \mathsf{result}' \neq \bot & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ \wedge & (\mathsf{s}, u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ w_2 = w_2' & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}) \\ & w_1 \leftarrow \mathcal{E}(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st}) \end{array}\right] \geq \epsilon'(\mathcal{A}, \mathcal{P}^*) - \mathsf{negl}(\lambda). \tag{34}$$

Item 5 exactly checks that the simulated adversary in Item 4 succeeds. Thus, the event that $\mathsf{result}' \neq \bot$ occurs with probability $\epsilon'(\mathcal{A}, \mathcal{P}^*)$. Assume that the event $\mathsf{result}' \neq \bot$ occurs. By (33), $w_2 \neq w_2'$ with at most $\mathsf{negl}(\lambda)$ probability. Thus, all together, we have (34) holds.

Assume that the event $\mathsf{result}' \neq \bot \wedge w_2 = w_2'$ occurs, which implies the extractor outputs a witness $w_1 := (z_1, Z_2, \ldots, Z_k) \neq \bot$ (as the extractor passes the checks in Item 6 and Item 8). We will show that $(\mathsf{pp}, \mathsf{s}, u_1, w_1) \notin \mathcal{R}_1$ with probabilty at

most $\mathsf{negl}(\lambda)$. Hence,

$$\Pr\left[(\mathsf{pp},\mathsf{s},u_1,w_1)\in\mathcal{R}_1 \;\middle|\; \begin{array}{l}\mathsf{pp}\leftarrow\mathcal{G}(1^\lambda,\mathsf{sz})\\ (\mathsf{s},u_1,\mathsf{st})\leftarrow\mathcal{A}(\mathsf{pp})\\ (\mathsf{pk},\mathsf{vk})\leftarrow\mathcal{K}(\mathsf{pp},\mathsf{s})\\ w_1\leftarrow\mathcal{E}(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})\end{array}\right]$$

$\geq \big(\epsilon'(\mathcal{A},\mathcal{P}^*)-\mathsf{negl}(\lambda)\big)-\mathsf{negl}(\lambda)=\epsilon'(\mathcal{A},\mathcal{P}^*)-\mathsf{negl}(\lambda)$.

Since $\mathsf{result}'\neq\perp$, by construction (Item 5), we must have $(\mathsf{pp},\mathsf{s},u_2',w_2')\in\mathcal{R}_2'$ and during the simulation $\langle\mathcal{P}_1^*,\mathcal{V}_1\rangle((\mathsf{pk},\mathsf{vk}),u_1,\mathsf{st}^*)$ in Item 4, the verifier $\mathcal{V}_1$ did not abort. Namely, that the sum-check verifier in protocol (Item 2) did not abort and the evaluation checks Item 4 were satisfied. By definition of $\mathcal{R}_2'=\mathsf{ME}(q/2,\mathcal{L})^k$, we must know that for all $i\in[k]$, $X_i:=\mathcal{L}_x(Z_i)$ and $c_i:=\mathcal{L}(Z_i)$. This also implies, by construction (Item 2 and Item 10), that $X_1:=\mathsf{Decomp}_b(x_1)$ and $c_1=\mathcal{L}(\mathsf{Decomp}_b(z_1))$.

Assume that $(\mathsf{pp},\mathsf{s},u_1,w_1)\notin\mathcal{R}_1$. Recall that $\mathcal{R}_1:=\mathsf{MCS}(b,\mathcal{L})\times\mathsf{ME}(b,\mathcal{L})^{k-1}$. Since the commitments agree with the witnesses, we must have that $(\mathsf{pp},\mathsf{s},u_1,w_1)\notin\mathcal{R}_1$ implies that either (using notation from Lemma 11)

1. $\mathsf{F}(X_{[\delta+1,\zeta]})\notin\mathsf{ZS}_n$ (identically, $\mathsf{F}(X_{[1,n]})\notin\mathsf{ZS}_n$),
2. OR there exists an $i\in[k]$ for which $\|Z_i\|_\infty\geq b$,
3. OR there exists an $i\in[2,k]$ for which $\widetilde{y}_{(i,j)}(\boldsymbol{A})\neq\widetilde{M}_{i,j}(\boldsymbol{A},r)$ (identically, $y_{(i,j)}=Z_iM_j^\intercal\widehat{r}$)

By Lemma 11, we must have $T\neq\sum_{x\in\{0,1\}^\zeta}Q(x,\boldsymbol{A},\boldsymbol{B},C)$. By the construction of the verifier's checks in Item 4 and definition of $(\mathsf{pp},\mathsf{s},u_2',w_2')\in\mathcal{R}_2'$, we must have that the sum-check evaluation check $v=Q(\alpha',r')$ is true. Note, that the randomness used in the second simulation of the protocol (Item 4) is fresh and independent of the first simulation of the protocol (Item 3). Additionally, note that the witness from the first execution, w_2, agrees with the witness from the second execution, $w_2':=(z_1,Z_2,\ldots,Z_k)$. Thus, in order for the sum-check verifier to have passed, either the adversary $\mathcal{P}^*$

- violated the soundness of the sum-check protocol since $T\neq\sum_{x\in\{0,1\}^\zeta}Q(x,\boldsymbol{A},\boldsymbol{B},C)$
- OR the non-zero polynomial

$$\mathrm{T}-\textstyle\sum_{x\in\{0,1\}^\zeta}Q(x,\boldsymbol{A},\boldsymbol{B},C)$$

  for $\mathrm{T}:=C^k\sum_{j=1,i=2}^{t,k}C^{i+(j-1)k-1}\cdot\widetilde{y}_{(i,j)}(\boldsymbol{A})$ evaluated to zero on random point $(\alpha\in\mathbb{K}^{\log d},\beta\in\mathbb{K}^{\log(dn)},\ \gamma\in\mathbb{K})$.

By the soundness error of the sum-check protocol, the first event occurs with probability at most $\epsilon_{\mathsf{SC}}:=\max(u+1,2b+1,2)\cdot\log(dn)/|\mathbb{K}|$. By the Schwartz–Zippel lemma, the second event occurs with probability at most $\epsilon_{\mathsf{SZ}}:=\max(\log dn,(t+1)k-1)/|\mathbb{K}|$. Thus, all together, $(\mathsf{pp},\mathsf{s},u_1,w_1)\notin\mathcal{R}_1$ with probability at most $\mathsf{negl}(\lambda):=\epsilon_{\mathsf{SC}}+\epsilon_{\mathsf{SZ}}$. □

B.9 Proof of Lemma 8

Proof. Consider an arbitrary expected-polynomial time adversary $(\mathcal{A}, \mathcal{P}^*)$ for Π_{RLC} with success probability, $\epsilon(\mathcal{A}, \mathcal{P}^*) \geq 1/\mathsf{poly}(\lambda)$. First, we can construct an adversary and verification function for Theorem 6,

$A_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}(\boldsymbol{c})$:
1. Execute encoder $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s})$.
2. Simulate $(u_2, w_2) \leftarrow \langle \mathcal{P}^*(\mathsf{pk}, u_1, \mathsf{st}),\ \mathcal{V}(\mathsf{vk}, u_1)\rangle$ with verifier randomness $\boldsymbol{c}$.
3. Output w_2

$V_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}(\boldsymbol{c}, w_2) \to \{0,1\}$:
1. Execute encoder $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s})$.
2. Simulate $(u_2, \_) \leftarrow \langle \mathcal{P}^*(\mathsf{pk}, u_1, \mathsf{st}),\ \mathcal{V}(\mathsf{vk}, u_1)\rangle$ with verifier randomness $\boldsymbol{c}$.
3. Output accept if and only if $(u_2, w_2) \in \mathsf{ME}(B, \mathcal{L})$.

Let $E_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}$ be the corresponding extractor from Theorem 6. We define $E(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st})$ as the trivial algorithm that executes $E_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}$ by simulating calls to $A_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}$. We construct an extractor for adversary $(\mathcal{A}, \mathcal{P}^*)$ as follows:

$\mathcal{E}(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st})$:
1. $\mathsf{result} \leftarrow E(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st})$.
2. If $u_1 = \perp$ or $\mathsf{result} = \perp$, output $\perp$.
3. Parse $(\boldsymbol{c}, w_1'), (\boldsymbol{c}_1, w_1'), \ldots (\boldsymbol{c}_{k+1}, w_{k+1}') \leftarrow \mathsf{result}$.
4. Parse $\rho_1, \ldots, \rho_{k+1} \leftarrow \boldsymbol{c}$.
5. For $i \in [k+1]$,
   (a) Parse $\rho_1^{(i)}, \ldots, \rho_{k+1}^{(i)} \leftarrow \boldsymbol{c}_i$.
   (b) Parse $Z \leftarrow w_1'$ and $Z_i \leftarrow w_i'$.
   (c) Assign $Z_i \leftarrow \left(\rho_i - \rho_i^{(i)}\right)^{-1} \cdot (Z - Z^{(i)})$.
6. Parse $(c_i,\ X_i,\ r,\ \{y_{(i,j)}\}_{j\in[t]})_{i\in[k+1]} \leftarrow u_1$.
7. Output $w_1 := (Z_i)_{i\in[k+1]}$ if and only if $\left(\mathsf{s};\ c_i,\ X_i,\ r,\ \{y_{(i,j)}\}_{j\in[t]};\ Z_i\right)_{i\in[k+1]} \in \mathsf{ME}(q/2, \mathcal{L})^{k+1}$

Extractor runtime. By Theorem 6, we are guaranteed $E_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}$ makes in expectation at most $k+2$ calls to $A_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}$. Hence, our overall extractor $\mathcal{E}$ runs in expected polynomial time.

Extractor success probability. By Theorem 6, we are guaranteed that $E(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st})$ outputs $k+2$ pairs $(\boldsymbol{c}, w_1'), (\boldsymbol{c}_1, w_1'), \ldots (\boldsymbol{c}_{k+1}, w_{k+1}')$ such that

- $V(\boldsymbol{c}, w_1') = 1$,
- for all $i \in [k+1]$, $V(\boldsymbol{c}_i, w_i') = 1$, and
- $(\boldsymbol{c}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{k+1}) \in \mathsf{SS}(\mathcal{C}, k+1)$

with probability $\epsilon^{V_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}}(A_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}) - \frac{k+1}{|\mathcal{C}|}$. Since $A_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}$ and $V_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}$ simulate the interaction between $\mathcal{P}^*$ and $\mathcal{V}$ and checks if the output pair (u_2, w_1') belongs to $\mathsf{ME}(B, \mathcal{L})$, we must have $\epsilon^{V_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}}(A_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}) - \frac{k+1}{|\mathcal{C}|} = \epsilon(\mathcal{A}, \mathcal{P}^*) -$

$(k+1)/|\mathcal{C}|$. Assume that this event occurs. Since $V(\boldsymbol{c}, w_1') = 1$, by construction of Π_{RLC}, we have

$$\left(\mathsf{s};\ \begin{array}{c} c = \sum_{i=1}^{k+1} \rho_i c_i, \\ X = \sum_{i=1}^{k+1} \rho_i X_i, \\ r, \\ \left\{ y_j = \sum_{i=1}^{k+1} \rho_i y_{(i,j)} \right\}_{j \in [t]} \end{array} ;\ Z \right) \in \mathsf{ME}(B, \mathcal{L}) \tag{35}$$

Furthermore, since $V(\boldsymbol{c}_i, w_i') = 1$, we have for all $i \in [k+1]$,

$$\left(\mathsf{s};\ \begin{array}{c} c^{(i)} = \sum_{i=1}^{k+1} \rho_i^{(i)} c_i, \\ X^{(i)} = \sum_{i=1}^{k+1} \rho_i^{(i)} X_i, \\ r, \\ \left\{ y_j^{(i)} = \sum_{i=1}^{k+1} \rho_i^{(i)} y_{(i,j)} \right\}_{j \in [t]} \end{array} ;\ Z^{(i)} \right) \in \mathsf{ME}(B, \mathcal{L}) \tag{36}$$

Since $(\boldsymbol{c}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{k+1}) \in \mathsf{SS}(\mathcal{C}, k+1)$, we must have for all $i \in [k+1]$ that

$$(\rho_1, \ldots, \rho_{k+1}) \equiv_i (\rho_1^{(i)}, \ldots, \rho_{k+1}^{(i)}) \tag{37}$$

which means the challenges differ only on index i. Thus, we have for all $i \in [k+1]$,

$$\begin{aligned} \textstyle\sum_{i=1}^{k+1} \rho_i c_i - \sum_{i=1}^{k+1} \rho_i^{(i)} c_i &= \mathcal{L}(Z) - \mathcal{L}\big(Z^{(i)}\big), \\ \textstyle\sum_{i=1}^{k+1} \rho_i X_i - \sum_{i=1}^{k+1} \rho_i^{(i)} X_i &= \mathcal{L}_x(Z) - \mathcal{L}_x\big(Z^{(i)}\big) \end{aligned} \tag{38}$$

$$\begin{aligned} \big(\rho_i - \rho_i^{(i)}\big) \cdot c_i &= \mathcal{L}(Z) - \mathcal{L}\big(Z^{(i)}\big), \\ \big(\rho_i - \rho_i^{(i)}\big) \cdot X_i &= \mathcal{L}_x(Z) - \mathcal{L}_x\big(Z^{(i)}\big) \end{aligned} \tag{39}$$

$$\begin{aligned} c_i &= \mathcal{L}\Big(\big(\rho_i - \rho_i^{(i)}\big)^{-1} \cdot \big(Z - Z^{(i)}\big)\Big), \\ X_i &= \mathcal{L}_x\Big(\big(\rho_i - \rho_i^{(i)}\big)^{-1} \cdot \big(Z - Z^{(i)}\big)\Big) \end{aligned} \tag{40}$$

$$c_i = \mathcal{L}(Z_i), \qquad X_i = \mathcal{L}_x(Z_i)$$

where equation (38) follows from the definition of $\mathsf{ME}(B, \mathcal{L})$ and both (35) and (36). Equation (39) follows from the equivalence (37). Equation (40) follows from $\mathcal{L}, \mathcal{L}_x$ being $\mathcal{S}$-homomorphisms and $\mathcal{C} \subseteq \mathcal{S}$ being a strong sampling set (Definition 14) which because $\rho_i \neq \rho_i^{(i)}$ (guaranteed by (37)) means $\rho_i - \rho_i^{(i)}$ is invertible. Similarly, we must have for all $i \in [k+1]$ and $j \in [t]$,

$$\textstyle\sum_{i=1}^{k+1} \rho_i y_{(i,j)} - \sum_{i=1}^{k+1} \rho_i^{(i)} y_{(i,j)} = Z M_j^\intercal \widehat{r} - Z^{(i)} M_j^\intercal \widehat{r} \tag{41}$$

$$\big(\rho_i - \rho_i^{(i)}\big) \cdot y_{(i,j)} = \big(Z - Z^{(i)}\big) M_j^\intercal \widehat{r} \tag{42}$$

$$y_{(i,j)} = \big(\rho_i - \rho_i^{(i)}\big)^{-1} \cdot \big(Z - Z^{(i)}\big) M_j^\intercal \widehat{r} = Z_i M_j^\intercal \widehat{r} \tag{43}$$

where equation (41) follows from the definition of $\mathsf{ME}(B, \mathcal{L})$ and both (35) and (36), equations (42) follows from the equivalence (37), and (43) follows from $\mathcal{C} \subseteq \mathcal{S}$ being a strong sampling set (Definition 14) which because $\rho_i \neq \rho_i^{(i)}$ (guaranteed by (37)) means $\rho_i - \rho_i^{(i)}$ is invertible. Therefore, by (40) and (43), we must have with probability $\epsilon(\mathcal{A}, \mathcal{P}^*) - (k+1)/|\mathcal{C}|$, the extractor outputs $Z_1, \ldots, Z_{k+1}$ such that $\left(\mathsf{s};\ c_i,\ X_i,\ r,\ \{y_{(i,j)}\}_{j\in[t]};\ Z_i\right)_{i\in[k+1]} \in \mathsf{ME}(q/2, \mathcal{L})^{k+1}$, which has the trivial norm bound of $q/2$.

Now, assume that $\mathcal{A} := (\mathcal{B}, \mathcal{B}')$ such that

$$\Pr\left[\begin{array}{c|l} u_1, u_1' \neq \perp & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ \Downarrow & (\mathsf{s}, \mathsf{st}^*) \leftarrow \mathcal{B}(\mathsf{pp}) \\ \phi(u_1) = \phi(u_1') & (u_1, \mathsf{st}) \leftarrow \mathcal{B}'(\mathsf{st}^*) \\ & (u_1', \mathsf{st}') \leftarrow \mathcal{B}'(\mathsf{st}^*) \end{array}\right] = 1$$

We will show that

$$\Pr\left[\begin{array}{c|l} & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ & (\mathsf{s}, \mathsf{st}^*) \leftarrow \mathcal{B}(\mathsf{pp}) \\ w_1, w_1' \neq \perp & (u_1, \mathsf{st}) \leftarrow \mathcal{B}'(\mathsf{st}^*) \\ \wedge\ w_1 \neq w_1' & w_1 \leftarrow \mathcal{E}(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st}) \\ & (u_1', \mathsf{st}') \leftarrow \mathcal{B}'(\mathsf{st}^*) \\ & w_1' \leftarrow \mathcal{E}(\mathsf{pp}, \mathsf{s}, u_1', \mathsf{st}') \end{array}\right] \leq \epsilon_{\mathsf{bind}}(d, m, 2B, \mathcal{C})$$

Assume that the event $w_1, w_1' \neq \perp\ \wedge\ w_1 \neq w_1'$ occurs. Since $w_1, w_1' \neq \perp$, we have that $u_1, u_1' \neq \perp$, since the extractor would have failed otherwise. Thus, we must have that:

1. $\phi(u_1) = \phi(u_1')$, which guarantees the instances share identical commitments $(c_i)_{i\in[j]}$.
2. Define $(f_i)_{i\in[k+1]} = w_1$ and $(f_i')_{i\in[k+1]} = w_1'$. Then, $w_1 \neq w_1'$ implies that there exist an $i \in [k+1]$ such that $f_i \neq f_i'$.

During the execution of $\mathcal{E}(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st})$, the call to algorithm $E(\mathsf{pp}, \mathsf{s}, u_1, \mathsf{st})$ produces $\rho_i, \rho_i^{(i)}, Z, Z^{(i)}$ and $\mathcal{E}(\mathsf{pp}, \mathsf{s}', u_1', \mathsf{st}')$, the call to algorithm $E(\mathsf{pp}, \mathsf{s}', u_1', \mathsf{st}')$ produces $\rho_i', \rho_i^{(i)\prime}, Z', Z^{(i)\prime}$ such that

$$f_i \neq f_i' \iff \left(\rho_i - \rho_i^{(i)}\right)^{-1} \cdot \left(Z - Z^{(i)}\right) \neq \left(\rho_i' - \rho_i^{(i)\prime}\right)^{-1} \cdot \left(Z' - Z^{(i)\prime}\right) \tag{44}$$

$$\|Z\|_\infty, \left\|Z^{(i)}\right\|_\infty, \|Z'\|_\infty, \left\|Z^{(i)\prime}\right\|_\infty < B \tag{45}$$

where (44) follows from Item 2 and (45) follows from Theorem 6, which guarantees the verification function $V_{(\mathsf{pp},\mathsf{s},u_1,\mathsf{st})}$ accepts (by construction, this verification function checks if the output tuples belong to $\mathsf{ME}(B, \mathcal{L})$, which checks the output witnesses have norm bound B). By Item 1 and (40), we must have

$$c_i = \mathcal{L}\Big(\left(\rho_i - \rho_i^{(i)}\right)^{-1} \cdot \left(Z - Z^{(i)}\right)\Big) = \mathcal{L}\Big(\left(\rho_i' - \rho_i^{(i)\prime}\right)^{-1} \cdot \left(Z' - Z^{(i)\prime}\right)\Big)$$

Thus, since $\mathcal{L}$ is a $\mathcal{S}$-homomorphism, we have

$$\left(\rho_i - \rho_i^{(i)}\right) \cdot c_i = \mathcal{L}\left(Z - Z^{(i)}\right) \ \wedge \ \left(\rho_i' - \rho_i^{(i)\prime}\right) \cdot c_i = \mathcal{L}\left(Z' - Z^{(i)\prime}\right) \tag{46}$$

All together, by (44), (45), and (46), we have that $(c_i,\ \Delta_1 = \rho_i - \rho_i^{(i)},\ \Delta_2 = \rho_i' - \rho_i^{(i)\prime},\ Z_1 = Z - Z^{(i)},\ Z_2 = Z' - Z^{(i)\prime})$ is a $2B$-relaxed binding collision. Thus, the probability of the original event must be less than or equal to $\epsilon_{\mathsf{bind}}(d, m, 2B, \mathcal{C})$. Otherwise, we could construct a corresponding relaxed-binding adversary which executes the extractor $\mathcal{E}$ twice to retrieve the corresponding elements for the $2B$-relaxed binding collision with probability greater than $\epsilon_{\mathsf{bind}}(d, m, 2B, \mathcal{C})$. $\square$

B.10 Proof of Theorem 5

Proof. Consider an arbitrary expected polynomial-time adversary $(\mathcal{A}, \mathcal{P}^*)$ for the composition $\Pi := \Pi_2 \circ \Pi_1$ with success probability $\epsilon(\mathcal{A}, \mathcal{P}^*) \geq 1/\mathsf{poly}(\lambda)$. Without loss of generality, the adversary $\mathcal{P}^*$ can be split into two adversaries $(\mathcal{P}_1^*, \mathcal{P}_2^*)$ such that given $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$, $(\mathsf{s}, u_1, \mathsf{st}_1) \leftarrow \mathcal{A}(\mathsf{pp})$, and $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s})$,

- $\langle \mathcal{P}_1^*, \mathcal{V}_1 \rangle((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st}_1) \to (u_2, \mathsf{st}_2)$
- $\langle \mathcal{P}_2^*, \mathcal{V}_2 \rangle((\mathsf{pk}, \mathsf{vk}), u_2, \mathsf{st}_2) \to (u_3, w_3)$

Furthermore, we assume that $\mathcal{A}$ outputs st_1 which contains $(\mathsf{s}, \mathsf{pp})$; otherwise, we could trivially construct an adversary $\mathcal{A}'$ with an identical distribution of prior outputs that does so. First, we construct an adversary $\mathcal{A}_2 := (\mathcal{B}_2, \mathcal{B}_2')$ for Π_2:

$\mathcal{B}_2(\mathsf{pp}) \to (\mathsf{s}, \mathsf{st}^*)$:
1. $(\mathsf{s}, u_1, \mathsf{st}_1) \leftarrow A(\mathsf{pp})$.
2. Output $(\mathsf{s}, \mathsf{st}_1)$.

$\mathcal{B}_2'(\mathsf{st}^*) \to (u_2, \mathsf{st}_2)$:
1. Parse st^* to obtain $(\mathsf{s}, \mathsf{pp})$.
2. $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s})$.
3. Simulate $(u_2, \mathsf{st}_2) \leftarrow \langle \mathcal{P}_1^*, \mathcal{V}_1 \rangle((\mathsf{pk}, \mathsf{vk}), u_1, \mathsf{st}^*)$.
4. Output (u_2, st_2).

$A_2(\mathsf{pp}) \to (\mathsf{s}, u_2, \mathsf{st}_2)$:
1. $(\mathsf{s}, \mathsf{st}^*) \leftarrow \mathcal{B}_2(\mathsf{pp})$.
2. $(u_2, \mathsf{st}_2) \leftarrow \mathcal{B}_2'(\mathsf{st}^*)$.
3. Output $(\mathsf{s}, u_2, \mathsf{st}_2)$.

Observe that, by construction, the success probability $\epsilon(\mathcal{A}_2, \mathcal{P}_2^*)$ of adversary $(\mathcal{A}_2, \mathcal{P}_2^*)$ for Π_2 is equal to the success probabilty $\epsilon(\mathcal{A}, \mathcal{P}^*)$ of adversary $(\mathcal{A}, \mathcal{P}^*)$ for Π. Since Π_1 is ϕ-restricted, we must have

$$\Pr\left[\begin{array}{c} u_2, u_2' \neq \perp \\ \Downarrow \\ \phi(u_2) = \phi(u_2') \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, \mathsf{sz}) \\ (\mathsf{s}, \mathsf{st}^*) \leftarrow \mathcal{B}_2(\mathsf{pp}) \\ (u_2, \mathsf{st}_2) \leftarrow \mathcal{B}_2'(\mathsf{st}^*) \\ (u_2', \mathsf{st}_2') \leftarrow \mathcal{B}_2'(\mathsf{st}^*) \end{array}\right] = 1, \tag{47}$$

Thus, we have by (47) and the ϕ-relaxed knowledge soundness of Π_2 that there exists an expected polynomial-time extractor $\mathcal{E}_2$ such that

$$\Pr\left[(\mathsf{pp},\mathsf{s},u_2,w_2)\in\mathcal{R}_2' \;\middle|\; \begin{array}{l}\mathsf{pp}\leftarrow\mathcal{G}(1^\lambda,\mathsf{sz})\\ (\mathsf{s},u_2,\mathsf{st}_2)\leftarrow\mathcal{A}_2(\mathsf{pp})\\ (\mathsf{pk},\mathsf{vk})\leftarrow\mathcal{K}(\mathsf{pp},\mathsf{s})\\ w_2\leftarrow\mathcal{E}_2(\mathsf{pp},\mathsf{s},u_2,\mathsf{st}_2)\end{array}\right]\geq\epsilon(\mathcal{A},\mathcal{P}^*)-\mathsf{negl}(\lambda) \tag{48}$$

and

$$\Pr\left[\begin{array}{c}w_2,w_2'\neq\bot\\ \wedge\ w_2\neq w_2'\end{array} \;\middle|\; \begin{array}{l}\mathsf{pp}\leftarrow\mathcal{G}(1^\lambda,\mathsf{sz})\\ (\mathsf{s},\mathsf{st}^*)\leftarrow\mathcal{B}_2(\mathsf{pp})\\ (u_2,\mathsf{st})\leftarrow\mathcal{B}_2'(\mathsf{st}^*)\\ w_2\leftarrow\mathcal{E}_2(\mathsf{pp},\mathsf{s},u_2,\mathsf{st})\\ (u_2',\mathsf{st}')\leftarrow\mathcal{B}_2'(\mathsf{st}^*)\\ w_2'\leftarrow\mathcal{E}_2(\mathsf{pp},\mathsf{s},u_2',\mathsf{st}')\end{array}\right]\leq\mathsf{negl}(\lambda) \tag{49}$$

Next, we will construct an adversary $\mathcal{P}_1^{**}$ for Π_1:

$\mathcal{P}_1^{**}(\mathsf{pk},u_1,\mathsf{st}_1)\to w_2$:
1. Parse st_1 to obtain (s,pp).
2. $(\mathsf{pk},\mathsf{vk})\leftarrow\mathcal{K}(\mathsf{pp},\mathsf{s})$.
3. Simulate $(u_2,\mathsf{st}_2)\leftarrow\langle\mathcal{P}_1^*,\mathcal{V}_1\rangle((\mathsf{pk},\mathsf{vk}),u_1,\mathsf{st}_1)$.
4. $w_2\leftarrow\mathcal{E}_2(\mathsf{pp},\mathsf{s},u_2,\mathsf{st}_2)$.
5. Output w_2.

Observe that, by construction, the relaxed success probability $\epsilon'(\mathcal{A},\mathcal{P}_1^{**})$ of adversary $(\mathcal{A},\mathcal{P}_1^{**})$ for Π_1 is equal to $\epsilon(\mathcal{A},\mathcal{P}^*)-\mathsf{negl}(\lambda)\geq 1/\mathsf{poly}(\lambda)$ which is the success probability of the relaxed extractor $\mathcal{E}_2$ from equation (48). Furthermore, by equation (49) and construction of $(\mathcal{B}_2,\mathcal{B}_2')$, we must have that

$$\Pr\left[\begin{array}{c}w_2,w_2'\neq\bot\\ \wedge\\ w_2\neq w_2'\end{array} \;\middle|\; \begin{array}{l}\mathsf{pp}\leftarrow\mathsf{Gen}(1^\lambda)\\ (\mathsf{s},u_1,\mathsf{st}_1)\leftarrow\mathcal{A}(\mathsf{pp})\\ (\mathsf{pk},\mathsf{vk})\leftarrow\mathcal{K}(\mathsf{pp},\mathsf{s})\\ (u_2,w_2)\leftarrow\langle\mathcal{P}_1^{**},\mathcal{V}\rangle((\mathsf{pk},\mathsf{vk}),u_1,\mathsf{st}_1)\\ (u_2',w_2')\leftarrow\langle\mathcal{P}_1^{**},\mathcal{V}\rangle((\mathsf{pk},\mathsf{vk}),u_1,\mathsf{st}_1)\end{array}\right]\leq\mathsf{negl}(\lambda) \tag{50}$$

Thus, we have by (50) and the restricted knowledge soundness of Π_1 that there exists an expected polynomial-time extractor $\mathcal{E}_1$ such that

$$\Pr\left[(\mathsf{pp},\mathsf{s},u_1,w_1)\in\mathcal{R}_1 \;\middle|\; \begin{array}{l}\mathsf{pp}\leftarrow\mathcal{G}(1^\lambda,\mathsf{sz})\\ (\mathsf{s},u_1,\mathsf{st}_1)\leftarrow\mathcal{A}(\mathsf{pp})\\ (\mathsf{pk},\mathsf{vk})\leftarrow\mathcal{K}(\mathsf{pp},\mathsf{s})\\ w_1\leftarrow\mathcal{E}_1(\mathsf{pp},\mathsf{s},u_1,\mathsf{st}_1)\end{array}\right]\geq\epsilon(\mathcal{A},\mathcal{P}^*)-\mathsf{negl}(\lambda) \tag{51}$$

In conclusion, we have constructed an extractor $\mathcal{E}:=\mathcal{E}_1$ with respect to adversary $(\mathcal{A},\mathcal{P}^*)$ such that

$$\Pr\left[(\mathsf{pp},\mathsf{s},u_1,w_1)\in\mathcal{R}_1 \;\middle|\; \begin{array}{l}\mathsf{pp}\leftarrow\mathcal{G}(1^\lambda,\mathsf{sz})\\ (\mathsf{s},u_1,\mathsf{st}_1)\leftarrow\mathcal{A}(\mathsf{pp})\\ (\mathsf{pk},\mathsf{vk})\leftarrow\mathcal{K}(\mathsf{pp},\mathsf{s})\\ w_1\leftarrow\mathcal{E}(\mathsf{pp},\mathsf{s},u_1,\mathsf{st}_1)\end{array}\right]\geq\epsilon(\mathcal{A},\mathcal{P}^*)-\mathsf{negl}(\lambda).$$

Thus, $\Pi := \Pi_2 \circ \Pi_1$ is knowledge sound. $\square$

B.11 Finding choices of cyclotomic and fields

```
# [LS18, eprint 2017-523] pg 6
# m is the cyclotomic polynomial index
def tau(m):
    return m if (m % 2) != 0 else m / 2

# [LS18, eprint 2017-523] Thm 1.1, pg 4
# m is the cyclotomic polynomial index
# p is the prime
# z is any divisor of m
# This tests for the condition for thm 1.1 to hold
def thm1_1_cond(m, p, z):
    cond1 = (p % z) == 1
    cond2 = Mod(p,m).multiplicative_order() == m/z
    return cond1 and cond2

# [LS18, eprint 2017-523] Thm 1.1, pg 4
# p is the prime
# z is any divisor of m
# lInf bound for elements to be invertible
# given that m,p,z satisfy thm 1.1 cond
def thm1_1_inv_bound(p, z):
    return (1/s1(z)*p^(1/euler_phi(z))).n()

def thm1_1_num_factors(z):
    return euler_phi(z)

# Output divisors of m
def divisors(m):
    zs = list()
    for i in range(1,m+1):
        if m % i == 0:
            zs.append(i)
    return zs

# [LS18, eprint 2017-523] pg 6, pg 9
# We only consider prime power cyclotomics
# m is the cyclotomic polynomial index
def s1(m):
    return sqrt(tau(m))

# checks if cyclotomic index m is power of two
def is_pow2(m):
    return sum(m.digits(2)) == 1

# [MR09] lattice-based cryptography
```

```
# makes sure characteristic does not lead
# to trivial bound
def non_trivial(q, n, d, delta):
    return (q/2).n() >= (2^(2 * sqrt( n*d * log(q,2) * log(delta, 2)))).n()

# [AL21] eprint Prop 2. 2021/202
# for all u,v in R, |u*v| / |v| <= gamma*|u|
# outputs T = gamma * |u|
# assumes we are only testing prime powers
def expansion_factor(m, norm):
    if is_pow2(m):
        return euler_phi(m) * norm
    else:
        return 2 * euler_phi(m) * norm

# p is prime
# max_idx is max cyclotomic index
# outputs list of (m, z)
def candidates(p, min_idx=10, max_idx=200):
    # prime powers
    possible_indices = [i for i in range(min_idx, max_idx) if len(factor(i)) == 1]
    c = list()
    for m in possible_indices:
        zs = divisors(m)
        for z in zs:
            if thm1_1_cond(m, p, z):
                c.append((Integer(m), Integer(z)))
    return c

def pre_filter(q, cyclotomic_index, z, n, m, chals):
    chals_norm = max({abs(c) for c in chals})
    chals_max_diff = chals[-1] - chals[0]
    delta = 1.0045 # root hermite factor, chosen from [ESSLL19] eprint 2018/773
    phi = cyclotomic_polynomial(cyclotomic_index) # index cyclotomic polynomial
    d = phi.degree() # degree of cyclotomic

    # return non_trivial(q, n, d, delta) and chals_max_diff < thm1_1_inv_bound(q, z) and log(len(chals
    # We remove non_trivial(...) because we use the lattice estimator for hardness
    return chals_max_diff < thm1_1_inv_bound(q, z) and log(len(chals)^d,2).n() >= 120

def info(q, cyclotomic_index, z, n, m, chals):
    chals_norm = max({abs(c) for c in chals})
    chals_max_diff = chals[-1] - chals[0]
    delta = 1.0045 # root hermite factor, chosen from [ESSLL19] eprint 2018/773
    phi = cyclotomic_polynomial(cyclotomic_index) # index cyclotomic polynomial
    d = phi.degree() # degree of cyclotomic
    T = expansion_factor(cyclotomic_index, chals_norm)

    # Bounds for MSIS to be hard
    # [MR09] lattice-based cryptography pg 6
```

```
    # [CMNW24] pg 38 eprint 2024/281
    MSIS_B_l2_bound = min(q, 2^(2 * sqrt( n*d * log(q,2) * log(delta, 2))))
    MSIS_B_linf_bound = MSIS_B_l2_bound / sqrt(m*d)

    # We need MSIS infinity bound 8TB to be hard
    B = MSIS_B_linf_bound / (8*T)

    print("####")
    print("Cyclotomic idx:", cyclotomic_index)
    print("Cyclotomic Poly:", phi)
    print("z:", z)
    #print("Prime is non-trivial?", non_trivial(q, n, d, delta))
    print("Csmall norm is small enough?", chals_max_diff < thm1_1_inv_bound(q, z))
    print("Csmall large enough?", log(len(chals)^d,2).n() >= 120)
    print("Degree of Cyclotomic:", d)
    # print("log(B):", log(B, 2).n())
    print("Expansion Factor T:", T)
    print("Invertible Norm bound:", thm1_1_inv_bound(q, z))
    print("log(|C_Small|):", log(len(chals)^d,2).n())
    print("Factors of Cyclotomic:", thm1_1_num_factors(z))
    print()

def possible_settings(q, n, m, chals):
    for (cyclotomic_index, z) in candidates(q):
        if pre_filter(q, cyclotomic_index, z, n, m, chals):
            info(q, cyclotomic_index, z, n, m, chals)
        else:
            delta = 1.0045
            d = cyclotomic_polynomial(cyclotomic_index).degree()
            print("[Does not satisfy security requirements] index: {}, degree: {}, z: {}, non_trivial

# Primes:
GL = 2^64 - 2^32 + 1
AGL = GL - 32
print("################################")
print("AGL ################################")
print("################################")
# MSIS settings
n = 13 # rows, kappa in latticefold
m = 2^26 # cols
# Small Challenge set
chals = [-1, 0, 1, 2]
possible_settings(AGL, n, m, chals)
print("################################")
print("M61 ################################")
print("################################")
# MSIS settings
n = 16 # rows, kappa in latticefold
m = 2^22 # cols
# Small Challenge set
```

```
chals = [-2, -1, 0, 1, 2]
possible_settings(2^61-1, n, m, chals)
print("################################")
print("GL ################################")
print("################################")
# MSIS settings
n = 16 # rows, kappa in latticefold
m = 2^24 # cols
# Small Challenge set
chals = [-2, -1, 0, 1, 2]
possible_settings(GL, n, m, chals)
print("################################")
```

B.12 Lattice Estimator Script

```
from estimator import *
Logging.set_level(Logging.LEVEL0)

M61 = 2^61 -1
GL = 2^64 - 2^32 +1
AGL = GL - 32

n = 13
d = 64
T = 128
k = 11
b = 2
B = b^k
m = 2^26
q = AGL

n_sis = n*d
m_sis = m*d
B_l2 = sqrt(m*d)*(8*T*B)

params = SIS.Parameters(n=n_sis, q=q, m=m_sis,length_bound=B_l2, norm=2)
_ = SIS.estimate(params)
print((k+1)*T*(b-1) < B)

n = 16
d = 54
T = 216
k = 12
b = 2
B = b^k
m = 2^22
q = M61

n_sis = n*d
```

```
m_sis = m*d
B_l2 = sqrt(m*d)*(8*T*B)

params = SIS.Parameters(n=n_sis, q=q, m=m_sis,length_bound=B_l2, norm=2)
_ = SIS.estimate(params)
print((k+1)*T*(b-1) < B)

n = 16
d = 54
T = 216
k = 12
b = 2
B = b^k
m = 2^24
q = GL

n_sis = n*d
m_sis = m*d
B_l2 = sqrt(m*d)*(8*T*B)

params = SIS.Parameters(n=n_sis, q=q, m=m_sis,length_bound=B_l2, norm=2)
_ = SIS.estimate(params)
print((k+1)*T*(b-1) < B)
```