Lattice-based Σ-Protocols for Polynomial Relations with Standard Soundness

Lizhen ZHANG, Shang GAO, and Bin XIAO

The Hong Kong Polytechnic University, Hong Kong, China
`lizh6.zhang@connect.polyu.hk`
`shanggao@polyu.edu.hk`
`b.xiao@polyu.edu.hk`

Abstract. We propose new techniques for enhancing the efficiency of Σ-protocols in lattice settings. One major challenge in lattice-based Σ-protocols is restricting the norm of the extracted witness in soundness proofs. Most of existing solutions either repeat the protocol several times or opt for a relaxation version of the original relation. Recently, Boneh and Chen have propose an innovative solution called LatticeFold [BC24], which utilizes a sum-check protocol to enforce the norm bound on the witness. In this paper, we elevate this idea to efficiently proving multiple polynomial relations without relaxation. Simply incorporating the techniques from LatticeFold into Σ-protocols leads to inefficient results; therefore, we introduce several new techniques to ensure efficiency. First, to enable the amortization in [AC20] for multiple polynomial relations, we propose a general linearization technique to reduce polynomial relations to homomorphic ones. Furthermore, we generalize the folding protocol in LatticeFold, enabling us to efficiently perform folding and other complex operations multiple times without the need to repeatedly execute sum-checks. Moreover, we achieve zero-knowledge by designing hiding claims and elevating the zero-knowledge sum-check protocol [XZZ$^+$19] on rings. Our protocol achieves standard soundness, thereby enabling the efficient integration of the compressed Σ-protocol theory [AC20, ACF21] in lattice settings.

Keywords: Lattice-based cryptography, Σ-protocol, sum-check protocol

1 Introduction

Σ-protocol is a fundamental cryptographic primitive for constructing zero knowledge proofs. In discrete logarithm setting, we benefit from nice tools such as Bulletproofs [BBB$^+$18] and amortization techniques [AC20, ACF21] that enhance the efficiency of protocols. Unfortunately, the security of these implementations largely hinges on the discrete logarithm problem, rendering them vulnerable to potential threats posed by quantum computers.

This deficiency has impelled the development of "post-quantum" cryptographic solutions that can withstand the advent of quantum computing. Lattice-based cryptography stands out as a leading contender among post-quantum alternatives, with its security relying on the hardness of computational problems in lattices.

However, lattice-based approaches typically face extra challenges when contrasted with techniques in discrete logarithm environments. This is primarily attributed to the norm bound constraints in lattice commitments. More precisely, consider a lattice commitment scheme Com, the commitment relation for a public statement F and a witness $\boldsymbol{f}$ with small norm can be expressed as

$$\mathcal{R}_{\mathsf{com}}^{\mathcal{B}} := \{(F; \boldsymbol{f}) : \mathsf{Com}(\boldsymbol{f}) = F, \|\boldsymbol{f}\|_\infty \leq \mathcal{B}\}\,.$$

The norm claim is crucial since it forms the foundation upon which the lattice hardness problem is constructed. Applying this commitment scheme into a Σ-protocol presents two challenges: (1) folding two vectors (the witness $\boldsymbol{f}$ and the masking vector $\boldsymbol{t}$) into a single vector $\boldsymbol{g}$ by means of random linear combination would increase the norm of the folded result, potentially surpassing the norm constraint $\mathcal{B}$, at which point the commitment scheme is no longer binding. (2) In the soundness proof of Σ-protocol, the extractor can only extract $\boldsymbol{f'}$such that $\|\boldsymbol{f'}\|_\infty \leq \alpha\mathcal{B}$, where α is known as the *soundness slack*. To achieve standard soundness, one can use a small challenge space to restrict the value of α. However this approach requires multiple iterations of the protocol to ensure a negligible soundness error. Another option is to relax the commitment relation by $\|\boldsymbol{f}\|_\infty \leq \alpha\mathcal{B}$. Nonetheless, in multi-round protocols the soundness slack accumulates, necessitating a more rigorous analysis, particularly when incorporating Bulletproofs [BBB$^+$18] to further optimize the computation process.

In a recent development, Boneh and Chen [BC24] proposed a solution to the aforementioned problems using the "split-and-fold" technique in their framework, $\mathsf{LatticeFold}$. Their approach maintains the witness norm within bounds during folding process by first decomposing multiple witness vectors, each with a norm bound of $\mathcal{B}$, into k sub-vectors with a smaller norm b, where $b = \lceil \mathcal{B}^{1/k} \rceil$, and then folding these sub-vectors back into a single vector. It is worth mentioning that the "split-and-fold" technique is not a novel concept, rather, it has been widely used in various foundational building blocks and protocols, such as the sum-check protocol, the Fast Reed-Solomon interactive oracle proofs of proximity protocol (FRI), and others. Nonetheless, one major significance of $\mathsf{LatticeFold}$ is to achieve *standard soundness* for "split-and-fold" without the need for multiple repetitions. They incorporate a norm check on input vectors in the folding protocol to ensure that each extracted witness has a small norm. Notably, they employ a sum-check protocol to reduce the norm check into a multilinear polynomial evaluation check. These advancements would enable efficiently composing multiple protocols based on [KP23].

We consider their results to be both impressive and efficient, offering a solid foundation for constructing multi-round Σ-protocols in lattice settings. But directly applying their technique to construct lattice-based Σ protocols would lead to several issues (as detailed below), potentially reducing both efficiency and effectiveness.

1.1 Our Contributions

Inspired by LatticeFold [BC24], we introduce an efficient lattice-based Σ-protocol for multiple polynomial relations. Drawing from the advantages of LatticeFold, our protocol eliminates the need for repetition and avoids the issue of soundness slack. This enables further application of the compressed Σ-protocol theory in [AC20, ACF21] efficiently. Nonetheless, to effectively integrate and leverage the techniques from LatticeFold in a Σ-protocol for arbitrary polynomial relations, we must overcome additional challenges.

Not Applicable to Non-homomorphic Relations. The amortization technique in [AC20, ACF21] serves as a preprocessing step that folds multiple homomorphic relations into one. However, this method is not directly applicable to polynomial relations due to the non-homomorphic properties of high-degree polynomials. Even though [AC20] demonstrates that a non-linear relation can be reduced to a linear one by representing it with arithmetic circuits, the transformation process incurs a non-negligible cost due to Fast Fourier Transforms (FFT) [BCS21] and non-native operations [ZCYW23]. To address this problem, we introduce a new linearization technique that provides a *direct* solution without the circuit transformation. This method can be regarded as a generalized linearization in [BC24] for any polynomial relations. By leveraging a sum-check protocol, our approach ensures efficiency with only logarithmic overhead.

Multiple Sum-Checks. The amortization technique can be viewed as a process of folding multiple relations, which can be achieved by executing the folding protocol proposed in [BC24]. However, since the folding protocol in [BC24] applies norm enforcement on the input to ensure standard soundness, each invocation of the folding protocol would introduce an additional norm check, specifically a sum-check process. When a protocol requires multiple folds—like in Σ-protocols where the use of a masking vector to hide the amortized witness can be considered as an extra folding process—repeating the sum-check each time introduces additional costs. To address this issue, we are inspired by the folding protocol and generalize it into a more "sophisticated" protocol. This enhanced protocol enables a *single* norm check (i.e., sum-check) while accommodating multiple folds and other operations, all while ensuing the extracted witnesses to be of a small norm.

Not Zero Knowledge. The approach described in [BC24] does not provide zero-knowledge property, but this is actually crucial as the interactions and output claims of the sum-check protocol may leak witness information. To achieve zero-knowledge, we apply the zero-knowledge sum-check in [XZZ$^+$19] to rings and employ a hiding technique that pads the witness with a short random vector. These strategies allow us to preserve zero-knowledge while incurring only a logarithmic cost.

A comparison between our protocol's properties and other compressed Σ-protocols is shown in Table 1. As the soundness slack in our approach remains constant, we are able to use a much smaller parameter set compared to other methods like [BLNS20, ACK21]. We also provide an informal summary of our efficiency in Table 2 (the detailed one is given in Table 4).

Table 1: Comparison of our approach with other compressed Σ-protocols. We do not consider indirect approaches to address non-homomorphic relations in [AC20, ACF21]. SIS stands for short integer solution, and SIS relation is the knowledge of a SIS preimage.

	Relation	**Post-Quantum**	**Soundness**	**Slack**
[AC20]	linear	no	standard	-
[ACF21]	homomorphic	no	standard	-
[BLNS20]	SIS	yes	relaxed	increase
[ACK21]	homomorphic	yes	relaxed	increase
[AL21]	SIS	yes	relaxed	increase
This work	polynomial	yes	standard	fix

Table 2: Efficiency comparison (informal) between the direct approach and our protocol. The direct approach means sending all witnesses directly, which is the approach described in 2.4 without arithmetation and compression. k represents the number of instances, m represents the witness size for each instance, and D represents degree of the polynomial relation. Compression refers to further adopting the Bulletproofs compression in compressed Σ-protocol theory.

	Size	**Prover**	**Verifier**
Direct approach	$O(kmD)$	$O(kmD)$	$O(kmD)$
Ours (w/o compression)	$O(k+D\log m+m)$	$O(kmD\log^2 D)$	$O(D\log m+kD+m)$
Ours (w/ compression)	$O(k+D\log m)$	$O(kmD\log^2 D)$	$O(D\log m+kD+m)$

1.2 Related Work

Compressed Σ-protocol theory. Compressed Σ-protocols [AC20, ACF21] optimize the proof size of Σ-protocols to a logarithmic scale by utilizing the Bulletproofs compression [BBB$^+$18]. Through the use of a preprocessed amortization [AC20, ACF21], a prover can prove multiple homomorphic relations at the cost of one. However, since Bulletproofs is a multi-round protocol, integrating compressed Σ-protocol theory into Fiat-Shamir with abort protocols presents significant challenges. Bootle et al. [BLNS20] introduce new solutions for Bulletproofs compression in lattice settings, and Attema et al. [ACK21] further provide a tight proof for the knowledge soundness. Albrecht and Lai [AL21] study the tradeoffs between the soundness error and slack. By carefully selecting an appropriate ring, they can achieve better result than [ACK21]. Nonetheless, since these approaches are designed upon relaxed relations,[1] the soundness slack increases with the number of rounds in the protocol. Consequently, they require a larger parameter set to ensure the hardness of the underlying lattice problems.

Lattice-based proof systems. The fundamental hardness assumption upon which lattice-based cryptography rests is that it is computationally difficult to

[1] [AL21] can achieve a protocol with standard soundness, but incurs a large soundness error.

find a low-norm vector $\vec{s}$ satisfying $\boldsymbol{A}\vec{s} = \vec{t} \mod q$. Early methods for proving norm bounds can be categorized into two main types: those that proving ℓ_∞-norm of $\vec{s}$ and those that proving ℓ_2-norm of $\vec{s}$. There are two major approaches for proving ℓ_∞-norm: *Stern-type protocols* [KTX08,LNSW13,YAZ+19] and *Fiat-Shamir with abort protocols* [Lyu09,Lyu12,ESLL19,ESZ22]. Stern-type protocols are capable of achieving standard soundness but require a small challenge space. Consequently, their soundness error can be large, requiring the prover to repeat the protocol multiple times to achieve a negligible soundness error. Fiat-Shamir with abort protocols are similar to Σ-protocols in discrete logarithm settings, and can utilize a large challenge space, allowing for a negligible soundness error with a few (or even a single) rounds. But they do not have standard soundness since the extracted witness does not necessarily satisfy the original relation (i.e., the relation is relaxed for the extracted witness). Thus, Fiat-Shamir with abort protocols face efficiency challenges when composing multi-round protocols with techniques in [KP23] due to the accumulated soundness slack.

[LNP22] proposed a direct method for proving that $\vec{s}$ has a small ℓ_2-norm by leveraging the inner product of two related vectors. This method has the advantage of not requiring additional Number Theoretic Transform (NTT) or Chinese Remainder Theorem (CRT) transformations. However, it is challenging to apply this method to multiple-witness folding, and even more so to multi-round protocols as discussed in [KP23] as it introduces a variety of non-linear operations. Recently, LOVA [FKNP24] introduced a novel approach for proving the ℓ_2-norm by flexibly leveraging the Demillo-Lipton-Schwartz-Zippel lemma in their proposed lattice-based folding scheme. Unfortunately, this work has not yet been shown to support R1CS relations.

As quantum technology advances, lattice-based proof systems are increasingly being proposed and applied in modern cryptography protocols. LaBRADOR [BS22]has presented a succinct recursive lattice-based proof system with linear time verifier. It uses *random project* to prove norm bound, which is an appealing technique but has been shown to be inefficient than [BC24].

LatticeFold [BC24] is an innovative folding scheme for lattice commitments. It provides new insights to address the norm bound challenge by employing a sum-check protocol. To further mitigate the norm increment after folding multiple instances, LatticeFold adopts a split-and-fold technique to reduce the norm bound of an output relation. We are mostly motivated by the ideas from [BC24] in designing an efficient Σ-protocol that maintains standard soundness. However, we observe that straightforwardly adopting LatticeFold's techniques leads to inefficient designs, as we elaborate in Section 1.1. This requires novel solutions to enhance efficiency within our protocol.

2 Preliminaries

2.1 Notations

Let q be a modulus, $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ be the ring of integers modulo q, i.e., $\left\{-\frac{q-1}{2}, \cdots, \frac{q-1}{2}\right\}$. We use $\mathbb{Z}_q[X]$ to denote the set of polynomials over $\mathbb{Z}_q$, i.e., whose coefficients are in $\mathbb{Z}_q$. For $n \in \mathbb{N}^+$, we denote $[n]$ as the set $\{0, 1, \cdots, n-1\}$. Vectors are denoted as $\vec{f} := (f_0, \cdots, f_{n-1}) \in \mathbb{Z}_p^n$. We use $(\vec{f}, \vec{g})$ to denote appending vector $\vec{f}$ to $\vec{g}$. Given a constant $a \in \mathbb{Z}_q$, define $\vec{a}^k := (1, a, a^2, \cdots, a^{k-1})$. Furthermore, the Hadamard product is denoted as $\vec{f} \circ \vec{g} := (f_0 \cdot g_0, \cdots, f_{k-1} \cdot g_{k-1})$ and $\bigcirc_{i=0}^{k-1} \vec{f}_i := \vec{f}_0 \circ \cdots \circ \vec{f}_{k-1}$. Given a distribution $\mathbb{S}$, $a \xleftarrow{\$} \mathbb{S}$ denotes sampling a from $\mathbb{S}$, or uniformly sampling from a set $\mathbb{S}$.

Cyclotomic rings. Let $\mathcal{R} := \mathbb{Z}[X]/(X^d + 1)$ denotes the polynomial ring and $\mathcal{R}_q := \mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[X]/(X^d + 1)$, where $d > 1$ is a power of 2. We use bold-face letters to denote $\mathcal{R}_q$ elements except commitments and challenges, i.e., polynomials such as $\boldsymbol{f} := \sum_{i=0}^{d-1} f_i X^i \in \mathcal{R}_q$. Here we slightly abuse the notion of $\boldsymbol{f}$ to denote the *coefficients* of the polynomial, i.e., $\boldsymbol{f} = (f_0, \cdots, f_{d-1})$. Note that though $\mathbb{Z}_q$ elements can also be regarded as a constant polynomial, we do not use bold-face letters for them. Define $\mathbb{S}_{\mathcal{B}}$ as the subset of polynomials in $\mathcal{R}_q$ with ℓ_∞-norm at most $\mathcal{B}$ (ℓ_∞-norm is defined below).

We choose a prime q such that $\mathbb{Z}_q$ contains a primitive $2t$-th root of unity $\zeta \in \mathbb{Z}_q$ but no elements whose order is a higher power of two, i.e. $q - 1 \equiv 2t \mod 4t$, where $t \in \mathbb{N}^+$ be a divisor of d. Therefore, we have:

$$X^d + 1 = \prod_{j=0}^{t-1} (X^{d/t} - \zeta^{2j+1}) \mod q$$

where each factor $(X^{d/t} - \zeta^{2j+1})$ is *irreducible* and ζ^{2j+1} represents all the t primitive $2t$-th roots of unity. By applying the *Chinese Remainder Theorem* (CRT), $\mathcal{R}_q$ can be split into the product of t quotient rings:

$$\mathcal{R}_q \cong \prod_{j=0}^{t-1} \mathbb{Z}_q[X]/(X^{d/t} - \zeta^{2j+1}) \cong \mathbb{Z}_{q^{d/t}}^t.$$

Definition 1. ***Number Theoretic Transform.*** *For a polynomial $\boldsymbol{f} \in \mathcal{R}_q$, its Number Theoretic Transform (NTT) is defined as*

$$\mathsf{NTT}(\boldsymbol{f}) := (\hat{\boldsymbol{f}}_0, \cdots, \hat{\boldsymbol{f}}_{t-1}) \in \mathbb{Z}_{q^{d/t}}^t,$$

where $\hat{\boldsymbol{f}}_j := \boldsymbol{f} \mod (X^{d/t} - \zeta^{2j+1})$ (which is isomorphic to $\mathbb{Z}_{q^{d/t}}$).

In a special case where $t = d$, we have $\mathcal{R}_q \cong \prod_{j=0}^{d-1} \mathbb{Z}_q[X]/(X - \zeta^{2j+1}) \cong \mathbb{Z}_q^d$ and $\mathsf{NTT}(\boldsymbol{f}) = (\hat{\boldsymbol{f}}_0, \cdots, \hat{\boldsymbol{f}}_{d-1}) \in \mathbb{Z}_q^d$.

NTT has the following property: Since NTT is a variant of Discrete Fourier Transform (DFT) over the polynomial ring, we can apply DFT's convolution theorem to compute polynomial multiplications: given $\boldsymbol{f}, \boldsymbol{g} \in \mathcal{R}_q$, we have the following equation:

$$\mathsf{NTT}(\boldsymbol{f} \cdot \boldsymbol{g}) = \mathsf{NTT}(\boldsymbol{f}) \circ \mathsf{NTT}(\boldsymbol{g}).$$

We also define the inverse NTT operation. For a vector $\vec{\boldsymbol{u}} \in \mathbb{Z}^t_{q^{d/t}}$, $\mathsf{NTT}^{-1}(\vec{\boldsymbol{u}})$ returns a polynomial $\boldsymbol{p} \in \mathcal{R}_q$ such that $\mathsf{NTT}(\boldsymbol{p}) = \vec{\boldsymbol{u}}$. Given a vector of polynomial $\vec{\boldsymbol{f}} \in \mathcal{R}_q^m$, denote $\vec{\boldsymbol{f}'} := (\vec{\boldsymbol{f}}'_0, \cdots, \vec{\boldsymbol{f}}'_{m-1}) \in \mathcal{R}_q^m$, such that $\mathsf{NTT}(\vec{\boldsymbol{f}'}) := \Big(\mathsf{NTT}(\vec{\boldsymbol{f}}'_0), \cdots, \mathsf{NTT}(\vec{\boldsymbol{f}}'_{m-1})\Big) = (\boldsymbol{f}_0, \cdots, \boldsymbol{f}_{m-1})$ (i.e., $\vec{\boldsymbol{f}'} = \mathsf{NTT}^{-1}(\vec{\boldsymbol{f}})$).

Norms. Let $\mathcal{R}_q := \mathcal{R}/\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$. For a polynomial $\boldsymbol{f} := \sum_{i=0}^{d-1} f_i X^i \in \mathcal{R}_q$, the ℓ_1-norm, ℓ_2-norm and ℓ_∞-norm are defined as:

$$\|\boldsymbol{f}\|_1 := \sum_{i=0}^{d-1} |f_i|, \quad \|\boldsymbol{f}\|_2 := \sqrt{\sum_{i=0}^{d-1} f_i^2}, \quad \|\boldsymbol{f}\|_\infty := \max_{i\in[d]} |f_i|$$

For a vector of polynomial $\vec{\boldsymbol{f}} := (\boldsymbol{f}_0, \cdots, \boldsymbol{f}_{m-1}) \in \mathcal{R}_q^m$, its ℓ_1-norm, ℓ_2-norm and ℓ_∞-norm are:

$$\|\vec{\boldsymbol{f}}\|_1 := \sum_{i=0}^{m-1} \|\boldsymbol{f}_i\|_1, \quad \|\vec{\boldsymbol{f}}\|_2 := \sqrt{\sum_{i=0}^{m-1} \|\boldsymbol{f}_i\|_2^2}, \quad \|\vec{\boldsymbol{f}}\|_\infty := \max_{i\in[m]}(\|\boldsymbol{f}_i\|_\infty)$$

2.2 Sum-Checks and Multilinear Extensions over Rings

Challenge space. The definition of challenge space (sampling set) [CCKP19] is defined as follows.

Definition 2. *Define a subset $\mathcal{C}$ of $\mathcal{R}_q$ as a challenge space if the difference of any two distinct elements in $\mathcal{C}$ is not a zero divisor. $\mathcal{C}$ is further a strong challenge space if the difference of any two distinct elements in it is invertible in $\mathcal{R}_q$.*

A typical example of a strong challenge space is $\mathbb{Z}_q \subset \mathcal{R}_q$. Sometimes we need a strong challenge space $\mathcal{C}_{\mathsf{small}} \subset \mathcal{R}_q$ whose elements have small norms. The *expansion factor* of $\mathcal{C}_{\mathsf{small}}$ is

$$\|\mathcal{C}_{\mathsf{small}}\|_{\mathsf{op}} := \sup_{\rho\in\mathcal{C}_{\mathsf{small}}, \boldsymbol{v}\in\mathcal{R}} \frac{\|\rho\boldsymbol{v}\|_\infty}{\|\boldsymbol{v}\|_\infty}. \tag{1}$$

$\rho \cdot \boldsymbol{v}$ here is performed in $\mathcal{R}$. Attema et al. prove that $\mathcal{C}$ has a small expansion factor and we can find large strong challenge spaces in $\mathcal{R}_q$ [ACK21].

Sum-check over rings. We give a generalized sum-check protocol operating on an arbitrary ring $\bar{\mathcal{R}}$, utilizing challenges sampled from a strong sampling set.

Lemma 1. ***(Generalized sum-check [CCKP19])*** *For an arbitrary ring $\bar{\mathcal{R}}$, let $f \in \bar{\mathcal{R}}^{\leq d}[\boldsymbol{X}_0, \cdots, \boldsymbol{X}_{\mu-1}]$ be a μ-variate nonzero polynomial with per-variable degree at most d. Let $\mathcal{C} \subset \bar{\mathcal{R}}$ be a strong sampling set, the following protocol for proving $\boldsymbol{s} = \sum_{\vec{b}\in\{0,1\}^\mu} f(\vec{b})$ has soundness error $\frac{\mu d}{|\mathcal{C}|}$.*

1. *In the i-th round ($i \in [\mu]$):*

- *Upon receiving the challenges $\boldsymbol{r}_0, \cdots, \boldsymbol{r}_{i-1}$ from the previous rounds, $\mathcal{P}$ sends the univariate polynomial*

$$f_i(\boldsymbol{X}) := \sum_{\vec{b} \in \{0,1\}^{\mu-i}} f(\boldsymbol{r}_0, \cdots, \boldsymbol{r}_{i-1}, \boldsymbol{X}, \vec{b}) \quad \in \bar{\mathcal{R}}[\boldsymbol{X}].$$

More precisely, $\mathcal{P}$ sends $d+1$ evaluations of f_i at $d+1$ points in $\mathcal{C}$.

- *$\mathcal{V}$ checks $f_i(0)+f_i(1) = f_{i-1}(\boldsymbol{r}_{i-1})$ and sends a random challenge $\boldsymbol{r}_i \overset{\$}{\leftarrow} \mathcal{C}$.*

2. $\mathcal{V}$ checks $f_{\mu-1}(\boldsymbol{r}_{\mu-1}) = f(\boldsymbol{r}_0, \cdots, \boldsymbol{r}_{\mu-1})$.

Sometimes the last check is deferred. Accordingly, the protocol outputs a claim of $f_{\mu-1}(\boldsymbol{r}_{\mu-1}) = f(\boldsymbol{r}_0, \cdots, \boldsymbol{r}_{\mu-1})$.

Multilinear extensions over rings The definition of multilinear extensions over rings is defined below.

Definition 3. ***(Multilinear extensions over rings [CCKP19])****. For an arbitrary ring $\bar{\mathcal{R}}$, given a function $f : \{0,1\}^\mu \to \bar{\mathcal{R}}$, the multilinear extension (MLE) of f is defined as*

$$\mathsf{mle}[f](\vec{\boldsymbol{X}}) := \sum_{\vec{b} \in \{0,1\}^\mu} f(\vec{b}) \cdot \mathsf{eq}(\vec{b}, \vec{\boldsymbol{X}}) \quad \in \bar{\mathcal{R}}^{\le 1}[\boldsymbol{X}_0, \cdots, \boldsymbol{X}_{\mu-1}],$$

where $\mathsf{eq}(\vec{b}, \vec{\boldsymbol{X}}) := \prod_{i=0}^{\mu-1} \big((1-b_i)(1-\boldsymbol{X}_i) + b_i \boldsymbol{X}_i\big)$.

When given an m-size vector $\vec{\boldsymbol{f}} \in \bar{\mathcal{R}}^m$, denote the MLE of $\vec{\boldsymbol{f}}$ as $\mathsf{mle}[\vec{\boldsymbol{f}}](\vec{\boldsymbol{X}}) := \sum_{\vec{b} \in \{0,1\}^{\log m}} \boldsymbol{f}_i \cdot \mathsf{eq}(\vec{b}, \vec{\boldsymbol{X}})$.

Similar to the field setting, the multilinear extension over rings of a Boolean function is unique [CCKP19] and has the following properties.

Corollary 1. ***(Corollary 2.1 in [BC24])****. For a multilinear polynomial $\boldsymbol{f} \in \bar{\mathcal{R}}^{\le 1}[\boldsymbol{X}_0, \cdots, \boldsymbol{X}_{\mu-1}]$ over a ring $\bar{\mathcal{R}}$, we have $\boldsymbol{f}(\vec{\boldsymbol{X}}) = \sum_{\vec{b} \in \{0,1\}^\mu} \boldsymbol{f}(\vec{b}) \cdot \mathsf{eq}(\vec{b}, \vec{\boldsymbol{X}})$.*

Lemma 2. ***(Lemma 3.3 in [BC24])****. Let $\vec{r} \in \mathcal{C}^{\log m}$. Given k instances $(\vec{\boldsymbol{f}}_i, \boldsymbol{s}_i)_{i=0}^{k-1}$ such that $\mathsf{mle}[\vec{\boldsymbol{f}}_i'](\vec{r}) = \boldsymbol{s}_i$, where $\mathsf{NTT}(\vec{\boldsymbol{f}}_i') = \vec{\boldsymbol{f}}_i$, set $(\vec{\boldsymbol{f}}, \boldsymbol{s})$ as follows for any $\vec{\rho} \in \mathcal{C}^k$*

$$\vec{\boldsymbol{f}} := \sum_{i=0}^{k-1} \rho_i \cdot \vec{\boldsymbol{f}}_i, \qquad \mathsf{NTT}(\boldsymbol{s}) = \sum_{i=0}^{k-1} \rho_i \cdot \mathsf{NTT}(\boldsymbol{s}_i).$$

We have $\mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{r}) = \boldsymbol{s}$, where $\mathsf{NTT}(\vec{\boldsymbol{f}}') = \vec{\boldsymbol{f}}$.

2.3 Lattice Problems and Ajtai Commitment

MSIS. We define the lattice problem known as the *module short integer solution* (MSIS) [LS15], upon which the security of our schemes relies. Specifically, this definition is a variant that utilizes the ℓ_∞-norm [ACK21, BC24].

Definition 4. $\mathsf{MSIS}(\kappa, m, q, \gamma)$. *Given a random matrix* $A \xleftarrow{\$} \mathcal{R}_q^{\kappa \times m}$*, the goal of the problem is to find non-zero* $\vec{\boldsymbol{z}} \in \mathcal{R}_q^m$ *such that* $A\vec{\boldsymbol{z}} = \vec{0}$ *over* $\mathcal{R}_q$ *and* $\|\vec{\boldsymbol{z}}\|_\infty \leqslant \gamma_{\mathsf{MSIS}}$.

In this paper, we regard γ_{MSIS} as γ. Based on the result in [MR09, BC24], MSIS is expected to have 128-bit security when

$$\min\{q, 2^{2\sqrt{d\kappa \log q \log(1.0045)}}\} > \gamma_{\mathsf{MSIS}}.$$

.

Ajtai commitment scheme. We review the Ajtai commitment scheme [Ajt96] where the messages are ring elements with *small* norms.

Definition 5. ***Ajtai commitment [Ajt96]****. Let* λ *be the security parameter and* κ, q, γ *be positive integers. Suppose a prover commits to a* m*-dimensional* short *vector* $\vec{\boldsymbol{f}} \in \mathcal{R}_q^m$ *(with small norms). The Ajtai commitment is as follows:*

- $\mathsf{Setup}(1^\lambda)$*: Sample a matrix* $G \xleftarrow{\$} \mathcal{R}_q^{\kappa \times m}$*. Output* $\mathsf{ck} := G$.
- $\mathsf{Commit}_{\mathsf{ck}}(\vec{\boldsymbol{f}})$*: Given* $\vec{\boldsymbol{f}}$ *such that* $\|\vec{\boldsymbol{f}}\|_\infty \leq \gamma$*, output* $\mathsf{Com}(\vec{\boldsymbol{f}}) := G\vec{\boldsymbol{f}}$.

A commitment scheme is *binding* if it is difficult to find two different openings for the same commitment and *hiding* if it reveals no information about the input message (formally defined in Appendix A). The Ajtai commitment is clearly binding if $\mathsf{MSIS}(\kappa, m, q, 2\gamma)$ is hard. It can also support hiding by appending a small random vector to the message [BDL$^+$18, ACK21]. For simplicity, we consider the randomness as part of the witness in our protocols, as noted in [BLNS20, ACK21].

2.4 Lattice-based Σ-Protocol

Σ-Protocol. A Σ-protocol is a public coin interactive proof system to allow a prover to convince a verifier that a statement is true. It has three properties: completeness, special soundness, and special honest-verifier zero-knowledge (special HVZK), which are formally defined in Appendix A. Here we start from a simple example in discrete logarithm settings to show the knowledge of a secret $\vec{\boldsymbol{f}}$ such that $F = \mathsf{Com}(\vec{\boldsymbol{f}})$ for a public F. Specifically, the prover samples a masking vector $\vec{\boldsymbol{t}}$ and sends $T = \mathsf{Com}(\vec{\boldsymbol{t}})$. The verifier challenges with a random challenge λ and the prover responds with $\vec{\boldsymbol{g}} = \lambda\vec{\boldsymbol{f}} + \vec{\boldsymbol{t}}$. Finally, the verifier checks $\mathsf{Com}(\vec{\boldsymbol{g}}) = \lambda F + T$.

Rejection sampling [Lyu12]. In lattice settings, the verifier also needs to check the norm bound of $\vec{\boldsymbol{g}}$ to ensure the hardness of MSIS problem in Definition 4. Therefore, $\vec{\boldsymbol{t}}$ must be sampled from a distribution (denoted as $D_{\gamma/2}$) with a larger range to hide $\lambda\vec{\boldsymbol{f}}$ term, while the bound of the distribution should be manageable for the hardness of MSIS. Besides, the prover cannot output $\vec{\boldsymbol{g}}$ directly since the distribution of $\vec{\boldsymbol{g}}$ leaks the information of $\vec{\boldsymbol{f}}$ when $\vec{\boldsymbol{t}}$ is not uniformly sampled from the field. For instance, consider one-bit message $f \in \{0,1\}$, the distribution of $\boldsymbol{g}$ is the same as the distribution of $\boldsymbol{t}$ when $f = 0$, and will shift by λ if $f = 1$. Anyone can infer f by observing the distribution of $\boldsymbol{g}$. Existing solutions adopt an additional *rejection sampling* algorithm to reject responses that are "out-of-bounds". Specifically, only $\vec{\boldsymbol{g}}$'s that can be "touched" by all possible values of $\vec{\boldsymbol{f}}$ and follow an expected distribution are acceptable. As we directly use the results of rejection sampling in this paper, we only briefly summarize rejection sampling [Lyu12] in Algorithm 1, where $\tau := \|\lambda\vec{\boldsymbol{f}}\|$ and $\phi := \gamma/(2\tau)$. Returning 1 means $\vec{\boldsymbol{g}}$ passes the rejection sampling.

Algorithm 1 Rejection Sampling [Lyu12]

```
Rej($\vec{\boldsymbol{g}}, \lambda\vec{\boldsymbol{f}}, \phi, \tau$)
1: $\gamma := 2\phi\tau$; $\mu(\phi) := \exp(\frac{12}{\phi} + \frac{1}{2\phi^2})$; $u \xleftarrow{\$} [0,1)$
2: if $u > \frac{1}{\mu(\phi)} \cdot \exp(\frac{-4\langle\vec{\boldsymbol{g}},\lambda\vec{\boldsymbol{f}}\rangle + 2\|\lambda\vec{\boldsymbol{f}}\|^2}{\gamma^2})$ then
3:     Return $\perp$
4: end if
5: Return 1
```

Lyubashevsky also proves the probability of Rej outputting 1 is within 2^{-100} of $1/\mu(\phi)$; and when the output is 1, the statistical distance between the distribution of $\vec{\boldsymbol{g}}$ and $D^m_{\gamma/2}$ is at most 2^{-100} [Lyu12].

Lattice-based *Σ*-Protocol. We summarize the lattice-based Σ-protocol in Protocol 1. The major differences with that in discrete logarithm settings is highlighted in blue.

Protocol 1 Lattice-Based *Σ*-Protocol

1. $\mathcal{P}$: Sample $\vec{\boldsymbol{t}} \xleftarrow{\$} D^m_{\gamma/2}$ and set $T := \mathsf{Com}(\vec{\boldsymbol{t}})$.
2. $\mathcal{P} \to \mathcal{V}$: T.
3. $\mathcal{V} \to \mathcal{P}$: $\lambda \xleftarrow{\$} \mathcal{C}$.
4. $\mathcal{P}$: Set $\vec{\boldsymbol{g}} := \lambda\vec{\boldsymbol{f}} + \vec{\boldsymbol{t}}$ and run $\mathsf{Rej}(\vec{\boldsymbol{g}}, \lambda\vec{\boldsymbol{f}}, \phi, \tau)$.
5. $\mathcal{P} \to \mathcal{V}$: $\vec{\boldsymbol{g}}$.
6. $\mathcal{V}$: Check $\mathsf{Com}(\vec{\boldsymbol{g}}) = \lambda F + T$ and $\|\vec{\boldsymbol{g}}\|_\infty \leq \gamma$.

The approach described above remains inefficient, as straightforward soundness proofs under lattice assumptions are not viable. Consider a simple case with two accepting transcripts, $(T, \lambda_1, \vec{\boldsymbol{g}}_1)$ and $(T, \lambda_2, \vec{\boldsymbol{g}}_2)$. Though the extractor can derive $(\lambda_2 - \lambda_1)F = \mathsf{Com}(\vec{\boldsymbol{g}}_2 - \vec{\boldsymbol{g}}_1)$, it cannot directly output $\vec{\boldsymbol{f}}' = (\lambda_2 - \lambda_1)^{-1}(\vec{\boldsymbol{g}}_2 - \vec{\boldsymbol{g}}_1)$ because: (1) $(\lambda_2 - \lambda_1)$ may not be invertible; and (2) even if $(\lambda_2 - \lambda_1)$ is invertible, $(\lambda_2 - \lambda_1)^{-1}$ may not be sufficiently short, resulting in an invalid extracted opening. The former issue can be mitigated by restricting challenges are sampled from a strong challenge space $\mathcal{C}$. For the latter, Boneh and Chen proposed a solution in [BC24], ensuring that the extracted witness maintains a small norm by imposing a norm restriction on the input vector $\vec{\boldsymbol{f}}$ through the use of **sum-check norm enforcement technique**.

Sum-check norm enforcement technique. The core idea of the norm bound reduction process is to transform the range proof into a product check, and subsequently into a sum-check using the NTT transformation. Since $\|\vec{\boldsymbol{f}}\|_\infty \le \mathcal{B}$ is equivalent to $\bigodot_{i=-\mathcal{B}}^{\mathcal{B}}(\vec{f}-\vec{i}) = \vec{0}$ in $\mathbb{Z}_q^{md}$, where $\vec{f} \in \mathbb{Z}_q^{md}$ is the concatenation of the coefficients of $\vec{\boldsymbol{f}} \in \mathcal{R}_q^m$ and $\vec{i} = i \cdot \vec{1}$. If we consider $\vec{f}$ as a NTT representation for some $\vec{\boldsymbol{f'}} \in \mathcal{R}_q^m$ (assuming $t = d$ in Definition 1 for simplicity), i.e., $\mathsf{NTT}(\vec{\boldsymbol{f'}}) = \vec{f}$, we can reformulate this as $\prod_{i=-\mathcal{B}}^{\mathcal{B}} \left(\mathsf{mle}[\vec{\boldsymbol{f'}}](\vec{b}) - i\right) = 0$ for all $\vec{b} \in \{0,1\}^{\log m}$. Consequently, the prover and verifier can engage in a sum-check protocol over $\mathcal{R}_q$ to ensure $\|\vec{\boldsymbol{f}}\|_\infty \le \mathcal{B}$.

This technique can be applied to Σ-Protocols by regarding the masking procedure as the process of folding $(F, \vec{\boldsymbol{f}})$ and $(T, \vec{\boldsymbol{t}})$ (of different rounds). We present the sum-check based Σ-Protocol in Protocol 2.

Protocol 2 sum-check basedΣ-Protocol

1. $\mathcal{P}$: Set $\vec{\boldsymbol{f'}} := \mathsf{NTT}^{-1}(\vec{\boldsymbol{f}})$, $\vec{\boldsymbol{t'}} := \mathsf{NTT}^{-1}(\vec{\boldsymbol{t}})$, and $g(\vec{\boldsymbol{x}})$ as

$$g(\vec{\boldsymbol{x}}) := \lambda \cdot \prod_{i=-\mathcal{B}}^{\mathcal{B}} \left(\mathsf{mle}[\vec{\boldsymbol{f'}}](\vec{\boldsymbol{x}}) - i\right) + \prod_{i=-\gamma}^{\gamma} \left(\mathsf{mle}[\vec{\boldsymbol{t'}}](\vec{\boldsymbol{x}}) - i\right).$$

2. $\mathcal{P}$ and $\mathcal{V}$: Engage in a sum-check on $\sum_{\vec{b}\in\{0,1\}^{\log m}} g(\vec{b}) = 0$ to reduce to an evaluation claim $g(\vec{r}) = \boldsymbol{s}_g$, where $\vec{r} \xleftarrow{\$} \mathcal{C}^{\log m}$.
3. $\mathcal{P} \to \mathcal{V}$: $\boldsymbol{s}_f := \mathsf{mle}[\vec{\boldsymbol{f'}}](\vec{r})$ and $\boldsymbol{s}_t := \mathsf{mle}[\vec{\boldsymbol{t'}}](\vec{r})$. *// additionally check the following constraints in verification*
4. $\mathcal{V}$: Set $\vec{\boldsymbol{g}}' := \mathsf{NTT}^{-1}(\vec{g})$. Check

$$\mathsf{mle}[\vec{\boldsymbol{g}}'](\vec{r}) = \lambda \boldsymbol{s}_f + \boldsymbol{s}_t, \quad \boldsymbol{s}_g = \lambda \cdot \prod_{i=-\mathcal{B}}^{\mathcal{B}} (\boldsymbol{s}_f - i) + \prod_{i=-\gamma}^{\gamma} (\boldsymbol{s}_t - i).$$

Remarks. The above approach does not ensure the zero-knowledge property since $\boldsymbol{s}_f$ and the interaction of sum-check may leak information about $\vec{\boldsymbol{f}}$.

2.5 Reduction of Knowledge

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two relations. A reduction of knowledge protocol Π [KP23] allows a prover to convince a verifier on input u_1 to derive an output u_2, such that for anyone who knows w_2 where $(u_2, w_2) \in \mathcal{R}_2$, one can extract w_1 where $(u_1, w_1) \in \mathcal{R}_1$. The protocol Π should satisfy three properties: completeness, knowledge soundness, and public reducibility (formally defined in Appendix A).

For two (or multiple) reduction of knowledge protocols Π_1 and Π_2, they can be composed in two ways [KP23]: *sequential composition* $\Pi_1 \diamond \Pi_2$ and *parallel composition* $\Pi_1 \times \Pi_2$.

Theorem 1. ***Sequential composition (Theorem 5 in [KP23]).*** *Let $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$ be three relations. Given two reduction of knowledge protocols, Π_1 from $\mathcal{R}_1$ to $\mathcal{R}_2$ and Π_2 from $\mathcal{R}_2$ to $\mathcal{R}_3$, the composed protocol $\Pi_1 \diamond \Pi_2$ is a reduction of knowledge from $\mathcal{R}_1$ to $\mathcal{R}_3$.*

Theorem 2. ***Parallel composition (Theorem 6 in [KP23]).*** *Let $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4$ be four relations. Given two reduction of knowledge protocols, Π_1 from $\mathcal{R}_1$ to $\mathcal{R}_2$ and Π_2 from $\mathcal{R}_3$ to $\mathcal{R}_4$, the composed protocol $\Pi_1 \times \Pi_2$ is a reduction of knowledge from $(\mathcal{R}_1 \times \mathcal{R}_3)$ to $(\mathcal{R}_2 \times \mathcal{R}_4)$.*

3 Technique Overview

In this section, we provide an overview of the techniques used in our protocol and the enhancements made to [BC24]

3.1 LatticeFold Soundness Proof Review

LatticeFold is a lattice-based folding protocol built upon Ajtai commitment scheme. One of our core contribution is to apply and generalize such folding protocol to Σ-Protocols by leveraging its standard soundness property. In Section 2.4, we briefly outlined how a simple Σ-Protocol can be viewed as a folding process for two instances while highlighting how LatticeFold addresses the well-known challenges of knowledge soundness.

In the following parts, we focus on handling a combination of instances that will present more complex challenges regarding soundness. To clarify our technique, we first review the folding protocol in LatticeFold which is used to fold multiple instances into one while achieving a standard soundness. The relation handled by the folding protocol of LatticeFold is as follows:

$$\mathcal{R}^b := \left\{ \begin{array}{c} (F \in \mathcal{R}_q^\kappa, \boldsymbol{s} \in \mathcal{R}_q, \vec{r} \in \mathcal{R}_q^{\log m}; \vec{\boldsymbol{f}} \in \mathcal{R}_q^m) : \\ F = \mathsf{Com}(\vec{\boldsymbol{f}}), \left\| \vec{\boldsymbol{f}} \right\| \leq b, \mathsf{mle}[\hat{\boldsymbol{f}}](\vec{r}) = \boldsymbol{s} \end{array} \right\},$$

where $\mathsf{NTT}(\hat{\boldsymbol{f}}) = \vec{\boldsymbol{f}}$. The framework of the folding protocol in LatticeFold consists of two components: (1) the initial linearization phase, which reduces the norm constraint on the witness to an evaluation claim, (2) the main folding phase, which is used to fold k-many instances of $\mathcal{R}^b$ (denoted as $(F_i, \boldsymbol{s}_i, \vec{r}_i; \vec{\boldsymbol{f}}_i)_{i=1}^k$) into a single instance of $\mathcal{R}^{\mathcal{B}}$ (which is similar to $\mathcal{R}^b$ but with a larger norm bound $\mathcal{B}$, i.e., $\left\|\vec{\boldsymbol{f}}\right\| \le \mathcal{B}$). The complete interaction between $\mathcal{P}$ and $\mathcal{V}$ in the folding protocol proceeds as follows:

1. $\mathcal{V} \to \mathcal{P}$: $(a_i, d_i)_{i=1}^k \overset{\$}{\leftarrow} (\mathcal{C} \times \mathcal{C})^k$ and $\vec{\beta} \overset{\$}{\leftarrow} \mathcal{C}^{\log m}$.
2. $\mathcal{P} \leftrightarrow \mathcal{V}$: $\mathcal{P}$ and $\mathcal{V}$ engage in a sum-check protocol on such claim:

$$\sum_{\vec{b} \in \{0,1\}^{\log m}} g(\vec{b}) = \sum_{i=1}^k a_i \boldsymbol{s}_i,$$

where

$$\begin{aligned} g(\vec{\boldsymbol{x}}) &:= g_{\mathsf{norm}}(\vec{\boldsymbol{x}}) + g_{\mathsf{eval}}(\vec{\boldsymbol{x}}), \\ g_{\mathsf{norm}}(\vec{\boldsymbol{x}}) &:= \sum_{i=1}^k \Big(d_i \cdot (\mathsf{eq}(\vec{\beta}, \vec{\boldsymbol{x}}) \cdot \prod_{j=-b}^{b} (\mathsf{mle}[\hat{\boldsymbol{f}}_i](\vec{\boldsymbol{x}}) - j)) \Big), \\ g_{\mathsf{eval}}(\vec{\boldsymbol{x}}) &:= \sum_{i=1}^k \Big(a_i \cdot (\mathsf{eq}(\vec{r}_i, \vec{\boldsymbol{x}}) \cdot \mathsf{mle}[\hat{\boldsymbol{f}}_i](\vec{\boldsymbol{x}})) \Big). \end{aligned}$$

The sum-check protocol reduces to an evaluation claim: $g(\vec{r}) = \boldsymbol{v}$.

3. $\mathcal{P} \to \mathcal{V}$: $\Big(\boldsymbol{\nu}_i := \mathsf{mle}[\hat{\boldsymbol{f}}_i](\vec{r})\Big)_{i=1}^k$
4. $\mathcal{V}$: Compute $\Big(\boldsymbol{e}_i := \mathsf{eq}(\vec{r}_i, \vec{r})\Big)_{i=1}^k$ and $\boldsymbol{e}' := \mathsf{eq}(\vec{\beta}, \vec{r})$, then check

$$\boldsymbol{v} \overset{?}{=} \sum_{i=1}^k \Big(a_i \boldsymbol{e}_i \boldsymbol{\nu}_i + d_i \boldsymbol{e}' \prod_{j=-b}^{b} (\boldsymbol{\nu}_i - j) \Big).$$

5. $\mathcal{V} \to \mathcal{P}$: $(\rho_i)_{i=1}^k \overset{\$}{\leftarrow} \mathcal{C}^k$.
6. $\mathcal{V}$: Output $(F, \boldsymbol{s}, \vec{r})$, where

$$F := \sum_{i=1}^k \rho_i F_i, \quad \mathsf{NTT}(\boldsymbol{s}) = \sum_{i=1}^k \rho_i \mathsf{NTT}(\boldsymbol{s}_i).$$

7. $\mathcal{P}$: Output $\vec{\boldsymbol{f}} = \sum_{i=1}^k \rho_i \vec{\boldsymbol{f}}_i$.

This framework ensures knowledge soundness, as the extractor can derive witness vectors with a specified norm bound from the transcripts and outputs during extraction. We illustrate this by first considering the simpler case when $k = 2$ (also see in Theorem 3.2 in [BC24]).

The extractor first rewinds twice with a *same* sum-check challenge $\vec{r}$ and *different* folding challenges, $(\rho_1^{(1)}, \rho_2^{(1)})$ and $(\rho_1^{(2)}, \rho_2^{(2)})$, to obtain $\vec{\boldsymbol{f}}^{(1)}$ and $\vec{\boldsymbol{f}}^{(2)}$ of norm bounded by $\mathcal{B}$ (ensured by the careful selection of the random challenge set). Then, $\vec{\boldsymbol{f}}_1$ and $\vec{\boldsymbol{f}}_2$ can be extracted by solving the following system:

$$\begin{aligned}
\vec{\boldsymbol{f}}^{(1)} &= \rho_1^{(1)} \cdot \vec{\boldsymbol{f}}_1 + \rho_2^{(1)} \cdot \vec{\boldsymbol{f}}_2;\\
F^{(1)} &= \rho_1^{(1)} \cdot F_1 + \rho_2^{(1)} \cdot F_2;\\
\vec{\boldsymbol{f}}^{(2)} &= \rho_1^{(2)} \cdot \vec{\boldsymbol{f}}_1 + \rho_2^{(2)} \cdot \vec{\boldsymbol{f}}_2;\\
F^{(2)} &= \rho_1^{(2)} \cdot F_1 + \rho_2^{(2)} \cdot F_2.
\end{aligned} \tag{2}$$

Since $\vec{\boldsymbol{f}}^{(1)}$ and $\vec{\boldsymbol{f}}^{(2)}$ should be verified with small norms, the MSIS assumption ensures that no other set of vectors can bind to $F^{(1)}$ and $F^{(2)}$. Therefore, $(\vec{\boldsymbol{f}}_1, \vec{\boldsymbol{f}}_2)$ is the unique solution to the system above.

Next, we prove $\vec{\boldsymbol{f}}_1$ and $\vec{\boldsymbol{f}}_2$ have a small norm. Rewind twice with $(\rho_1^{(1)}, \rho_2^{(1)})$ and $(\rho_1^{(2)}, \rho_2^{(2)})$ (same ρ as before) but under a different sum-check challenge $\vec{r}'$. Since the folding challenges are identical, $F^{(1)}$ and $F^{(2)}$ must also be identical, leading to the same vectors $\vec{\boldsymbol{f}}_1$ and $\vec{\boldsymbol{f}}_2$ within the same system. Therefore, even under different independent sum-check challenges, $(\vec{\boldsymbol{f}}_1, \vec{\boldsymbol{f}}_2)$ are expected to satisfy the sum-check relation reduced from the norm bound constraint. This means that the sum-check verification is independent of the selected challenge, that is: $(\vec{\boldsymbol{f}}_1, \vec{\boldsymbol{f}}_2)$ are determined *before* the sum-check challenges are sampled in extraction (i.e., they are *independent* from $\vec{r}$). Therefore, based on the soundness of the sum-check and the original MLE relation, the norm of $\vec{\boldsymbol{f}}_1$ and $\vec{\boldsymbol{f}}_2$ are guaranteed to be bounded. In the more complex case where $k > 2$, the analysis follows a similar approach, except that we must fix the other challenge parameters and examine different witnesses individually.

LatticeFold undoubtedly provides key insights into the soundness problem in lattice schemes for multi-instance folding operations. However, it has certain limitations—when folding protocols are invoked multiple times within a single system, each call necessitates a check on the input norm, adding extra sum-check processes. We believe this process can be optimized in practice, allowing the folding protocol to be generalized for broader applications.

3.2 Technique Overview

Our goal is to develop a lattice-based Σ-protocol with a standard soundness for handling multiple relation instances, similar to [AC20, ACF21], but with enhancements that relax restrictions on homomorphic relations and the use of discrete logarithm settings by redefining a new relation $\mathcal{R}_{\mathsf{poly}}^{\mathcal{B}}$ in lattice for arbitrary polynomial relations. Our protocol consists of two steps:

(1) Amortization of multiple instances. Our first step is to fold multiple instances of $\mathcal{R}_{\mathsf{poly}}^{\mathcal{B}}$ into one. The main challenges in this process come from the non-linear nature of polynomial relations and issues introduced by lattice settings. To address these challenges, first, we convert non-linear polynomial relations into linear ones, and then amortize these instances by running an optimized protocol based on [BC24].

For the linearization aspect, we propose a linearization protocol that generalizes the work in [BC24] by adding a process for handling polynomial constraints. Specifically, we address the non-linear property of arbitrary polynomial functions by applying an MLE transformation on both sides. This converts the polynomial constraint $h(\vec{\boldsymbol{x}}) = \vec{\boldsymbol{y}}$ into $h(\mathsf{mle}[\vec{\boldsymbol{x}}'](\vec{b})) = \mathsf{mle}[\vec{\boldsymbol{y}}'](\vec{b})$ for all $\vec{b}$ in its binary domain, where $\mathsf{NTT}(\vec{\boldsymbol{x}}') = \vec{\boldsymbol{x}}$ and $\mathsf{NTT}(\vec{\boldsymbol{y}}') = \vec{\boldsymbol{y}}$. As a result, the original polynomial constraint reduces to a sum-check constraint, and then to an evaluation check. Ultimately, the non-linear relation $\mathcal{R}^{\mathcal{B}}_{\mathsf{poly}}$ is reduced into a new linear relation $\mathcal{R}^{\mathcal{B}}_{\mathsf{mle}}$.

After linearization, multiple instances of $\mathcal{R}^{\mathcal{B}}_{\mathsf{mle}}$ can be folded together to form an amortized instance. To address norm growth and knowledge soundness challenges, we utilize the same "split-and-fold" approach as in [BC24]. Considering the limitations discussed in 3.1, we propose an optimized protocol that extends LatticeFold's folding protocol.

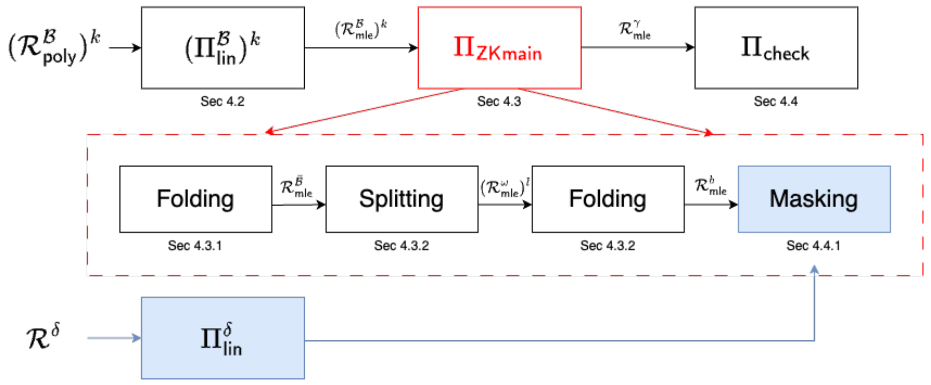
By analyzing the soundness proof of the folding protocol in LatticeFold (see Section 3.1), we found that ensuring the extracted witness has a regulated norm relies on two factors: (1) When the output witness has a small norm that binds to the output commitment in Equation 2, there exists a unique set of solutions that corresponds to the output. (2) The extracted witnesses must be independent of the sum-check challenge to ensure the extracted witnesses satisfy the sum-check claim (i.e., the small norm constraint). Based on this observation, we propose a lemma to show that if a lattice-based protocol with multiple operations meets these two criteria, then when its output is constrained, connecting it with a pre-check protocol that performs sum-check based norm enforcement can ensure that the entire system maintains knowledge soundness property (see details in 4.3, Lemma 3).

With Lemma 3, we can "concatenate" multiple split-and-fold operations together to propose a *single, "large"* protocol. In this protocol, each splitting and folding process can be invoked multiple times with just one norm check (i.e, one sum-check), while preserving security properties.

(2) Σ-Protocol. Our second step is to run a lattice-based Σ-protocol to verify the validity of the folded instance. Recall that the first phase of a Σ-protocol is to sample a masking vector to hide the witness. Unlike [AC20], the masking vector in our protocol is applied to the folded instance, rather than directly to the original multiple instances, to avoid inefficient sum-checks on instances with a large norm. Accordingly, the masking process is actually a folding process for two linear relation instances and can therefore be integrated into the aforementioned "*large*" protocol.

The final issue we need to address is to add zero-knowledge characteristics into the entire system. We accomplish this by replacing the MLE claim and sum-check process with zero-knowledge approaches, specifically by adding a hiding factor on the original claims and introducing a masking polynomial into the sum-check procedure.

Figure 1 illustrates the entire system's workflow.



$(\mathcal{R}^{\mathcal{B}}_{\text{poly}})^k$
$(\Pi^{\mathcal{B}}_{\text{lin}})^k$
Sec 4.2
$(\mathcal{R}^{\mathcal{B}}_{\text{mle}})^k$
Π_{ZKmain}
Sec 4.3
$\mathcal{R}^{\gamma}_{\text{mle}}$
Π_{check}
Sec 4.4
Folding
Sec 4.3.1
$\mathcal{R}^{\bar{\mathcal{B}}}_{\text{mle}}$
Splitting
Sec 4.3.2
$(\mathcal{R}^{\omega}_{\text{mle}})^l$
Folding
Sec 4.3.2
$\mathcal{R}^{b}_{\text{mle}}$
Masking
Sec 4.4.1
$\mathcal{R}^{\delta}$
$\Pi^{\delta}_{\text{lin}}$

Fig. 1: The entire system's work flow.

4 Efficient Σ-Protocol for Polynomial Relations

In this section, we construct a Σ-protocol that can efficiently handle multiple polynomial relations in lattice settings. Section 4.1 to 4.4 outline the subcomponents dedicated to achieving specific sub-goals, while Section 4.5 offers a complete lattice-based Σ-protocol for polynomial relations.

4.1 Polynomial Relation in Lattice Settings

Our starting point is the amortization technique for Σ-protocols [AC20, ACF21], which allows a prover to prove multiple instances of a homomorphic relation at the cost of a single instance. Attema et al. describe this technique in discrete logarithm settings [ACF21]. Consider a homomorphic relation

$$\mathcal{R}_{\text{hom}} := \left\{ \begin{array}{c} (F \in \mathbb{G}, \vec{v} \in \mathbb{Z}_q^m; \vec{f} \in \mathbb{Z}_q^m) : \\ \mathsf{Com}(\vec{f}) = F, h(f_i) = v_i \quad \forall i \in [m] \end{array} \right\}, \tag{3}$$

where h is a homomorphic function over $\mathbb{Z}_q$ and $\mathbb{G}$ is an elliptic curve group with $\mathbb{Z}_q$ as its scalar field. For k instances where $\left(F_j, \vec{v}_j; \vec{f}_j\right) \in \mathcal{R}_{\text{hom}}$ for all $j \in [k]$, it is equivalent to set $F := \sum_{j=0}^{k-1} \rho^j F_j, \vec{v} := \sum_{j=0}^{k-1} \rho^j \vec{v}_j, \vec{f} := \sum_{j=0}^{k-1} \rho^j \vec{f}_j$ with a random challenge ρ. The prover then proves $\left(F, \vec{v}; \vec{f}\right) \in \mathcal{R}_{\text{hom}}$. This process, denoted as the amortization in [AC20], can be regarded as folding k-many $\mathcal{R}_{\text{hom}}$ relations to one.

The requirement of h being a *homomorphic* function limits its practical applications. In many real-world scenarios, such as range proofs, h is a non-linear polynomial. In this paper, we consider a more general case where h can be any *polynomial*. We begin by generalizing Equation (3) to polynomial relations in lattice settings. A straightforward approach is to regard each f_i as $\boldsymbol{f}_i \in \mathcal{R}_q$.

Consequently, $\mathcal{R}^{\mathcal{B}}_{\mathsf{poly}}$ is defined as

$$\mathcal{R}^{\mathcal{B}}_{\mathsf{poly}} := \left\{ \begin{array}{c} \big((F \in \mathcal{R}_q^{\kappa}, \vec{\boldsymbol{v}} \in \mathcal{R}_q^m); \vec{\boldsymbol{f}} \in \mathcal{R}_q^m\big) : \\ \|\vec{\boldsymbol{f}}\|_\infty \le \mathcal{B}, \mathsf{Com}(\vec{\boldsymbol{f}}) = F, h(\boldsymbol{f}_i) = \boldsymbol{v}_i \quad \forall i \in [m] \end{array} \right\}. \tag{4}$$

Alternatively, we can divide a *single* $\boldsymbol{f}_i \in \vec{\boldsymbol{f}}$ into d/t secrets $(\boldsymbol{f}_{i,j})_{j=0}^{d/t-1}$ and find $\boldsymbol{f}'_i := (\boldsymbol{f}'_{i,0}, \cdots, \boldsymbol{f}'_{i,d/t-1}) \in \mathcal{R}_q^{d/t}$ as well as $\boldsymbol{v}'_i := (\boldsymbol{f}'_{i,0}, \cdots, \boldsymbol{f}'_{i,d/t-1}) \in \mathcal{R}_q^{d/t}$ where $\mathsf{NTT}(\boldsymbol{f}'_{i,j}) = \boldsymbol{f}_{i,j}$, $\mathsf{NTT}(\boldsymbol{v}'_{i,j}) = \boldsymbol{v}_{i,j}$. The polynomial claim $h(\boldsymbol{f}_i) = \boldsymbol{v}_i$ then becomes $h(\boldsymbol{f}'_{i,j}) = \boldsymbol{v}'_{i,j}$ for all $i \in [m]$ and $j \in [d/t]$.

For ease of expression, let us assume $q - 1 = 2d \mod 4d$, and thus $\mathcal{R}_q \cong \mathbb{Z}_q^d$ and $t = d$. Note that this can be generalized to an arbitrary prime modulus as described in [BC24] (also see in Section 6). Accordingly, define $\vec{\boldsymbol{f}}' := (\boldsymbol{f}'_0, \cdots, \boldsymbol{f}'_{m-1})$ and $\vec{\boldsymbol{v}}' := (\boldsymbol{v}'_0, \cdots, \boldsymbol{v}'_{m-1})$ such that $\mathsf{NTT}(\vec{\boldsymbol{f}}') = \vec{\boldsymbol{f}}$ and $\mathsf{NTT}(\vec{\boldsymbol{v}}') = \vec{\boldsymbol{v}}$. We can then redefine $\mathcal{R}^{\mathcal{B}}_{\mathsf{poly}}$ as follows

$$\mathcal{R}^{\mathcal{B}}_{\mathsf{poly}} := \left\{ \begin{array}{c} \big(F \in \mathcal{R}_q^{\kappa}, \vec{\boldsymbol{v}} \in \mathcal{R}_q^m; \vec{\boldsymbol{f}} \in \mathcal{R}_q^m\big) : \\ \|\vec{\boldsymbol{f}}\|_\infty \le \mathcal{B}, \mathsf{Com}(\vec{\boldsymbol{f}}) = F, h(\boldsymbol{f}'_i) = \boldsymbol{v}'_i \quad \forall i \in [m] \end{array} \right\}. \tag{5}$$

4.2 Linearization

The amortization technique [AC20, ACF21] cannot be applied directly on $\mathcal{R}^{\mathcal{B}}_{\mathsf{poly}}$ relations since h is not homomorphic. To address this issue, we use a sumcheck based protocol to convert the polynomial claim into a linear claim (more precisely, an MLE claim). Our linearization can be regarded as a generalization of the norm enforcement technique in [BC24], which is specifically designed for grand product and Customizable Constraint System (CCS) relations [STW23].

First, we apply the same approach as in [BC24] to convert the norm claim in $\mathcal{R}^{\mathcal{B}}_{\mathsf{poly}}$ into a high-degree polynomial relation. We then linearize this high-degree polynomial with h simultaneously.

Let $\vec{f} \in \mathbb{Z}_q^{md}$ be the vector of concatenating all coefficients of $\vec{\boldsymbol{f}}$. Then, $\|\vec{\boldsymbol{f}}\|_\infty \le \mathcal{B}$ is equivalent to $\bigodot_{i=-\mathcal{B}}^{\mathcal{B}} (\vec{f} - \vec{i}) = \vec{0}$, where $\vec{i} = i \cdot \vec{1}$. Regarding $\vec{f}$ as the NTT representation of a ring vector $\vec{\boldsymbol{f}}' \in \mathcal{R}_q^m$ such that $\mathsf{NTT}(\vec{\boldsymbol{f}}') = \vec{f}$, we then have $\prod_{i=-\mathcal{B}}^{\mathcal{B}} \big(\boldsymbol{f}'_j - i\big) = 0$, for all $j \in [m]$.[2] Therefore, by conducting an MLE on $\vec{\boldsymbol{f}}'$, we can rewrite the equation as $\prod_{i=-\mathcal{B}}^{\mathcal{B}} \Big(\mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{b}) - i\Big) = 0$ for all $\vec{b} \in \{0,1\}^{\log m}$. As a result, we reduce the norm claim to a polynomial claim.

Second, consider the polynomial claim of $h(\boldsymbol{f}'_i) = \boldsymbol{v}'_i$ for all $i \in [m]$. Similarly, by applying MLEs on $\vec{\boldsymbol{f}}'$ and $\vec{\boldsymbol{v}}'$, the polynomial claim is equivalent to $h\big(\mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{b})\big) = \mathsf{mle}[\vec{\boldsymbol{v}}'](\vec{b})$ for all $\vec{b} \in \{0,1\}^{\log m}$.

Note that to prove a polynomial $p(x)$ is zero on its domain, we cannot just prove $\sum_{\vec{b} \in \{0,1\}^{\log m}} p(\vec{b}) = 0$, as some terms may cancel out each other. Instead,

[2] Since we assume $t = d$ in the NTT transform, $i = \mathsf{NTT}(i)$ holds.

we can pair each term with $\mathsf{eq}(\vec{\mu}, \vec{b})$, where $\vec{\mu}$ is a random challenge. And then to prove $\sum_{\vec{b}\in\{0,1\}^{\log m}} \mathsf{eq}(\vec{\mu}, \vec{b}) \cdot p(\vec{b}) = 0$, to imply $p(\vec{b}) = 0$ for all $\vec{b} \in \{0,1\}^{\log m}$.

With random challenges $\alpha \xleftarrow{\$} \mathcal{C}$ and $\vec{\mu} \xleftarrow{\$} \mathcal{C}^{\log m}$, we can compose the two polynomials as $g(\vec{\boldsymbol{x}}) := g_1(\vec{\boldsymbol{x}}) + \alpha g_2(\vec{\boldsymbol{x}})$, where

$$\begin{aligned} g_1(\vec{\boldsymbol{x}}) &:= \mathsf{eq}(\vec{\mu}, \vec{\boldsymbol{x}}) \cdot \prod_{i=-\mathcal{B}}^{\mathcal{B}} \Big(\mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{\boldsymbol{x}}) - i\Big), \\ g_2(\vec{\boldsymbol{x}}) &:= \mathsf{eq}(\vec{\mu}, \vec{\boldsymbol{x}}) \cdot \Big(h\Big(\mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{\boldsymbol{x}})\Big) - \mathsf{mle}[\vec{\boldsymbol{v}}'](\vec{\boldsymbol{x}})\Big). \end{aligned} \tag{6}$$

Accordingly, the prover and verifier can engage in a sum-check protocol to show $\sum_{\vec{b}\in\{0,1\}^{\log m}} g(\vec{b}) = 0$, which outputs an MLE claim.

We define the output MLE relation $\mathcal{R}^{\mathcal{B}}_{\mathsf{mle}}$ as follows

$$\mathcal{R}^{\mathcal{B}}_{\mathsf{mle}} := \left\{ \begin{array}{c} \left(F \in \mathcal{R}_q^{\kappa}, \boldsymbol{s} \in \mathcal{R}_q, \vec{r} \in \mathcal{C}^{\log m}; \vec{\boldsymbol{f}} \in \mathcal{R}_q^m\right) : \\ \|\vec{\boldsymbol{f}}\|_\infty \le \mathcal{B}, \mathsf{Com}(\vec{\boldsymbol{f}}) = F, \mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{r}) = \boldsymbol{s} \end{array} \right\}. \tag{7}$$

We provide a detailed description of the protocol $\Pi^{\mathcal{B}}_{\mathsf{lin}}$ in Protocol 3. $\Pi^{\mathcal{B}}_{\mathsf{lin}}$ converts an input instance $(F, \boldsymbol{v}; \vec{\boldsymbol{f}}) \in \mathcal{R}^{\mathcal{B}}_{\mathsf{poly}}$ to an output $(F, \boldsymbol{s}, \vec{r}; \vec{\boldsymbol{f}}) \in \mathcal{R}^{\mathcal{B}}_{\mathsf{mle}}$.

Protocol 3 $\Pi^{\mathcal{B}}_{\mathsf{lin}}$: reduce $\mathcal{R}^{\mathcal{B}}_{\mathsf{poly}}$ to $\mathcal{R}^{\mathcal{B}}_{\mathsf{mle}}$

$\mathcal{P}(F, \vec{\boldsymbol{v}}, \vec{\boldsymbol{f}}), \mathcal{V}(F, \vec{\boldsymbol{v}})$

1: $\mathcal{P}$: $\alpha \xleftarrow{\$} \mathcal{C}, \vec{\mu} \xleftarrow{\$} \mathcal{C}^{\log m}$.
2: $\mathcal{P}$ and $\mathcal{V}$: Set $g(\vec{\boldsymbol{x}}) := g_1(\vec{\boldsymbol{x}}) + \alpha g_2(\vec{\boldsymbol{x}})$ where g_1 and g_2 are defined in Equation (6). Engage in a sum-check for the claim

$$\sum_{\vec{b}\in\{0,1\}^{\log m}} g(\vec{b}) = 0.$$

The protocol reduces to check an evaluation claim $g(\vec{r}) = \boldsymbol{s}_g$, where $\vec{r} \xleftarrow{\$} \mathcal{C}^{\log m}$.
3: $\mathcal{P} \to \mathcal{V}$: $\boldsymbol{s} := \mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{r})$.
4: $\mathcal{V}$: Set $\boldsymbol{s}_v := \mathsf{mle}[\vec{\boldsymbol{v}}'](\vec{r})$ and $\boldsymbol{e}_\mu := \mathsf{eq}(\vec{\mu}, \vec{r})$. Check

$$\boldsymbol{s}_g \stackrel{?}{=} \boldsymbol{e}_\mu \cdot \prod_{i=-\mathcal{B}}^{\mathcal{B}} \Big(\boldsymbol{s} - i\Big) + \alpha \cdot \boldsymbol{e}_\mu \Big(h(\boldsymbol{s}) - \boldsymbol{s}_v\Big). \tag{8}$$

5: $\mathcal{P}$: Output $\vec{\boldsymbol{f}}$.
6: $\mathcal{V}$: Output $(F, \boldsymbol{s}, \vec{r})$.

Efficiency. Let $D := \max(\deg(h), 2\mathcal{B} + 1)$. The sum-check protocol requires $\mathcal{P}$ to send $(D + 1)\log m$-many $\mathcal{R}_q$ messages. Additionally, $\mathcal{P}$ also needs to send an MLE claim at step 3. Thus, $\Pi_{\mathsf{lin}}^{\mathcal{B}}$ has $(1 + (D+1)\log m)$ prover messages in $\mathcal{R}_q$. For the time efficiency, the complexity of the sum-check is approximately $O(mD\log^2 D)$ for $\mathcal{P}$ and $O(D \log m)$ for $\mathcal{V}$. Furthermore, checking the equation in step 4 takes $O(D)$ time.

Parallel composition. For k instances of the $(F_j, \vec{\boldsymbol{v}}_j; \vec{\boldsymbol{f}}_j) \in \mathcal{R}_{\mathsf{poly}}^{\mathcal{B}}$ for all $j \in [k]$, we can leverage the parallel composition theorem in Theorem 9 to concurrently execute k-many $\Pi_{\mathsf{lin}}^{\mathcal{B}}$ protocols (denoted as $(\Pi_{\mathsf{lin}}^{\mathcal{B}})^k$) with a single sum-check. Accordingly, the communication cost is $(k + (D+1)\log m)$ (instead of $(k + k(D+1)\log m)$) and the verification cost for the sum-check is $O(D\log m)$ (instead of $O(kD\log m)$). Specifically, this is achieved by setting $g(\vec{\boldsymbol{x}}) := \sum_{j=0}^{k-1}\left(\alpha_j g_{1,j}(\vec{\boldsymbol{x}}) + \beta_j g_{2,j}(\vec{\boldsymbol{x}})\right)$ with random challenges $\vec{\alpha}, \vec{\beta} \xleftarrow{\$} \mathcal{C}^k$, where $g_{1,j}(\vec{\boldsymbol{x}})$ and $g_{2,j}(\vec{\boldsymbol{x}})$ are defined as

$$\begin{aligned} \forall j \in [k]: \quad & g_{1,j}(\vec{\boldsymbol{x}}) := \mathsf{eq}(\vec{\mu}, \vec{\boldsymbol{x}}) \cdot \prod_{i=-\mathcal{B}}^{\mathcal{B}} \left(\mathsf{mle}[\vec{\boldsymbol{f}}'_j](\vec{\boldsymbol{x}}) - i\right), \\ \forall j \in [k]: \quad & g_{2,j}(\vec{\boldsymbol{x}}) := \mathsf{eq}(\vec{\mu}, \vec{\boldsymbol{x}}) \cdot \left(h\left(\mathsf{mle}[\vec{\boldsymbol{f}}'_j](\vec{\boldsymbol{x}})\right) - \mathsf{mle}[\vec{\boldsymbol{v}}'_j](\vec{\boldsymbol{x}})\right). \end{aligned} \tag{9}$$

Accordingly, after receiving k-many $\boldsymbol{s}_j = \mathsf{mle}[\vec{\boldsymbol{f}}'_j](\vec{r})$, $\mathcal{V}$ checks

$$\boldsymbol{s}_g \stackrel{?}{=} \sum_{j=0}^{k-1}\left(\alpha_j \boldsymbol{e}_\mu \prod_{i=-\mathcal{B}}^{\mathcal{B}}\left(\boldsymbol{s}_j - i\right) + \beta_j \boldsymbol{e}_\mu\left(h(\boldsymbol{s}_j) - \boldsymbol{s}_{v,j}\right)\right) \tag{10}$$

where $\boldsymbol{s}_g = g(\vec{r})$ is the output claim and $\boldsymbol{s}_{v,j} := \mathsf{mle}[\vec{\boldsymbol{v}}'_j](\vec{r})$.

Theorem 3. *$\Pi_{\mathsf{lin}}^{\mathcal{B}}$ is a reduction of knowledge from $\mathcal{R}_{\mathsf{poly}}^{\mathcal{B}}$ to $\mathcal{R}_{\mathsf{mle}}^{\mathcal{B}}$.*

Proof. *Public reducibility.* Given input public statements $(F, \vec{\boldsymbol{v}})$ and the transcript that includes $\vec{r}$, $\boldsymbol{s}$, α and μ, one can output $(F, \boldsymbol{s}, \vec{r})$ if the $\mathcal{V}$ checks pass and abort otherwise.

Completeness. Given a maliciously chosen input $(F, \vec{\boldsymbol{v}}; \vec{\boldsymbol{f}}) \in \mathcal{R}_{\mathsf{poly}}^{\mathcal{B}}$, the protocol outputs $(F, \boldsymbol{s}, \vec{r})$ and $\vec{\boldsymbol{f}}$ if the verifier check passes. Since g is defined as the composite of two polynomials $g_1(\vec{\boldsymbol{x}})$ and $g_2(\vec{\boldsymbol{x}})$, based on the completeness of the sum-check, Equation (19) holds and step 4 passes. Furthermore, $\mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{r}) = \boldsymbol{s}$ holds by definition. Additionally, since $(F, \vec{\boldsymbol{v}}; \vec{\boldsymbol{f}}) \in \mathcal{R}_{\mathsf{poly}}^{\mathcal{B}}$, the norm claim (i.e., $\|\vec{\boldsymbol{f}}\|_\infty \leq \mathcal{B}$) and the commitment claim (i.e., $\mathsf{Com}(\vec{\boldsymbol{f}}) = F$) hold. Thus, the output $(F, \boldsymbol{s}, \vec{r}; \vec{\boldsymbol{f}}) \in \mathcal{R}_{\mathsf{mle}}^{\mathcal{B}}$.

Knowledge soundness. Given a maliciously chosen input public statement $(F, \vec{\boldsymbol{v}})$, the extractor simulates the protocol with the malicious prover and abort if $\mathcal{V}$ rejects. Otherwise, denote the output as $(F, \boldsymbol{s}, \vec{r}; \vec{\boldsymbol{f}})$. The extractor outputs $\vec{\boldsymbol{f}}$ as the extracted witness.

Next, we show if the extractor does not abort and $(F, \boldsymbol{s}, \vec{r}; \vec{\boldsymbol{f}}) \in \mathcal{R}_{\mathsf{mle}}^{\mathcal{B}}$, then $(F, \vec{\boldsymbol{v}}; \vec{\boldsymbol{f}}) \in \mathcal{R}_{\mathsf{poly}}^{\mathcal{B}}$ for the extracted witness. Since $(F, \boldsymbol{s}, \vec{r}; \vec{\boldsymbol{f}}) \in \mathcal{R}_{\mathsf{mle}}^{\mathcal{B}}$, we have $\mathsf{Com}(\vec{\boldsymbol{f}}) = F$ and $\mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{r}) = \boldsymbol{s}$. As the binding property of the commitment scheme ensures $\vec{\boldsymbol{f}}$ binds with F and F is fixed before receiving $\vec{r}$, the extracted $\vec{\boldsymbol{f}}$ (and the polynomial g) are independent from $\vec{r}$ with overwhelming probability. Additionally, since step 4 passes, by sum-check soundness, the following equation holds with overwhelming probability over the sum-check challenges.

$$\sum_{\vec{b}\in\{0,1\}^{\log m}} g(\vec{b}) = 0. \tag{11}$$

Define $p_1(\vec{\boldsymbol{x}}) := \prod_{i=-\mathcal{B}}^{\mathcal{B}} \left(\mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{\boldsymbol{x}}) - i\right)$. By the uniqueness of MLE, we define $\widetilde{p}_1(\vec{\mu}) = \mathsf{mle}[p_1](\vec{\mu}) = \sum_{\vec{b}\in\{0,1\}^{\log m}} \mathsf{eq}(\vec{\mu},\vec{b}) \cdot p_1(\vec{b})$. Recall the definition of g_1 in Equation (6), we have $\widetilde{p}_1(\vec{\mu}) = \sum_{\vec{b}\in\{0,1\}^{\log m}} g_1(\vec{b})$. Similarly, define $p_2(\vec{\boldsymbol{x}}) := h\left(\mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{\boldsymbol{x}})\right) - \mathsf{mle}[\vec{\boldsymbol{v}}'](\vec{\boldsymbol{x}})$, we have $\widetilde{p}_2(\vec{\mu}) = \sum_{\vec{b}\in\{0,1\}^{\log m}} g_2(\vec{b})$. Recall the definition of g at step 2, we can rewrite Equation (11) as

$$\sum_{\vec{b}\in\{0,1\}^{\log m}} g(\vec{b}) = \widetilde{p}_1(\vec{\mu}) + \alpha \cdot \widetilde{p}_2(\vec{\mu}) = 0. \tag{12}$$

Since α and $\vec{\mu}$ are uniformly chosen from the challenge space, by the generalized Schwartz-Zippel lemma, we have $\widetilde{p}_1(\vec{\mu}) = 0$ and $\widetilde{p}_2(\vec{\mu}) = 0$ with overwhelming probability over α and $\vec{\mu}$. Therefore, $p_1(\vec{b}) = 0$ and $p_2(\vec{b}) = 0$ for all $\vec{b} \in \{0,1\}^{\log m}$. Accordingly, $\prod_{i=-\mathcal{B}}^{\mathcal{B}} \left(\boldsymbol{f}'_j - i\right) = 0$ for all $j \in [m]$ (which implies $\|\vec{\boldsymbol{f}}\|_\infty \leq \mathcal{B}$), and $h(\boldsymbol{f}'_i) = \boldsymbol{v}_i$ for all $i \in [m]$. Thus, the extracted $(F, \vec{\boldsymbol{v}}; \vec{\boldsymbol{f}}) \in \mathcal{R}_{\mathsf{poly}}^{\mathcal{B}}$. □

4.3 Main Protocol on Multiple Instances

After linearization, we can apply the amortization technique to multiple $\mathcal{R}_{\mathsf{mle}}^{\mathcal{B}}$ instances to get a single one. As mentioned in 3.1, directly using the folding technique from [BC24] for amortization would require additional sum-checks when multiple folding operations are invoked. To address this issue and allow multiple operations within a "large" protocol while minimizing sum-checks (ideally to just one), we first present a lemma to generalize the soundness proof of $\mathsf{Latticefold}$, then provide our main protocol according to this lemma.

Definition 6. *We define a mapping as* ***extractive injective*** *if and only if during the extraction process, the extracted system—comprising re-winding factors, private inputs, and public outputs —is "perfectly injective". In other words, each output can only extract exactly one input set.*

Remarks. The extracted system derived from Equation 2 is perfectly injective, as only one set of $(\vec{\boldsymbol{f}}_1, \vec{\boldsymbol{f}}_2)$ can be extracted, indicating that no other solutions can satisfy the extracted system. Therefore, the protocol presented in 3.1 is constitutes an ***extractive injective*** mapping.

Lemma 3. *Given a lattice-based protocol $\mathit{\Pi}_{\mathsf{preCheck}} \diamond \mathit{\Pi}_{\mathsf{G}}$, where $\mathit{\Pi}_{\mathsf{preCheck}}$ includes the sum-check based norm enforcement and $\mathit{\Pi}_{\mathsf{G}}$ performs some additional operations (e.g., folding operations). If $\mathit{\Pi}_{\mathsf{G}}$ is* ***extractive injective*** *and ensures that* ***operations on witnesses remain independent of the challenge used in the sum-check in $\boldsymbol{\mathit{\Pi}}_{\mathsf{preCheck}}$****, then when the output witness is bounded, the protocol $\mathit{\Pi}_{\mathsf{preCheck}} \diamond \mathit{\Pi}_{\mathsf{G}}$ can maintain knowledge soundness.*

Proof. Without loss of generality, we assume $\mathit{\Pi}_{\mathsf{G}}$ takes k inputs $((X_i); (\vec{\boldsymbol{x}}_i))_{i=1}^{k}$ and outputs one instance $(Y; \vec{\boldsymbol{y}})$. Moreover, we represent the process as $\mathsf{G}(\vec{\rho}, (\vec{\boldsymbol{x}}_i)_{i=1}^{k}) = \vec{\boldsymbol{y}}$ and $\mathsf{G}(\vec{\rho}, (X_i)_{i=1}^{k}) = Y$, where $\mathsf{G}(\cdot)$ is an injective function and $\vec{\rho}$ denotes the challenge sent by the verifier to conduct G. Regard $\vec{r}$ as the sum-check challenge in the final MLE claim of $\mathit{\Pi}_{\mathsf{preCheck}}$.

We illustrate the knowledge soundness proof of $\mathit{\Pi}_{\mathsf{preCheck}} \diamond \mathit{\Pi}_{\mathsf{G}}$ by first considering the simple case when $k = 2$ (the general case follows the same approach). The extractor first rewinds twice with the *same* sum-check challenge $\vec{r}$ and *different* operation challenges $(\vec{\rho}^{(1)}, \vec{\rho}^{(2)})$, producing output witnesses $(\vec{\boldsymbol{y}}^{(1)}, \vec{\boldsymbol{y}}^{(2)})$ and their corresponding commitments $(Y^{(1)}, Y^{(2)})$:

$$\begin{aligned} \vec{\boldsymbol{y}}^{(1)} &= \mathsf{G}(\vec{\rho}^{(1)}, \vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2); \\ Y^{(1)} &= \mathsf{G}(\vec{\rho}^{(1)}, X_1, X_2); \\ \vec{\boldsymbol{y}}^{(2)} &= \mathsf{G}(\vec{\rho}^{(2)}, \vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2); \\ Y^{(2)} &= \mathsf{G}(\vec{\rho}^{(2)}, X_1, X_2). \end{aligned} \tag{13}$$

When the norm of the output witness is bounded, the MSIS assumption ensures that $(\vec{\boldsymbol{y}}^{(1)}, \vec{\boldsymbol{y}}^{(2)})$ uniquely binds to the output commitments $(Y^{(1)}, Y^{(2)})$. Additionally, since G is extractive injective, there is only one set of $(\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2)$ can be extracted corresponding to the unique output set $(\vec{\boldsymbol{y}}^{(1)}, \vec{\boldsymbol{y}}^{(2)})$.

Additionally, we need to show that the extracted witnesses $(\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2)$ have a small norm. Similar to $\mathsf{LatticeFold}$, for each $\vec{\rho}^{(1)}$ and we $\vec{\rho}^{(2)}$, we rewind using the *same* operation challenges (i.e., $\vec{\rho}^{(1)}$ or $\vec{\rho}^{(2)}$) but with *different* sum-check challenge $\vec{r}'$. Since G does not take the sum-check challenge as an input, and the operation challenges $(\vec{\rho}^{(1)}, \vec{\rho}^{(2)})$ are the same, the output commitments $(Y^{(1)}, Y^{(2)})$ must also be identical. This leads to the same $(\vec{\boldsymbol{y}}^{(1)}, \vec{\boldsymbol{y}}^{(2)})$ and extracted witnesses $(\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2)$ within the system in Equation 3. Therefore, since the extracted witnesses are independent from the sum-check challenges, they are expected to satisfy the sum-check relation reduced from the norm bound constraint. This implies that the extracted witnesses are determined before the sum-check challenges are sampled in extraction. Based on the soundness of sum-check, the norms of the extracted witnesses are guaranteed to be bounded.

□

In the folding protocol of $\mathsf{LatticeFold}$, $\mathit{\Pi}_{\mathsf{preCheck}}$ corresponds to the initial linearization phase, while $\mathit{\Pi}_{\mathsf{G}}$ is the folding phase. In our design, $\mathit{\Pi}_{\mathsf{preCheck}}$ aligns with $\mathit{\Pi}_{\mathsf{lin}}^{\mathcal{B}}$, and $\mathit{\Pi}_{\mathsf{G}}$ is the $\mathit{\Pi}_{\mathsf{main}}$ described below, which consists of the following two operations.

(1) Folding Operation on k instances When dealing with k-many $\mathcal{R}_{\mathsf{mle}}^{\mathcal{B}}$ instances $(F_j, \boldsymbol{s}_j, \vec{r}; \vec{\boldsymbol{f}}_j)_{j=0}^{k-1}$, one can adopt the amortization technique [AC20, ACF21] to fold them into one. Considering that the $\mathcal{R}_{\mathsf{mle}}^{\mathcal{B}}$ instances are the output of $\Pi_{\mathsf{lin}}^{\mathcal{B}}$ process, whose witnesses are expected to have smaller norms—no more than $\mathcal{B}$, and in many applications, $\mathcal{B}$ is quite small (such as $\mathcal{B}$ is 1 in binary proofs)—we can directly fold these witnesses together without needing to split and fold each one individually.

The folding operation on k witnesses is performed as follows. First, the verifier samples a random challenge $\vec{\rho}$ from a small strong challenge space (as defined in Definition 2). Then, the prover computes the random linear combination $\vec{\boldsymbol{f}} = \sum_{j=0}^{k-1} \rho_j \vec{\boldsymbol{f}}_j$. As for the public parameters, the verifier could compute $F := \sum_{j=0}^{k-1} \rho_j \cdot F_j$, and $\boldsymbol{s}$ such that $\mathsf{NTT}(\boldsymbol{s}) = \sum_{i=0}^{k-1} \rho_j \cdot \mathsf{NTT}(\boldsymbol{s}_j)$.

(2) Splitting and Folding Operation Note that the output folded witness $\vec{\boldsymbol{f}}$ may have a larger norm $\bar{\mathcal{B}}$, which scales linearly with the number of inputs. Consequently, we may require a larger set of parameters to uphold the MSIS assumption, which will be inefficient for real-world applications. To avoid this, we can apply the "split-and-fold" technique, which divides a single $\vec{\boldsymbol{f}}$ with a large norm bound $\bar{\mathcal{B}}$ into ℓ witnesses with a smaller norm bound ω, and then folds them into one vector with a relatively small norm bound b. Accordingly, we can choose a smaller parameter set for our final protocol.

For positive integers $\bar{\mathcal{B}} = k\mathcal{B}$ and $b < \gamma$, choose ω, ℓ such that $\omega^{\ell} = \bar{\mathcal{B}}$ and $c\ell\omega = b$ (c is the norm bound of challenges, see Definition 2). For an m-size vector $\vec{\boldsymbol{f}} \in \mathcal{R}_q^m$ where $\|\vec{\boldsymbol{f}}\|_\infty \leq \bar{\mathcal{B}}$, we can split it into an $m \times \ell$ matrix $\mathsf{split}(\vec{\boldsymbol{f}}) := (\vec{\boldsymbol{f}}_0, \cdots, \vec{\boldsymbol{f}}_{\ell-1}) \in \mathcal{R}_q^{m\times\ell}$, such that $\|\vec{\boldsymbol{f}}_j\|_\infty \leq \omega$ for all $j \in [\ell]$ and $\vec{\boldsymbol{f}} := \sum_{j=0}^{\ell-1} \omega^j \cdot \vec{\boldsymbol{f}}_j$. Furthermore, we can fold them into one $\vec{\boldsymbol{f}}_*$ such that $\|\vec{\boldsymbol{f}}_*\|_\infty \leq b$ by conducting the same folding operation as mentioned above.

A detailed description of our main protocol Π_{main} is shown in Protocol 4.

Efficiency. $\mathcal{P}$ needs to send ℓ commitments in $\mathcal{R}_q^{\kappa}$ and ℓ messages in $\mathcal{R}_q$.

In terms of time efficiency, the prover requires $O(mk + m\ell)$ time for the folding operations and $O(\kappa m\ell)$ for the splitting operations. The verifier takes $O(\kappa k + \kappa\ell)$ time to compute the folded instances and $O(\kappa\ell)$ time to verify the validity of the splitting operations.

Theorem 4. *Π_{main} satisfies public reducibility and completeness.*

Proof. *Public reducibility.* Given input public statements $(F_j, \boldsymbol{s}_j, \vec{r})_{j=0}^{k-1}$ and the transcript that includes folding challenges $\vec{\rho_1}, \vec{\rho_2}$, and $(F_i, \boldsymbol{s}_i)_{i=0}^{\ell-1}$, one can output $(F_*, \boldsymbol{s}_*, \vec{r})$ if $\mathcal{V}$ passes and abort otherwise.

Completeness. Given maliciously chosen inputs $(F_j, \boldsymbol{s}_j, \vec{r}; \vec{\boldsymbol{f}}_j)_{j=0}^{k-1} \in (\mathcal{R}_{\mathsf{mle}}^{\mathcal{B}})^k$, the protocol $< \mathcal{P}, \mathcal{V} >$ proceeds as follows:

1. $\mathcal{P}$ computes $\vec{\boldsymbol{f}} = \sum_{j=0}^{k-1} \rho_{1,j} \vec{\boldsymbol{f}}_j$, then splits $\vec{\boldsymbol{f}}$ into $(\vec{\boldsymbol{f}}_0, \ldots, \vec{\boldsymbol{f}}_{\ell-1})$ and sends $F_i := \mathsf{Com}(\vec{\boldsymbol{f}}_i)$ and $\boldsymbol{s}_i := \mathsf{mle}[\vec{\boldsymbol{f}}_i'](\vec{r})$ for each $i \in [\ell]$.

Protocol 4 $\mathit{\Pi}_{\mathsf{main}}$: Main Protocol on $(\mathcal{R}^{\mathcal{B}}_{\mathsf{mle}})^k$

```
$\mathcal{P}((F_j, \boldsymbol{s}_j)_{j=0}^{k-1}, \vec{r}, (\vec{\boldsymbol{f}}_j)_{j=0}^{k-1}), \mathcal{V}((F_j, \boldsymbol{s}_j)_{j=0}^{k-1}, \vec{r})$
1: $\mathcal{V} \to \mathcal{P}$: $\vec{\rho}_1 \xleftarrow{\$} \mathcal{C}^k_{\mathsf{small}}$, $\vec{\rho}_2 \xleftarrow{\$} \mathcal{C}^\ell_{\mathsf{small}}$.
2: $\mathcal{P}$: Compute $\vec{\boldsymbol{f}} = \sum_{j=0}^{k-1} \rho_{1,j} \vec{\boldsymbol{f}}_j$.
3: $\mathcal{V}$: Compute:
       $$F := \sum_{j=0}^{k-1} \rho_{1,j} \cdot F_j, \quad \mathsf{NTT}(\boldsymbol{s}) = \sum_{i=0}^{k-1} \rho_{1,j} \cdot \mathsf{NTT}(\boldsymbol{s}_j).$$
4: $\mathcal{P}$: Set $(\vec{\boldsymbol{f}}_0, \cdots, \vec{\boldsymbol{f}}_{\ell-1}) := \mathsf{split}(\vec{\boldsymbol{f}})$.
5: $\mathcal{P} \to \mathcal{V}$: $(F_i, \boldsymbol{s}_i)_{i=0}^{\ell-1}$ such that:
       $$F_i := \mathsf{Com}(\vec{\boldsymbol{f}}_i), \quad \boldsymbol{s}_i := \mathsf{mle}[\vec{\boldsymbol{f}}_i'](\vec{r}).$$
6: $\mathcal{V}$: Check $\sum_{i=0}^{\ell-1} \omega^i \cdot F_i = F$ and $\sum_{i=0}^{\ell-1} \omega^i \cdot \boldsymbol{s}_i = \boldsymbol{s}$.
7: $\mathcal{P}$: Compute $\vec{\boldsymbol{f}}_* = \sum_{i=0}^{\ell-1} \rho_{2,i} \vec{\boldsymbol{f}}_i$
8: $\mathcal{P}$: Output $\vec{\boldsymbol{f}}_*$.
9: $\mathcal{V}$: Output $F_*, \boldsymbol{s}_*, \vec{r}$ such that:
       $$F_* := \sum_{i=0}^{\ell-1} \rho_{2,i} \cdot F_i, \quad \mathsf{NTT}(\boldsymbol{s}_*) = \sum_{i=0}^{\ell-1} \rho_{2,i} \cdot \mathsf{NTT}(\boldsymbol{s}_i). \qquad (14)$$
```

2. $\mathcal{V}$ verifies that $\sum_{i=0}^{\ell-1} \omega^i \cdot F_i \overset{?}{=} \sum_{j=0}^{k-1} \rho_{1,j} \cdot F_j$ and $\sum_{i=0}^{\ell-1} \omega^i \cdot \mathsf{NTT}(\boldsymbol{s}_i) \overset{?}{=} \sum_{j=0}^{k-1} \rho_{1,j} \cdot \mathsf{NTT}(\boldsymbol{s}_j)$. If either check fails, $\mathcal{V}$ aborts.

3. If the checks pass, $\mathcal{P}$ outputs $\vec{\boldsymbol{f}}_* = \sum_{i=0}^{\ell-1} \rho_{2,i} \vec{\boldsymbol{f}}_i$ and $\mathcal{V}$ outputs $F_*, s_*, \vec{r}$.

It can be proved that $\mathcal{V}$ accepts in the honest execution. First, since the commitment scheme is homomotphic,

$$\sum_{i=0}^{\ell-1} \omega^i \cdot F_i = \sum_{i=0}^{\ell-1} \omega^i \cdot \mathsf{Com}(\vec{\boldsymbol{f}}_i) = \mathsf{Com}(\sum_{i=0}^{\ell-1} \omega^i \cdot \vec{\boldsymbol{f}}_i) = \mathsf{Com}(\sum_{j=0}^{k-1} \rho_{1,j} \vec{\boldsymbol{f}}_j) = \sum_{j=0}^{k-1} \rho_{1,j} F_j$$

Similarly,

$$\sum_{i=0}^{\ell-1} \omega^i \cdot \mathsf{NTT}(\boldsymbol{s}_i) = \sum_{i=0}^{\ell-1} \omega^i \cdot \mathsf{NTT}(\mathsf{mle}[\vec{\boldsymbol{f}}_i'](\vec{r})) = \sum_{j=0}^{k-1} \rho_{1,j} \cdot \mathsf{NTT}(\boldsymbol{s}_j).$$

The first equality follows from the definition of $\boldsymbol{s}_i$, while the second is based on Lemma 2. $\square$

Theorem 5. *For any c, ℓ, ω and b such that $c\ell\omega = b$ and $b < \gamma$, when the norm of the output witness of Π_{main} is bounded, $\Pi_{\mathsf{lin}}^{\mathcal{B}} \diamond \Pi_{\mathsf{main}}$ satisfies knowledge soundness.*

Proof. We prove this by demonstrating that Π_{main} satisfies the properties outlined in Lemma 3.

First, we demonstrate that Π_{main} is *extractive injective*—there is just one unique valid input set corresponding to the output. For a given output $\vec{\boldsymbol{f}}_*$ with a small norm bound, the analysis in Section 3.1 guarantees that there is only one set of $(\vec{\boldsymbol{f}}_i)_{i=0}^{\ell-1}$ corresponding to $\vec{\boldsymbol{f}}_*$. Hence, computing $\sum_{i=0}^{\ell} \omega^i \vec{\boldsymbol{f}}_i$ will only yield one result vector, $\vec{\boldsymbol{f}}$. Similarly, for this fixed $\vec{\boldsymbol{f}}$, there is only one set of input solutions $(\vec{\boldsymbol{f}}_j)_{j=0}^{k-1}$ corresponding to it. Thus, Π_{main} is indeed *extractive injective*.

Furthermore, it is evident that the extracted witnesses of Π_{main} are independent of the sum-check challenge in $\Pi_{\mathsf{lin}}^{\mathcal{B}}$. According to Lemma 3, we can conclude that $\Pi_{\mathsf{lin}}^{\mathcal{B}} \diamond \Pi_{\mathsf{main}}$ possesses knowledge soundness. □

4.4 Achieving Zero-Knowledge

After executing $(\Pi_{\mathsf{lin}}^{\mathcal{B}})^k \diamond \Pi_{\mathsf{main}}$, we have an output instance $(F_*, \boldsymbol{s}_*, \vec{r}; \vec{\boldsymbol{f}}_*) \in \mathcal{R}_{\mathsf{mle}}^b$. The verification of $(F_*, \boldsymbol{s}_*, \vec{r}; \vec{\boldsymbol{f}}_*)$ can be performed by sending $\vec{\boldsymbol{f}}_*$ to the verfier and checking $(F_*, \boldsymbol{s}_*, \vec{r}; \vec{\boldsymbol{f}}_*) \in \mathcal{R}_{\mathsf{mle}}^b$ directly. However, to achieve zero knowledge for Σ-protocols, we require an additional masking step. In [AC20], the masking instance is treated as one of the k-many instances in the input of Protocol 4. This approach works well in discrete logarithm settings, as there is no norm constraint. While in lattice scenarios, adding the masking vector to the result of the first folding (denoted as $\vec{\boldsymbol{f}}$ in Protocol 4) poses a challenge: since the norm of $\vec{\boldsymbol{f}}$ may be large, fully masking it would require a vector with a much larger norm, leading to a larger parameter for MSIS and more processing overhead. To ensure efficiency, we choose to add the masking phase to the output vector $\vec{\boldsymbol{f}}_*$, which has a relatively small norm. But this introduces another issue: during the phase in which we obtain $\vec{\boldsymbol{f}}_*$, the interaction between the prover and verifier may leak some information about the witness (specifically, the prover's messages and evaluation claims of sum-check protocols may reveal the information about the witness). Thus, to make this process zero-knowledge, we need to incorporate zero-knowledge characteristics into the original sum-check process and the evaluation claims.

Masking Operation In order to mask the witness $\vec{\boldsymbol{f}}_*$, we use a masking vector $\vec{\boldsymbol{t}}$ with a norm σ. The relation of the masking instance is defined as follows:

$$\mathcal{R}^{\sigma} := \left\{ \left(T \in \mathcal{R}_q^{\kappa}; \vec{\boldsymbol{t}} \in \mathcal{R}_q^m\right) : \|\vec{\boldsymbol{t}}\|_\infty \leq \sigma, \mathsf{Com}(\vec{\boldsymbol{t}}) = T \right\}. \tag{15}$$

Then, followed the linearization process described before, we can convert $\mathcal{R}^{\sigma}$ into a linearized instance $\mathcal{R}_{\mathsf{mle}}^{\sigma}$:

$$\mathcal{R}_{\mathsf{mle}}^{\sigma} := \left\{ \begin{array}{c} \left(T \in \mathcal{R}_q^{\kappa}, \boldsymbol{s}_t \in \mathcal{R}_q, \vec{r} \in \mathcal{C}^{\log m}; \vec{\boldsymbol{t}} \in \mathcal{R}_q^m\right) : \\ \|\vec{\boldsymbol{t}}\|_\infty \leq \sigma, \mathsf{Com}(\vec{\boldsymbol{t}}) = T, \mathsf{mle}[\vec{\boldsymbol{t}}](\vec{r}) = \boldsymbol{s}_t \end{array} \right\}. \tag{16}$$

The linearization process is almost identical to the one used for $\mathcal{R}^{\mathcal{B}}_{\mathsf{poly}}$, except that the norm bound for $\vec{\boldsymbol{t}}$ is σ, and $\vec{\boldsymbol{t}}$ only needs to satisfy the norm constraint without requiring additional polynomial constraints.

The masking operation on $\vec{\boldsymbol{f}}_*$ can be regarded as a folding operation between $\mathcal{R}^{\sigma}_{\mathsf{mle}}$ and $\mathcal{R}^{b}_{\mathsf{mle}}$. Additionally, recall that in a lattice-based Σ-protocol, the prover must run a rejection sampling algorithm to obscure the distribution of the composed vector. We need to incorporate this phase into the masking process: After the prover folds witness as $\vec{\boldsymbol{g}} := \lambda\vec{\boldsymbol{f}}_* + \vec{\boldsymbol{t}}$ based on a challenge λ, he runs $\mathsf{Rej}(\vec{\boldsymbol{g}}, \lambda\vec{\boldsymbol{f}}_*, \phi, \tau)$ to ensure $\vec{\boldsymbol{g}}$ is within the expected distribution. In order to condense the overall structure, this masking operation can be incorporated into the Π_{main} protocol. We regard the updated version of Π_{main} as Π_{ZKmain}, which is shown in Protocol 5.

Protocol 5 Π_{ZKmain}: zero-knowledge version of Π_{main}

```
$\mathcal{P}((F_j, \boldsymbol{s}_j)_{j=0}^{k-1}, \vec{r}, (\vec{\boldsymbol{f}}_j)_{j=0}^{k-1}), \mathcal{V}((F_j, \boldsymbol{s}_j)_{j=0}^{k-1}, \vec{r})$
1: $\mathcal{V} \to \mathcal{P}$: $\vec{\rho}_1 \xleftarrow{\$} \mathcal{C}^k_{\mathsf{small}}$, $\vec{\rho}_2 \xleftarrow{\$} \mathcal{C}^{\ell}_{\mathsf{small}}$.
2: $\mathcal{P}$: Compute $\vec{\boldsymbol{f}} = \sum_{j=0}^{k-1} \rho_{1,j} \vec{\boldsymbol{f}}_j$.
3: $\mathcal{P}$: Set $(\vec{\boldsymbol{f}}_0, \cdots, \vec{\boldsymbol{f}}_{\ell-1}) := \mathsf{split}(\vec{\boldsymbol{f}})$.
4: $\mathcal{P} \to \mathcal{V}$: $(F_i, \boldsymbol{s}_i)_{i=0}^{\ell-1}$ such that $F_i := \mathsf{Com}(\vec{\boldsymbol{f}}_i)$, $\boldsymbol{s}_i := \mathsf{mle}[\vec{\boldsymbol{f}}_i'](\vec{r})$.
5: $\mathcal{V}$: Check:
$$\sum_{i=0}^{\ell-1} \omega^i \cdot F_i \overset{?}{=} \sum_{j=0}^{k-1} \rho_{1,j} \cdot F_j, \quad \sum_{i=0}^{\ell-1} \omega^i \cdot \mathsf{NTT}(\boldsymbol{s}_i) \overset{?}{=} \sum_{j=0}^{k-1} \rho_{1,j} \cdot \mathsf{NTT}(\boldsymbol{s}_j). \quad (17)$$
6: $\mathcal{P}$: Compute $\vec{\boldsymbol{f}}_* = \sum_{i=0}^{\ell-1} \rho_{2,i} \vec{\boldsymbol{f}}_i$
7: $\mathcal{V} \to \mathcal{P}$: $\lambda \xleftarrow{\$} \mathcal{C}$.
8: $\mathcal{P}$: $\vec{\boldsymbol{t}} \xleftarrow{\$} \mathcal{R}_q^m$ with $\left\|\vec{\boldsymbol{t}}\right\|_\infty \leq \delta$, then compute $\vec{\boldsymbol{g}} = \lambda\vec{\boldsymbol{f}}_* + \vec{\boldsymbol{t}}$ and run $\mathsf{Rej}(\vec{\boldsymbol{g}}, \lambda\vec{\boldsymbol{f}}_*, \phi, \tau)$.
9: $\mathcal{P} \to \mathcal{V}$: $\vec{\boldsymbol{g}}, T = \mathsf{Com}(\vec{\boldsymbol{t}}), \boldsymbol{s}_t = \mathsf{mle}[\vec{\boldsymbol{t}}'](\vec{r})$.
```

Theorem 6. *Π_{ZKmain} still satisfies Theorem 4 and Theorem 5.*

Proof. The zero-knowledge version of Π_{main} is padded with an additional masking operation, which is essentially a folding operation. According to Section 4.3, Π_{ZKmain} will still maintain completeness and public reducibility. Moreover, it continues to fulfill the requirements stated in Lemma 3, which implies that $\Pi^{\mathcal{B}}_{\mathsf{lin}} \diamond \Pi_{\mathsf{ZKmain}}$ still preserves knowledge soundness. $\square$

When $(\sigma + cb) \leq \gamma$, the output of Π_{ZKmain} would be an instance in $\mathcal{R}^{\gamma}_{\mathsf{mle}}$, which can be further verified by running Protocol 6.

Protocol 6 Π_{check}: check the properties of $\vec{\boldsymbol{g}}$

```
$\mathcal{P}(F_*, \boldsymbol{s}_*, \vec{r}; \vec{\boldsymbol{f}}_*), \mathcal{V}(F_*, \boldsymbol{s}_*, \vec{r})$
1: $\mathcal{P} \to \mathcal{V}$: the output of $\mathsf{Rej}(\vec{\boldsymbol{g}}, \lambda\vec{\boldsymbol{f}}_*, \phi, \tau)$, denote as $\vec{\boldsymbol{g}}$.
2: $\mathcal{V}$: Check
       $\mathsf{Com}(\vec{\boldsymbol{g}}) \overset{?}{=} \lambda F_* + T, \quad \mathsf{NTT}(\boldsymbol{s}_{\mathcal{G}}) \overset{?}{=} \lambda\mathsf{NTT}(\boldsymbol{s}_*) + \mathsf{NTT}(\boldsymbol{s}_t), \quad \|\vec{\boldsymbol{g}}\|_\infty \le \gamma$
```

ZK Sum-Check and Zk Claims In $\Pi_{\mathsf{lin}}^{\mathcal{B}}$, the MLE claim of $\mathsf{mle}[\vec{\boldsymbol{f}}'_j](\vec{r}) = \boldsymbol{s}_j$ can leak the information of $\vec{\boldsymbol{f}}_j$. Additionally, since the sum-check is executed as a sub-protocol, the interaction along with the output of the sum-check is not zero-knowledge, which leaks the information about the witness.

We first show how to hide $\vec{\boldsymbol{f}}_j$ within the MLE claim $\mathsf{mle}[\vec{\boldsymbol{f}}'_j](\vec{r}) = \boldsymbol{s}_j$ and the output of a sum-check. Consider μ claims on a same $\vec{\boldsymbol{f}}_j$. We concatenate $\vec{\boldsymbol{f}}_j$ with a randomly sampled vector $\vec{\boldsymbol{r}}_f \xleftarrow{\$} \mathcal{R}_q^{\mu}$ as $\vec{\boldsymbol{f}}_{j,r} := (\vec{\boldsymbol{f}}_j, \vec{\boldsymbol{r}}_f)$. Accordingly, the claim on $\vec{\boldsymbol{f}}_r$ is $\mathsf{mle}[\vec{\boldsymbol{f}}'_{j,r}](\vec{r}, \vec{r}_1) = \boldsymbol{s}_j + \mathsf{mle}[\vec{\boldsymbol{r}}'_f](\vec{r}_1)$, which is uniform in $\mathcal{R}_q$. Additionally, μ claims on a same $\vec{\boldsymbol{f}}_r$ are indistinguishable from μ uniformly sampled points in $\mathcal{R}_q$. This allows the simulator to sample uniformly from $\mathcal{R}_q$ to simulate the claims. Note that, the norm of $\vec{\boldsymbol{r}}_f$ can be large, it should not be included in the norm enforcement in $\Pi_{\mathsf{lin}}^{\mathcal{B}}$ and the commitment. (The commitment scheme should use a different randomness of small norm.)

Moreover, to achieve a zero-knowledge sum-check, we employ the idea in [XZZ+19] and generalize it on rings. For any arbitrary ring $\bar{\mathcal{R}}$, to mask a multilineal polynomial $g(\vec{\boldsymbol{X}}) \in \bar{\mathcal{R}}^{\le 1}[\boldsymbol{X}_0, \cdots, \boldsymbol{X}_{\mu-1}]$, we introduce a masking polynomial

$$g_r(\boldsymbol{x}_0, \cdots, \boldsymbol{x}_{\mu-1}) := \boldsymbol{a} + \boldsymbol{a}_0\boldsymbol{x}_0 + \cdots + \boldsymbol{a}_{\mu-1}\boldsymbol{x}_{\mu-1},$$

where $\boldsymbol{a}$ and $\boldsymbol{a}_i$ are uniformly sampled from $\bar{\mathcal{R}}$ for all $i \in [\mu]$. Denote the original claim as $\boldsymbol{v} := \sum_{\vec{b}\in\{0,1\}^\mu} g(\vec{b})$ and $\boldsymbol{v}_r := \sum_{\vec{b}\in\{0,1\}^\mu} g_r(\vec{b})$. With a challenge $\zeta \xleftarrow{\$} \mathcal{C}$, the prover and verifier engage in the sum-check protocol on $\boldsymbol{v} + \zeta\boldsymbol{v}_r = \sum_{\vec{b}\in\{0,1\}^\mu} \big(g(\vec{b}) + \zeta g_r(\vec{b})\big)$. After the protocol, the verifier obtains a claim on $g(\vec{r}) + \zeta g_r(\vec{r})$. The prover then sends $g_r(\vec{r})$, which allow the verifier to compute the claim on $g(\vec{r})$. Since ζ is from $\mathcal{C}$, the proof follows the same logic as the result in [XZZ+19], Theorem 3.

4.5 Putting Everything Together

By substituting the sum-checks in $\Pi_{\mathsf{lin}}^{\mathcal{B}}$ with zero-knowledge sum-checks, and replacing the evaluation claims with hiding claims, we obtain the desired protocol $\Pi := \Pi_{\mathsf{lin}}^{\delta} \times ((\Pi_{\mathsf{lin}}^{\mathcal{B}})^k \diamond \Pi_{\mathsf{ZKmain}}) \diamond \Pi_{\mathsf{check}}$.

The full protocol is shown in Protocol 7.

Protocol 7 Π: Σ-Protocol for k polynomial relation instances

$\mathcal{P}((F_j, \vec{\boldsymbol{v}}_j)_{j=0}^{k-1}, (\vec{\boldsymbol{f}}_j)_{j=0}^{k-1}), \mathcal{V}((F_j, \vec{\boldsymbol{v}}_j)_{j=0}^{k-1})$

(1) Run zero knowledge $\Pi_{\mathsf{lin}}^{\mathcal{B}}$:

1: $\mathcal{V} \to \mathcal{P}$: challenge $\zeta \xleftarrow{\$} \mathcal{C}$

2: $\mathcal{P}$: $(\alpha_j, \beta_j)_{j=0}^{k-1} \xleftarrow{\$} (\mathcal{C} \times \mathcal{C})^k$, $\vec{\mu} \xleftarrow{\$} \mathcal{C}^{\log m}$, $\vec{\boldsymbol{r}}_f \xleftarrow{\$} \mathcal{R}_q^{\log m}$, $(\boldsymbol{a}_\theta)_{\theta=0}^{\log m} \xleftarrow{\$} \mathcal{R}_q^{\log m+1}$.

3: $\mathcal{P}$: Set $g_r(\vec{\boldsymbol{x}}) = \boldsymbol{a}_0 + \boldsymbol{a}_1 \boldsymbol{x}_0 + \cdots + \boldsymbol{a}_{\log m} \boldsymbol{x}_{\log m-1}$, and compute $\boldsymbol{s}_r = \sum_{\vec{b} \in \{0,1\}^{\log m}} g_r(\vec{b})$.

4: $\mathcal{P}$ and $\mathcal{V}$: $\mathcal{P}$ Set $\vec{\boldsymbol{f}}_{j,r} = (\vec{\boldsymbol{f}}_j, \vec{\boldsymbol{r}}_f)$, $\vec{\boldsymbol{v}}_{j,r} = (\vec{\boldsymbol{v}}_j, h(\vec{\boldsymbol{r}}_f))$, $g(\vec{\boldsymbol{x}}, \vec{r}_1) := \sum_{j=0}^{k-1} (\alpha_j g_{1,j}(\vec{\boldsymbol{x}}, , \vec{r}_1) + \beta_j g_{2,j}(\vec{\boldsymbol{x}}, \vec{r}_1))$ where $g_{1,j}$ and $g_{2,j}$ are defined as below. Let $g_f(\vec{\boldsymbol{x}}, \vec{r}_1) := g(\vec{\boldsymbol{x}}, \vec{r}_1) + \zeta g_r(\vec{\boldsymbol{x}})$.

$$\forall j \in [k], g_{1,j}(\vec{\boldsymbol{x}}, \vec{r}_1) := eq(\vec{\mu}, \vec{\boldsymbol{x}}) \cdot \prod_{i=-\mathcal{B}}^{\mathcal{B}} (\mathsf{mle}[\vec{\boldsymbol{f}}'_{j,r}](\vec{\boldsymbol{x}}, \vec{r}_1) - (i + \mathsf{mle}[\vec{\boldsymbol{r}}'_f](\vec{r}_1)))$$

$$\forall j \in [k], g_{2,j}(\vec{\boldsymbol{x}}, \vec{r}_1) := eq(\vec{\mu}, \vec{\boldsymbol{x}}) \cdot (h(\mathsf{mle}[\vec{\boldsymbol{f}}'_{j,r}](\vec{\boldsymbol{x}}, \vec{r}_1)) - \mathsf{mle}[\vec{\boldsymbol{v}}'_{j,r}](\vec{\boldsymbol{x}}, \vec{r}_1))$$

Engage in a sum-check for the claim

$$\sum_{\vec{b} \in \{0,1\}^{\log m}} g_f(\vec{b}, \vec{r}_1) = \zeta \boldsymbol{s}_r. \tag{18}$$

The protocol reduces to check an evaluation claim $g_f(\vec{r}, \vec{r}_1) = \boldsymbol{s}_g$.

5: $\mathcal{P} \to \mathcal{V}$: $(\boldsymbol{s}_j := \mathsf{mle}[\vec{\boldsymbol{f}}'_j](\vec{r}))_{j=0}^{k-1}$, and $g_r(\vec{r})$.

6: $\mathcal{V}$: Set $\boldsymbol{s}_{v,j} := \mathsf{mle}[\vec{\boldsymbol{v}}'_j](\vec{r})$ and $\boldsymbol{e}_\mu := \mathsf{eq}(\vec{\mu}, \vec{r})$. Check

$$\boldsymbol{s}_g - \zeta g_r(\vec{r}) \stackrel{?}{=} \sum_{i=0}^{k-1} \Big(\alpha_j \boldsymbol{e}_\mu \cdot \prod_{i=-\mathcal{B}}^{\mathcal{B}} (\boldsymbol{s}_j - i) + \beta_j \boldsymbol{e}_\mu \cdot (h(\boldsymbol{s}_j) - \boldsymbol{s}_{v,j}) \Big).$$

(2) Run Π_{ZKmain}. (See Protocol 5)

(3) Run Π_{check}:

1: $\mathcal{V}$: Compute $\boldsymbol{s}_{\mathcal{G}} = \mathsf{mle}[\vec{\boldsymbol{g}}'](\vec{r})$. Check:

$$\mathsf{Com}(\vec{\boldsymbol{g}}) \stackrel{?}{=} \lambda \sum_{i=0}^{\ell-1} \rho_{2,i} \cdot F_i + T,$$

$$\mathsf{NTT}(\boldsymbol{s}_{\mathcal{G}}) \stackrel{?}{=} \lambda \sum_{i=0}^{\ell-1} \rho_{2,i} \cdot \mathsf{NTT}(\boldsymbol{s}_i) + \boldsymbol{s}_t, \quad \|\vec{\boldsymbol{g}}\|_\infty \le \gamma.$$

Theorem 7. *Given a ring $\mathcal{R}_q$, let* $\mathsf{pp} := (\kappa, m, \boldsymbol{G}, \gamma < q/2)$ *be the public parameters such that* $\mathsf{MSIS}(\kappa, m, q, 2\gamma)$ *is hard,* $\mathcal{C}, \mathcal{C}_{\mathsf{small}} \subset \mathcal{R}_q$ *be super-poly large strong sampling sets where* $\|\mathcal{C}_{\mathsf{small}}\|_{\mathsf{op}} = c$*. Choose* b, ℓ, ω, δ *such that* $\mathcal{B} = \omega^\ell$*,* $c\ell\omega = b$ *and* $(\delta + cb) \leq \gamma$*.* $\mathit{\Pi}$ *satisfies completeness, special soundness, and special HVZK.*

Proof. *Completeness* Based on Theorem 3 and Theorem 6, $(\mathit{\Pi}^{\mathcal{B}}_{\mathsf{lin}})^k \diamond \mathit{\Pi}_{\mathsf{ZKmain}}$ serves as a public-coin reduction of knowledge from the tuple $((\mathcal{R}^{\mathcal{B}}_{\mathsf{poly}})^k, \mathcal{R}^\delta)$ to $\mathcal{R}^\gamma_{\mathsf{mle}}$. Meanwhile, $\mathit{\Pi}^\delta_{\mathsf{lin}}$ represents a reduction of knowledge from $\mathcal{R}^\delta$ to $\mathcal{R}^\delta_{\mathsf{mle}}$. Thus, $\mathit{\Pi}^\delta_{\mathsf{lin}} \times ((\mathit{\Pi}^{\mathcal{B}}_{\mathsf{lin}})^k \diamond \mathit{\Pi}_{\mathsf{ZKmain}})$ ensures completeness according to the knowledge composition theorems in [KP23]. Furthermore, since $\mathit{\Pi}_{\mathsf{check}}$ satisfies completeness, it can be concluded that $\mathit{\Pi}$ also satisfies completeness.

Special Soundness According to Theorem 6, $\mathit{\Pi}_{\mathsf{ZKmain}}$ meets the criteria specified in Lemma 3. As the norm of the output witness from $(\mathit{\Pi}^{\mathcal{B}}_{\mathsf{lin}})^k \diamond \mathit{\Pi}_{\mathsf{ZKmain}}$ is verified by $\mathit{\Pi}_{\mathsf{check}}$, $((\mathit{\Pi}^{\mathcal{B}}_{\mathsf{lin}})^k \diamond \mathit{\Pi}_{\mathsf{ZKmain}}) \diamond \mathit{\Pi}_{\mathsf{check}}$ demonstrates knowledge soundness. Specifically, the extractor first requires *two* accepting transcripts of the same T but with different λ values to recover the amortized witness. This amortized witness is then used to extract the original witnesses with a regularized norm based on ℓ and k accepting transcripts, respectively. Therefore, $\mathit{\Pi}$ demonstrates $(k, \ell, 2)$-special soundness.

Special HVZK. For $(\mathit{\Pi}^{\mathcal{B}}_{\mathsf{lin}})^k \diamond \mathit{\Pi}_{\mathsf{ZKmain}}$, denote the folding challenge in the first folding operation as $\vec{\rho}^{(1)}$. Randomly sample $\boldsymbol{s}_j \xleftarrow{\$} \mathcal{R}_q$ for all $j \in [k]$. Compute the folded commitment as $F := \sum_{j=0}^{k-1} \rho^{(1)}_j F_j$ and the folded MLE claim $\boldsymbol{s}$ such that $\mathsf{NTT}(\boldsymbol{s}) = \sum_{j=0}^{k-1} \rho^{(1)}_j \cdot \mathsf{NTT}(\boldsymbol{s}_j)$. Set $\boldsymbol{s}_g := g(\vec{r})$ where g is defined as Equation (9) by replacing $\mathsf{mle}[\vec{\boldsymbol{f}}'_j](\vec{r})$ with $\boldsymbol{s}_j$. Thus, the check of Equation (10) passes and the simulated $\boldsymbol{s}_j$'s are indistinguishable from the real ones.

For splitting operation, sample $F^*_1 \cdots, F^*_{\ell-1} \xleftarrow{\$} \mathcal{R}^\kappa_q$ and $\boldsymbol{s}^*_1 \cdots, \boldsymbol{s}^*_{\ell-1} \xleftarrow{\$} \mathcal{R}_q$. Set $F^*_0 := F - \sum_{j=1}^{\ell-1} \omega^j F^*_j$ and $\boldsymbol{s}^*_0 := \boldsymbol{s} - \sum_{j=1}^{\ell-1} \omega^j \boldsymbol{s}^*_j$. Clearly, $(F^*_j, \boldsymbol{s}^*_j)_{j=0}^{\ell-1}$ pass the check at step 5 of $\mathit{\Pi}_{\mathsf{ZKmain}}$ in $\mathit{\Pi}$. Moreover, $(F^*_j, \boldsymbol{s}^*_j)_{j=0}^{\ell-1}$ are indistinguishable from real ones due to the hiding property of claim and commitment.

Similarly, denote the folding challenge as $\vec{\rho}^{(2)}$ in the second folding operation. The simulator simply compute $F_* := \sum_{j=0}^{\ell-1} \rho^{(2)}_j F^*_j$ and $\boldsymbol{s}_*$ such that $\mathsf{NTT}(\boldsymbol{s}_*) = \sum_{i=0}^{\ell-1} \rho^{(2)}_j \cdot \mathsf{NTT}\left(\boldsymbol{s}^*_j\right)$. Finally, the simulator selects λ uniformly at random and samples $\vec{\boldsymbol{g}}$ randomly to compute $T = \mathsf{Com}(\vec{\boldsymbol{g}}) - \lambda F_*$.

Since all steps pass, the simulated transcript is a valid one and indistinguishable from a real transcript. □

5 Evaluation

Efficiency. Table 3 summarizes the efficiency of each sub-protocol used in our system. Where $D := \max(\deg(h), 2\mathcal{B} + 1)$, $D' := 2\delta + 1$, m is the length of the witness, κ is the length of the commitment.

Table 3: Efficiency of sub protocols

	prover cost	**verifier cost**	**proof size**
$\Pi_{\mathsf{lin}}^{\mathcal{B}}$ for $\mathcal{R}_{\mathsf{poly}}^{\mathcal{B}}$	$O(mD\log^2 D)$	$O(D\log m)$	$(1+(D+1)\log m)\mathcal{R}_q$
$\Pi_{\mathsf{lin}}^{\delta}$ for $\mathcal{R}^{\delta}$	$O(mD'\log^2 D')$	$O(D'\log m)$	$(1+(D'+1)\log m)\mathcal{R}_q$
Π_{ZKmain} for ($(\mathcal{R}_{\mathsf{mle}}^{\mathcal{B}})^k$,$\mathcal{R}_{\mathsf{mle}}^{\delta}$)	$O(mk+\kappa m\ell)$	$O(\kappa k)$ + $O(\kappa\ell)$	ℓ $\mathcal{R}_q^{\kappa}$ + $(\ell+m)\mathcal{R}_q$
Π_{check}	/	$O(\kappa\ell+m)$	/

Table 4: Efficiency of our protocol.

	w/o compression	**w/ compression**
Proof size	$((D+1)\log m+k+$ $\ell+(D'+1)\log m+1+m)$ $\mathcal{R}_q$ ℓ $\mathcal{R}_q^{\kappa}$	$((D+1)\log m+2\log m+k+$ $\ell+1+(D'+1)\log m)$ $\mathcal{R}_q$ $(2\log m+\ell)$ $\mathcal{R}_q^{\kappa}$
Prover time	$O(kmD\log^2 D+mD'\log^2 D'+$ $km+\kappa m\ell)$	$O(kmD\log^2 D+mD'\log^2 D'+$ $km+\kappa m\ell)$
Verifier time	$O(D\log m+kD+D'\log m$ $+\kappa k+\kappa\ell+m)$	$O(D\log m+kD+D'\log m$ $+\kappa k+\kappa\ell+\kappa m)$

As explained in the parallel composition in Section 4.2, when linearizing k instances simultaneously, we can use the parallel composition theorem to concurrently execute k-many $\Pi_{\mathsf{lin}}^{\mathcal{B}}$ protocols with a single sum-check.$(\Pi_{\mathsf{lin}}^{\mathcal{B}})^k$ has $(k+(D+1)\log m)$ prover messages in $\mathcal{R}_q$. The complexity of prover and verifier are $O(kmD\log^2 D)$ and $O(D\log m+kD)$, respectively.

When adopting the compressed Σ-protocol theory, our scheme can be further optimized. We give the overall efficiency of our protocol in Table 4.

Instantiation. Set $m=2^{16}$, $d=64$, $\kappa=9$ and let q be a 64-bit prime. By the MSIS hardness bound we can achieve 128-bit security as long as the norm bound $\log\mathcal{B}\leq 27-0.5\log m$. We choose $\gamma=2^{19}$.

Let the challenge set $\mathcal{C}_{\mathsf{small}}$ be a set of elements with $\{-1,0,1,2\}$ in $\mathcal{R}_q$. Then, we have $|\mathcal{C}_{\mathsf{small}}|=4^d=4^{64}=2^{128}$. Additionally, $\mathcal{C}_{\mathsf{small}}$ is a strong sampling set because the difference between any two distinct elements has an infinity norm of at most 3, which is less than $q^{1/16}/\sqrt{16}=4$. Thus, it is invertible.

In this set, our proof size is $16(D+D'+2)+k+l+m$. Considering k and ℓ is much less than D, D' and m, the approximate proof size is 64KB.

6 Optimization and Extension

Supporting small prime modulus. As mentioned in Section 4.1, our protocol is also compatible with a small prime modulus. By choosing a q that is significantly smaller than 2^{128}, our protocol can achieve better efficiency while still maintain 128-bit security at the same time. This adaptation is a straightforward application of the challenge space in [BC24], which we briefly summarize here.

Recall the NTT transform in Section 2.1. To ensure 128-bit security, we can choose a prime q such that $q-1 = 2t \mod 4t$ and $q^{d/t} \approx 2^{128}$. The challenge space $\mathcal{C}$ is (re)defined as follows

$$\mathcal{C} := \left\{ \boldsymbol{a}_i \in \mathcal{R}_q : \mathsf{NTT}(\boldsymbol{a}_i) = \vec{i} \right\}_{i \in \mathbb{Z}_{q^{d/t}}},$$

where $\vec{i} \in \mathbb{Z}^t_{q^{d/t}}$ is a vector containing t-many i's.

Linearizing inner-product relations. Unlike methods in [AC20, ACK21], our approach cannot directly extend to support inner-product relations due to the high-degree polynomial h. Nonetheless, since the inner-product relation is essentially a sum-check argument [BCS21], this provides us with the opportunity to seamlessly integrate the inner-product claims within the sum-check protocol.

Define the inner-product polynomial relation $\mathcal{R}^{\mathcal{B}}_{\mathsf{polyIP}}$ as follows:

$$\mathcal{R}^{\mathcal{B}}_{\mathsf{polyIP}} := \left\{ \begin{array}{c} \left(F \in \mathcal{R}^{\kappa}_q, \vec{\boldsymbol{a}} \in \mathcal{R}^m_q, \boldsymbol{v} \in \mathcal{R}_q; \vec{\boldsymbol{f}} \in \mathcal{R}^m_q\right) : \\ \|\vec{\boldsymbol{f}}\|_\infty \le \mathcal{B}, \mathsf{Com}(\vec{\boldsymbol{f}}) = F, \sum_{i=0}^{m-1} h(\boldsymbol{a}'_i, \boldsymbol{f}'_i) = \boldsymbol{v}' \end{array} \right\}, \tag{19}$$

where $\vec{\boldsymbol{a}}'$ and $\boldsymbol{v}'$ are the NTT transform of $\vec{\boldsymbol{a}}$ and $\boldsymbol{v}$, respectively.

We show how to adjust $\Pi^{\mathcal{B}}_{\mathsf{lin}}$ to convert $\mathcal{R}^{\mathcal{B}}_{\mathsf{polyIP}}$ into $\mathcal{R}^{\mathcal{B}}_{\mathsf{mle}}$. In particular, for the polynomial claim, g_2 in Equation (6) is modified as follows:

$$g_2(\vec{\boldsymbol{x}}) := h\Big(\mathsf{mle}[\vec{\boldsymbol{a}}'](\vec{\boldsymbol{x}}), \mathsf{mle}[\vec{\boldsymbol{f}}'](\vec{\boldsymbol{x}})\Big), \tag{20}$$

and in step 3 of Protocol 3, the sum-check claim becomes $\sum_{\vec{b} \in \{0,1\}^{\log m}} g(\vec{b}) = \alpha \cdot \boldsymbol{v}'$, where $g(\vec{\boldsymbol{x}}) := g_1(\vec{\boldsymbol{x}}) + \alpha \cdot g_2(\vec{\boldsymbol{x}})$. Consequently, the at step 5, Equation (19) becomes

$$\boldsymbol{s}_g = \boldsymbol{e}_\mu \cdot \prod_{i=-\mathcal{B}}^{\mathcal{B}} (\boldsymbol{s} - i) + \alpha \cdot h(\boldsymbol{s}_a, \boldsymbol{s}), \tag{21}$$

where $\boldsymbol{s}_a := \mathsf{mle}[\vec{\boldsymbol{a}}'](\vec{r})$ is computed by $\mathcal{V}$. The remaining steps are the same.

It is important to noted that the Ajtai commitment in Definition 5 can also be represented by an inner-product relation with the witness. This enables us to simultaneous linearize the commitment claim and polynomial claim simultaneously within $\Pi^{\mathcal{B}}_{\mathsf{lin}}$.

Generalize to different polynomial relations (and arithmetic circuit relations). Relation $\mathcal{R}^{\mathcal{B}}_{\mathsf{poly}}$ in Equation (5) is defined on a fixed h. Our linearization technique can be generalized to support different h_j polynomials without incurring extra costs. This can be simply achieved by setting $g_{2,j}$ in Equation (9) to

$$g_{2,j}(\vec{\boldsymbol{x}}) := \mathsf{eq}(\vec{\mu}, \vec{\boldsymbol{x}}) \cdot \left(h_j \left(\mathsf{mle}[\vec{\boldsymbol{f}}'_j](\vec{\boldsymbol{x}}) \right) - \mathsf{mle}[\vec{\boldsymbol{v}}'_j](\vec{\boldsymbol{x}}) \right).$$

The verification on Equation (10) is adjusted to h_j accordingly.

Note that arithmetic circuit relations such as R1CS (rank-1 constraint system) [GGPR13] and CCS [STW23] are special instances of the polynomial relations (with inner-product relations, which are captured by $\mathcal{R}^{\mathcal{B}}_{\mathsf{polyIP}}$). By incorporating an extra step that decomposes the witness into its binary representation and subsequently reconstructs it within arithmetic circuit relations, our approach inherently supports arithmetic circuit relations by definition. Since this is essentially the technique in [BC24] to support CCS relations, we omit the detailed description here. Besides, since we also present the case to support different h polynomials, our protocol can also handle multiple *different* arithmetic circuit relations directly, such as proving multiple R1CS and CCS relations at the same time.

7 Acknowledgement

We gratefully acknowledge Dr. Binyi CHEN from Stanford University for the technical discussions about LatticeFold. This research has received partial support from HK RGC GRF under Grants PolyU 15205624/QCFD, 15202123/QC1V, 15207522/Q93W, and 15216721/Q86A, and NSFC Youth 62302418/ZGJV.

References

AC20. Thomas Attema and Ronald Cramer. Compressed Σ-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2020.

ACF21. Thomas Attema, Ronald Cramer, and Serge Fehr. Compressing Proofs of k-out-of-n Partial Knowledge. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2021.

ACK21. Thomas Attema, Ronald Cramer, and Lisa Kohl. A Compressed Sigma-Protocol Theory for Lattices. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2021.

Ajt96. Miklós Ajtai. Generating Hard Instances of Lattice Problems. In *Proc. of the Annual ACM Symposium on Theory of Computing (STOC)*, pages 99–108. ACM, 1996.

AL21. Martin R Albrecht and Russell WF Lai. Subtractive Sets over Cyclotomic Rings: Limits of Schnorr-like Arguments over Lattices. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2021.

BBB$^+$18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short Proofs for Confidential Transactions and More. In *Proc. of the IEEE Symposium on Security and Privacy (Oakland)*. IEEE, 2018.

BC24. Dan Boneh and Binyi Chen. LatticeFold: A Lattice-based Folding Scheme and its Applications to Succinct Proof Systems. In *IACR Cryptology ePrint Archive*, 2024.

BCS21. Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Sumcheck Arguments and Their Applications. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2021.

BDL$^+$18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More Efficient Commitments from Structured Lattice Assumptions. In *Proc. of the International Conference on Security and Cryptography for Networks (SCN)*. Springer, 2018.

BLNS20. Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. A non-PCP Approach to Succinct Quantum-Safe Zero-Knowledge. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2020.

BS22. Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. 2022. `https://eprint.iacr.org/2022/1341`.

CCKP19. Shuo Chen, Jung Hee Cheon, Dongwoo Kim, and Daejun Park. Verifiable Computing for Approximate Computation. In *IACR Cryptology ePrint Archive*, 2019.

ESLL19. Muhammed F Esgin, Ron Steinfeld, Joseph K Liu, and Dongxi Liu. Lattice-Based Zero-Knowledge Proofs: New Techniques for Shorter and Faster Constructions and Applications. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2019.

ESZ22. Muhammed F Esgin, Ron Steinfeld, and Raymond K Zhao. MatRiCT+: More Efficient Post-Quantum Private Blockchain Payments. In *Proc. of the IEEE Symposium on Security and Privacy (Oakland)*, 2022.

FKNP24. Giacomo Fenzi, Christian Knabenhans, Khanh Ngoc Nguyen, and Duc Tu Pham. Lova: Lattice-based folding scheme from unstructured lattices. In *Proc. of the Annual International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2024.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic Span Programs and Succinct NIZKs without PCPs. In *Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2013.

KP23. Abhiram Kothapalli and Bryan Parno. Algebraic Reductions of Knowledge. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2023.

KTX08. Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently Secure Identification Schemes based on the Worst-Case Hardness of Lattice Problems. In *Proc. of the Annual International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer, 2008.

LNP22. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plancon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. 2022. `https://eprint.iacr.org/2022/284`.

LNSW13. San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved Zero-Knowledge Proofs of Knowledge for the ISIS Problem, and Applications. In *Proc. of the International Conference on Public Key Cryptography (PKC)*. Springer, 2013.

LS15. Adeline Langlois and Damien Stehlé. Worst-Case to Average-Case Reductions for Module Lattices. In *Designs, Codes and Cryptography*. Springer, 2015.

Lyu09. Vadim Lyubashevsky. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-based Signatures. In *Proc. of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer, 2009.

Lyu12. Vadim Lyubashevsky. Lattice Signatures Without Trapdoors. In *Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2012.

MR09. Daniele Micciancio and Oded Regev. Lattice-based Cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

STW23. Srinath Setty, Justin Thaler, and Riad Wahby. Customizable Constraint Systems for Succinct Arguments. In *IACR Cryptology ePrint Archive*, 2023.

XZZ$^+$19. Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2019.

YAZ$^+$19. Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient Lattice-based Zero-Knowledge Arguments with Standard Soundness: Construction and Applications. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2019.

ZCYW23. Min Zhang, Yu Chen, Chuanzhou Yao, and Zhichao Wang. Sigma Protocols from Verifiable Secret Sharing and Their Applications. In *Proc. of the Annual International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 208–242. Springer, 2023.

A Formal Definitions

A.1 Hiding and Binding

Given a commitment scheme, let $\mathsf{Commit}_{\mathsf{ck}}(\vec{\boldsymbol{f}};\vec{\boldsymbol{r}})$ represent the operation of concatenating the vector $\vec{\boldsymbol{r}}$ to $\vec{\boldsymbol{f}}$ and then committing to the combined vector $(\vec{\boldsymbol{f}},\vec{\boldsymbol{r}})$.

Definition 7. ***Hiding***. *The commitment scheme is hiding if for For all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{matrix}\mathsf{ck}\leftarrow\mathsf{Setup}(1^\lambda),(\vec{\boldsymbol{m}}_0,\vec{\boldsymbol{m}}_1)\leftarrow\mathcal{A}(\mathsf{ck}),\\ b\overset{\$}{\leftarrow}\{0,1\},\vec{\boldsymbol{r}}\overset{\$}{\leftarrow}D_{\mathsf{ck}},C\leftarrow\mathsf{Commit}_{\mathsf{ck}}(\vec{\boldsymbol{m}}_b;\vec{\boldsymbol{r}})\end{matrix}:\mathcal{A}(C)=b\right]\approx\frac{1}{2}.$$

where D_{ck} is the space of the randomness controller by ck.

Definition 8. ***Binding***. *The commitment scheme is hiding if for For all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{matrix}\mathsf{ck}\leftarrow\mathsf{Setup}(1^\lambda),\\ (\vec{\boldsymbol{m}}_0,\vec{\boldsymbol{r}}_0,\vec{\boldsymbol{m}}_1,\vec{\boldsymbol{r}}_1)\leftarrow\mathcal{A}(ck)\end{matrix}:\begin{matrix}\vec{\boldsymbol{m}}_0\neq\vec{\boldsymbol{m}}_1\quad\wedge\\ \mathsf{Commit}_{\mathsf{ck}}(\vec{\boldsymbol{m}}_0;\vec{\boldsymbol{r}}_0)=\mathsf{Commit}_{\mathsf{ck}}(\vec{\boldsymbol{m}}_b;\vec{\boldsymbol{r}}_b)\end{matrix}\right]\approx 0.$$

A.2 Properties of $\varSigma$-Protocols

For any Nondeterministic Polynomial time (NP) relation $\mathcal{R}_*$, let $\mathcal{G}$ be a setup algorithm that generates public parameters pp for the $\varSigma$-protocol. Given an instance $(u;w)$ where u is the public statement and w is the witness, the $\varSigma$-protocol to prove $(u;w)\in\mathcal{R}_*$ works as follows: (i) $\mathcal{P}$ sends an initial message t, (ii) $\mathcal{V}$ issues with a random challenge $\lambda\overset{\$}{\leftarrow}\mathcal{C}$ from the challenge space, and (iii) $\mathcal{P}$ provides with a response g. The properties of the $\varSigma$-protocol are defined as follows.

Completeness. When $(u;w)\in\mathcal{R}_*$, the interaction between an honest prover and an honest verifier yields an accepting transcript with probability larger than $1-\epsilon_c$, where ϵ_c is denoted as the completeness error.

Formally, for all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\begin{matrix}\mathsf{pp}\leftarrow\mathcal{G}(1^\lambda),(u,w)\leftarrow\mathcal{A}(\mathsf{pp}),\\ t\leftarrow\mathcal{P}(\mathsf{pp},u,w),\lambda\overset{\$}{\leftarrow}\mathcal{C},g\leftarrow\mathcal{P}(\lambda)\end{matrix}:\mathcal{V}(\mathsf{pp},u,t,\lambda,g)=1\right]=1-\epsilon_c.$$

Special soundness. There exists an efficient extractor $\mathcal{E}$ that on any statement u and k accepting transcripts $(t,\lambda_i,g_i)_{i=0}^{k-1}$ with common t and distinct λ_i's and g_i's, outputs a witness w' such that $(u,w')\in\mathcal{R}_*$ with probability larger than $1-\epsilon_s$, where ϵ_s is denoted as the soundness error. This is also known as the k-special soundness prosperity. A more generalized version, $(k_1,\cdots,k_\mu)$-special soundness, allows $\mathcal{E}$ to run with a $(k_1,\cdots,k_\mu)$-tree of accepting transcripts tr, where tr is a set of $\prod_{i=1}^{\mu}k_i$ accepting transcripts with the following structure. The nodes in tr correspond to the prover's messages and the edges correspond to the verifier's challenges. Every node at depth i has precisely k_i children corresponding to k_i pairwise distinct challenges. Every transcript corresponds to exactly one path from the root node to a leaf node.

Formally, for all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), (u, \mathsf{tr}) \leftarrow \mathcal{A}(\mathsf{pp}), w \leftarrow \mathcal{E}(\mathsf{pp}, u, \mathsf{tr}) : (u; w) \in \mathcal{R}\right] = 1 - \epsilon_s.$$

Special HVZK. There exists an efficient simulator that on any statement u and a challenge λ from the challenge space, outputs (t, g) such that (t, g) is indistinguishable from an accepting transcript produced by running the protocol.

Formally, for all PPT adversaries $\mathcal{A}$,

$$\begin{aligned}&\Pr\left[\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), (u, w, \lambda) \leftarrow \mathcal{A}(\mathsf{pp}), t \leftarrow \mathcal{P}(\mathsf{pp}, u, w), g \leftarrow \mathcal{P}(\lambda) : \mathcal{A}(t, \lambda, g) = 1\right] \\ \approx &\Pr\left[\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), (u, w, \lambda) \leftarrow \mathcal{A}(\mathsf{pp}), (t, \lambda, g) \leftarrow \mathcal{S}(\mathsf{pp}, u) : \mathcal{A}(t, \lambda, g) = 1\right].\end{aligned}$$

A.3 Reduction of Knowledge Protocols

Definition 9. ***Reduction of knowledge [KP23].*** *Let λ be a security parameter. Given two relations $\mathcal{R}_1$ and $\mathcal{R}_2$, denote $\langle \mathcal{P}, \mathcal{V} \rangle$ be an interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. A reduction of knowledge protocol Π from $\mathcal{R}_1$ to $\mathcal{R}_2$ consists of the following algorithms:*

- $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$*: Output public parameters* pp.
- $\langle \mathcal{P}(\mathsf{pp}, u_1, w_1), \mathcal{V}(\mathsf{pp}, u_1) \rangle \rightarrow (u_2, w_2)$*: On input public parameters* pp*, a shared public statement u_1, $\mathcal{P}$ (with a witness w_1 such that $(u_1, w_1) \in \mathcal{R}_1$) and $\mathcal{V}$ engage in Π. At the end of Π, $\mathcal{P}$ and $\mathcal{V}$ $\mathcal{V}$ output u_2 (or $\perp$ if abort), $\mathcal{P}$ additionally outputs w_2.*

Reduction of knowledge protocols satisfy the following properties:

Completeness. For every PPT adversary $\mathcal{A}$ that adaptively chooses an $\mathcal{R}_1$ instance $(u_1, w_1) \leftarrow \mathcal{A}(\mathsf{pp})$ when given $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, the protocol execution $(u_2, w_2) \leftarrow \langle \mathcal{P}(\mathsf{pp}, u_1, w_1), \mathcal{V}(\mathsf{pp}, u_1) \rangle$ satisfies $(u_2, w_2) \in \mathcal{R}_2$ if $(u_1, w_1) \in \mathcal{R}_1$.

Knowledge soundness. For every expected polynomial-time adversary $\mathcal{A}$ and malicious prover $\mathcal{P}^*$, there is an expected polynomial-time extractor $\mathcal{E}$ such that given $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(u_1, w_1^*) \leftarrow \mathcal{A}(\mathsf{pp})$,

$$\Pr\left[\left(u_1, \mathcal{E}(\mathsf{pp}, u_1, w_1^*)\right) \in \mathcal{R}_1\right] \approx \Pr\left[\langle \mathcal{P}^*(\mathsf{pp}, u_1, w_1^*), \mathcal{V}(\mathsf{pp}, u_1) \rangle \in \mathcal{R}_2\right].$$

Public reducibility. There is a deterministic polynomial-time algorithm f, such that for any PPT adversary $\mathcal{A}$ and a malicious expected polynomial-time prover $\mathcal{P}^*$, given

$$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), \quad (u_1, w_1^*) \leftarrow \mathcal{A}(\mathsf{pp}), \quad (u_2, w_2) \leftarrow \langle \mathcal{P}^*(\mathsf{pp}, u_1, w_1^*), \mathcal{V}(\mathsf{pp}, u_1) \rangle,$$

and the transcript tr, we have $f(\mathsf{pp}, u_1, \mathsf{tr}) = u_2$.

By the composition theory [KP23], the reduction of knowledge protocols can be composed.

Theorem 8. ***Sequential composition (Theorem 5 in [KP23]).*** *Let $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$ be three relations. Given two reduction of knowledge protocols, Π_1 from $\mathcal{R}_1$ to $\mathcal{R}_2$ and Π_2 from $\mathcal{R}_2$ to $\mathcal{R}_3$, the composed protocol $\Pi_1 \diamond \Pi_2$ is a reduction of knowledge from $\mathcal{R}_1$ to $\mathcal{R}_3$.*

Theorem 9. ***Parallel composition (Theorem 6 in [KP23]).*** *Let $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4$ be four relations. Given two reduction of knowledge protocols, Π_1 from $\mathcal{R}_1$ to $\mathcal{R}_2$ and Π_2 from $\mathcal{R}_3$ to $\mathcal{R}_4$, the composed protocol $\Pi_1 \times \Pi_2$ is a reduction of knowledge from $(\mathcal{R}_1 \times \mathcal{R}_3)$ to $(\mathcal{R}_2 \times \mathcal{R}_4)$.*