# Algebraic Reductions of Knowledge

Abhiram Kothapalli
Carnegie Mellon University
akothapalli@cmu.edu

Bryan Parno
Carnegie Mellon University
parno@cmu.edu

**Abstract**

We introduce *reductions of knowledge*, a generalization of arguments of knowledge, which reduce checking knowledge of a witness in one relation to checking knowledge of a witness in another (simpler) relation. Reductions of knowledge unify a growing class of modern techniques as well as provide a compositional framework to modularly reason about individual steps in complex arguments of knowledge. As a demonstration, we simplify and unify recursive arguments over linear algebraic statements by decomposing them as a sequence of reductions of knowledge. To do so, we develop the *tensor reduction of knowledge*, which generalizes the central reductive step common to most recursive arguments. Underlying the tensor reduction of knowledge is a new information-theoretic reduction, which, for any modules $U$, $U_1$, and $U_2$ such that $U \cong U_1 \otimes U_2$, reduces the task of evaluating a homomorphism in $U$ to evaluating a homomorphism in $U_1$ and evaluating a homomorphism in $U_2$.

## Contents

## 1 Introduction

Arguments of knowledge [GMR89] are powerful cryptographic primitives that allow a verifier to efficiently check that a prover *knows* a satisfying witness for a claimed statement. Such arguments provide strong integrity and privacy guarantees that enable a large class of cryptographic applications [DLFKP16, SCG+14, KMS+16, ZKP15, TKPS21].

However, a growing body of work challenges the traditional paradigm by describing interactions in which the verifier does not fully resolve the prover's statement to true or false, but rather reduces it to a simpler statement to be checked:

- The well-studied *inner-product argument* [BCC+16] (along with subsequent optimizations [BBB+18] and generalizations [BMM+21,BCS21]) relies on recursively applying an interactive reduction from the task of checking knowledge of size $n$ vectors to the task of checking knowledge of size $n/2$ vectors.

- *Aggregation schemes* for polynomial commitments [BGH19,BDFG20] and *unbounded aggregation schemes* for linear-map vector commitments [CNR+22] can both be viewed as interactive reductions from checking proofs of several openings to a commitment to checking a proof of a single opening to a commitment.

- *Split-accumulation schemes* [BCL+21] can be viewed as interactive reductions from checking several proofs of knowledge and several accumulators to checking a single accumulator. *Folding schemes* [KST22] can be viewed as interactive reductions from checking knowledge of two instances in a relation to checking knowledge of a single instance in the relation.

- As observed by Ràfols and Zapico [RZ21], most argument systems with a universal and updatable trusted setup (e.g., [CHM+20, KMP20, Set20, CFF+21]) construct an interactive reduction from the task of checking knowledge of a preimage of a matrix evaluation to the task of checking knowledge of a preimage of a vector evaluation.

Such interactive reductions, although central to modern arguments, lack a unifying theoretical foundation. As evidenced above, these reductions typically have case-by-case security definitions (if any at all) that are tailored towards the larger systems that rely on them. The lack of a common language makes it difficult to relate comparable techniques hidden under incomparable abstractions. Moreover, stitching together various techniques requires remarkably delicate (and often tedious) reasoning for how the soundness of the larger protocol reduces to the soundness of each subprotocol.

**Contributions** Towards a unifying language, we formalize the notion of an interactive reduction over statements of knowledge, in which the verifier reduces the task of checking the original statement to the task of checking a new (simpler) statement. We refer to such a protocol as a *reduction of knowledge*. We prove that reductions of knowledge can be composed sequentially and in parallel. As such, reductions of knowledge serve as both a crisp abstraction and a theory of composition. In particular, they can be stitched together to modularly construct complex arguments of knowledge. Under this treatment, each step of an argument is instilled with a meaningful (and composable) soundness guarantee. This enables significantly simpler soundness proofs and allows each subcomponent to be reused independently in other protocols.

As a technical contribution, we employ reductions of knowledge to unify recursive algebraic arguments and simplify the corresponding analysis. In particular, we develop the *tensor reduction* *of knowledge* as a generalization of the central recursive step for arguments in this class. By instantiating and recursively composing the tensor reduction of knowledge over appropriate spaces, we derive both new and existing arguments of knowledge for various linear algebraic structures. Most notably, we derive a new argument of knowledge for *bilinear forms* which are expressive enough to encode weighted (and permuted) inner-products and more generally any degree-two gate over vectors of inputs.

Throughout our development, we provide various examples to demonstrate how reductions of knowledge offer a promising route towards taming the complexity of modern arguments. Most notably, we compose our linear algebraic reductions to construct an argument of knowledge for $\mathsf{NP}$ with logarithmic communication with minimal additional reasoning.

## 1.1 Reductions of Knowledge

Recall that arguments of knowledge are defined over a relation $\mathcal{R}$ and allow a prover to show for some statement $u$ that it knows witness $w$ such that $(u, w) \in \mathcal{R}$. In contrast, a reduction of knowledge is defined over a pair of relations $\mathcal{R}_1$ and $\mathcal{R}_2$, and enables a verifier to reduce the task of checking knowledge of a satisfying witness for a statement in $\mathcal{R}_1$ to the task of checking knowledge of a satisfying witness for a new statement in $\mathcal{R}_2$.

**Definition 1 (Reduction of Knowledge, Informal).** *A reduction of knowledge from $\mathcal{R}_1$ to $\mathcal{R}_2$ is an interactive protocol between a prover and a verifier. Both parties take as input a claimed statement $u_1$ 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 statement $u_2$ to be checked in place of the original statement, and the prover additionally outputs a corresponding witness $w_2$ such that $(u_2, w_2) \in \mathcal{R}_2$. A reduction of knowledge satisfies the following properties.*

*(i) Completeness: If the prover is provided a satisfying witness $w_1$ for the verifier's input statement $u_1$, then the prover outputs a satisfying witness $w_2$ for the verifier's output statement $u_2$.*

*(ii) Knowledge Soundness: If an arbitrary prover provides a satisfying witness $w_2$ for the verifier's output statement $u_2$, then the prover almost certainly knows a satisfying witness $w_1$ for the verifier's input statement $u_1$.*

*We write $\Pi : \mathcal{R}_1 \to \mathcal{R}_2$ to denote that protocol $\Pi$ is a reduction of knowledge from $\mathcal{R}_1$ to $\mathcal{R}_2$.*

There are two ways to conceptually reconcile reductions of knowledge with arguments of knowledge. First, arguments of knowledge can be viewed as a special case of reductions of knowledge where the second relation $\mathcal{R}_2$ is fixed to encode $\mathsf{true}$ or $\mathsf{false}$. This interpretation helps naturally translate existing tooling used to study arguments of knowledge to study reductions of knowledge. For instance, we can expect reductions of knowledge to be compatible with idealized soundness models such as the random oracle model [BR93] and the algebraic group model [FKL18], idealized communication models such as interactive oracle proofs [BSCS16] and variants [BFS20, CHM$^+$20, CFF$^+$21], and heuristic transformations such as Fiat-Shamir [FS86].

Second, reductions of knowledge can be interpreted as arguments for *conditional* statements in which a prover shows for some $u_1$ that it knows $w_1$ such that $(u_1, w_1) \in \mathcal{R}_1$ *contingent* on the fact that for $u_2$ output by the verifier it knows $w_2$ such that $(u_2, w_2) \in \mathcal{R}_2$. Put more plainly, reductions of knowledge are arguments for statements of the form "If you believe that I know a witness for statement $u_2$ in $\mathcal{R}_2$, then you should believe that I know a witness for statement $u_1$ in $\mathcal{R}_1$". This interpretation helps characterize the sorts of statements that reductions of knowledge can handle more naturally than arguments of knowledge.

Reductions of knowledge can also be viewed as a probabilistic variant of Levin reductions (i.e., Karp reductions that map witnesses as well as statements) that verifiably proceed through interaction. Under this interpretation, Levin reductions can be understood as deterministic reductions of knowledge with no interaction. Just as standard reductions are used for principled algorithm design, reductions of knowledge are intended for principled argument design.

Under any interpretation, we are interested in proving that reductions of knowledge can be composed sequentially and in parallel. Such a requirement holds immediately for standard notions of reductions, but requires subtle reasoning when considering knowledge soundness: To ensure that sequential composability holds, we additionally require that reductions of knowledge are *publicly reducible*. That is, given the input statement $u_1$ and the interaction transcript, any party should be able to reconstruct the output statement $u_2$. As we detail in Section 4, this seemingly innocuous requirement becomes the linchpin in arguing sequential composability. With public reducibility, we have the following.

**Theorem 1 (Sequential Composition, Informal).** *Consider relations $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$. For 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 $\Pi_2 \circ \Pi_1$ is a reduction of knowledge from $\mathcal{R}_1$ to $\mathcal{R}_3$ where $\Pi_2 \circ \Pi_1$ denotes the protocol that results from first running $\Pi_1$, and then running $\Pi_2$ on the statement and witness output by $\Pi_1$.*

By parallel composition, we do not mean running both protocols at the same time, but rather we mean that the composed protocol takes as input instance-witness pairs in parallel and and outputs instance-witness pairs in parallel. For relations $\mathcal{R}_1$ and $\mathcal{R}_2$, let relation $\mathcal{R}_1 \times \mathcal{R}_2$ be such that $((u_1, u_2), (w_1, w_2)) \in \mathcal{R}_1 \times \mathcal{R}_2$ if and only if $(u_1, w_1) \in \mathcal{R}_1$ and $(u_2, w_2) \in \mathcal{R}_2$. Then, we have the following.

**Theorem 2 (Parallel Composition, Informal).** *Consider relations $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, and $\mathcal{R}_4$. For reductions of knowledge $\Pi_1 : \mathcal{R}_1 \to \mathcal{R}_2$ and $\Pi_2 : \mathcal{R}_3 \to \mathcal{R}_4$ we have that $\Pi_1 \times \Pi_2$ is a reduction of knowledge from $\mathcal{R}_1 \times \mathcal{R}_3$ to $\mathcal{R}_2 \times \mathcal{R}_4$ where $\Pi_1 \times \Pi_2$ denotes the protocol that results from running $\Pi_1$ on the statement-witness pair in $\mathcal{R}_1$, running $\Pi_2$ on the statement-witness pair in $\mathcal{R}_3$, and outputting the pair of results.*

## 1.2 A Theory of Composition for Arguments of Knowledge

Reductions of knowledge can be viewed as the first compositional framework that can feasibly capture and tame the growing complexity of modern arguments. Regardless of how reductions are stitched together, our composition results abstract out the pedantic reasoning for how exactly to use the extractors for each subcomponent to build an extractor for the composed reduction. We develop several examples to concretely demonstrate how the reductions of knowledge framework opens up new possibilities.

In more detail, the requirement that the prover "knows" a witness is formally stated as an extractability property: Given an expected polynomial-time prover that can produce a satisfying interaction, there must exist a corresponding expected polynomial-time extractor that can extract the alleged witness (e.g., by running and rewinding the prover internally). This definition, while undoubtedly natural, requires subtle reasoning when constructing large arguments which rely on several sub-arguments: In general, the soundness analysis must meticulously detail how to use the successful prover to construct successful provers for each sub-argument and then use the corresponding extractors to derive an extractor for the overall argument.

In the public-coin setting (where the verifier only sends random challenges), Bootle et al. [BCC$^+$16] abstract away some low-level reasoning by proving that *tree special soundness* implies the standard notion of knowledge soundness. Tree special soundness holds when a *tree of accepting transcripts* contains sufficient information to reconstruct the witness, with each path representing a unique transcript and each branch representing diverging verifier randomness. Both Lee [Lee21] and Attema and Cramer [AC20] show that tree special soundness implies modularity by observing that tree special sound protocols can be sequentially composed to produce a tree special sound protocol.

As demonstrated by these works, tree special soundness is a remarkably useful abstraction for simplifying sequentially composed, uniformly structured arguments (e.g., arguments that recursively invoke themselves). However, when dealing with larger arguments that invoke various *independent* sub-arguments, such as modern arguments for $\mathsf{NP}$, tree special soundness is no longer an appropriate abstraction: having a single transcript that weaves through all such sub-arguments and globally forks with each local challenge undermines the intended semantics and unnecessarily blows up the knowledge error (i.e., the extractors failure probability).

Reductions of knowledge are designed precisely to reason about such arguments. Unlike prior work, our parallel composition operator enables us to capture arguments with arbitrary dependence topologies. For instance, most argument systems for $\mathsf{NP}$, such as Spartan [Set20], Poppins [KMP20], and Marlin [CHM$^+$20], reduce a statement in an $\mathsf{NP}$-complete relation such as R1CS [GGPR13] to several simpler linear algebraic statements (such as inner-product and polynomial evaluation claims), each of which is then checked using a tailored argument [RZ21]. As a concrete example, we show that an argument for $\mathsf{NP}$ can be captured modularly in our framework by utilizing both sequential and parallel composition.

Moreover, because we demonstrate that *any* two publicly verifiable reductions can be composed, this opens up the ability to modularly reason about knowledge-assumption-based non-interactive arguments such as SNARKs [BCCT12, GW11] and incrementally verifiable computation [Val08], which currently fall back on composing extractors in intricate ways [KMP20, KST22, BCL$^+$21]. As a concrete example, we demonstrate how to succinctly express non-interactive $\ell$-*folding schemes* [KST22, RZ22] (i.e, folding schemes reducing $\ell$ initial instances) by utilizing a tree-like dependence topology in our reductions of knowledge framework.

In the public-coin setting, we incorporate prior progress into our framework by proving that tree special soundness implies our notion of knowledge soundness. As such, analysis for public-coin reductions can proceed using standard techniques.

## 1.3 A Unified Theory for Recursive Algebraic Arguments

Reductions of knowledge provide the necessary abstraction to view various techniques under a unifying lens. As a demonstration, we consolidate recursive arguments over homomorphic structures by recasting their central recursive step as instantiations of the tensor reduction of knowledge.

In more detail, modern arguments are designed around leveraging homomorphic structure to achieve better asymptotics and concrete efficiency. An influential line of work [BCC$^+$16, BBB$^+$18, Lee21, AC20, ACR20] studies the consequences of arguments over structurally nested homomorphic objects such as vectors, matrices and hypercubes. A key insight is that such objects contain sufficient algebraic structure for *recursive* arguments in which larger composed statements can be reduced to smaller constituent statements of the same form. For instance, Bootle et al. [BCC+16] show that the task of checking an inner-product over committed size $n$ vectors can be split into the task of checking two inner-products over committed size $n/2$ vectors which can then be "folded" into the task of checking a single inner-product over committed size $n/2$ vectors. Homomorphic structures that enable recursive techniques have become a staple in constructing efficient argument systems for NP [Set20, WTS+18, BBB+18, KMP20]. However, while arguments over recursive homomorphic structures have become an essential tool in practice, the literature detailing such techniques is becoming increasingly dissonant with sparse progress on unifying the disparate approaches.

Bünz et al. [BMM+21] initiate the study of a unified theory by observing that existing inner-product arguments [BCC+16, BBB+18] only require a commitment scheme that is homomorphic over both the commitment keys and messages. Thus, such inner-product arguments can be viewed as instantiations of a generic inner-product argument that only leverages these properties. Bootle, Chiesa, and Sotiraki [BCS21] further relax this requirement by observing that split-and-fold style techniques in general [BBB+18, AC20, BMM+21, BFS20] only require a commitment scheme that can be computed by summing over a hypercube. Leveraging this insight, Bootle et al. show that such techniques can be interpreted as instantiations of the familiar sum-check protocol [LFKN92].

We considerably sharpen the sufficient conditions with the following observation: Protocols such as the sumcheck protocol and the inner-product argument only require that the underlying linear-algebraic objects (e.g., polynomials, vectors, and matrices) form a module (i.e., have a notion of addition and scalar multiplication). Abstracting away the specific details of the associated modules, all such protocols reduce a claim in a "tensored" module to claims in constituent modules. Leveraging this insight, we design an information-theoretic protocol, the *tensor reduction*, as a sweeping generalization of protocols in this class. Conceptually, the tensor reduction explains why such a broad class of protocols look different but feel the same.

**Theorem 3 (Tensor Reduction, Informal).** *For modules $U$, $U_1$, and $U_2$ such that $U \cong U_1 \otimes U_2$, there exists an interactive reduction between a prover and a verifier that reduces the task of evaluating a homomorphism in $U$ to the task of evaluating a homomorphism in $U_1$ and evaluating a homomorphism in $U_2$.*

We explain in detail how the tensor reduction generalizes familiar patterns in Section 5. Essentially, the versatility of the tensor reduction stems from its ability to work over any pair of modules and any valid notion of a tensor product between these modules. In particular, the tensor product can be defined as *any* operator that satisfies the prescribed universality property: the tensor product of any two modules $U_1$ and $U_2$ must result in a new module, denoted $U_1 \otimes U_2$, such that any bilinear mapping $\varphi : U_1 \times U_2 \to V$ induces a unique homomorphism $\widetilde{\varphi} : U_1 \otimes U_2 \to V$ such that $\widetilde{\varphi}(u_1 \otimes u_2) = \varphi(u_1, u_2)$.

For instance, for field $\mathbb{F}$, let the tensor product denote the outer product and consider an arbitrary vector in $\mathbb{F}^n$. This vector can be interpreted as a matrix in $\mathbb{F}^{(n/2)\times 2}$ or equivalently as an element of $\mathbb{F}^{n/2} \otimes \mathbb{F}^2$ which consists of sums of outer products of vectors in $\mathbb{F}^{n/2}$ and $\mathbb{F}^2$. Thus, the tensor reduction can reduce a claim over a vector in $\mathbb{F}^n$ to a claim over a vector in $\mathbb{F}^{n/2}$ and a vector in $\mathbb{F}^2$. Similarly, by taking the tensor product to be polynomial multiplication, the tensor reduction can reduce a claim over a degree $(m, n)$ bivariate polynomial in $\mathbb{F}[X, Y] \cong \mathbb{F}[X] \otimes \mathbb{F}[Y]$ to a claim over a degree $m$ univariate polynomial in $\mathbb{F}[X]$ and a degree $n$ univariate polynomial in $\mathbb{F}[Y]$. By taking the tensor product to be the Kronecker product, the tensor reduction can reduce a claim

| Structure | Module | Decomposition | |
|---|---|---|---|
| | | $k = 2$ | $k = \sqrt[4]{n}$ |
| Vector Commitment | PO Groups | [BCC+16] | ✓ |
| | Bil Groups | [BCS21] | ✓ |
| Linear Forms | PO Groups | [AC20] | [AC20] |
| | Bil Groups | [ACR20] | ✓ |
| Bilinear Forms | Bil Groups | ✓ | ✓ |

Table 1: Protocols synthesized by instantiating the tensor reduction of knowledge. We denote previously unexplored protocols with ✓. PO Group indicates prime order groups. Bil Group indicates symmetric bilinear groups. $k$ denotes the number of chunks tensors are decomposed into in the tensor reduction of knowledge. For vectors of size $n$, $k = 2$ results in protocols with $O(\log n)$ rounds of communication and $O(1)$ messages per round. $k = \sqrt[4]{n}$ results in protocols with $O(1)$ rounds of communication and $O(\sqrt{n})$ messages per round.

over a matrix in $\mathbb{F}^{mp \times nq}$ to a claim over a matrix in $\mathbb{F}^{m \times n}$ and a matrix in $\mathbb{F}^{p \times q}$. Alternatively, by taking the tensor product to be a pairing operation mapping groups $\mathbb{G}_1$ and $\mathbb{G}_2$ to $\mathbb{G}_\mathsf{T}$, the tensor reduction can reduce a claim over $\mathbb{G}_\mathsf{T}$ to claims regarding $\mathbb{G}_1$ and $\mathbb{G}_2$.

Just as the sum-check protocol can be used to design arguments of knowledge, the tensor reduction can be used to design reductions of knowledge. By instantiating the tensor reduction over vector spaces, we derive the *tensor reduction of knowledge*, an unconditionally secure protocol that generalizes the central reductive step common to most recursive algebraic arguments.

**Theorem 4 (Tensor Reduction of Knowledge, Informal).** *For any vector space $(W \to V)$, denoting homomorphisms from vector space $W$ to vector space $V$, and length $n$, there exists a reduction of knowledge that reduces the task of checking knowledge of a preimage of a vector in $(W^n \to V)$ to the task of checking knowledge of a preimage of a vector in $(W \to V)$.*

Leveraging our composition result, we show that tensor reductions of knowledge can be recursively composed to recover various recursive arguments. In particular, we appropriately instantiate the vector spaces to recover a family of reductions of knowledge for vector commitments [BCC+16, BCS21, BDFG20], and linear forms [AC20, ACR20]. Table 1 summarizes the concrete protocols synthesized under the various instantiations of the tensor reduction of knowledge.

We also develop a new family of arguments for bilinear forms which falls out naturally from our prior generalizations. In particular, consider prime order group $\mathbb{G}$ and corresponding scalar field $\mathbb{F}$. For public key $(G, H) \in (\mathbb{G}^m, \mathbb{G}^n)$, public matrix $M \in \mathbb{F}^{m \times n}$, commitments $(\overline{A}, \overline{B}) \in (\mathbb{G}, \mathbb{G})$, and scalar $\sigma \in \mathbb{F}$, a bilinear forms argument allows a verifier to check that a prover knows $(A, B) \in (\mathbb{F}^m, \mathbb{F}^n)$ such that $A^\top M B = \sigma$, $\langle G, A \rangle = \overline{A}$, and $\langle H, B \rangle = \overline{B}$.

In practice, the matrix $M$ in the bilinear forms relation can encode a variety of constraints. For instance, if $M$ is the identity matrix then the verifier can check the inner-product of $A$ and $B$ (and more generally the inner product of any rearrangement of $A$ and $B$). If instead $M$ assigns weights to the diagonal, then the verifier can check a weighted inner-product [CHJ+22, BMM+21]. More generally, $M$ can encode any degree-two custom-gate [GWC19] enabling an expressive constraint system for NP as we show in Section 7.

## 1.4 Overview of the Upcoming Sections

The remainder of this work formally treats all of the introduced concepts. In Section 2, we study two concrete examples, the vector commitment argument [BCC$^+$16] and $\ell$-folding schemes [KST22, RZ22], to both preface the tensor reduction of knowledge and demonstrate how our framework simplifies the corresponding analysis. In Section 4, we formally treat reductions of knowledge and the corresponding composition results. In Section 5, we formally introduce the tensor reduction, followed by the tensor reduction of knowledge as a generalization of the core reductive step common to most recursive algebraic arguments. In Section 6, we instantiate the tensor reduction of knowledge arguments for vector commitments, linear forms, and bilinear forms. In Section 7, we show that the linear algebraic reductions derived from the tensor reduction of knowledge can be composed to derive an argument of knowledge for $\mathsf{NP}$ with minimal effort.

# 2 Technical Overview

In this section, we demonstrate how reductions of knowledge can be used to modularly reason about the vector commitment argument of Bootle et al. [BCC$^+$16] and folding schemes [KST22] in the non-interactive setting [RZ22]. The former example, being a special case of the tensor reduction of knowledge, provides an introductory overview of its mechanics. The latter example demonstrates how the reductions of knowledge framework can significantly simplify arguments with non-linear dependence topologies. We additionally demonstrate how reductions of knowledge provide a unifying language by formally defining arguments of knowledge and folding schemes as particular types of reductions.

## 2.1 First Example: Vector Commitment Argument

The vector commitment argument allows a prover to show that it knows the opening to a Pedersen vector commitment [Ped91]. In particular, consider group $\mathbb{G}$ of prime order $p$ where the discrete logarithm is hard and corresponding scalar field $\mathbb{F} = \mathbb{Z}_p$. Consider some public key $G \in \mathbb{G}^n$ where $n = 2^i$ for some $i \in \mathbb{N}$. Suppose a prover would like to *succinctly* demonstrate to a verifier that it knows $A \in \mathbb{F}^n$ such that $\langle G, A\rangle = \overline{A}$. That is, we would like to design an argument of knowledge for the following relation.

**Definition 2 (Vector Commitment Relation).** *The vector commitment relation is defined as* $\mathcal{R}_{\mathsf{VC}}(n) = \{((G, \overline{A}), A) \in ((\mathbb{G}^n, \mathbb{G}), \mathbb{F}^n) \mid \langle G, A\rangle = \overline{A}\}$.

Bootle et al. [BCC$^+$16] provide an argument system with sublinear communication cost for the above relation. At a high level, the verifier splits the task of checking knowledge of vector $A$ into the task of checking knowledge of the first and second half of $A$. Instead of checking each separately, the verifier "folds" the two checks into a single check using a random linear combination. The prover computes the corresponding random linear combination of the first and second half of $A$ to produce a folded witness vector that is half the original size. This folding procedure is recursively run until the length of the vector to be checked is 1. At this point the prover directly sends the vector to the verifier.

While the vector commitment argument can be described in a straightforward manner, proving its soundness is considerably more involved. Recursive arguments typically require recursive extractors and the vector commitment argument is no exception. To prove knowledge soundness, the full malicious prover, which produces a length 1 witness vector as its final message, is used to build an extractor that can produce a length 2 folded witness vector (which is allegedly the result of folding the original witness vector $\log n - 1$ times). Such an extractor is used to produce an extractor that can produce a length 4 vector, and so on. Ensuring that the extractor can successively unfold the witness vector in each recursive step while also ensuring that it's runtime remains expected polynomial-time requires tedious low-level reasoning. Bootle et al. [BCC+16], and following works [BBB+18, Lee21, AC20] work with tree special soundness precisely to avoid such low-level reasoning.

We show that the reductions of knowledge framework is equally as effective in simplifying the analysis for the vector commitment argument. In particular, we start with the simpler goal of designing a reduction of knowledge that reduces the task of checking knowledge of a size $n$ vector to checking knowledge of a size $n/2$ vector. We can recursively compose such a reduction to design an argument of knowledge for the vector commitment relation.

**Construction 1 (Vector Commitment Reduction of Knowledge).** We construct a reduction of knowledge from $\mathcal{R}_{\mathsf{VC}}(n)$ to $\mathcal{R}_{\mathsf{VC}}(n/2)$ for $n = 2^i$ where $i \geq 1$. Suppose that the prover $\mathcal{P}$ and verifier $\mathcal{V}$ take as input statement $(G, \overline{A}) \in (\mathbb{G}^n, \mathbb{G})$ and that the prover additionally takes as input alleged witness vector $A \in \mathbb{F}^n$ such that

$$((G, \overline{A}), A) \in \mathcal{R}_{\mathsf{VC}}(n).$$

The reduction proceeds as follows.

1. $\mathcal{P}$: Let $G_1$ and $G_2$ (respectively $A_1$ and $A_2$) denote the first and second half of vector $G$ (respectively $A$). The prover begins by sending $\overline{A}_{ij} \leftarrow \langle G_i, A_j \rangle$ for $i, j \in \{1, 2\}$. Here, $\overline{A}_{11}$ and $\overline{A}_{22}$ represent the first and second "half" of the original commitment $\overline{A}$, and $\overline{A}_{12}$ and $\overline{A}_{21}$ represent cross terms which will assist the verifier in folding the original statement.

2. $\mathcal{V}$: The verifier first checks the consistency of $\overline{A}_{11}$ and $\overline{A}_{22}$ with $\overline{A}$ by checking that $\overline{A}_{11} + \overline{A}_{22} = \overline{A}$. The verifier must still check that the prover knows $A_1$ and $A_2$ such that $\overline{A}_{11} = \langle G_1, A_1 \rangle$ and $\overline{A}_{22} = \langle G_2, A_2 \rangle$. Instead of checking each individually, the verifier folds them into a single check by using a random linear combination. In particular, the verifier sends random $r \in \mathbb{F}$ to $\mathcal{P}$.

3. $\mathcal{P}, \mathcal{V}$: Together, the prover and verifier output the folded key and corresponding commitment $(G', \overline{A}') \in (\mathbb{G}^{n/2}, \mathbb{G})$ where $G' \leftarrow G_1 + r \cdot G_2$ and $\overline{A}' \leftarrow \overline{A}_{11} + r \cdot (\overline{A}_{12} + \overline{A}_{21}) + r^2 \cdot \overline{A}_{22}$.

4. $\mathcal{P}$: The prover outputs the folded witness $A' \in \mathbb{F}^{n/2}$ where $A' \leftarrow A_1 + r \cdot A_2$.

Now, to check the original statement, it is sufficient for the verifier to check that the prover knows $A'$ such that

$$((G', \overline{A}'), A') \in \mathcal{R}_{\mathsf{VC}}(n/2).$$

To prove knowledge soundness, we must show that given a prover that produces a witness for the output statement with non-negligible probability we can derive an extractor that can use this prover to derive a witness for the input statement with nearly the same probability. Because the above reduction is public-coin, by our tree extractability lemma (Lemma 6), it is sufficient to show that there exists an extractor that can derive a satisfying input witness given a tree of transcripts and corresponding satisfying outputs. Intuitively, the original extractor can generate such a tree by repeatedly rewinding the prover and collecting transcripts in which the prover outputs a satisfying witness.

**Lemma 1 (Vector Commitment Reduction of Knowledge).** *For $n = 2^i$ where $i \geq 1$, Construction 1 is a reduction of knowledge from $\mathcal{R}_{\mathsf{VC}}(n)$ to $\mathcal{R}_{\mathsf{VC}}(n/2)$.*

*Proof.* We reason via tree extractability (Lemma 6). Suppose an extractor is provided with a tree of transcripts which consists of three transcripts where the $k$th transcript has the same initial message $\overline{A}_{ij}$ for $i, j \in \{1, 2\}$, random challenge $r_k$, and satisfying output instance-witness pairs $((G'_k, \overline{A}'_k), A'_k)$. Using any *two* transcripts, the extractor solves for $A_0$ and $A_1$ such that $A_1 + r_k \cdot A_2 = A'_k$ and outputs the unfolded witness $A \leftarrow (A_1, A_2)$. By the discrete logarithm assumption, we must have that the extractor derives the same $A_1$ and $A_2$ for all pairs of transcripts because otherwise, for some $k \in \{1, 2, 3\}$, we have that the extractor can derive two different preimages for the same commitment $\overline{A}'_k$ with respect to key $G'_k$. Then, because

$$\langle G_1 + r_k \cdot G_2, A_1 + r_k \cdot A_2 \rangle = \langle G'_k, A'_k \rangle = \overline{A}'_k = \overline{A}_{11} + r_k \cdot (\overline{A}_{12} + \overline{A}_{21}) + r_k^2 \cdot \overline{A}_{22}$$

for $k \in \{1, 2, 3\}$, we have that $\langle G_1, A_1 \rangle = \overline{A}_{11}$ and $\langle G_2, A_2 \rangle = \overline{A}_{22}$. Therefore, we have that

$$\langle G, A \rangle = \langle G_1, A_1 \rangle + \langle G_2, A_2 \rangle = \overline{A}_{11} + \overline{A}_{22} = \overline{A}.$$

□

Later, in Section 6, we show that the vector commitment reduction of knowledge is precisely the tensor reduction of knowledge from homomorphisms in $\mathbb{G}^n \cong (\mathbb{G}^{n/2})^2$ to homomorphisms in $\mathbb{G}^{n/2}$ thereby further demonstrating that knowledge soundness holds unconditionally.

We are still tasked with isolating the base case of the original vector commitment argument. Below we specify an *argument of knowledge* for $\mathcal{R}_{\mathsf{VC}}(1)$. An argument of knowledge can be succinctly formalized as a reduction of knowledge that reduces to the relation $\mathcal{R}_\top$ encoding $\mathsf{true}$. A verifier reducing to $\mathcal{R}_\top$ can output $\mathsf{true}$ if it accepts and any other string (e.g., $\mathsf{false}$) otherwise.

**Definition 3 (Argument of Knowledge).** *Let $\mathcal{R}_\top = \{(\mathsf{true}, \bot)\}$. An argument of knowledge for relation $\mathcal{R}$ is a reduction of knowledge from $\mathcal{R}$ to $\mathcal{R}_\top$.*

**Construction 2 (Base Case).** We construct an argument of knowledge for $\mathcal{R}_{\mathsf{VC}}(1)$. Given statement $(G, \overline{A})$ and corresponding witness $A$, the prover sends $A$ directly to the verifier. The verifier outputs $\mathsf{true}$ if $\langle G, A \rangle = \overline{A}$.

We can compose the above reductions to modularly recover the original argument of knowledge for the vector commitment relation. By formalizing each step as a reduction of knowledge, our composition result abstracts away the brunt of the proof effort. In particular, the following corollary holds immediately.

**Corollary 1 (Vector Commitment Argument of Knowledge).** *Let $\Pi_{\mathsf{VC}}$ denote a reduction of knowledge from $\mathcal{R}_{\mathsf{VC}}(n)$ to $\mathcal{R}_{\mathsf{VC}}(n/2)$ and let $\Pi_{\mathsf{base}}$ denote an argument of knowledge for $\mathcal{R}_{\mathsf{VC}}(1)$. Then*

$$\Pi_{\mathsf{base}} \circ \underbrace{\Pi_{\mathsf{VC}} \circ \ldots \circ \Pi_{\mathsf{VC}}}_{\log n \text{ times}}$$

*is an argument of knowledge for $\mathcal{R}_{\mathsf{VC}}(n)$ where $n = 2^i$ for $i \in \mathbb{N}$.*

## 2.2 Second Example: Folding Schemes

The vector commitment reduction of knowledge can be further decomposed into two reductions of knowledge: The first reduction of knowledge splits the original instance into two half-sized instances (i.e., a reduction from $\mathcal{R}_{\mathsf{VC}}(n)$ to $\mathcal{R}_{\mathsf{VC}}(n/2) \times \mathcal{R}_{\mathsf{VC}}(n/2)$). The second folds the two instances into a single instance of the same size (i.e., a reduction from $\mathcal{R}_{\mathsf{VC}}(n/2) \times \mathcal{R}_{\mathsf{VC}}(n/2)$ to $\mathcal{R}_{\mathsf{VC}}(n/2)$).

Kothapalli, Setty, and Tzialla [KST22] abstract the latter pattern to arbitrary relations and refer to such protocols as *folding schemes*. In particular, a folding scheme is an interactive protocol that reduces the task of checking two instances in a relation to the task of checking a single instance in the relation. Folding schemes provide a minimal abstraction for various protocols in the literature. For instance, Kothapalli et al. show that there exists a folding scheme for $\mathsf{NP}$ instances with some fixed size and show that such a construction implies incrementally verifiable computation [Val08].

More recently, Ràfols and Zacharakis [RZ22] provide non-interactive $\ell$-folding schemes (i.e., folding schemes for $\ell$ initial statements) for the vector commitment relation, inner-product relation, and polynomial commitment relation. Such folding schemes help amortize the verifier's work over multiple instances in larger non-interactive arguments of knowledge which typically involve checking multiple instances of the same form.

As these folding schemes rely on knowledge assumptions rather than interaction, prior techniques cannot help modularize the corresponding soundness analysis. As promised, we can still achieve modularity by decomposing them as a sequence of *non-interactive* reductions of knowledge. Formally, a non-interactive reduction of knowledge is one in which the interaction only consists of messages from the prover. Non-interactive $\ell$-folding schemes can be succinctly formalized as a particular class of non-interactive reductions of knowledge. Letting $\mathcal{R}^\ell$ denote $\mathcal{R} \times \ldots \times \mathcal{R}$ for $\ell$ times, we define the following.

**Definition 4 ($\ell$-Folding Schemes).** *A (non-interactive) $\ell$-folding scheme for relation $\mathcal{R}$ is a (non-interactive) reduction of knowledge from $\mathcal{R}^\ell$ to $\mathcal{R}$.*

Ràfols and Zacharakis achieve $\ell$-folding schemes for various relations by recursively composing 2-folding schemes in a tree-like fashion. In particular, $\ell$ instances are treated as leaves in a tree. A 2-folding scheme is then used to fold each pair of adjacent instances to produce a total of $\ell/2$ instances. These $\ell/2$ instances are once more folded in a pairwise fashion to produce $\ell/4$ instances and so on until a single instance remains.

Once again, as demonstrated by Ràfols and Zacharakis, while the tree-folding protocol can be stated in a straightforward manner, the corresponding knowledge soundness analysis requires careful attention to detail. In particular, the corresponding proof involves demonstrating that the malicious prover induces a corresponding expected polynomial-time extractor that unfolds once. Such an extractor is then shown to induce a pair of expected polynomial-time malicious provers for the previous layer of the tree, and so on. Alternatively, by working in the reductions of knowledge framework, nearly all of this reasoning is abstracted away. Indeed, we condense the original three-page proof into several lines.

**Lemma 2 ($\ell$-Folding Scheme).** *Consider a (non-interactive) 2-folding scheme $\Pi_{\mathsf{TF}}$ for relation $\mathcal{R}$ and $\ell = 2^i$ for $i \in \mathbb{N}$ where $i \geq 1$. Then, $\Pi_\ell$, inductively defined as follows, is a (non-interactive) $\ell$-folding scheme for $\mathcal{R}$.*

$$\begin{aligned} \Pi_\ell &= \Pi_{\mathsf{TF}} \circ (\Pi_{\ell/2} \times \Pi_{\ell/2}) \\ \Pi_2 &= \Pi_{\mathsf{TF}} \end{aligned}$$

*Proof.* We reason inductively over $i$. In the base case, suppose $i = 1$. Then, by construction, $\Pi_2$ is a 2-folding scheme. Suppose instead $i \geq 2$. Suppose that for $\ell = 2^i$ we have that $\Pi_{\ell/2}$ is a $(\ell/2)$-folding scheme. Then, $\Pi_{\ell/2} \times \Pi_{\ell/2}$ is a reduction of knowledge from $\mathcal{R}^{\ell/2} \times \mathcal{R}^{\ell/2} = \mathcal{R}^\ell$ to $\mathcal{R}^2$. Thus, $\Pi_{\mathsf{TF}} \circ (\Pi_{\ell/2} \times \Pi_{\ell/2})$ is a reduction of knowledge from $\mathcal{R}^\ell$ to $\mathcal{R}$. $\square$

## 3 Preliminaries

### 3.1 Module Theory

In this section, we introduce our notation, intuit the direct sum and tensor product, and recall several useful properties of these operations. In Appendix A, we present formal definitions for rings, modules, direct sums, and tensor products.

**Notation (Modules).** We assume finite, unital, commutative rings and modules with a finite basis throughout. We use $\cong$ to denote that two modules are isomorphic. For ring $\mathsf{R}$ and $\mathsf{R}$-modules $W$, $V$, let $W \to V$ denote the module of homomorphisms from $W$ to $V$. For $n \in \mathbb{N}$ we let $W^n$ denote $W \otimes \mathsf{R}^n$ (equivalently $W \oplus \ldots \oplus W$ for $n$ times). We use $\{\delta_i\}$ to denote an orthonormal basis. We refer to elements of modules as tensors. As we use tensors to represent both homomorphisms and objects, for tensors $g$ and $a$, we use $g(a)$ to denote evaluating the homomorphism tensor $g$ on the object tensor $a$. For $n \in \mathbb{N}$, let $[n]$ denote $\{1, 2, \ldots, n\}$ and let $[i, n]$ for $i \leq n$ denote $\{i, i+1, \ldots, n\}$ When summing over a variable, we will omit the bounds when clear from context.

**Modules** Intuitively, modules are vector spaces over rings. That is, they support a notion of addition and can be scaled by ring elements. We say a module is an $\mathsf{R}$-module if it is scaled by ring $\mathsf{R}$. Vectors, polynomials, matrices, tensors and scalars all form modules.

**The Direct Sum** Intuitively, a *direct sum* of two $\mathsf{R}$-modules $U$ and $V$, forms a new $\mathsf{R}$-module denoted $U \oplus V$, which is essentially a Cartesian product of the original modules. Elements of $U \oplus V$ consist of pairs of elements in $U$ and $V$ which are denoted as $u \oplus v$ for $u \in U$ and $v \in V$. For example, for field $\mathbb{F}$, if $U \cong \mathbb{F}^n$ and $V \cong \mathbb{F}^m$ we have that $U \oplus V \cong \mathbb{F}^{n+m}$. We have that $U \oplus V$ forms a module because we can naturally compute $u_1 \oplus v_1 + u_2 \oplus v_2 = (u_1 + u_2) \oplus (v_1 + v_2)$ and $r \cdot (u \oplus v) = (r \cdot u) \oplus (r \cdot v)$ for $r \in \mathsf{R}$.

**The Tensor Product** Intuitively, the tensor product, denoted $\otimes$, can be considered a generalized outer-product that distributes with respect to the direct sum. The tensor product of two modules $U$ and $V$, forms a new module denoted $U \otimes V$. Elements of $U \otimes V$ include *simple tensors* which are outer products of elements in $U$ and $V$ and are denoted as $u \otimes v$ for $u \in U$ and $v \in V$. The module $U \otimes V$ also contains arbitrary sums of these simple tensors, which are denoted as $\sum_{i \in [\ell]} u_i \otimes v_i$ for $u_1, \ldots, u_\ell \in U$ and $v_1, \ldots, v_\ell \in V$. If $U \cong \mathbb{F}^n$ and $V \cong \mathbb{F}^m$ we have that $U \otimes V \cong \mathbb{F}^{n \times m}$ (i.e., $n \times m$ matrices over $\mathbb{F}$). Simple tensors in $\mathbb{F}^n \otimes \mathbb{F}^m$ consist of outer products of vectors in $\mathbb{F}^n$ and $\mathbb{F}^m$, however, the entire space is generated by sums over such outer products. We have that $U \otimes V$ forms a module because we can naturally add two sums and compute $r \cdot \sum_i u_i \otimes v_i = \sum_i (r \cdot u_i) \otimes v_i = \sum_i u_i \otimes (r \cdot v_i)$.

**The Direct Sum and Tensor Product as Abstract Operations** Formally, the particular definitions of the direct sum and tensor product depend on the particular modules they are working over. For instance, the tensor product could mean the outer product when working over vectors and the Kronecker product when working over matrices. Even for a fixed pair of modules, there could exist multiple valid definitions. For instance, for vectors $v_1, v_2 \in \mathbb{F}^n$, we can define $v_1 \oplus v_2$ to be a vector in $\mathbb{F}^{2n}$ or a matrix in $\mathbb{F}^{2\times n}$. To account for these considerations, we treat the direct sum and tensor product as *abstract* operations that can be implemented by any concrete operations that satisfy certain axioms (which we detail in Appendix A). In practice, much like how abstract groups and rings must be instantiated with concrete objects such as elliptic curves and polynomials, the direct sum and tensor product must be instantiated with concrete operations that respect the prescribed properties.

For the majority of our development, we are interested in taking the direct sum and tensor product of homomorphisms (represented as tensors). In this situation, we do not need to invoke the abstract definitions of these operations, but rather the identities that follow from their axioms.

**Lemma 3 (Direct Sum of Homomorphisms).** *Homomorphisms $r \in (U_1 \to V)$ and $s \in (U_2 \to V)$ over $\mathsf{R}$-modules (where $\mathsf{R}$ is a commutative ring) induce a homomorphism $r \oplus s \in (U_1 \oplus U_2) \to V$ such that $(r \oplus s)(u_1 \oplus u_2) = r(u_1) + s(u_2)$. Symmetrically, homomorphisms $r \in (U \to V_1)$ and $s \in (U \to V_2)$ over $\mathsf{R}$-modules induce a homomorphism $r \oplus s \in U \to (V_1 \oplus V_2)$ such that $(r \oplus s)(u) = r(u) \oplus s(u)$.*

**Example 1 (Direct Sum of Homomorphisms).** Consider group $\mathbb{G}$ of prime order $p$ and corresponding scalar field $\mathbb{F} \cong \mathbb{Z}_p$. We can interpret $\mathbb{G}^n$ as the module of homomorphisms from $\mathbb{F}^n$ to $\mathbb{G}$. In particular, for $g \in \mathbb{G}^n$ we can define $g(a) = \langle g, a \rangle$ for $a \in \mathbb{F}^n$. Then, for $g \in \mathbb{G}^n$ and $h \in \mathbb{G}^m$ we have that $g \oplus h \in \mathbb{G}^n \oplus \mathbb{G}^m \cong \mathbb{G}^{n+m}$ can be interpreted as a map from $\mathbb{F}^{n+m} \cong \mathbb{F}^n \oplus \mathbb{F}^m$ to $\mathbb{G}$. By definition, for $u \in \mathbb{F}^n$ and $v \in \mathbb{F}^m$, we have $(g \oplus h)(u \oplus v) = \langle g \oplus h, u \oplus v \rangle = \langle g, u \rangle + \langle h, v \rangle = g(u) + h(v)$.

**Lemma 4 (Tensor Product of Homomorphisms).** *Homomorphisms $r \in (U \to X)$ and $s \in (V \to Y)$ over $\mathsf{R}$-modules (where $\mathsf{R}$ is a commutative ring) induce a homomorphism $r \otimes s \in (U \otimes V) \to (X \otimes Y)$, such that $(r \otimes s)(u \otimes v) = r(u) \otimes s(v)$.*

**Example 2 (Tensor Product of Homomorphisms).** Let $\otimes$ denote the outer product. For prime $p$ and field $\mathbb{F} \cong \mathbb{Z}_p$ we can interpret $\mathbb{F}^n$ as the module of homomorphisms from $\mathbb{F}^n$ to $\mathbb{F}$. In particular, for $f \in \mathbb{F}^n$ we can define $f(a) = \langle f, a \rangle$ for $a \in \mathbb{F}^n$. Then, $f \in \mathbb{F}^n$ and $g \in \mathbb{F}^m$ induce a new map $f \otimes g \in \mathbb{F}^n \otimes \mathbb{F}^m \cong \mathbb{F}^{nm}$ from $\mathbb{F}^{nm}$ to $\mathbb{F} \otimes \mathbb{F} \cong \mathbb{F}$. By definition, for $u \in \mathbb{F}^n$ and $v \in \mathbb{F}^m$, we have $(f \otimes g)(u \otimes v) = (f \cdot g_1 \oplus \ldots \oplus f \cdot g_m)(u \cdot v_1 \oplus \ldots \oplus u \cdot v_m) = \sum_{j \in [m]} f(u) \cdot g_i(v_i) = f(u) \otimes g(v)$.

**Lemma 5 (Useful Identities).** *For commutative ring $\mathsf{R}$ and $\mathsf{R}$-modules $U, V, W$, we have that $(U \otimes V) \otimes W \cong U \otimes (V \otimes W)$, $U \otimes V \cong V \otimes U$, $U \otimes (V \oplus W) \cong (U \otimes V) \oplus (U \otimes W)$, and $\mathsf{R} \otimes U \cong U \otimes \mathsf{R} \cong U$.*

## 3.2 Cryptographic Preliminaries

**Notation (Cryptographic Variables).** We use $\lambda$ globally to denote the security parameter, and $\mathsf{negl}(\lambda)$ to denote negligible functions. For events $A$ and $B$, we let $\Pr[A] \approx \Pr[B]$ denote that $|\Pr[A] - \Pr[B]| = \mathsf{negl}(\lambda)$.

For soundness to hold when randomly sampling over rings, the set of admissible values must be constrained. We define a valid sampling set over rings.

**Definition 5 (Sampling Set [BCS21]).** *For ring $\mathsf{R}$ and $\mathsf{R}$-module $M$, subset $Q \subseteq \mathsf{R}$ is a sampling set for $M$ if for every $q_1, q_2 \in Q$, the map $\varphi_{q_1,q_2}(m) = (q_1 - q_2) \cdot m$ for $m \in M$ is injective.*

For certain relations, to be able to prove knowledge soundness, we will need to rely on computational hardness assumptions. We adapt the bilinear relation assumption [BCS21], which can be viewed as a generalization of the discrete logarithm assumption, and the double pairing assumption [AFG$^+$10].

**Definition 6 (Bilinear Relation Assumption).** *For ring $\mathsf{R}$, length parameter $n$, and security parameter $\lambda$, consider $\mathsf{R}$-modules $U$ and $V$ such that $|U| = O(2^\lambda)$ and $|V| = O(2^\lambda)$. The bilinear relation assumption holds for $(U, V)$ (w.r.t. tensor product $\otimes$) if given random $u_1, \ldots, u_n \in U$, there exists no polynomial-time algorithm to find non-trivial $v_1, \ldots, v_n \in V$ such that $\sum_{i\in[n]} u_i \otimes v_i = 0$.*

Symmetrically, we can consider composite spaces such that given elements from both of the constituent spaces, it is *easy* to check that they satisfy the above relation. This ensures that that the verifier is able to perform its requisite checks efficiently. Throughout our development, we assume the coset equality assumption holds as necessary.

**Definition 7 (Coset Equality Assumption).** *For ring $\mathsf{R}$ and length parameter $n$ consider $\mathsf{R}$-modules $U$ and $V$. The coset equality assumption holds for $(U, V)$ (w.r.t. tensor product $\otimes$) if for any $u_1, \ldots, u_n \in U$ and $v_1, \ldots, v_n \in V$, there exists a polynomial-time algorithm to check $\sum_{i\in[n]} u_i \otimes v_i = 0$.*

**Example 3 (Bilinear Relation Assumption).** Suppose $U$ is a group of prime order $p$ and $V$ is the corresponding scalar field $\mathbb{Z}_p$. Let the tensor product between these two modules be defined as scalar multiplication. In this setting, the bilinear relation assumption is equivalent to the discrete logarithm assumption. Alternatively, suppose $U$ and $V$ are prime order groups such that there exists a corresponding pairing operation $e$ from $U \times V$ into some target group. Let the tensor product be defined as this pairing operation. In this setting, the bilinear relation assumption is equivalent to the double pairing assumption.

## 4 Reductions of Knowledge

Recall that in contrast to arguments of knowledge, reductions of knowledge are defined over a pair of relations $\mathcal{R}_1$ and $\mathcal{R}_2$. A prover can use a reduction of knowledge to show for some $u_1$ that it knows $w_1$ such that $(u_1, w_1) \in \mathcal{R}_1$ contingent on the fact that it knows $w_2$ for some statement $u_2$ (derived from its interaction with the verifier) such that $(u_2, w_2) \in \mathcal{R}_2$. We start by intuiting the desired notion of knowledge soundness needed to capture such an interaction before presenting a formal definition (Definition 8). We show that any two reductions of knowledge that respect this definition can be composed sequentially and in parallel (Theorems 5 and 6). We then observe that a more restricted — but simpler — notion of soundness, known as tree extractability, implies our definition of knowledge soundness (Lemma 6). In the following sections, we leverage this observation to prove that our reductions of knowledge for linear-algebraic statements are secure.

## 4.1 Defining Reductions of Knowledge

Intuitively, we would like that if a prover is able to convince a verifier on input $u_1$ to output some derived statement $u_2$ such that it knows a corresponding satisfying witness $w_2$, then it must have known a corresponding satisfying witness $w_1$ for $u_1$. We can capture this notion formally by stating that if a malicious prover can output a satisfying witness $w_2$ for the verifier's output statement $u_2$, then there must exist a corresponding extractor that can output a satisfying witness $w_1$ for the verifier's input statement $u_1$.

While this presents a stand-alone notion of knowledge soundness, we require a more nuanced definition to capture technical difficulties that arise when reasoning about sequential composability. In particular, existing definitions implicitly assume that the environment is provided access to the inputs and outputs of the prover and the verifier, and that some of this material (such as an adversarially chosen statement) is forwarded to the extractor. Unfortunately, when composing such arguments, we end up in situations where intermediate inputs expected by the extractor are never exposed to the environment.

Concretely, consider a reduction of knowledge $\Pi_1$ with prover $\mathcal{P}_1$, verifier $\mathcal{V}_1$, and extractor $\mathcal{E}_1$, and a second reduction of knowledge $\Pi_2$ with corresponding $\mathcal{P}_2$, $\mathcal{V}_2$, and $\mathcal{E}_2$. Ideally, we would want to use $\mathcal{E}_1$ and $\mathcal{E}_2$ in a black-box manner to construct an extractor $\mathcal{E}$ for $\Pi_2 \circ \Pi_1$. A typical knowledge soundness definition would dictate that the statement provided to the verifier is forwarded to the extractor as well. Unfortunately, in the composed setting, the statement $u_2$ output by $\mathcal{V}_1$ as input to $\mathcal{V}_2$ is never exposed to the environment, and thus it is unclear how $\mathcal{E}$ can simulate the intermediate statement $u_2$ expected by $\mathcal{E}_2$.

To alleviate this issue, we stipulate an additional requirement that the verifier's output statement can be deterministically recovered from the mutual view of both the prover and verifier. Specifically, the mutual view consists of the public parameters, initial input statement, and interaction transcript. We refer to this property as *public reducibility*, which can be viewed as analogous to the public verifiability property common to most modern arguments. With public reducibility, we are afforded sequential composability.

We formally define reductions of knowledge as interactive protocols in the common reference string model.[1]

**Definition 8 (Reduction of Knowledge).** *Consider binary relations $\mathcal{R}_1$ and $\mathcal{R}_2$. A reduction of knowledge from $\mathcal{R}_1$ to $\mathcal{R}_2$ is defined by PPT algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ denoting the generator, the prover, and the verifier respectively with the following interface.*

- $\mathcal{G}(\lambda) \to \mathsf{pp}$*: Takes as input security parameter $\lambda$. Outputs public parameters* $\mathsf{pp}$.
- $\mathcal{P}(\mathsf{pp}, u_1, w_1) \to (u_2, w_2)$*: Takes as input public parameters* $\mathsf{pp}$*, and statement-witness pair $(u_1, w_1) \in \mathcal{R}_1$. Interactively reduces the statement $(u_1, w_1) \in \mathcal{R}_1$ to a new statement $(u_2, w_2) \in \mathcal{R}_2$.*
- $\mathcal{V}(\mathsf{pp}, u_1) \to u_2$*: Takes as input public parameters* $\mathsf{pp}$*, and statement $u_1$ associated with $\mathcal{R}_1$. Interactively reduces the task of checking $u_1$ to the task of checking a new statement $u_2$ associated with $\mathcal{R}_2$.*

[1]The formal definition allows for an adversarially chosen statement, which, according to our constructions also includes public parameters. We implicitly assume that the verifier in such reductions checks that the honestly generated public parameters are consistent with the statement.

*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 $(\mathsf{pp}, u_1, w_1)$ and runs the interaction on prover input $(\mathsf{pp}, u_1, w_1)$ and verifier input $(\mathsf{pp}, u_1)$. At the end of the interaction, $\langle \mathcal{P}, \mathcal{V} \rangle$ outputs the verifier's statement $u_2$ and the prover's witness $w_2$. A reduction of knowledge $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ satisfies the following conditions.*

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

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

*(ii) Knowledge Soundness: For any expected polynomial-time adversaries $\mathcal{A}$ and $\mathcal{P}^*$, there exists an expected polynomial-time extractor $\mathcal{E}$ such that given $\mathsf{pp} \leftarrow \mathcal{G}(\lambda)$ and $(u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$, we have that*

$$\Pr[(u_1, \mathcal{E}(\mathsf{pp}, u_1, \mathsf{st})) \in \mathcal{R}_1] \approx \Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle(\mathsf{pp}, u_1, \mathsf{st}) \in \mathcal{R}_2].$$

*(iii) Public Reducibility: There exists a deterministic polynomial-time function $\varphi$ such that for any* $\mathsf{PPT}$ *adversary $\mathcal{A}$ and expected polynomial-time adversary $\mathcal{P}^*$, given $\mathsf{pp} \leftarrow \mathcal{G}(\lambda)$, $(u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$, and $(u_2, w_2) \leftarrow \langle \mathcal{P}^*, \mathcal{V} \rangle(\mathsf{pp}, u_1, \mathsf{st})$ with interaction transcript $\mathsf{tr}$, we have that $\varphi(\mathsf{pp}, u_1, \mathsf{tr}) = u_2$.*

*We write $\Pi : \mathcal{R}_1 \to \mathcal{R}_2$ to denote that protocol $\Pi$ is a reduction of knowledge from relation $\mathcal{R}_1$ to relation $\mathcal{R}_2$.*

**Definition 9 (Public-Coin).** *A reduction of knowledge is public-coin if the verifier only sends uniformly random challenges to the prover.*

## 4.2 Composing Reductions of Knowledge

We now prove a sequential and parallel composition theorem for reductions of knowledge. This allows us to construct complex arguments by stitching together simpler reductions sequentially and in parallel. In the case of sequential composition, much like recursive composition techniques [BCCT13, Val08, KST22, BCL+21], each composition step induces a polynomial blowup in the corresponding extractor. Thus, sequential composition cannot be used more than a constant number of times without additional computational assumptions.[2] Our parallel composition operator is not parallel in the sense that both protocols are being run at the same time, but rather parallel in the sense that the composed protocol takes incoming instance-witness pairs in parallel and produces outgoing instance-witness pairs in parallel.

**Theorem 5 (Sequential Composition).** *Consider binary relations $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$. For reductions of knowledge $\Pi_1 = (\mathcal{G}_1, \mathcal{P}_1, \mathcal{V}_1) : \mathcal{R}_1 \to \mathcal{R}_2$ and $\Pi_2 = (\mathcal{G}_2, \mathcal{P}_2, \mathcal{V}_2) : \mathcal{R}_2 \to \mathcal{R}_3$, we have that $\Pi_2 \circ \Pi_1 = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a reduction of knowledge from $\mathcal{R}_1$ to $\mathcal{R}_3$ where*

$$\begin{aligned} \mathcal{G}(\lambda) &= (\mathcal{G}_1(\lambda), \mathcal{G}_2(\lambda)) \\ \mathcal{P}((\mathsf{pp}_1, \mathsf{pp}_2), u_1, w_1) &= \mathcal{P}_2(\mathsf{pp}_2, \mathcal{P}_1(\mathsf{pp}_1, u_1, w_1)) \\ \mathcal{V}((\mathsf{pp}_1, \mathsf{pp}_2), u_1) &= \mathcal{V}_2(\mathsf{pp}_2, \mathcal{V}_1(\mathsf{pp}_1, u_1, w_1)) \end{aligned}$$

[2]We recommend Bitansky et al. [BCCT13, Remark 6.3] for a detailed discussion on such assumptions.

*Proof Intuition.* Completeness and public reducibility follow by observation. As for knowledge soundness, assume there exists an adversarial prover $\mathcal{P}^*$ for $\Pi$ that succeeds in producing an accepting witness $w_3$ with non-negligible probability. Using the second half of $\mathcal{P}^*$ (i.e. the part that interacts with $\mathcal{V}_2$) we can construct an adversary $\mathcal{P}_2^{**}$ for $\Pi_2$ that succeeds in producing an accepting witness $w_3$ with the same probability. By the knowledge soundness of $\Pi_2$ this implies an extractor $\mathcal{E}_2$ that succeeds in producing an intermediate witness $w_2$ with nearly the same probability. We can then leverage $\mathcal{E}_2$ to construct an adversary $\mathcal{P}_1^{**}$ for $\Pi_1$ that succeeds in producing an accepting witness $w_2$ with nearly the same probability: In particular $\mathcal{P}_1^{**}$ first runs the first half of $\mathcal{P}^*$ and then runs extractor $\mathcal{E}_2$ on the intermediate statement $u_2$ (derived by the public reducibility of $\Pi_1$) and the intermediate state of $\mathcal{P}^*$ to produce the output $w_2$. By the knowledge soundness of $\Pi_1$ this implies the desired extractor $\mathcal{E}_1$ that succeeds in producing the witness $w_1$ with nearly the same probability. We present a formal proof in Appendix B.1. □

**Definition 10 (Relation Pair).** *Consider relations $\mathcal{R}_1$ and $\mathcal{R}_2$. We define the relation $\mathcal{R}_1 \times \mathcal{R}_2 = \{((u_1, u_2), (w_1, w_2)) \mid (u_1, w_1) \in \mathcal{R}_1, (u_2, w_2) \in \mathcal{R}_2\}$. We let $\mathcal{R}^\ell$ denote $\mathcal{R} \times \ldots \times \mathcal{R}$ for $\ell$ times.*

**Theorem 6 (Parallel Composition).** *Consider relations $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$. For reductions of knowledge $\Pi_1 = (\mathcal{G}_1, \mathcal{P}_1, \mathcal{V}_1) : \mathcal{R}_1 \to \mathcal{R}_2$ and $\Pi_2 = (\mathcal{G}_2, \mathcal{P}_2, \mathcal{V}_2) : \mathcal{R}_3 \to \mathcal{R}_4$, we have that $\Pi_1 \times \Pi_2 = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a reduction of knowledge from $\mathcal{R}_1 \times \mathcal{R}_3$ to $\mathcal{R}_2 \times \mathcal{R}_4$ where*

$$\begin{aligned}
\mathcal{G}(\lambda) &= (\mathcal{G}_1(\lambda), \mathcal{G}_2(\lambda)) \\
\mathcal{P}((\mathsf{pp}_1, \mathsf{pp}_2), (u_1, u_3), (w_1, w_3)) &= (\mathcal{P}_1(\mathsf{pp}_1, u_1, w_1), \mathcal{P}_2(\mathsf{pp}_2, u_3, w_3)) \\
\mathcal{V}((\mathsf{pp}_1, \mathsf{pp}_2), (u_1, u_3)) &= (\mathcal{V}_1(\mathsf{pp}_1, u_1), \mathcal{V}_2(\mathsf{pp}_2, u_3))
\end{aligned}$$

*Proof Intuition.* For $i \in \{1, 2\}$, we leverage $\mathcal{A}$ and $\mathcal{P}^*$ to construct expected polynomial time adversaries $\mathcal{A}_i$ and $\mathcal{P}_i^*$ for protocol $\Pi_i$ that succeeds in producing a satisfying output witness with the same probability. By the knowledge soundness of $\Pi_i$, this implies an expected polynomial-time extractor $\mathcal{E}_i$ that succeeds in producing a satisfying input witness with nearly the same probability. These extractors imply the desired extractor $\mathcal{E}$. We present a formal proof in Appendix B.2. □

## 4.3 Knowledge Soundness from Tree Extraction

When proving constructions secure, reasoning about knowledge soundness directly is typically cumbersome. To alleviate this issue, prior work [BCC+16] observes that most protocols are algebraic: The corresponding extractor typically runs the malicious prover multiple times with refreshed verifier randomness to retrieve accepting transcripts, which can be interpolated to retrieve the witness. Leveraging this insight, Bootle et al. [BCC+16] provide a general extraction lemma, which states that to prove knowledge soundness for algebraic protocols, it is sufficient to show that there exists an extractor that can produce a satisfying witness when provided a *tree of accepting transcripts* with refreshed verifier randomness at each layer. This proof technique has been adapted to various settings [BCL+21, KST22, BBB+18, BCS21], and we similarly provide the corresponding lemma for reductions of knowledge.

**Definition 11 (Tree of Transcripts).** *Consider an $m$-round public-coin interactive protocol $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ that satisfies the interface described in Definition 8. A $(n_1, \ldots, n_m)$-tree of accepting transcripts for statement $u_1$ is a tree of depth $m$ where each vertex at layer $i$ has $n_i$ outgoing edges* *such that (1) each vertex in layer $i \in [m]$ is labeled with a prover message for round $i$; (2) each outgoing edge from layer $i \in [m]$ is labeled with a different choice of verifier randomness for round $i$; (3) each leaf is labeled with an accepting statement-witness pair output by the prover and verifier corresponding to the interaction along the path.*

**Lemma 6 (Tree Extraction [BCS21]).** *Consider an $m$-round public-coin interactive protocol $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ that satisfies the interface described in Definition 8 and satisfies completeness. Then $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a reduction of knowledge if there exists a* PPT *extractor $\chi$ that, for all instances $u_1$, outputs a satisfying witness $w_1$ with probability $1 - \mathsf{negl}(\lambda)$, given an $(n_1, \ldots, n_m)$-tree of accepting transcripts for $u_1$ where the verifier's randomness is sampled from space $Q$ such that $|Q| = O(2^\lambda)$, and $\prod_i n_i = \mathsf{poly}(\lambda)$.*

*Proof Intuition.* Our proof closely follows that of Bootle et al. [BCC+16]. At a high level, we construct an expected polynomial-time extractor $\mathcal{E}$ that repeatedly runs the malicious prover $\mathcal{P}^*$ and collects corresponding accepting transcripts and associated output statement-witness pairs. The extractor then passes these collected transcripts to $\chi$ which retrieves the desired witness by assumption. We present a formal proof in Appendix B.3. □

## 5 The Tensor Reduction of Knowledge

We start by defining a general tensor-based language to capture a large class of linear algebraic statements. We then design a general reduction, the tensor reduction, for such statements, by extending the sum-check protocol [LFKN92]. Next, we leverage the tensor reduction to construct the tensor reduction of knowledge, which, for any length vector space of homomorphisms $(W \to V)$ and length $n$, reduces the task of checking knowledge of a preimage of a vector in $(W^n \to V)$ to checking knowledge of a preimage in $(W \to V)$.

### 5.1 Tensor Evaluation Statements

We observe that arguments of knowledge built around statements over linear algebraic objects — such as matrices, vectors, polynomials, and homomorphisms — typically share hints of symmetry. Our goal is to generalize such statements, and more interestingly generalize interactive reductions for such statements.

Regardless of the underlying "linear-algebraic" objects, arguments over them tend to only rely on the fact they support some notion of addition and that they can be scaled by elements in a field (and more generally rings). This seems to suggest that designing a reduction over the most general objects that support these operations, namely tensors, would give a single universal protocol for such objects. From an algebraic standpoint, tensors unify objects such as scalars, vectors, matrices, and polynomials. More generally, tensors provide a unifying algebraic object for describing both functions (when viewed as homomorphisms) and objects (when viewed as elements of a module).

Take for instance the vector commitment relation: Given a prime order group $\mathbb{G}$ and an underlying scalar field $\mathbb{F}^n$, a prover claims that for public commitment key $G \in \mathbb{G}^n$ and commitment $\overline{A}$, it knows a vector $A \in \mathbb{F}^n$ such that $\langle G, A \rangle = \overline{A}$. As the spaces $\mathbb{G}^n$, $\mathbb{F}^n$ and $\mathbb{G}$ are all modules, we can build a corresponding "tensor evaluation" statement

$$G(A) = \overline{A}$$

where $G$ is a tensor in $\mathbb{G}^n$ that maps tensors in $\mathbb{F}^n$ to tensors in $\mathbb{G}$.

Alternatively, suppose in addition to claiming that it knows a vector $A$ underlying a commitment $\overline{A}$ with respect to commitment key $G$, the prover additionally claims that taking the inner-product of $A$ against some public vector $B \in \mathbb{F}^n$ results in a scalar $\sigma \in \mathbb{F}$. Following our prior reasoning, this can be represented as two tensor evaluation statements: A claim that $G(A) = \overline{A}$ and a claim that $B(A) = \sigma$. But, under the rules of the direct sum (which can be interpreted as a Cartesian product), this is equivalent to applying the tensor $G \oplus B \in \mathbb{G}^n \oplus \mathbb{F}^n$ to $A$ and checking that this results in $\overline{A} \oplus \sigma \in \mathbb{G} \oplus \mathbb{F}$. Namely, we have that the composite statement can be encoded as the following tensor evaluation statement:

$$(G \oplus B)(A) = \overline{A} \oplus \sigma.$$

The flexibility of tensor evaluation statements becomes more salient with the sum-check protocol [LFKN92]. In the sum-check protocol, the prover claims for multivariate polynomial $P \in \mathbb{F}^n \to \mathbb{F}$ with degree $d$ in each variable that

$$\sum_{x_1,\ldots,x_n \in \{0,1\}} P(x_1, \ldots, x_n) = \sigma \tag{1}$$

for some claimed sum $\sigma \in \mathbb{F}$. For $i \in [n]$ consider the tensor $\bigoplus_{j \in [0,d]} x_i^j$ which is just shorthand for the vector $(x_i^0, x_i^1, \ldots, x_i^d)$. Now, consider $\bigotimes_{i \in [n]} \bigoplus_{j \in [0,d]} x_i^j$, which is an $n$-dimensional matrix populated with all possible products of powers of $x_1, \ldots, x_n$. We can now define a tensor $\mathbf{X} = \sum_{x_1,\ldots,x_n \in \{0,1\}} \bigotimes_{i \in [n]} \bigoplus_{j \in [0,d]} x_i^j \in (\mathbb{F}^d)^n$ which encodes all desired evaluation points. Additionally, let $\mathbf{P} \in (\mathbb{F}^d)^n$ denote an $n$-dimensional tensor constituting of the coefficients of $P$. Specifically, let $\mathbf{P}$ contain at index $(j_1, \ldots, j_n)$ the coefficient of $P$ associated with term $x_1^{j_1} x_2^{j_2} \ldots x_n^{j_n}$. Now, we have that checking the original sum-check statement is equivalent to checking the tensor evaluation statement

$$\mathbf{P}(\mathbf{X}) = \sigma.$$

The three examples above suggest that seemingly disparate linear-algebraic claims can be uniformly viewed as tensor evaluation claims. In light of this, we are interested in designing a reduction for statements of the form $u(w) = v$ for tensors $u$, $w$, and $v$.

## 5.2 The Tensor Reduction

To design a general reduction for tensor statements of the form $u(w) = v$, we start by generalizing the sum-check protocol for tensor evaluation statements. Recall that the sum-check protocol reduces the task of checking the claim in Equation (1) to the task of checking a sum-check claim over a polynomial with one less variable. In particular, the prover begins by sending

$$p(X) = \sum_{x_1,\ldots,x_{n-1} \in \{0,1\}} P(x_1, \ldots, x_{n-1}, X)$$

The verifier then checks that $p(0) + p(1) = \sigma$. The verifier must now check that $p$ is consistent with $P$. To do so, the verifier samples a random $r \leftarrow \mathbb{F}$, and reduces to checking

$$\sum_{x_1,\ldots,x_{n-1}} P(x_1, \ldots, x_{n-1}, r) = p(r).$$

In essence, the sum-check protocol leverages the nested structure of polynomials to reduce the task of checking $n$-variate polynomials to $(n-1)$-variate polynomials. This intuition can be more lucidly expressed with the corresponding tensor evaluation statements: the sum-check protocol reduces the task of checking the evaluation of $\mathbf{P} \in (\mathbb{F}^d)^n \cong (\mathbb{F}^d)^{n-1} \otimes \mathbb{F}^d$ (representing $P$) to the task of checking the evaluation of $\mathbf{P}_r \in (\mathbb{F}^d)^{n-1}$ (representing $P$ evaluated on $r$) and $\mathbf{p} \in \mathbb{F}^d$ (representing $p$). That is, the sum-check protocol factors the original statement with respect to the tensor product.

The tensor reduction, which we detail below, follows from generalizing the involved spaces to handle arbitrary tensor evaluation statements: for any modules $U$, $U_1$, and $U_2$ such that $U \cong U_1 \otimes U_2$, we derive a mechanism to reduce an evaluation claim in $U$ to an evaluation claim in $U_1$ and an evaluation claim in $U_2$. In Appendix C, we show that we can recover the sum-check protocol when instantiating the tensor reduction over multivariate polynomials.

**Construction 3 (Tensor Reduction).** Suppose for tensors $u \in (W_1 \to V_1) \otimes (W_2 \to V_2)$ of rank $I$, $w \in W_1 \otimes W_2$ of rank $J$, and $v \in V_1 \otimes V_2$ over ring $\mathsf{R}$, a verifier would like to check

$$u(w) = v \tag{2}$$

where $u = \sum_{i\in[I]} u_{1,i} \otimes u_{2,i}$, and $w = \sum_{j\in[J]} w_{1,j} \otimes w_{2,j}$. By definition, the verifier can check (2) by checking $\sum_{i,j} u_{1,i}(w_{1,j}) \otimes u_{2,i}(w_{2,j}) = v$. Therefore, the prover begins by computing and sending $v_{1,ij} \leftarrow u_{1,i}(w_{1,j})$ and $v_{2,ij} \leftarrow u_{2,i}(w_{2,j})$ for all $i \in [I], j \in [J]$. The verifier directly checks

$$\sum_{i\in[I],j\in[J]} v_{1,ij} \otimes v_{2,ij} = v.$$

The verifier must still check that $v_{1,ij} = u_{1,i}(w_{1,j})$ and $v_{2,ij} = u_{2,i}(w_{2,j})$ for all $i, j$. To do so, the verifier takes a random linear combination of these checks by sending random $\alpha, \beta$ from a valid sampling set $Q \subseteq \mathsf{R}$, and computing $v_1 = \sum_{i,j} \alpha^i \beta^j v_{1,ij}$ and $v_2 = \sum_{i,j} \alpha^i \beta^j v_{2,ij}$. The verifier then outputs $(\alpha, \beta, v_1, v_2)$, reducing the original check to the task of checking

$$\Big(\sum_i \alpha^i u_{1,i}\Big)\Big(\sum_j \beta^j w_{1,j}\Big) = v_1 \quad \text{and} \quad \Big(\sum_i \alpha^i u_{2,i}\Big)\Big(\sum_j \beta^j w_{2,j}\Big) = v_2.$$

**Theorem 7 (Tensor Reduction).** *For tensors $u = \sum_i u_{1,i} \otimes u_{2,i} \in (W_1 \to V_1) \otimes (W_2 \to V_2)$ of rank $I$, $w = \sum_j w_{1,j} \otimes w_{2,j} \in W_1 \otimes W_2$ of rank $J$, and $v \in V_1 \otimes V_2$ over ring $\mathsf{R}$, the tensor reduction reduces the task of checking*

$$u(w) = v$$

*to the task of checking*

$$\Big(\sum_i \alpha^i u_{1,i}\Big)\Big(\sum_j \beta^j w_{1,j}\Big) = v_1 \quad \textit{and} \quad \Big(\sum_i \alpha^i u_{2,i}\Big)\Big(\sum_j \beta^j w_{2,j}\Big) = v_2$$

*for verifier output $(\alpha, \beta, v_1, v_2)$. Formally, if the former is true, then the latter is true with probability 1, and if the former is false, then the latter is false with probability at least $1 - \frac{IJ}{|Q|}$. The prover complexity, verifier complexity, and communication complexity are all proportional to $IJ$.*

*Proof.* This follows from the Schwartz-Zippel Lemma [Sch80] extended to modules [BCS21]. $\square$

At first glance, it may seem that the communication cost of the tensor reduction is *greater* than the size of the witness: the witness only consists of $J$ elements in $W_1 \otimes W_2$, but the prover sends $IJ$ elements in $V_1$ and $V_2$. This is reconciled by the fact that elements of $V_1$ and $V_2$ are intended to be significantly smaller than elements in $W_1 \otimes W_2$. For instance, elements in $W_1 \otimes W_2$ may be long vectors that are mapped to short commitments in $V_1$ and $V_2$.

To build intuition for where tensor reductions are useful, we explain how to instantiate the tensor reduction to reconstruct the vector commitment reduction of knowledge presented in Section 2.

**Example 4 (Vector Commitment Reduction of Knowledge).** We construct a reduction of knowledge from $\mathcal{R}_{\mathsf{VC}}(n)$ to $\mathcal{R}_{\mathsf{VC}}(n/2)$ for $n = 2^i$ where $i \geq 1$. Consider group $\mathbb{G}$ of prime order $p$ where the discrete logarithm problem is hard, and corresponding scalar field $\mathbb{F} \cong \mathbb{Z}_p$. Consider some public key $G \in \mathbb{G}^n$. Suppose a verifier would like to check for some commitment $\overline{A} \in \mathbb{G}$, that the prover knows vector $A \in \mathbb{F}^n$ such that $G(A) = \overline{A}$ where $G(A)$ is defined to be $\langle G, A \rangle$.

We observe that $\mathbb{G}^n \cong \mathbb{G}^{n/2} \otimes \mathbb{F}^2$ and $\mathbb{F}^n \cong \mathbb{F}^{n/2} \otimes \mathbb{F}^2$. Let $\{\delta_1, \delta_2\}$ be an orthonormal basis for $\mathbb{F}^2$ (i.e., we have that $\delta_i(\delta_j) = 1$ when $i = j$ and 0 otherwise). Then, we have that $G = G_1 \otimes \delta_1 + G_2 \otimes \delta_2$ and $A = A_1 \otimes \delta_1 + A_2 \otimes \delta_2$ for some $G_1, G_2 \in \mathbb{G}^{n/2}$ and $A_1, A_2 \in \mathbb{F}^{n/2}$. These terms can be interpreted as the first and second half of vectors $G$ and $A$. Therefore, the verifier can equivalently check

$$\Big(\sum_i G_i \otimes \delta_i\Big)\Big(\sum_j A_j \otimes \delta_j\Big) = \overline{A}.$$

Applying the tensor reduction with respect to this decomposition, we have that the prover sends to the verifier $G_i(A_j), \delta_i(\delta_j)$ for $i, j \in \{1, 2\}$. Explicitly, letting $\overline{A}_{ij} = G_i(A_j)$, the prover sends the terms $(\overline{A}_{11}, 1)$, $(\overline{A}_{12}, 0)$, $(\overline{A}_{21}, 0)$, and $(\overline{A}_{22}, 1)$. We recognize that the first and last terms correspond with the first and second half of commitment $\overline{A}$, and the middle two terms are cross terms.

Upon receiving these terms, the verifier checks that

$$\overline{A}_{11} \otimes 1 + \overline{A}_{12} \otimes 0 + \overline{A}_{21} \otimes 0 + \overline{A}_{22} \otimes 1 = \overline{A}.$$

The verifier then samples and sends random $\alpha, \beta \leftarrow \mathbb{F}$, and sets the new statements to be checked to be $(G_1 + \alpha G_2)(A_1 + \beta A_2) = \sum_{i,j \in \{1,2\}} \overline{A}_{ij} \cdot \alpha^i \beta^j$ and $(\delta_1 + \alpha\delta_2)(\delta_1 + \beta\delta_2) = 1 + (\beta + \alpha) \cdot 0 + \alpha\beta \cdot 1$. The latter check holds immediately. As for the former check, the prover and verifier compute and output the new statement $G' \leftarrow G_1 + \alpha G_2 \in \mathbb{G}^{n/2}$ and $\overline{A} \leftarrow \sum_{i,j \in \{1,2\}} \overline{A}_{ij} \cdot \alpha^i \beta^j$. The prover privately computes and outputs the new witness vector $A' \leftarrow A_1 + \beta A_2 \in \mathbb{F}^{n/2}$. Now, it is sufficient for the verifier to check that the prover knows $A' \in \mathbb{F}^{n/2}$ such that

$$G'(A') = \overline{A}'.$$

## 5.3 The Tensor Reduction of Knowledge

By generalizing Example 4 for arbitrary tensor statements, we arrive at the tensor reduction of knowledge, which we show to be unconditionally secure. We start by defining the tensor relation which fixes the homomorphism and image as a statement and the preimage as the witness. We then construct the tensor reduction of knowledge, which for a vector space of homomorphisms $U$ and length $n$, reduces the task of checking knowledge of a preimage of a homomorphism in $U^n$ to the task of checking knowledge of a preimage of a homomorphism in $U$. In the upcoming section, we show that the tensor reduction of knowledge can be instantiated to derive reductions of knowledge for various linear algebraic statements.

Recall that the tensor relation statement consists of a homomorphism and an image, and a witness consists of the corresponding preimage. We first recall the formal definition for the tensor relation.

**Definition 12 (Tensor Relation).** *For* $\mathsf{R}$*-modules* $U$*,* $W$ *and* $V$*, such that* $U \cong (W \to V)$ *we define the tensor relation for* $U$ *as follows*

$$\mathcal{R}(U) = \left\{ ((u,v),w) \;\middle|\; \begin{array}{l} u \in U, v \in V, w \in W, \\ u(w) = v \end{array} \right\}$$

**Construction 4 (Tensor Reduction of Knowledge).** Consider field $\mathbb{F}$, length parameter $n$, and $\mathbb{F}$-modules $W$ and $V$. We construct a reduction of knowledge from $\mathcal{R}(W^n \to V)$ to $\mathcal{R}(W \to V)$. Let $\{\delta_i\}$ be an orthonormal basis for $\mathbb{F}^n$. Suppose the prover and verifier are provided statement $u = \sum_i u_i \otimes \delta_i \in (W^n \to V)$, and $v \in V$. Additionally, suppose the prover is provided an alleged witness $w = \sum_j w_j \otimes \delta_j \in W^n$ such that

$$((u,v),w) \in \mathcal{R}(W^n \to V).$$

The prover and verifier run a single tensor reduction on the equivalent statement

$$\Big(\sum_{i\in[n]} u_i \otimes \delta_i\Big)\Big(\sum_{j\in[n]} w_j \otimes \delta_j\Big) = v.$$

At the end of tensor reduction, the verifier outputs $(\alpha, \beta, v', c)$ where $c \in \mathbb{F}$. The prover and verifier compute $u' = \sum_i \alpha^i \cdot u_i$ and set the output statement to be $(u', v')$. The prover additionally computes the output witness $w' = \sum_j \beta^j \cdot w_j$ as dictated by the tensor reduction. Now, to check the original statement, it is sufficient for the verifier to check that the prover knows $w'$ such that

$$((u',v'),w') \in \mathcal{R}(W \to V).$$

**Theorem 8 (Tensor Reduction of Knowledge).** *For field* $\mathbb{F}$*, length parameter* $n$*, and* $\mathbb{F}$*-modules* $W$ *and* $V$*, Construction 4 is a reduction of knowledge from* $\mathcal{R}(W^n \to V)$ *to* $\mathcal{R}(W \to V)$*.*

*Proof Intuition.* Consider instance $u = \sum_{i\in[n]} u_i \otimes \delta_i$ and $v$. We prove knowledge soundness via tree extraction (Lemma 6). That is, we construct extractor $\chi$ that outputs $w$ such that $u(w) = v$ given a tree of accepting transcripts and corresponding output prover witnesses.

Suppose the extractor $\chi$ is provided with $n^2$ accepting transcripts $\tau_{mk}$ with the same prover's first message $\{(v_{1,ij}, v_{2,ij}) | i, j \in [n]\}$ and with randomness $(\alpha_m, \beta_{mk})$ for $m \in [n], k \in [n]$. Let $w'_{mk} \in W'$ for $k \in [n^2]$ denote the corresponding satisfying witnesses. For $m \in [n]$, the extractor solves for $w_{mj} \in W$ for $j \in [n]$ such that $\sum_{j\in[n]} \beta^j_{mk} w_{mj} = w'_{mk}$ for $k \in [n]$ using an inverse Vandermonde matrix (where invertibility is afforded by working over a field). The extractor then computes $a_{mj}$ for all $m \in [n], j \in [n]$ such that for all $i \in [n]$, $\sum_{m\in[n]} \alpha^i_m a_{mj} = v_{2,ij}$. Next, the extractor computes

$$w \leftarrow \sum_{l\in[n]} \sum_{m\in[n]} \sum_{j\in[n]} a_{mj} \cdot \alpha^l_m \cdot w_{mj} \otimes \delta_l.$$

By textbook algebra, we can show that $w$ is indeed a satisfying witness. We present a formal proof in Appendix B.4. □

## 6 Instantiating the Tensor Reduction of Knowledge

In this section, we demonstrate a unifying view of existing recursive algebraic arguments by deriving them by instantiating the tensor reduction of knowledge over the appropriate structures. We additionally derive new reductions of knowledge for bilinear forms by extending our techniques. We additionally discuss concrete modules each of these reductions can be instantiated over. In Section 7, we show how to stitch together these reductions to derive an argument for $\mathsf{NP}$.

### 6.1 Vector Commitments and Linear Forms

We start by generalizing the vector commitment relation from Section 2 and then discuss how succinctly derive the vector commitment reduction of knowledge via the tensor reduction of knowledge. We then augment the vector commitment reduction for the linear forms relation. The high level approach for both reductions is to first split all checks over size $n$ vectors into $k$ checks over size $n/k$ vectors. These checks are then folded using a random linear combination. How exactly the vectors are split and folded is abstracted away by the tensor reduction of knowledge.

Consider size parameter $n \in \mathbb{N}$, and consider $\mathbb{F}$-modules $\mathbb{G}$ and $\mathbb{H}$ for field $\mathbb{F}$ such that $\mathbb{G} \cong \mathbb{H} \to \mathbb{G} \otimes \mathbb{H}$. For public key $G \in \mathbb{G}^n \cong \mathbb{H}^n \to \mathbb{G} \otimes \mathbb{H}$, and commitment $\overline{H} \in \mathbb{G} \otimes \mathbb{H}$, suppose a verifier would like to check that a prover knows $H \in \mathbb{H}^n$ such that $\sum_i G_i \otimes H_i = \overline{H}$. For example, suppose $\mathbb{G}$ is a group of prime order $p$ where the discrete logarithm is hard, $\mathbb{H}$ and $\mathbb{F}$ are $\mathbb{Z}_p$, and $\otimes$ represents scalar multiplication. Then, this amounts to checking knowledge of the opening for a Pedersen commitment. Recall that the prover's claim can be expressed as a tensor statement $G(H) = \overline{H}$. Therefore, we can define the generalized vector commitment relation as the tensor relation over homomorphisms in $\mathbb{G}^n$.

**Definition 13 (Generalized Vector Commitment Relation).** *For length $n \in \mathbb{N}$ and group $\mathbb{G}$, the vector commitment relation is defined to be $\mathcal{R}(\mathbb{G}^n)$.*

**Construction 5 (Vector Commitment Reduction of Knowledge).** Because $\mathbb{G}^n \cong (\mathbb{G}^{n/k})^k$, we can directly apply the tensor reduction of knowledge to get a reduction from $\mathcal{R}(\mathbb{G}^n)$ to $\mathcal{R}(\mathbb{G}^{n/k})$.

Suppose that in addition to checking that the prover knows a vector opening to a commitment, the verifier would like to additionally check some public linear combination of the provers opening. In particular, for public vector $A \in \mathbb{F}^n$, and $\sigma \in \mathbb{H}$ the suppose verifier would like to additionally check that $A(H) = \sigma$ where $A(H)$ is defined to be $\sum_{i\in[n]} A_i \otimes H_i$. For example, if $\otimes$ represents scalar multiplication, then this amounts to checking an inner-product. Recall, from Section 5, that this is equivalent to checking $(G \oplus A)(H) = \overline{H} \oplus \sigma$. Therefore, because $G \oplus A \in \mathbb{G}^n \oplus \mathbb{F}^n$, we can define the linear forms relation as the tensor relation over $\mathbb{G}^n \oplus \mathbb{F}^n$.

**Definition 14 (Linear Forms Relation).** *For length $n$ and $\mathbb{F}$-module $\mathbb{G}$ for field $\mathbb{F}$, let $\mathsf{LF}_n = \mathbb{G}^n \oplus \mathbb{F}^n$. The linear forms relation is defined to be $\mathcal{R}(\mathsf{LF}_n)$.*

**Construction 6 (Linear Forms Reduction of Knowledge).** Consider $n, k \in \mathbb{N}$ such that $k$ divides $n$. We construct a reduction of knowledge from $\mathcal{R}(\mathsf{LF}_n)$ to $\mathcal{R}(\mathsf{LF}_{n/k})$. In particular, we have that

$$\mathsf{LF}_n = (\mathbb{G} \oplus \mathbb{F})^n \cong (\mathbb{G} \oplus \mathbb{F})^{(n/k)\cdot k} = (\mathsf{LF}_{n/k})^k.$$

Therefore, the prover and verifier can apply the tensor reduction of knowledge with respect to this decomposition to reduce the task of checking a statement in $\mathcal{R}(\mathsf{LF}_n)$ to the task of checking a statement in $\mathcal{R}(\mathsf{LF}_{n/k})$.

**Lemma 7 (Linear Forms Reduction of Knowledge).** *For $n, k \in \mathbb{N}$ such that $k$ divides $n$, Construction 6 is a reduction of knowledge from $\mathsf{LF}_n$ to $\mathsf{LF}_{n/k}$ with $O(n)$ prover and verifier time complexity and $O(k^2)$ communication complexity.*

As discussed in Section 2, we can construct a base case argument for $\mathsf{LF}_1$ where the prover directly sends its witness to the verifier. Thus, we have the following.

**Corollary 2 (Linear Forms Argument of Knowledge).** *Consider $n, k \in \mathbb{N}$ such that $k$ divides $n$. Let $\Pi_{\mathsf{LF}}$ be a reduction of knowledge from $\mathcal{R}(\mathsf{LF}_n)$ to $\mathcal{R}(\mathsf{LF}_{n/k})$. Let $\Pi_{\mathsf{base}}$ be an argument of knowledge for $\mathcal{R}(\mathsf{LF}_1)$. Then*

$$\Pi_{\mathsf{base}} \circ \underbrace{\Pi_{\mathsf{LF}} \circ \ldots \circ \Pi_{\mathsf{LF}}}_{\log_k n \text{ times}}$$

*is an argument of knowledge for $\mathsf{LF}_n$ with $O(n)$ prover and verifier time complexity and $O(k^2 \cdot \log_k n)$ communication complexity.*

## 6.2 Bilinear Forms

We extend our methodology to encode vector commitments and linear forms as tensor relations to develop a new reduction for bilinear forms. Recall from Section 1 that we define the bilinear forms relation, $\mathcal{R}_{\mathsf{Bil}(m,n)}$, as follows.

**Definition 15 (Bilinear Forms, Original).** *Consider $\mathbb{F}$-modules $\mathbb{G}$ and $\mathbb{H}$ for field $\mathbb{F}$. We define the bilinear forms relation, $\mathcal{R}_{\mathsf{Bil}}$, characterized by $m$ rows and $n$ columns.*

$$\mathcal{R}_{\mathsf{Bil}(m,n)} = \left\{ ((G, H, M, \overline{A}, \overline{B}, \sigma), (A, B)) \;\middle|\; \begin{array}{l} G \in \mathbb{G}^m, H \in \mathbb{H}^n, (\overline{A}, \overline{B}) \in (\mathbb{G}, \mathbb{H}), \\ M \in \mathbb{F}^{m \times n}, \sigma \in \mathbb{F}, \\ A^\top M B = \sigma, G(A) = \overline{A}, H(B) = \overline{B} \end{array} \right\}$$

Unlike vector commitments and linear forms, the bilinear forms relation cannot directly be encoded as a tensor evaluation statement. Our approach is to encode the original statement as the *related* statement,

$$(G \otimes H \oplus \mathbf{M})(A \otimes B) = (\overline{A} \otimes \overline{B} \oplus \sigma), \tag{3}$$

where $\mathbf{M} \in \mathbb{F}^m \otimes \mathbb{F}^n$ is a tensor such that $\mathbf{M}(A \otimes B) = A^\top M B$. The tensor-based statement implies checking the original statement so long as we additionally stipulate that the bilinear relation assumption holds for $(\mathbb{G}, \mathbb{F})$, $(\mathbb{H}, \mathbb{F})$, and $(\mathbb{G}, \mathbb{H})$. Then, we can utilize the tensor reduction of knowledge to reduce the corresponding tensor relation $\mathcal{R}(\mathbb{G}^m \otimes \mathbb{H}^n \oplus \mathbb{F}^m \otimes \mathbb{F}^n)$.

In practice, $\mathbb{G}$ and $\mathbb{H}$ can be prime order bilinear groups with the pairing operation acting as the tensor product between $\mathbb{G}$ and $\mathbb{H}$ and $\mathbb{G} \otimes \mathbb{H}$ denoting the target group. In this setting, the bilinear relation assumptions are equivalent to the discrete logarithm assumption over $\mathbb{G}$ and $\mathbb{H}$ and the double pairing assumption [AFG+10] over $(\mathbb{G}, \mathbb{H})$.

The computational hardness assumptions are a critical detail for arguing that checking Equation (3) is sufficient to check the original relation: the unconditional knowledge soundness property of the tensor reduction of knowledge only guarantees that the prover knows *some* satisfying witness in $\mathbb{F}^m \otimes \mathbb{F}^n$ which may be of the form $\sum_i A_i \otimes B_i$ (i.e., not a simple tensor). While this is a valid witness for the corresponding tensor statement, it is *not* a valid witness for the original statement.

However, by assuming that the commitment scheme is computationally binding, we can argue that all $A_i$ values must be the same. Leveraging this, we can show that the prover must know a single $A$ and $B$ vector that satisfies the statement. Formally, we define the bilinear forms relation as follows.

**Definition 16 (Bilinear Forms, Tensor).** *Consider size parameters $n, m \in \mathbb{N}$, and consider $\mathbb{F}$-modules $\mathbb{G}$ and $\mathbb{H}$ for field $\mathbb{F}$ such that the bilinear relation assumption holds for $(\mathbb{G}, \mathbb{F})$, $(\mathbb{H}, \mathbb{F})$, and $(\mathbb{G}, \mathbb{H})$. Let $\mathsf{BF}_{m,n} = (\mathbb{G}^m \otimes \mathbb{H}^n) \oplus (\mathbb{F}^m \otimes \mathbb{F}^n)$. We define the (tensor-based) bilinear form relation as the corresponding tensor relation $\mathcal{R}(\mathsf{BF}_{m,n})$.*

Next, we show how to recursively reduce $\mathcal{R}_{\mathsf{Bil}(m,n)}$ to $\mathcal{R}(\mathsf{LF}_m)$. To do so, we construct a reduction from $\mathcal{R}_{\mathsf{Bil}(m,n)}$ to $\mathcal{R}_{\mathsf{Bil}(m,n/k)}$, which internally uses the tensor reduction of knowledge from $\mathcal{R}(\mathsf{BF}_{m,n})$ to $\mathcal{R}(\mathsf{BF}_{m,n/k})$. We then construct a base case reduction from $\mathcal{R}_{\mathsf{Bil}(m,1)}$ to $\mathcal{R}(\mathsf{LF}_m)$.

**Construction 7 (Bilinear Forms Reduction of Knowledge).** Consider $n, k \in \mathbb{N}$ such that $k$ divides $n$. We construct a reduction of knowledge from $\mathcal{R}_{\mathsf{Bil}(m,n)}$ to $\mathcal{R}_{\mathsf{Bil}(m,n/k)}$.

Consider the following input statement-witness pair.

$$((G, H, M, \overline{A}, \overline{B}, \sigma), (A, B)) \in \mathcal{R}_{\mathsf{Bil}(m,n)}$$

The prover and verifier begin by encoding the statement and witness as

$$((G \otimes H \oplus \mathbf{M}, \overline{A} \otimes \overline{B} \oplus \sigma), A \otimes B) \in \mathcal{R}(\mathsf{BF}_{m,n})$$

where $\mathbf{M} \in \mathbb{F}^m \otimes \mathbb{F}^n$ is a tensor such that $\mathbf{M}(A \otimes B) = A^\top M B$.

We observe that

$$\mathsf{BF}_{m,n} = \mathbb{G}^m \otimes \mathbb{H}^n \oplus \mathbb{F}^m \otimes \mathbb{F}^n \cong (\mathbb{G}^m \otimes \mathbb{H}^{n/k} \oplus \mathbb{F}^m \otimes \mathbb{F}^{n/k})^k = (\mathsf{BF}_{m,n/k})^k.$$

Therefore, the prover and verifier can apply the tensor reduction of knowledge with respect to this decomposition and reduce to the task of checking a statement in $\mathcal{R}(\mathsf{BF}_{m,n/k})$. At a high level, the tensor reduction prover and verifier section $\mathbf{M}$ and $H$ into $k$ sets of columns and the prover sections $B$ into $k$ corresponding sets of rows. The prover and verifier then fold these sets using a random linear combination. By linearity, we have that the output statement is of the form

$$((G \otimes H' \oplus \mathbf{M}', \overline{A} \otimes \overline{B}' \oplus \sigma'), A \otimes B') \in \mathcal{R}(\mathsf{BF}_{m,n/k})$$

for some $H' \in \mathbb{H}^{n/k}$, $M' \in \mathbb{F}^m \otimes \mathbb{F}^{n/k}$, $\overline{B}' \in \mathbb{H}$, $\sigma' \in \mathbb{F}$, and $B' \in \mathbb{F}^{n/k}$. Together, the prover and verifier output the decoded statement witness pair

$$((G, H', M', \overline{A}, \overline{B}', \sigma'), (A, B')) \in \mathcal{R}_{\mathsf{Bil}(m,n/k)}.$$

**Lemma 8 (Bilinear Forms Reduction of Knowledge).** *For $n, k \in \mathbb{N}$ such that $k$ divides $n$, Construction 7 is a reduction of knowledge form $\mathcal{R}_{\mathsf{Bil}(m,n)}$ to $\mathcal{R}_{\mathsf{Bil}(m,n/k)}$ with $O(n)$ prover and verifier time complexity and $O(k^2)$ communication complexity.*

*Proof Intuition.* Completeness, prover and verifier time complexity, and communication complexity follow from the corresponding properties of the tensor reduction of knowledge. By the composability of reductions (Theorem 5), the extractor can retrieve a satisfying witness $\sum_i A'_i \otimes B'_i \in \mathbb{F}^n \otimes \mathbb{F}^m$ for the underlying tensor reduction of knowledge. By the bilinear relation assumption over $(\mathbb{H}, \mathbb{F})$ and $(\mathbb{G}, \mathbb{H})$, the witness must be of the form $A \otimes B$ for some efficiently computable $A \in \mathbb{F}^m$ and $B \in \mathbb{F}^n$. We provide a formal proof in Appendix B.5. □

**Construction 8 (Bilinear Forms Base Case).** We construct a reduction of knowledge from $\mathcal{R}_{\mathsf{Bil}(m,1)}$ to $\mathcal{R}(\mathsf{LF}_m)$. Consider statement $(G, H, M, \overline{A}, \overline{B}, \sigma)$ and alleged witness $(A, B)$. The prover begins the reduction by directly sending $B$ to the verifier. The verifier immediately checks that $H(B) = \overline{B}$. Additionally, as $M \in \mathbb{F}^m$ and $B \in \mathbb{F}$, the verifier computes the vector $V \leftarrow M \cdot B$. The verifier is left with checking that the prover knows $A \in \mathbb{F}^m$ such that $G(A) = \overline{A}$ and $V(A) = \sigma$. This is equivalent to checking that the prover knows $A \in \mathbb{F}^m$ such that $((G \oplus V, \overline{A} \oplus \sigma), A) \in \mathcal{R}(\mathsf{LF}_m)$.

**Lemma 9 (Bilinear Forms Base Case).** *Construction 8 is a reduction of knowledge from $\mathcal{R}_{\mathsf{Bil}(m,1)}$ to $\mathcal{R}(\mathsf{LF}_m)$ with $O(m)$ prover and verifier time complexity and $O(1)$ communication complexity.*

**Corollary 3 (Bilinear Forms to Linear Forms).** *Consider $n, k \in \mathbb{N}$ such that $k$ divides $n$. Let $\Pi_{\mathsf{Bil}}$ be the reduction of knowledge from $\mathcal{R}_{\mathsf{Bil}(m,n)}$ to $\mathcal{R}_{\mathsf{Bil}(m,n/k)}$. Let $\Pi_{\mathsf{base}}$ be the reduction of knowledge from $\mathcal{R}(\mathsf{Bil}(m,1))$ to $\mathcal{R}(\mathsf{LF}_m)$. Then*

$$\Pi_{\mathsf{base}} \circ \underbrace{\Pi_{\mathsf{Bil}} \circ \ldots \circ \Pi_{\mathsf{Bil}}}_{\log_k n \text{ times}}$$

*is a reduction of knowledge from $\mathcal{R}_{\mathsf{Bil}(m,n)}$ to $\mathcal{R}(\mathsf{LF}_m)$ with $O(n+m)$ prover and verifier time complexity and $O(k^2 \cdot \log_k n)$ communication complexity.*

## 6.3 Instantiating Spaces

In practice, we are tasked with instantiating the underlying vector spaces and corresponding tensor product. This also instantiates the corresponding computational assumptions. Because $\mathbb{F}$ has multiplication built in, when considering the tensor product against vectors over the underlying field, $\otimes$ always corresponds to the outer product: For instance, $\mathbb{G}^m \otimes \mathbb{F}^n$ is equivalent to $\mathbb{G}^{mn}$. Thus, our remaining task is to instantiate $\mathbb{G}$, $\mathbb{H}$, and $\mathbb{G} \otimes \mathbb{H}$. We highlight two options.

- *Prime Order Groups:* We can set $\mathbb{G}$ to be a group of prime order $p$ and set $\mathbb{H}$ to be the underlying field $\mathbb{F} = \mathbb{Z}_p$. In this setting, $\otimes$ corresponds to group scalar multiplication, and $\mathbb{G} \otimes \mathbb{H} \cong \mathbb{G}$. The corresponding computational assumption (if needed) corresponds to the discrete logarithm assumption.

- *Bilinear Groups:* We can set $\mathbb{G} = \mathbb{H}$ to be a symmetric bilinear group with target group $\mathbb{G}_\top$. In this case $\otimes$ corresponds to the pairing operation $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\top$, and $\mathbb{G} \otimes \mathbb{H} \cong \mathbb{G}_\top$. The corresponding computational assumption, (if needed) corresponds to the double-pairing assumption [AFG$^+$10].

We note that instantiating bilinear forms over prime order groups reverts back down to linear forms over prime order groups. In particular, we have that $\mathsf{BF}_{m,n} \cong \mathbb{G}^m \otimes \mathbb{F}^n \oplus \mathbb{F}^m \otimes \mathbb{F}^n \cong \mathbb{G}^{mn} \oplus \mathbb{F}^{mn} \cong \mathsf{LF}_{mn}$.

# 7 An Argument of Knowledge for NP

In this section, we develop an argument of knowledge for $\mathsf{NP}$ with logarithmic communication by leveraging our reductions of knowledge for linear algebraic statements. In particular, we first show that an $\mathsf{NP}$-complete relation, $\mathcal{R}_{\mathsf{ACS}}$, can be encoded as a sequence of linear and bilinear forms constraints over the same commitment. We then develop helper reductions of knowledge that reduce the task of checking many linear forms (respectively bilinear forms) over the same commitment to a single linear form (respectively bilinear form). We then apply our reductions of knowledge for linear forms and bilinear forms.

**Definition 17 (NP-Complete Relation [KMP20]).** *Consider group $\mathbb{G}$ and corresponding field $\mathbb{F}$ such that the bilinear relation assumption holds for $(\mathbb{G}, \mathbb{F})$ and $(\mathbb{G}, \mathbb{F})$. We define the $\mathsf{NP}$-complete algebraic constraint relation, $\mathcal{R}_{\mathsf{ACS}}$, characterized by $n$ variables, $m = O(n)$ constraints, and $\ell$ inputs as follows. The statement consists of public key $G \in \mathbb{G}^n$, $m$ sparse constraint matrices $M_1, \ldots, M_m \in \mathbb{F}^{n\times n}$ such that the total number of non-zero values in* all *matrices combined is $O(n)$, public inputs and outputs vector $X \in \mathbb{F}^\ell$, and witness commitment $\overline{Z} \in \mathbb{G}$. A witness vector $W \in \mathbb{F}^{n-\ell}$ is satisfying if for $Z = (X, W)$, $Z^\top M_i Z = 0$ for all $i \in [m]$, and $G(Z) = \overline{Z}$.*

We can encode $\mathcal{R}_{\mathsf{ACS}}$ to tensor relations as follows: First, the verifier can check that $((G \oplus \delta_i, \overline{Z} \oplus X_i), Z) \in \mathcal{R}(\mathsf{LF}_n)$ for all $i \in [\ell]$ to ensure that $Z$ contains public vector $X$. To check the commitment and constraints, it is sufficient for the verifier to check that the prover knows $Z_1, Z_2 \in \mathbb{F}^n$ such that $((G, G, M_i, \overline{Z}, \overline{Z}, 0), (Z_1, Z_2)) \in \mathcal{R}_{\mathsf{Bil}(n,n)}$ for all $i \in [m]$. The bilinear relation assumptions ensure that $Z$, $Z_1$ and $Z_2$ are equal.

Next, we leverage the fact that all linear form checks and all bilinear form checks are over the same commitment to reduce these checks. We formally capture the set of linear form checks over the same commitment (respectively bilinear forms) as the multiple linear forms relation (respectively multiple bilinear forms relation).

**Definition 18 (Multiple Linear Forms).** *We define $\mathcal{R}_{\mathsf{MLF}(n,\ell)}$ such that*

$$((G, (V_1, \ldots, V_\ell), (\sigma_1, \ldots, \sigma_\ell), \overline{Z}), Z) \in \mathcal{R}_{\mathsf{MLF}(n,\ell)}$$

*if and only if*

$$((G \oplus V_i, \overline{Z} \oplus \sigma_i), Z) \in \mathcal{R}(\mathsf{LF}_n)$$

*for all $i$ in $[\ell]$.*

**Definition 19 (Multiple Bilinear Forms).** *We define $\mathcal{R}_{\mathsf{MBil}(m,n,\ell)}$ such that*

$$((G, H, (M_1, \ldots, M_\ell), (\sigma_1, \ldots, \sigma_\ell), \overline{Z}_1, \overline{Z}_2), (Z_1, Z_2)) \in \mathcal{R}_{\mathsf{MBil}(m,n,\ell)}$$

*if and only if*

$$((G, H, M_i, \overline{Z}_1, \overline{Z}_2, \sigma_i), (Z_1, Z_2)) \in \mathcal{R}_{\mathsf{Bil}(m,n)}$$

*for all $i$ in $[\ell]$.*

With these relations, the above encoding can be captured as a reduction of knowledge in which the prover and verifier do not interact but rather take as input an $\mathcal{R}_{\mathsf{ACS}}$ statement-witness pair and output the corresponding tensor-based statements and witnesses in the multiple linear forms and bilinear forms relations. This step can be interpreted as a Levin reduction.

**Lemma 10 (Encoding $\mathsf{NP}$ as Tensor Relations).** *There exists a reduction of knowledge from $\mathcal{R}_{\mathsf{ACS}(m,n,\ell)}$ to $\mathcal{R}_{\mathsf{MBil}(n,n,m)} \times \mathcal{R}_{\mathsf{MLF}(n,\ell)}$ with $O(n)$ prover and verifier complexity, and no communication.*

Because all $\ell$ checks for $\mathcal{R}_{\mathsf{MLF}(n,\ell)}$ concern the same committed value, we observe that they can be batched into a single check for $\mathcal{R}(\mathsf{LF}_n)$ using a random linear combination. In particular, the verifier can send a random challenge $r \in \mathbb{F}$. Together the prover and verifier can compute $V \leftarrow \sum_i V_i \cdot r^i$ and $\sigma \leftarrow \sum_i \sigma_i \cdot r^i$ and reduce to checking that the prover knows $Z$ such that $((G \oplus V, \overline{Z} \oplus \sigma), Z) \in \mathcal{R}(\mathsf{LF}_n)$. Similarly, we can reduce multiple bilinear forms over the same commitment to a single bilinear form. Formally, we have the following reductions.

**Lemma 11 (Linear Forms Batch Reduction).** *For $n, m, \ell \in \mathbb{N}$, there exists a reduction of knowledge from $\mathcal{R}_{\mathsf{MLF}(n,\ell)}$ to $\mathcal{R}_{\mathsf{LF}(n)}$ with $O(n\ell)$ prover and verifier time complexity, and $O(1)$ communication complexity.*

**Lemma 12 (Bilinear Forms Batch Reduction).** *For $n, m, \ell \in \mathbb{N}$, there exists a reduction of knowledge from $\mathcal{R}_{\mathsf{MBil}(m,n,\ell)}$ to $\mathcal{R}_{\mathsf{Bil}(m,n)}$ with $O(mn\ell)$ prover and verifier time complexity, and $O(1)$ communication complexity.*

Putting everything together, we arrive at an argument of knowledge for $\mathsf{NP}$.

**Corollary 4 (An Argument of Knowledge for $\mathsf{NP}$).** *Let $\Pi_{\mathsf{encode}}$ be the reduction of knowledge from $\mathcal{R}_{\mathsf{ACS}(n,m,\ell)}$ to $\mathcal{R}_{\mathsf{MBil}(n,n,m)} \times \mathcal{R}_{\mathsf{MLF}(n,\ell)}$ (Lemma 10). Let $\Pi_{\mathsf{batchLF}}$ be the batching scheme for linear forms (Lemma 11). Let $\Pi_{\mathsf{batchBil}}$ be the batching scheme for bilinear forms (Lemma 12). Let $\Pi_{\mathsf{LF}_n}$ be the argument of knowledge for $\mathcal{R}(\mathsf{LF}_n)$ with decomposition parameter $k$ (Construction 2). Let $\Pi_{\mathsf{Bil}(n,n)}$ be the reduction of knowledge from $\mathcal{R}_{\mathsf{Bil}(n,n)}$ to $\mathcal{R}_{\mathsf{LF}(n)}$ with decomposition parameter $k$ (Corollary 3). Let $\Pi_{\mathsf{id}}$ be the identity reduction of knowledge (i.e., the prover and verifier output their inputs). Let $\Pi_{\mathsf{foldBool}}$ be a 2-folding scheme for $\mathcal{R}_\top$ (i.e., the verifier outputs* $\mathsf{true}$ *if both its inputs are* $\mathsf{true}$*). Then*

$$\Pi_{\mathsf{foldBool}} \circ (\Pi_{\mathsf{id}} \times \Pi_{\mathsf{LF}_m}) \circ (\Pi_{\mathsf{LF}_n} \times \Pi_{\mathsf{Bil}(n,n)}) \circ (\Pi_{\mathsf{batchLF}} \times \Pi_{\mathsf{batchBil}}) \circ \Pi_{\mathsf{encode}}$$

*is an argument of knowledge for $\mathcal{R}_{\mathsf{ACS}(n,m,\ell)}$ with $O(n)$ prover and verifier time complexity, and $O(k^2 \log_k n)$ communication complexity.*

## Acknowledgments

We thank Jonathan Bootle, Quang Dao, Vipul Goyal, Yael Kalai, Jonathan Lee, Srinath Setty, Elaine Shi, and Zoe Wellner for comments on earlier versions of this work. Abhiram Kothapalli was supported by a fellowship from Protocol Labs, a gift from Bosch, NSF Grant No. 1801369, and by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

## References

[AC20] Thomas Attema and Ronald Cramer. Compressed sigma-protocol theory and practical application to plug & play secure algorithmics. In *Annual International Cryptology Conference*, pages 513–543. Springer, 2020.

[ACR20] Thomas Attema, Ronald Cramer, and Matthieu Rambaud. Compressed sigma-protocols for bilinear group arithmetic circuits and applications. Technical report, Cryptology ePrint Archive, Report 2020/1447, 2020.

[AFG$^+$10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In *Annual Cryptology Conference*, pages 209–236. Springer, 2010.

[BBB$^+$18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE S&P*, 2018.

[BCC$^+$16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT*, 2016.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, 2012.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 111–120, 2013.

[BCL$^+$21] Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data without succinct arguments. In *Annual International Cryptology Conference*, pages 681–710. Springer, 2021.

[BCS21] Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Sumcheck arguments and their applications. In *Advances in Cryptology–CRYPTO 2021*, 2021.

[BDFG20] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Halo infinite: Recursive zk-snarks from any additive polynomial commitment scheme. Cryptology ePrint Archive, Report 2020/1536, 2020.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In *EUROCRYPT*, 2020.

[BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. Recursive proof composition without a trusted setup. *Cryptol. ePrint Arch., Tech. Rep*, 1021:2019, 2019.

[BMM$^+$21] Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. Proofs for inner pairing products and applications. In *Advances in Cryptology–ASIACRYPT 2021*, 2021.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BSCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *TCC*, 2016.

[CFF$^+$21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: a toolbox for more efficient universal and updatable zksnarks and commit-and-prove extensions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 3–33. Springer, 2021.

[CHJ$^+$22] Heewon Chung, Kyoohyung Han, Chanyang Ju, Myungsun Kim, and Jae Hong Seo. Bulletproofs+: Shorter proofs for a privacy-enhanced distributed ledger. *IEEE Access*, 10:42067–42082, 2022.

[CHM$^+$20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *EUROCRYPT*, 2020.

[CNR$^+$22] Matteo Campanelli, Anca Nitulescu, Carla Rafols, Alexandros Zacharakis, and Arantxa Zapico. Linear-map vector commitments and their practical applications. *Cryptology ePrint Archive*, 2022.

[DLFKP16] Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. Cinderella: Turning shabby X. 509 certificates into elegant anonymous credentials with the magic of verifiable computation. In *IEEE S&P*, 2016.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In *CRYPTO*, 2018.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *EUROCRYPT*, 1986.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT*, 2013.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SICOMP*, 18(1), 1989.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, 2011.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019.

[KMP20] Abhiram Kothapalli, Elisaweta Masserova, and Bryan Parno. Poppins: A direct construction for asymptotically optimal zkSNARKs. Cryptology ePrint Archive, Report 2020/1318, 2020.

[KMS$^+$16] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE S&P*, 2016.

[KST22] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. In *Advances in Cryptology–CRYPTO*, 2022.

[Lee21] Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In *Theory of Cryptography: 19th International Conference, TCC 2021*, 2021.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *JACM*, 39(4), 1992.

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, 1991.

[RZ21] Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable snarks. In *Advances in Cryptology–CRYPTO 2021*, 2021.

[RZ22] Carla Ràfols and Alexandros Zacharakis. Folding schemes with selective verification. Cryptology ePrint Archive, Paper 2022/1576, 2022.

[SCG$^+$14] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *IEEE S&P*, 2014.

[Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *JACM*, 27(4), 1980.

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

[TKPS21] Ioanna Tzialla, Abhiram Kothapalli, Bryan Parno, and Srinath Setty. Transparency dictionaries with succinct proofs of correct operation. *Cryptology ePrint Archive*, 2021.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *TCC*, 2008.

[WTS$^+$18] Riad S Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *IEEE S&P*, 2018.

[ZKP15] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. IntegriDB: Verifiable SQL for outsourced databases. In *CCS*, 2015.

## A Supplementary Definitions

We start by defining rings and modules. We then define the direct sum and tensor product operations for modules over rings, which is used throughout our development.

**Definition 20 (Ring).** *A ring is a set* $\mathsf{R}$ *together with two binary operations* $+$ *and* $\cdot$ *over* $\mathsf{R}$ *that satisfy the following conditions.*

*(i)* $(\mathsf{R}, +)$ *is a commutative group.*

*(ii) Associativity: For all* $a, b, c \in \mathsf{R}$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

*(iii) Distributivity: For all* $a, b, c \in \mathsf{R}$, $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$ *and* $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.

*The ring is commutative if* $a \cdot b = b \cdot a$ *for all* $a, b \in \mathsf{R}$*. The ring is unital if it contains an identity element (denoted* $1$*) such that* $1 \cdot a = a \cdot 1 = a$ *for all* $a \in \mathsf{R}$.

**Definition 21 (Module).** *Consider commutative ring* $\mathsf{R}$*. An* $\mathsf{R}$*-module is a set* $M$ *together with binary operations* $+$ *from* $M \times M$ *to* $M$ *and* $\cdot$ *from* $\mathsf{R} \times M$ *to* $M$ *that satisfy the following conditions.*

*(i)* $(M, +)$ *is a commutative group.*

*(ii) For all* $r, s \in R$ *and* $m, n \in M$*, we have that* $(r + s) \cdot m = r \cdot m + s \cdot m$, $(r \cdot s) \cdot m = r \cdot (s \cdot m)$, *and* $r \cdot (m + n) = r \cdot m + r \cdot n$.

*(iii) If* $\mathsf{R}$ *is unital, then* $1 \cdot m = m$ *for all* $m \in M$.

Given the definition of rings and modules, we define the direct sum and tensor product operations over rings and modules via their universality properties.

**Definition 22 (Direct Sum).** *Consider* $\mathsf{R}$*-modules* $U_1$ *and* $U_2$*. A direct sum for* $U_1$ *and* $U_2$*, denoted* $\oplus$*, is any operation mapping from* $U_1 \times U_2$ *into a new module, denoted* $U_1 \oplus U_2$*, such that for natural embedding* $\iota_i \in U_i \to U_1 \oplus U_2$ *(where* $\iota_1(u_1) \mapsto u_1 \oplus 0$ *and* $\iota_2(u_2) \mapsto 0 \oplus u_2$*), there exists a unique linear map* $\varphi \in U_1 \oplus U_2 \to V$ *such that for any linear maps* $\varphi_i \in U_i \to V$ *the following diagram commutes* [3] *for* $i \in \{1, 2\}$.

$$\begin{tikzcd}
U_i \arrow[r, "\varphi_i"] \arrow[d, "\iota_i"'] & V \\
U_1 \oplus U_2 \arrow[ur, "\varphi"] &
\end{tikzcd}$$

**Example 5 (Direct Sum).** Consider field $\mathbb{F}$ and vector spaces $\mathbb{F}^m$ and $\mathbb{F}^n$. Vector concatenation mapping $u_1 \in \mathbb{F}^m$ and $u_2 \in \mathbb{F}^n$ to $(u_1, u_2) \in \mathbb{F}^{m+m}$ is valid direct sum over $\mathbb{F}^m$ and $\mathbb{F}^n$. This is because for any linear maps $\varphi_1$ and $\varphi_2$, we have that for $\varphi = (\varphi_1, \varphi_2)$

$$\varphi \circ \iota_1(u_1) = (\varphi_1, \varphi_2)(u_1, 0) = \varphi_1(u_1)$$
$$\varphi \circ \iota_2(u_2) = (\varphi_1, \varphi_2)(0, u_2) = \varphi_2(u_2).$$

[3] A diagram is said to commute if all paths along the arrows lead to the same result

**Definition 23 (Tensor Product).** *Consider* $\mathsf{R}$*-modules* $U_1$ *and* $U_2$*. A tensor product for* $U_1$ *and* $U_2$*, denoted* $\otimes$*, is any operation mapping from* $U_1 \times U_2$ *into a new module, denoted* $U_1 \otimes U_2$*, such that for any bilinear map* $\varphi \in U_1 \times U_2 \to V$ *there exists a unique linear map* $\widetilde{\varphi} \in U_1 \otimes U_2 \to V$ *such that the following diagram commutes.*

$$\begin{tikzcd} U_1 \times U_2 \arrow[r, "\varphi"] \arrow[d, "\otimes"] & V \\ U_1 \otimes U_2 \arrow[ur, "\widetilde{\varphi}"] & \end{tikzcd}$$

**Example 6 (Tensor Product).** Consider field $\mathbb{F}$ and vector spaces $\mathbb{F}^m$ and $\mathbb{F}^n$. The outer-product mapping $u_1 \in \mathbb{F}^m$ and $u_2 \in \mathbb{F}^n$ to matrix $u_1 u_2^\top \in \mathbb{F}^{m \times n}$ is a valid tensor product over $\mathbb{F}^m$ and $\mathbb{F}^n$. This is because for any bilinear map from vectors $(u_1, u_2)$, we can derive a corresponding linear map from the matrix $u_1 u_2^\top$ that behaves identically.

## B Deferred Proofs

### B.1 Proof of Theorem 5 (Sequential Composition)

**Theorem.** *Consider binary relations* $\mathcal{R}_1, \mathcal{R}_2,$ *and* $\mathcal{R}_3$*. For reductions of knowledge* $\Pi_1 = (\mathcal{G}_1, \mathcal{P}_1, \mathcal{V}_1) : \mathcal{R}_1 \to \mathcal{R}_2$ *and* $\Pi_2 = (\mathcal{G}_2, \mathcal{P}_2, \mathcal{V}_2) : \mathcal{R}_2 \to \mathcal{R}_3$*, we have that* $\Pi_2 \circ \Pi_1 = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is a reduction of knowledge from* $\mathcal{R}_1$ *to* $\mathcal{R}_3$ *where*

$$\begin{aligned} \mathcal{G}(\lambda) &= (\mathcal{G}_1(\lambda), \mathcal{G}_2(\lambda)) \\ \mathcal{P}((\mathsf{pp}_1, \mathsf{pp}_2), u_1, w_1) &= \mathcal{P}_2(\mathsf{pp}_2, \mathcal{P}_1(\mathsf{pp}_1, u_1, w_1)) \\ \mathcal{V}((\mathsf{pp}_1, \mathsf{pp}_2), u_1) &= \mathcal{V}_2(\mathsf{pp}_2, \mathcal{V}_1(\mathsf{pp}_1, u_1, w_1)) \end{aligned}$$

*Proof.* Completeness and public reducibility follow by observation. As for knowledge soundness, consider arbitrary expected polynomial-time adversaries $\mathcal{A}$ and $\mathcal{P}^*$. Let $\mathsf{pp} = (\mathsf{pp}_1, \mathsf{pp}_2) \leftarrow \mathcal{G}(\lambda)$ and let $(u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$. Suppose that

$$\Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle (\mathsf{pp}, u_1, \mathsf{st}) \in \mathcal{R}_3] = \epsilon. \tag{4}$$

We must construct an expected polynomial-time extractor $\mathcal{E}$ such that

$$\Pr[(u_1, \mathcal{E}(\mathsf{pp}, u_1, \mathsf{st})) \in \mathcal{R}_1] = \epsilon - \mathsf{negl}(\lambda).$$

At a high level, we proceed as follows: We leverage $\mathcal{A}$ and $\mathcal{P}^*$ to construct expected polynomial-time adversaries $\mathcal{A}_2$ and $\mathcal{P}_2^{**}$ for protocol $\Pi_2$ that succeed in producing a satisfying witness $w_3$ with probability $\epsilon$. By the knowledge soundness of $\Pi_2$, this implies an expected polynomial-time extractor $\mathcal{E}_2$ that succeeds in producing a satisfying intermediate witness $w_2$ with probability $\epsilon - \mathsf{negl}(\lambda)$. We then leverage $\mathcal{E}_2$ (in addition to $\mathcal{A}$ and $\mathcal{P}^*$) to construct expected polynomial-time adversaries $\mathcal{A}_1$ and $\mathcal{P}_1^{**}$ for protocol $\Pi_1$ that succeed in producing a satisfying witness $w_2$ with probability $\epsilon - \mathsf{negl}(\lambda)$. This implies an expected polynomial-time extractor $\mathcal{E}_1$ that succeeds in producing witness $w_1$ with probability $\epsilon - \mathsf{negl}(\lambda)$.

Indeed, we start by constructing adversaries $\mathcal{A}_2$ and $\mathcal{P}_2^{**}$ for $\Pi_2$. By construction, we have that $\mathcal{V}$ first runs $\mathcal{V}_1$ to produce an intermediate statement $u_2$ and then runs $\mathcal{V}_2$ with input $u_2$. As such, we have that $\mathcal{P}^*$ first runs some $\mathcal{P}_1^*$ and then runs some $\mathcal{P}_2^*$ such that $\mathcal{P}_1^*$ interacts with $\mathcal{V}_1$ and then passes some state to $\mathcal{P}_2^*$ which interacts with $\mathcal{V}_2$ before the two parties collectively produce the output $(u_3, w_3)$. Therefore, for $(u_2, \mathsf{st}_2) \leftarrow \langle \mathcal{P}_1^*, \mathcal{V}_1 \rangle(\mathsf{pp}_1, u_1, \mathsf{st})$, by Equation (4) we have that

$$\Pr[\langle \mathcal{P}_2^*, \mathcal{V}_2 \rangle(\mathsf{pp}_2, u_2, \mathsf{st}_2) \in \mathcal{R}_3] = \epsilon. \tag{5}$$

Thus, we define $\mathcal{A}_2$ and $\mathcal{P}_2^{**}$ as follows.

$\underline{\mathcal{A}_2(\mathsf{pp}_2) \rightarrow (u_2, \mathsf{st}_2)}$:

1. Compute $\mathsf{pp}_1 \leftarrow \mathcal{G}_1(\lambda)$.
2. Compute $(u_1, \mathsf{st}) \leftarrow \mathcal{A}((\mathsf{pp}_1, \mathsf{pp}_2))$.
3. Compute $(u_2, \mathsf{st}_2) \leftarrow \langle \mathcal{P}_1^*, \mathcal{V}_1 \rangle(\mathsf{pp}_1, u_1, \mathsf{st})$.
4. Output $(u_2, \mathsf{st}_2)$.

$\underline{\mathcal{P}_2^{**}(\mathsf{pp}_2, u_2, \mathsf{st}_2) \rightarrow (u_3, w_3)}$:

1. Run $\mathcal{P}_2^*(\mathsf{pp}_2, u_2, \mathsf{st}_2)$, which, at the end of interaction, produces output $(u_3, w_3)$.
2. Output $(u_3, w_3)$.

Suppose now that $\mathsf{pp}_2 \leftarrow \mathcal{G}_2(\lambda)$ and $(u_2, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{pp}_2)$. By construction, $\mathcal{A}_2$ produces the same distribution of outputs as $\langle \mathcal{P}_1^*, \mathcal{V}_1 \rangle(\mathsf{pp}_1, u_1, \mathsf{st})$. Then, because $\mathcal{P}_2^{**}$ runs $\mathcal{P}_2^*$, by Equation (5), we have that

$$\Pr[\langle \mathcal{P}_2^{**}, \mathcal{V}_2 \rangle(\mathsf{pp}_2, u_2, \mathsf{st}_2) \in \mathcal{R}_3] = \epsilon. \tag{6}$$

Then, by the knowledge soundness of $\Pi_2$, Equation (6) implies that there exists expected polynomial-time extractor $\mathcal{E}_2$ such that

$$\Pr[(u_2, \mathcal{E}_2(\mathsf{pp}_2, u_2, \mathsf{st}_2)) \in \mathcal{R}_2] = \epsilon - \mathsf{negl}(\lambda). \tag{7}$$

We leverage $\mathcal{E}_2$ to construct adversaries $\mathcal{A}_1$ and $\mathcal{P}_1^{**}$ for $\Pi_1$ as follows.

$\underline{\mathcal{A}_1(\mathsf{pp}_1) \rightarrow (u_1, \mathsf{st}_1)}$:

1. Compute $\mathsf{pp}_2 \leftarrow \mathcal{G}_2(\lambda)$.
2. Compute $(u_1, \mathsf{st}) \leftarrow \mathcal{A}((\mathsf{pp}_1, \mathsf{pp}_2))$.
3. Let $\mathsf{st}_1 \leftarrow (\mathsf{st}, \mathsf{pp}_2)$
4. Output $(u_1, \mathsf{st}_1)$.

$\underline{\mathcal{P}_1^{**}(\mathsf{pp}_1, u_1, \mathsf{st}_1) \rightarrow (u_3, w_3)}$:

1. Parse $\mathsf{st}_1$ as $(\mathsf{st}, \mathsf{pp}_2)$.

2. Run $\mathcal{P}_1^*(\mathsf{pp}_1, u_1, \mathsf{st})$, which, at the end of interaction produces intermediate state $\mathsf{st}_2$. Record the corresponding interaction transcript as $\mathsf{tr}$.

3. Use deterministic polynomial-time function $\varphi$ guaranteed by the public reducibility property of $\Pi_1$ to compute $u_2 \leftarrow \varphi(\mathsf{pp}_1, u_1, \mathsf{tr})$.

4. Compute $w_2 \leftarrow \mathcal{E}_2(\mathsf{pp}_2, u_2, \mathsf{st}_2)$.

5. Output $(u_2, w_2)$.

Suppose now that $\mathsf{pp}_1 \leftarrow \mathcal{G}_1(\lambda)$ and $(u_1, \mathsf{st}_1) \leftarrow \mathcal{A}_1(\mathsf{pp}_1)$. By construction of $\mathcal{A}_1$ and $\mathcal{P}_1^{**}$, the extractor $\mathcal{E}_2$ run by $\mathcal{P}_1^{**}$ is provided the same distribution of inputs as the extractor $\mathcal{E}_2$ in Equation (7). Thus, by Equation (7), we have that

$$\Pr[\langle \mathcal{P}_1^{**}, \mathcal{V}_1 \rangle(\mathsf{pp}_1, u_1, \mathsf{st}_1) \in \mathcal{R}_2] = \epsilon - \mathsf{negl}(\lambda). \tag{8}$$

Then, by the knowledge soundness of $\Pi_1$, Equation (8) implies that there exists expected polynomial-time extractor $\mathcal{E}_1$ such that

$$\Pr[(u_1, \mathcal{E}_1(\mathsf{pp}_1, u_1, \mathsf{st}_1)) \in \mathcal{R}_1] = \epsilon - \mathsf{negl}(\lambda). \tag{9}$$

Thus, we can construct the desired extractor $\mathcal{E}$ as follows.

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

1. Parse $\mathsf{pp}$ as $(\mathsf{pp}_1, \mathsf{pp}_2)$.

2. Let $\mathsf{st}_1 \leftarrow (\mathsf{st}, \mathsf{pp}_2)$

3. Compute $w_1 \leftarrow \mathcal{E}_1(\mathsf{pp}_1, u_1, \mathsf{st}_1)$.

4. Output $w_1$.

Suppose now that $\mathsf{pp} \leftarrow \mathcal{G}(\lambda)$ and $(u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$. By construction of $\mathcal{E}$, the extractor $\mathcal{E}_1$ run by $\mathcal{E}$ is provided the same distribution of inputs as the extractor $\mathcal{E}_1$ in Equation (9). Thus, by Equation (9), we have that

$$\Pr[(u_1, \mathcal{E}(\mathsf{pp}, u_1, \mathsf{st})) \in \mathcal{R}_1] = \epsilon - \mathsf{negl}(\lambda).$$

□

### B.2 Proof of Theorem 6 (Parallel Composition)

**Theorem.** *Consider relations $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$. For reductions of knowledge $\Pi_1 = (\mathcal{G}_1, \mathcal{P}_1, \mathcal{V}_1) : \mathcal{R}_1 \to \mathcal{R}_2$ and $\Pi_2 = (\mathcal{G}_2, \mathcal{P}_2, \mathcal{V}_2) : \mathcal{R}_3 \to \mathcal{R}_4$, we have that $\Pi_1 \times \Pi_2 = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a reduction of knowledge from $\mathcal{R}_1 \times \mathcal{R}_3$ to $\mathcal{R}_2 \times \mathcal{R}_4$ where*

$$\begin{aligned} \mathcal{G}(\lambda) &= (\mathcal{G}_1(\lambda), \mathcal{G}_2(\lambda)) \\ \mathcal{P}((\mathsf{pp}_1, \mathsf{pp}_2), (u_1, u_3), (w_1, w_3)) &= (\mathcal{P}_1(\mathsf{pp}_1, u_1, w_1), \mathcal{P}_2(\mathsf{pp}_2, u_3, w_3)) \\ \mathcal{V}((\mathsf{pp}_1, \mathsf{pp}_2), (u_1, u_3)) &= (\mathcal{V}_1(\mathsf{pp}_1, u_1), \mathcal{V}_2(\mathsf{pp}_2, u_3)) \end{aligned}$$

*Proof.* Completeness and public reducibility follow by observation. As for knowledge soundness, consider arbitrary expected polynomial-time adversaries $\mathcal{A}$ and $\mathcal{P}^*$. Let $\mathsf{pp} = (\mathsf{pp}_1, \mathsf{pp}_2) \leftarrow \mathcal{G}(\lambda)$ and let $(u_1, u_3) \leftarrow \mathcal{A}(\mathsf{pp})$. Suppose that

$$\Pr[\langle \mathcal{P}^*, \mathcal{V}\rangle(\mathsf{pp}, (u_1, u_3), \mathsf{st}) \in \mathcal{R}_2 \times \mathcal{R}_4] = \epsilon.$$

We must construct expected polynomial-time extractor $\mathcal{E}$ such that

$$\Pr[((u_1, u_3), \mathcal{E}(\mathsf{pp}, (u_1, u_3), \mathsf{st})) \in \mathcal{R}_1 \times \mathcal{R}_3] = \epsilon - \mathsf{negl}(\lambda).$$

At a high level, we proceed as follows: For $i \in \{1, 2\}$, we leverage $\mathcal{A}$ and $\mathcal{P}^*$ to construct expected polynomial time adversaries $\mathcal{A}_i$ and $\mathcal{P}_i^*$ for protocol $\Pi_i$ that succeeds in producing a satisfying output witness with probability $\epsilon$. By the knowledge soundness of $\Pi_i$, this implies an expected polynomial-time extractor $\mathcal{E}_i$ that succeeds in producing a satisfying input witness with probability $\epsilon - \mathsf{negl}(\lambda)$. Together these extractors imply the desired extractor $\mathcal{E}$.

Indeed, we construct adversaries $\mathcal{A}_1$ and $\mathcal{P}_1^*$ as follows.

$\underline{\mathcal{A}_1(\mathsf{pp}_1) \to (u_1, \mathsf{st}_1)}$:

1. Compute $\mathsf{pp}_2 \leftarrow \mathcal{G}_2(\lambda)$.
2. Compute $((u_1, u_3), \mathsf{st}) \leftarrow \mathcal{A}((\mathsf{pp}_1, \mathsf{pp}_2))$.
3. Output statement $u_1$ and state $\mathsf{st}_1 \leftarrow (\mathsf{pp}_2, u_3, \mathsf{st})$.

$\underline{\mathcal{P}_1(\mathsf{pp}_1, u_1, \mathsf{st}_1) \to (u_2, w_2)}$:

1. Parse $\mathsf{st}_1$ as $(\mathsf{pp}_2, u_3, \mathsf{st})$.
2. Run $\mathcal{P}^*((\mathsf{pp}_1, \mathsf{pp}_2), (u_1, u_3), \mathsf{st})$. At the end of interaction $\mathcal{P}^*$ produces statement pair $(u_2, u_4)$ and corresponding witness pair $(w_2, w_4)$.
3. Output $(u_2, w_2)$.

Suppose now that $\mathsf{pp}_1 \leftarrow \mathcal{G}_1(\lambda)$ and $(u_1, \mathsf{st}_1) \leftarrow \mathcal{A}_1(\mathsf{pp}_1)$. By construction, we have that

$$\Pr[\langle \mathcal{P}_1^*, \mathcal{V}_1\rangle(\mathsf{pp}_1, u_1, \mathsf{st}_1) \in \mathcal{R}_2] = \epsilon. \tag{10}$$

Then, by the knowledge soundness of $\Pi_1$, Equation 10 implies that there exists expected polynomial-time extractor $\mathcal{E}_1$ such that

$$\Pr[(u_1, \mathcal{E}_1(\mathsf{pp}_1, u_1, \mathsf{st}_1)) \in \mathcal{R}_1] = \epsilon - \mathsf{negl}(\lambda). \tag{11}$$

Similarly, we can design adversaries $\mathcal{A}_2$ and $\mathcal{P}_2^*$ for $\Pi_2$ such that $\mathcal{P}_2^*$ succeeds with probability $\epsilon$. Then, by the knowledge soundness of $\Pi_2$, there exists expected polynomial-time extractor $\mathcal{E}_2$ such that for $\mathsf{pp}_2 \leftarrow \mathcal{G}(\lambda)$ and $(u_3, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{pp}_2)$ we have that

$$\Pr[(u_3, \mathcal{E}_2(\mathsf{pp}_2, u_3, \mathsf{st}_2)) \in \mathcal{R}_3] = \epsilon - \mathsf{negl}(\lambda). \tag{12}$$

Thus, we can construct the desired extractor $\mathcal{E}$ as follows.

$\underline{\mathcal{E}((\mathsf{pp}_1, \mathsf{pp}_2), (u_1, u_3), \mathsf{st}) \to (w_1, w_3)}$:

1. Compute $w_1 \leftarrow \mathcal{E}_1(\mathsf{pp}_1, u_1, (\mathsf{pp}_2, u_3, \mathsf{st}))$.
2. Compute $w_3 \leftarrow \mathcal{E}_2(\mathsf{pp}_2, u_2, (\mathsf{pp}_1, u_1, \mathsf{st}))$.
3. Output $(w_1, w_3)$.

Suppose now that $\mathsf{pp} \leftarrow \mathcal{G}(\lambda)$ and $((u_1, u_3), \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$. By construction of $\mathcal{E}$ the extractors $\mathcal{E}_1$ and $\mathcal{E}_2$ are provided the same distribution of inputs as the extractors in Equations (11) and (12) respectively. Therefore, by Equations (11) and (12), we have that

$$\Pr[((u_1, u_3), \mathcal{E}(\mathsf{pp}, (u_1, u_3), \mathsf{st})) \in \mathcal{R}_1 \times \mathcal{R}_3] = \epsilon - \mathsf{negl}(\lambda).$$

□

## B.3 Proof of Lemma 6 (Tree Extraction)

**Lemma.** *Consider an $m$-round public-coin interactive protocol $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ that satisfies the interface described in Definition 8 and satisfies completeness. Then $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a reduction of knowledge if there exists* PPT *extractor $\chi$ such that for all instances $u_1$, outputs a satisfying witness $w_1$ with probability $1 - \mathsf{negl}(\lambda)$, given an $(n_1, \ldots, n_m)$-tree of accepting transcripts for $u_1$ where the verifier's randomness is sampled from space $Q$ such that $|Q| = O(2^\lambda)$, and $\prod_i n_i = \mathsf{poly}(\lambda)$.*

*Proof.* Public reducibility follows from the completeness and public-coin properties. As for knowledge soundness, our proof closely follows that of [KST22] and [BCC$^+$16]. At a high level, we construct an expected polynomial-time extractor $\mathcal{E}$ that repeatedly runs the malicious prover $\mathcal{P}^*$ and collects corresponding accepting transcripts and associated output statement-witness pairs. The extractor then passes these collected transcripts to $\chi$ which retrieves the desired witness by assumption.

In more detail, consider an $m$-round public-coin interactive protocol $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ that presumably reduces from $\mathcal{R}_1$ to $\mathcal{R}_2$. Suppose there exists adversary $\mathcal{P}^*$ that succeeds with probability $\epsilon$ and extractor $\chi$ that succeeds with probability $1 - \mathsf{negl}(\lambda)$. We are tasked with constructing extractor $\mathcal{E}$ that succeeds with probability $\epsilon - \mathsf{negl}(\lambda)$. Indeed, let $\mathsf{pp} \leftarrow \mathcal{G}(\lambda)$, and $(u_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$. We construct extractor $\mathcal{E}$ as follows.

$\underline{\mathcal{E}(\mathsf{pp}) \to w_1}$:

1. Compute $\mathsf{tree} \leftarrow \mathsf{TreeGen}(1)$.
2. If $\mathsf{tree}$ is not a valid $(n_1, \ldots, n_m)$-tree (i.e., there are collisions in the verifier's randomness), output $\perp$.
3. Output $w_1 \leftarrow \mathcal{X}(\mathsf{tree})$

where we define function TreeGen as follows.

$\underline{\mathsf{TreeGen}(i) \to \mathsf{tree}}$:

1. Sample fresh verifier randomness $r_i$ for round $i$.
2. Compute the interaction $\langle \mathcal{P}^*, \mathcal{V} \rangle(\mathsf{pp}, u_1, \mathsf{st})$ up to round $i$.

3. If $i = m$, then the interaction is complete. Let $\mathsf{tr}$ be the corresponding transcript. Let $(u_2, w_2)$ be the verifier's output statement and the prover's corresponding output witness. If $(u_2, w_2) \in \mathcal{R}_2$, output $\{(\mathsf{tr}, (u_2, w_2))\}$. Otherwise output $\perp$.

4. Compute $\mathsf{tree} \leftarrow \mathsf{TreeGen}(i+1)$. If $\mathsf{tree} = \perp$, then return $\perp$.

5. Repeatedly run $\mathsf{TreeGen}(i+1)$ to retrieve $n_{i+1} - 1$ additional accepting subtrees. Append all results to $\mathsf{tree}$ and output $\mathsf{tree}$.

We now argue that $\mathcal{E}$ succeeds with probability $\epsilon - \mathsf{negl}(\lambda)$. Let $E_{\mathsf{nempty}}$ denote the event that $\mathsf{TreeGen}$ outputs $\mathsf{tree} \neq \perp$ in less than $T$ time steps (we will specify $T$ later). Given $E_{\mathsf{nempty}}$, let $E_{\mathsf{valid}}$ denote the event that $\mathsf{tree}$ is valid (i.e., there are no collisions in the verifer's randomness). Given $E_{\mathsf{nempty}}$ and $E_{\mathsf{valid}}$, let $E_{\mathsf{ext}}$ denote the event that $\chi$ successfully extracts a valid witness with input $\mathsf{tree}$. Then, we have that $\mathcal{E}$ succeeds with probability

$$P_{\mathcal{E}} = \Pr[E_{\mathsf{nempty}}] \cdot \Pr[E_{\mathsf{valid}}] \cdot \Pr[E_{\mathsf{ext}}].$$

We will compute each of these probabilities.

To compute $\Pr[E_{\mathsf{nempty}}]$ we observe that $\mathsf{TreeGen}(1)$ succeeds so long as its first call to $\mathsf{TreeGen}(2)$ succeeds. Likewise, $\mathsf{TreeGen}(2)$ succeeds so long as its first call to $\mathsf{TreeGen}(3)$ succeeds. Chaining these assertions, we have that $\mathsf{TreeGen}(1)$ succeeds if $\mathsf{TreeGen}(m)$ succeeds, which happens with with probability $\epsilon$. Moreover, the expected number of times $\mathsf{TreeGen}(i)$ calls $\mathsf{TreeGen}(i+1)$ is

$$\begin{aligned}
&1 + \Pr[\text{First call to } \mathsf{TreeGen}(i+1) \text{ succeeds}] \cdot \frac{n_{i+1} - 1}{\Pr[\mathsf{TreeGen}(i+1) \text{ succeeds}]} \\
&= 1 + \epsilon \cdot \frac{n_{i+1} - 1}{\epsilon} \\
&= n_{i+1}.
\end{aligned}$$

Hence, the total runtime of $\mathsf{TreeGen}(1)$ is expected to be $t = O(\prod_{i=1}^{m} n_i)$ which is bounded by $\mathsf{poly}(\lambda)$ by assumption. Then, by Markov's inequality, we have that $\mathsf{TreeGen}(1)$ runs for time $T > t$ with probability $\frac{t}{T}$. Thus, we have that

$$\Pr[E_{\mathsf{nempty}}] = \left(1 - \frac{t}{T}\right) \cdot \epsilon$$

Given $E_{\mathsf{nempty}}$, we have that $\mathsf{TreeGen}(1)$ runs for at most $T$ steps. But this means that there are at most $T$ random challenges produced for the verifier implying that the probability of collision is at most $\frac{T^2}{|Q|}$. Thus, we have

$$\Pr[E_{\mathsf{valid}}] = 1 - \frac{T^2}{|Q|}.$$

Finally, given $E_{\mathsf{nempty}}$ and $E_{\mathsf{valid}}$, we have that $\Pr[E_{\mathsf{ext}}]$ is $1 - \mathsf{negl}(\lambda)$ by assumption.

Now, setting $T = \sqrt[3]{|Q|}$, we have

$$\begin{aligned}
P_{\mathcal{E}} &= \Pr[E_{\mathsf{nempty}}] \cdot \Pr[E_{\mathsf{valid}}] \cdot \Pr[E_{\mathsf{ext}}] \\
&= \left(1 - \frac{t}{T}\right) \cdot \epsilon \cdot \left(1 - \frac{T^2}{|Q|}\right) \cdot (1 - \mathsf{negl}(\lambda))
\end{aligned}$$

$$= \left(1 - \frac{t}{\sqrt[3]{|Q|}}\right) \cdot \epsilon \cdot \left(1 - \frac{1}{\sqrt[3]{|Q|}}\right) \cdot (1 - \mathsf{negl}(\lambda))$$
$$= \epsilon - \mathsf{negl}(\lambda).$$

Thus, we have that $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ satisfies knowledge soundness. □

## B.4 Proof of Theorem 8 (Tensor Reduction of Knowledge)

**Theorem (Tensor Reduction of Knowledge).** *Consider field $\mathbb{F}$, length parameter $n$, and $\mathbb{F}$-modules $W$ and $V$. Construction 4 is a reduction of knowledge for*

$$(\mathcal{R}(W^n \to V), \mathcal{R}(W \to V)).$$

*Proof.* Consider instance $u = \sum_{i\in[n]} u_i \otimes \delta_i$ and $v$. We prove knowledge soundness via tree extraction (Lemma 6). That is, we construct extractor $\chi$ that outputs $w$ such that $u(w) \cong v$ given a tree of accepting transcripts and corresponding output prover witnesses.

Suppose the extractor $\chi$ is provided with $n^2$ accepting transcripts $\tau_{mk}$ with the same prover's first message

$$\{(v_{1,ij}, v_{2,ij}) | i \in [n], j \in [n]\}$$

and with randomness $(\alpha_m, \beta_{mk})$ for $m, k \in [n]$. Let $w'_{mk} \in W$ for $m, k \in [n]$ denote the corresponding satisfying witnesses. For $m \in [n]$, the extractor solves for $w_{mj} \in W$ for $j \in [n]$ such that

$$\sum_{j\in[n]} \beta_{mk}^j w_{mj} = w'_{mk} \tag{13}$$

for $k \in [n]$ using an inverse Vandermonde matrix (where invertibility is afforded by working over a field). Because $w'_{mk}$ is a satisfying witness, by construction of the tensor reduction, for all $m, k \in [n]$ we have that

$$\Big(\sum_i \alpha_m^i u_i\Big)\Big(w'_{mk}\Big) = \Big(\sum_{i,j} \alpha_m^i \beta_{mk}^j v_{1,ij}\Big).$$

Then, by Equation (13) we have for all $m, k \in [n]$

$$\Big(\sum_i \alpha_m^i u_i\Big)\Big(\sum_j \beta_{mk}^j w_{mj}\Big) = \Big(\sum_{i,j} \alpha_m^i \beta_{mk}^j v_{1,ij}\Big).$$

Rearranging terms, we have that

$$\sum_j \Big(\sum_i \alpha_m^i u_i(w_{mj})\Big) \cdot \beta_{mk}^j = \sum_j \Big(\sum_i \alpha_m^i v_{1,ij}\Big) \cdot \beta_{mk}^j. \tag{14}$$

Thus, for each $m \in [n]$, we can treat both sides of Equation 14 as polynomials evaluated over $\{\beta_{mk} | k \in [n]\}$. Because equality holds for $k \in [n]$ distinct evaluations, we have that for all $m, j \in [n]$

$$\Big(\sum_i \alpha_m^i u_i\Big)\Big(w_{mj}\Big) = \Big(\sum_i \alpha_m^i v_{1,ij}\Big). \tag{15}$$

To compute a satisfying witness, the extractor first computes $a_{mj}$ for all $m, j \in [n]$ such that for all $i \in [n]$

$$\sum_{m \in [n]} \alpha_m^i a_{mj} = v_{2,ij}. \tag{16}$$

Next, the extractor computes

$$w \leftarrow \sum_{l \in [n]} \sum_{m \in [n]} \sum_{j \in [n]} a_{mj} \cdot \alpha_m^l \cdot w_{mj} \otimes \delta_l. \tag{17}$$

We must now show that $w$ is a satisfying witness. Indeed, we have

$$\begin{aligned}
u(w) &= \Big(\sum_i u_i \otimes \delta_i\Big)\Big(\sum_{l,m,j} a_{mj} \cdot \alpha_m^l \cdot w_{mj} \otimes \delta_l\Big) && \text{By (17).}\\
&= \sum_{i,l,m,j} a_{mj} \cdot \alpha_m^l \cdot u_i(w_{mj}) \otimes \delta_i(\delta_l) \\
&= \sum_{i,m,j} a_{mj} \cdot \alpha_m^i \cdot u_i(w_{mj}) && \text{By } \delta_i(\delta_l) = 0 \text{ for } i \neq l.\\
&= \sum_{m,j} a_{mj} \cdot \sum_i \alpha_m^i \cdot u_i(w_{mj}) \\
&= \sum_{m,j} a_{mj} \cdot \sum_i \alpha_m^i \cdot v_{1,ij} && \text{By (15).}\\
&= \sum_{i,j} v_{1,ij} \cdot \sum_m \alpha_m^i a_{mj} \\
&= \sum_{i,j} v_{1,ij} \cdot v_{2,ij} && \text{By (16).}\\
&= v. && \text{By the verifier's check.}
\end{aligned}$$

□

## B.5 Proof of Lemma 8 (Bilinear Forms Reduction of Knowledge)

**Lemma.** *For $n, k \in \mathbb{N}$ such that $k$ divides $n$, Construction 7 is a reduction of knowledge form $\mathcal{R}_{\mathsf{Bil}(m,n)}$ to $\mathcal{R}_{\mathsf{Bil}(m,n/k)}$ with $O(n)$ prover and verifier time complexity and $O(k^2)$ communication complexity.*

*Proof.* Completeness, prover time complexity, verifier time complexity and communication complexity follow by the corresponding properties of the tensor reduction of knowledge.

As for knowledge soundness, consider statement $(G, H, M, \overline{A}, \overline{B}, \sigma)$. Suppose a malicious prover $\mathcal{P}^*$ succeeds in producing an accepting reduced witness with probability $\epsilon$. We must construct an extractor $\mathcal{E}$ that succeeds with probability $\epsilon - \mathsf{negl}(\lambda)$.

By translating statements between $\mathcal{R}_{\mathsf{Bil}}$ and $\mathcal{R}(\mathsf{BF})$ as described in Construction 7, Prover $\mathcal{P}^*$ implies a malicious prover $\mathcal{P}^{**}$ that succeeds with probability $\epsilon$ for the underlying tensor reduction of knowledge. Thus, $\mathcal{E}$ runs the corresponding extractor for the tensor reduction of knowledge which succeeds with probability $\epsilon - \mathsf{negl}(\lambda)$. By construction, the tensor reduction extractor is provided a tree of accepting transcripts. Any branch, after appropriate translation of statements, can be parsed to retrieve $A, B' \in \mathbb{F}^m, \mathbb{F}^{n/k}$ such that

$$((G, H, M', \overline{A}, \overline{B}, \sigma'), (A, B')) \in \mathcal{R}_{\mathsf{Bil}(m,n)}.$$

for some $M' \in \mathbb{F}^m \otimes \mathbb{F}^{n/2}$, $\overline{B}' \in \mathbb{H}$, and $\sigma' \in \mathbb{F}$. By the bilinear relation assumption over $(\mathbb{G}, \mathbb{F})$ the vector $A$ in any given branch is the same with probability $1 - \mathsf{negl}(\lambda)$. Let $A = (a_1, \ldots, a_n)$.

With probability $\epsilon - \mathsf{negl}(\lambda)$, the tensor reduction extractor succeeds in producing $\sum_i A'_i \otimes B'_i \in \mathbb{F}^n \otimes \mathbb{F}^m$ such that

$$\Big(G \otimes H \oplus \mathbf{M}\Big)\Big(\sum_i A'_i \otimes B'_i\Big) = \overline{A} \otimes \overline{B} \oplus \sigma$$

with probability $\epsilon - \mathsf{negl}(\lambda)$. We will show that due to the bilinear relation assumption over $(\mathbb{H}, \mathbb{F})$ and $(\mathbb{G}, \mathbb{H})$, the witness must be of the form $A \otimes B$ for some efficiently computable $B$. Indeed, rearranging we have that

$$\sum_i A'_i \otimes B'_i = \sum_i \delta_i \otimes B_i$$

for canonical basis $\{\delta_i\}$ for $\mathbb{F}^n$ and some $B_i \in \mathbb{F}^m$. Then, we have

$$\Big(G \otimes H\Big)\Big(\sum_i \delta_i \otimes B_i\Big) = \sum_i G_i \otimes \overline{B}_i$$

where $\overline{B}_i = H(B_i)$. Additionally, we have

$$\overline{A} \otimes \overline{B} = \Big(\sum_i a_i \cdot G_i\Big) \otimes \overline{B} = \sum_i G_i \otimes (a_i \cdot \overline{B}).$$

By the bilinear relation assumption over $(\mathbb{G}, \mathbb{H})$ we have $\overline{B}_i = a_i \cdot \overline{B}$ for all $i \in [n]$ with overwhelming probability. This in turn implies $H(a_i^{-1} \cdot B_i) = \overline{B}$ for all $i \in [n]$. Then, by the bilinear relation assumption over $(\mathbb{H}, \mathbb{F})$, we have that

$$a_1^{-1} \cdot B_1 = \ldots = a_n^{-1} \cdot B_n$$

with overwhelming probability. Let $B = a_i^{-1} \cdot B_i$ denote the above value. Then we have that $A \otimes B$ is a satisfying witness because

$$A \otimes B = \sum_i a_i \cdot \delta_i \otimes B = \sum_i \delta_i \otimes B_i = \sum_i A'_i \otimes B'_i.$$

□

## C Recovering the Sum-Check Protocol

In this section, we show how to express the sum-check protocol as a tensor reduction over (linearized) polynomials.

Bootle, Chiesa, and Sotiraki [BCS21] show that a large class of split-and-fold techniques can be viewed as a special case of sum-check protocols over commitments, which they call sum-check arguments. We loosely show the converse of this result: That is, we show that tensor reductions, which can be interpreted as an abstracted folding technique, generalize sum-check protocols. Bootle et al. further show that sum-check arguments can be instantiated with any commitment scheme which satisfies a certain structural decomposability property, and thus show that sum-check arguments generalize folding techniques over prime-order groups, bilinear groups, and unknown-order groups. The following generalization lemma formally interprets these results as tensor reductions.

Our high level approach is as follows: First, we recall a simplified definition of the sum-check protocol. Next, we define a linearized sum-check protocol which represents running the tensor reduction on linearized multivariate polynomials decomposed as univariate polynomials. This effectively fixes the modules and decomposition rules necessary to fully specify the tensor reduction. Finally, we prove that a single step of the sum-check protocol is structurally equivalent to a single step of the linearized sum-check protocol.

We begin by recalling the sum-check protocol generalized to modules [BCS21]. We make several simplifications for the sake of a more lucid presentation: First, we only define and consider a single recursive step of the sum-check and prove that it is structurally equivalent to a single recursive step of the tensor reduction instantiated over multivariate polynomials. Equivalence between the full sum-check protocol and the full recursive tensor reduction follows by induction. Second, we have the verifier immediately compute the statement polynomial in each recursive step, as opposed to deferring this computation until the end. The purpose of this modification is to avoid having to carry the randomness generated by both the tensor reduction and sum-check protocol throughout all the rounds in a global statement. Finally, we assume that both protocols use the standard monomial basis. Similar results hold for an arbitrary basis.

**Definition 24 (Sum-Check Relation).** *Consider ring* $\mathsf{R}$*,* $\mathsf{R}$*-module* $V$*, and subset* $H \subseteq \mathsf{R}$*. The sum-check relation* $\mathsf{R}_{\mathsf{SC}}$*, characterized by the number of variables* $n$*, is defined to be*

$$\mathcal{L}_{\mathsf{SC}}(n) = \left\{ ((P,\sigma),\bot) \;\middle|\; \begin{array}{l} P \in \mathsf{R}^n \to V, \sigma \in V, \\ \sum_{x_1,\dots,x_n \in H} P(x_1,\dots,x_n) = \sigma. \end{array} \right\}$$

*For notational simplicity, we omit* $\bot$*.*

**Construction 9 (Sum-check Protocol [LFKN92, BCS21]).** Consider ring $\mathsf{R}$, $\mathsf{R}$-module $V$, and subset $H = \{h_1,\dots,h_m\} \subseteq \mathsf{R}$. Suppose for some polynomial $P \in \mathsf{R}^n \to V$ with degree $K-1$ in each variable, and claimed sum $\sigma \in V$, the verifier would like to check

$$(P,\sigma) \in \mathcal{R}_{\mathsf{SC}}(n)$$

The prover sends to the verifier degree $K-1$ polynomial

$$p(X) = \sum_{x_1,\dots,x_{n-1}\in H} P(x_1,\dots,x_{n-1},X).$$

The verifier checks

$$\sum_{x_n \in H} p(x_n) = \sigma.$$

The verifier then samples and sends $\alpha$ from a sampling set $Q$ in $\mathsf{R}$. The prover and verifier then compute

$$\begin{aligned} \sigma' &\leftarrow p(\alpha) \\ P'(X_1, \dots, X_{n-1}) &\leftarrow P(X_1, \dots, X_{n-1}, \alpha), \end{aligned}$$

reducing the original task to the task of checking

$$(P', \sigma') \in \mathcal{R}_{\mathsf{SC}}(n-1).$$

**Lemma 13 (Sum-check Protocol [LFKN92, BCS21]).** *The sum-check protocol is a reduction from $\mathcal{R}_{\mathsf{SC}}(n)$ to $\mathcal{R}_{\mathsf{SC}}(n-1)$.*

Next we describe a linearized sum-check statement and a corresponding linearized sum-check protocol which leverages the tensor reduction.

Consider ring $\mathsf{R}$, $\mathsf{R}$-module $V$, and subset $H \subseteq \mathsf{R}$. Suppose for some polynomial map $P \in \mathsf{R}^n \to V$ with degree $K-1$ in each variable, and claimed sum $\boldsymbol{\sigma} \in V$, the verifier would like to check

$$\sum_{u_1, \dots, u_n \in H} P(u_1, \dots, u_n) = \boldsymbol{\sigma}. \tag{18}$$

By the universality of the tensor product (Definition 23), there exists tensor $\mathbf{P} \in V \otimes \bigotimes_{i \in [n]} \mathsf{R}^K$ such that $P = \mathbf{P} \circ \iota$ where $\iota$ is defined to be

$$\iota(u_1, \dots, u_n) = \bigotimes_{j \in [n]} \bigoplus_{k \in [K_j]} u_j^k.$$

Because $\mathbf{P}$ is linear in its inputs, by letting

$$\mathbf{U} = \sum_{u_1, \dots, u_n \in H} \iota(u_1, \dots, u_n),$$

the verifier can check equation (18) by checking

$$\mathbf{P}(\mathbf{U}) = \boldsymbol{\sigma}.$$

This motivates defining the corresponding *linearized* sum-check relation.

**Definition 25 (Linearized Sum-Check Relation).** *Consider ring $\mathsf{R}$, $\mathsf{R}$-module $V$, and subset $H \subseteq \mathsf{R}$. The linearlized sum-check relation $\mathcal{R}_{\mathsf{LSC}}$, characterized by the number of variables $n$, is defined to be*

$$\mathcal{R}_{\mathsf{LSC}}(n) = \left\{ ((\mathbf{P}, \boldsymbol{\sigma}), \perp) \;\middle|\; \begin{array}{l} \mathbf{P} \in V \otimes \bigotimes_{i \in [n]} \mathsf{R}^K, \boldsymbol{\sigma} \in V, \\ \mathbf{U} = \sum_{u_1, \dots, u_n \in H} \iota(u_1, \dots, u_n), \\ \mathbf{P}(\mathbf{U}) = \boldsymbol{\sigma} \end{array} \right\}$$

*For notational simplicity, we omit $\perp$.*

**Construction 10 (Linearized Sum-Check Protocol).** For arbitrary ring $\mathsf{R}$, $\mathsf{R}$-module $V$, subset $H \subseteq \mathsf{R}$, and degree bound $K$, we build a reduction for $(\mathcal{R}_{\mathsf{LSC}}(n), \mathcal{R}_{\mathsf{LSC}}(n-1))$. Let $\{\delta_1, \ldots, \delta_K\}$ represent a canonical basis for $\mathsf{R}^K$ and let $H = \{h_1, \ldots, h_m\}$. For $(\mathbf{P}, \boldsymbol{\sigma}) \in \mathsf{R}_{\mathsf{LSC}}(n)$, and $\mathbf{U} = \sum_{u_1 \ldots, u_n \in H} \iota(u_1, \ldots, u_n)$ we have that

$$\mathbf{P} = \sum_{i \in [K]} \mathbf{P}_i \otimes \delta_i$$

for some $(n-1)$-dimensional tensors $\mathbf{P}_i$, and

$$\mathbf{U} = \sum_{j \in [m]} \mathbf{U}' \otimes \mathbf{h}_j$$

where $\mathbf{U}' = \sum_{u_1, \ldots, u_{n-1} \in H} \iota(u_1, \ldots, u_{n-1})$, and $\mathbf{h}_j = (h_j^0, \ldots, h_j^{K-1})$. Applying the tensor reduction with respect to this decomposition reduces the verifier's task of checking the original check to the task of checking

$$\Big(\sum_i \alpha^i \delta_i\Big)\Big(\sum_j \beta^j \mathbf{h}_j\Big) = x$$

which the verifier checks immediately and

$$\Big(\sum_i \alpha^i \mathbf{P}_i\Big)\Big((\sum_j \beta^j)\mathbf{U}'\Big) = y$$

for $\alpha, \beta, x, y \in \mathsf{R}$ generated during the reduction. Thus, the verifier computes $\mathbf{P}' = \sum_i \alpha^i \mathbf{P}_i$ and $\boldsymbol{\sigma}' = y / (\sum_j \beta^j)$ and reduces the original check to the task of checking $(\mathbf{P}', \boldsymbol{\sigma}') \in \mathcal{R}_{\mathsf{LSC}}(n-1)$.

**Lemma 14 (Linearized Sum-Check Protocol).** *The linearized sum-check protocol is a reduction from $\mathcal{R}_{\mathsf{LSC}}(n)$, to $\mathcal{R}_{\mathsf{LSC}}(n-1)$.*

*Proof.* Completeness and soundness follow from Theorem 7. $\square$

Given constructions for both the sum-check protocol and the linearized sum-check protocol, we can now prove that the two are structurally equivalent. To do so we will show that a single iteration of the sum-check protocol is equivalent to first linearizing the statement polynomials, running the linearized sum-check protocol and mapping the resulting statement back into the original space, and additionally show that the generated transcript from the linearized sum-check protocol can be used to recover the transcript produced by the standard sum-check protocol. It is important to note that we *cannot* show that the linearized sum-check protocol transcript is equivalent to the sum-check protocol transcript. This is because the tensor reduction transcript inherently contains more structural information, which must be thrown out to recover the sum-check protocol transcript.

**Lemma 15 (Structural Correspondence).** *Let $\Pi_{\mathsf{LSC}}$ represent the linearized sum-check protocol and let $\Pi_{\mathsf{SC}}$ represent the sum-check protocol. Define the bijection $\Phi$ from a statement in $\mathcal{R}_{\mathsf{LSC}}$ to a statement in $\mathcal{R}_{\mathsf{SC}}$ as follows*

$$\Phi((\mathbf{P}, \sigma)) = (P, \sigma)$$

*where, given that $\mathbf{P} \in (\mathbb{F}^d)^n$ is an $n$-dimensional tensor, $P$ is a polynomial that encodes the value at index $(j_1, \ldots, j_n)$ as the coefficient of term $x_1^{j_1} x_2^{j_2} \ldots x_n^{j_n}$. Then, given that $\Pi_{\mathsf{LSC}}$ and $\Pi_{\mathsf{SC}}$ are instantiated on the same verifier randomness, then the following diagram commutes*

$$\begin{tikzcd}
\mathcal{R}_{\mathsf{LSC}}(n) \arrow[r, "\Pi_{\mathsf{LSC}}"] \arrow[d, "\Phi"] & \mathcal{R}_{\mathsf{LSC}}(n-1) \arrow[d, "\Phi"] \\
\mathcal{R}_{\mathsf{SC}}(n) \arrow[r, "\Pi_{\mathsf{SC}}"] & \mathcal{R}_{\mathsf{SC}}(n-1)
\end{tikzcd}$$

*and there exists PPT simulator $\mathcal{S}$ that can simulate the interaction of $\Pi_{\mathsf{SC}}$ given oracle access to the interaction of $\Pi_{\mathsf{LSC}}$.*

*Proof.* Given a transcript of $\Pi_{\mathsf{SC}}$, let the simulator produce a transcript of $\Pi_{\mathsf{LSC}}$ as follows

$$\mathcal{S}\Big(\Big\{r_{ij}, s_{ij} \big| i \in [K], j \in [m]\Big\}, \alpha, \beta\Big) \mapsto \{r_{i1} \mid i \in [K]\}, \alpha$$

Consider arbitrary $(\mathbf{P}, \sigma) \in \mathcal{R}_{\mathsf{LSC}}(n)$. Let

$$\mathbf{P} = \mathbf{P}_1 \otimes \delta_1 + \mathbf{P}_2 \otimes \delta_2 + \ldots + \mathbf{P}_K \otimes \delta_K$$

Then, by linearity of $\Phi$, we have

$$P(X_1, \ldots, X_n) = P_1(X_1, \ldots, X_{n-1}) \cdot X_n^0 + \ldots + P_K(X_1, \ldots, X_{n-1}) \cdot X_n^{K-1}$$

where $\Phi(\mathbf{P}) = P$ and $\Phi(\mathbf{P}_i) = P_i$. Moreover, for $\mathbf{h}_j = (h_j^0, \ldots, h_j^K)$, the $\Pi_{\mathsf{LSC}}$ prover sends as its first message

$$\{\mathbf{P}_i(\mathbf{U}'), \delta_i(\mathbf{h}_j) | i \in [K], j \in [m]\}$$

This means

$$\mathcal{S}(\{\mathbf{P}_i(\mathbf{U}'), \delta_i(\mathbf{h}_j) | i \in [K], j \in [m]\}) = \{\mathbf{P}_i(\mathbf{U}') | i \in [K]\}$$

which, under the monomial basis, is precisely equal to the coefficients of $p(X)$ sent by the $\Pi_{\mathsf{SC}}$ prover. Additionally, by assumption both $\Pi_{\mathsf{LSC}}$, and $\Pi_{\mathsf{SC}}$ are initialized with the same verifier randomness. This means that the challenge $\alpha$ is identical in both transcripts. Therefore, we have that the simulator produces a transcript identical to the transcript produced by $\Pi_{\mathsf{SC}}$.

Next, we observe that

$$\Phi(P') = \Phi\Big(\sum_i P_i \cdot \alpha^i\Big) = \sum_i \Phi(P_i) \cdot \alpha^i = \sum_i \mathbf{P}_i \cdot \alpha^i = \mathbf{P}'$$

Additionally, we observe that

$$\sigma' = \sum_{x_1, \ldots, x_n} P(x_1, \ldots, x_{n-1}, \alpha) = \sum_i \alpha^i \mathbf{P}_i(\mathbf{U}') = \sum_{i,j} \alpha^i \beta^j \mathbf{P}_i(\mathbf{U}') / \Big(\sum_j \beta^j\Big) = \boldsymbol{\sigma}'$$

Therefore, we have also have that $\Phi(P', \sigma') = (\mathbf{P}', \boldsymbol{\sigma}')$. □