Σ-Check: Compressed Σ-protocol Theory from Sum-check

Shang Gao[1], Chen Qian[1], Tianyu Zheng[1]⋆, Yu Guo[2], Bin Xiao[1]

[1] Department of Computing, The Hong Kong Polytechnic University
[2] SECBIT Labs
{shanggao, b.xiao}@polyu.edu.hk
{chenqian.qian, tian-yu.zheng}@connect.polyu.hk
yu.guo@secbit.io

Abstract. The theory of compressed Σ-protocols [AC20, ACF21] provides a standardized framework for creating efficient Σ-protocols. This method involves two main phases: first, amortization, which combines multiple instances that satisfy a homomorphic relation into a single instance; and second, Bulletproofs compression [BBB+18], which minimizes communication overhead to a logarithmic scale during the verification of the combined instance. For high-degree polynomial (non-homomorphic) relations, a circuit-based linearization technique is used to transform each instance into a linear relation, resulting in a protocol with at least linear complexity.

In this paper, we provide a *direct* method to extend the compressed Σ-protocol theory to *polynomial* relations, named Σ-Check. One major contribution of Σ-Check is to eliminate the linear cost associated with linearization in amortization. To achieve this, we employ a sum-check during this phase to ensure a logarithmic communication cost. To the best of our knowledge, Σ-Check is the first work to achieve a *logarithmic amortization* for polynomial relations. Nevertheless, without linearization, the amortized relation may not be linear, which hinders us from using Bulletproofs compression. To overcome this problem, we employ another sum-check during the compression phase to effectively manage high-degree relations. Additionally, we propose several variants of our techniques and adapt them for arithmetic circuit relations. We also demonstrate the practicality of our compressed Σ-protocol theory through applications such as binary proofs, range proofs, and partial knowledge proofs. Our basic protocols are initially based on the Discrete Logarithm (DL) assumption, and we have further extended these to incorporate the Strong-RSA assumption and the Generalized Discrete Logarithm Representation (GDLR) assumption. Our work expands the scope of compressed Σ-protocol theory and provides a robust foundation for real-world cryptographic applications.

Keywords: Σ-protocol, compressed Σ-protocol theory, sum-check protocol, arithmetic circuit satisfiability.

⋆ Tianyu Zheng has the same contribution to the work as Chen Qian, who should be regarded as the co-second author.

1 Introduction

The Σ-protocol is a fundamental cryptographic primitive with extensive applications in zero-knowledge proofs. Many efforts have been invested in reducing the communication overhead of Σ-protocols. The most significant advancement in this area is the compressed Σ-protocol theory introduced by Attema et al. [2,3], which provides a general framework for constructing various Σ-protocols with a logarithmic communication cost.

The compressed Σ-protocol theory contains two major phases: (i) the *amortization* to fold multiple instances of a homomorphic relation into one, and (ii) the *compression* based on Bulletproofs [9] to decrease the communication cost of verifying a single instance to a logarithmic scale. In addition to the discrete logarithm assumption inherited from Bulletproofs, Attema et al. also describe the compressed Σ-protocol theory under Strong-RSA [2], knowledge-of-exponent [2], and module short integer solution [4] assumptions.

However, when dealing with high-degree polynomials, it is challenging to fold multiple instances into one due to cross-terms. For example, consider two instances $\vec{f_1}, \vec{f_2}$ such that $\vec{f_i}^2 = 0$ for all $i \in \{1, 2\}$. During the amortization phase, if we use a random challenge r to fold $\vec{f_1}$ and $\vec{f_2}$, we obtain $\vec{f} = \vec{f_1} + r\vec{f_2}$. To ensure the verification relation holds for $\vec{f}^2$, the prover must additionally send the cross-term $2\vec{f_1}\vec{f_2}$. Since the number of cross-terms increases linearly with the polynomial degree and number of instances, a straightforward amortization becomes inefficient for multiple high-degree relations. To handle high-degree polynomial (non-homomorphic) relations, Attema et al. [2] employs an arithmetic circuit to represent them and subsequently converts them into a linear relation through arithmetic secret-sharing techniques. After this "linearization" process, the resulting relation takes the form of $h(\vec{f}) = v$ for a public linear function h, where $\vec{f}$ is the witness and v is a public scalar. It is important to note that this indirect approach introduces (at least) a linear communication overhead when processing multiple instances, since the prover is required to send the public inputs for each instance. Besides, the linearization also requires the prover to engage in polynomial arithmetic on high-degree polynomials using Fast Fourier Transforms (FFTs), which prevents the prover from being implemented in logarithmic space [8]. Lastly, when the non-homomorphic function is expressed in non-native groups, such as elliptic curve groups, the cost of encoding these non-native operations into the circuit can be significant [26,30,36].

1.1 Our contributions

In this work, we adopt a more *direct* approach to avoid the linear overhead for processing multiple polynomial relations. Specifically, we face two major challenges. First, any form of linearization is insufficient due to the inherent costs of public inputs. This requires us to design an efficient amortization technique that bypasses linearization. Second, without linearization, the amortized relation may include high-degree polynomials, which prevents a direct application of Bulletproofs in the compression phase.

Table 1: Comparison between our approach with other methods. "Amortization" indicates the costs of handling polynomial relations, including the cost of linearization in AC20 [2] and ACF21 [3]. "size" states the proof size. In particular, we consider handling k instances with m-size witnesses. The degree-related terms are omitted for simplicity.

	Relation	**Amortization**			**Compression**		
		Prover	Verifier	Size	Prover	Verifier	Size
AC20 [2]	linear	$O(km)$	$O(k)$	$O(k)$	$O(m)$	$O(m)$	$O(\log m)$
ACF21 [3]	homomorphic	$O(km)$	$O(k)$	$O(k)$	$O(m)$	$O(m)$	$O(\log m)$
Σ-Check	polynomial	$O(k)$	$O(k)$	$O(\log k)$	$O(m)$	$O(m)$	$O(\log m)$

To address these challenges, we resort to the sum-check protocol [31], a powerful tool that allows us to directly extend the compressed Σ-protocol theory to polynomial relations. Sum-check protocols have previously been utilized in certain contexts to manage high-degree polynomials effectively. This motivates us to design an efficient amortization directly (without linearization) using a sum-check protocol. To the best of our knowledge, Σ-Check is the first work to achieve *logarithmic communication overhead for the amortization* of polynomial relations.

Moreover, we observe that the two phases of amortization and compression essentially address the same issue: reducing the size of the witness(es) while maintaining a verifiable relation. This inspires us to substitute Bulletproofs with a sum-check protocol in compression, which addresses the challenge of high-degree polynomials in an amortized relation. We conclude our result as the *compressed Σ-protocol theory from sum-checks* and compare its performance in Table 1. All protocols in our design can be further converted into non-interactive forms using the Fiat-Shamir transform [22].

Furthermore, we introduce several variants of our technique and describe an important application of *arithmetic circuit satisfiability*, represented by a committed Customizable Constraint System (CCS) [34] relation. This enables the application of our technique to non-polynomial relations (e.g., SHA256) and the design of (Zero-Knowledge) Succinct Non-interactive Arguments of Knowledge (zkSNARK) systems. Unlike existing sum-check based (zk)SNARKs [13,33], Σ-Check *does not require an additional polynomial commitment scheme (PCS).* Traditional methods rely on such commitments to verify that the evaluation claim is derived from the initial witness. However, in our design, the Pedersen vector commitment ensures a binding relation with the witness. When proving *committed* relations, the final opening binds to the original witness through the commitment claim, and thus, avoids additional PCS. As a result, our protocol naturally has the commit-and-prove capability [12], which enables it to be modularly used in proof compositions for non-linear constraints or cryptographic constraints, e.g., Discrete Logarithms (DL) relations.

Finally, we showcase further applications of Σ-Check, including binary proofs, range proofs, and partial knowledge proofs. In this paper, we primarily introduce our scheme based on Pedersen vector commitment under the DL assumption. Besides, our approach can be adapted to the integer PCS [10, 19] under the Strong-RSA assumption, as well as other Generalized Inner Product Argument (GIPA) based commitment schemes, such as the LMR commitment based on the Generalized Discrete Logarithm Representation (GDLR) assumption [28]. Experimental results indicate that our solution outperforms existing work [3] without requiring any expensive pairing operations.

1.2 Related work

Compressed Σ-protocol theory. To enhance the efficiency of Σ-protocols, Attema and Cramer [2] reconcile Bulletproofs with traditional Σ-protocol theory. This integration, known as the compressed Σ-protocol theory, inherits the flexibility and versatility of Σ-protocols while ensuring a logarithmic communication complexity. It involves two major phases in proving multiple instances of a linear relation. First, an "amortization" technique is employed to randomly combine all instances into a folded one, while preserving the linear relation within the folded instance. Second, a "compression" technique based on Bulletproofs is applied, which achieves a logarithmic communication overhead for verifing the folded instance. Attema et al. further extend the compressed Σ-protocol theory to homomorphic relations and demonstrate a practical application of partial knowledge proof in [3]. However, for non-homomorphic relations, which are represented by arithmetic circuits, a linearization technique is proposed to transform them into a linear form through arithmetic secret-sharing [2], which results in a linear communication cost.

In an effort to reduce the verifier's cost within the compressed Σ-protocol framework, Dutta et al. integrate an inner-product argument [11, 20] to achieve logarithmic verification [21]. This approach requires a trusted setup, which introduces new assumptions into the Σ-protocols and compromises the "plug-and-play" property. Although all protocols proposed in this paper can adopt this technique by delegating linear operations in verification to the prover, we do not incorporate this method in our designs.

Sum-check argument. The sum-check protocol [31] offers an efficient method for proving the summation of a function. It has been widely utilized in various applications to transform claims involving high-degree polynomials into linear ones without FFTs [6, 27]. The sum-check argument, proposed by Bootle et al. [8], establishes a connection between the sum-check protocol [31] and Bulletproofs [9]. By extending the concept of multilinear extension to modules, they demonstrate the versatility of sum-check arguments in a range of applications, including generic scalar-product protocols, zkSNARKs, and PCS.

zkSNARKs from Sumcheck. In recent years, the cryptographic community has witnessed a significant increase in the development of zkSNARKs based on both multivariate sum-checks like [13, 33, 34] and univariate sum-checks [15, 29]. Since our design developed on inner-product arguments, we mainly compare with multivariate-based zkSNARKs in this paper. A key innovation in these approaches is the adoption of the classical multivariate sum-check protocol to construct a Polynomial Interactive Oracle Proof (PIOP) system, which can avoid some costly operations such as FFT. Additionally, these approaches require evaluation proof from the PCS to verify that the responses of the polynomial queries are indeed derived from the original witness.

2 Preliminary

2.1 Notations

Let $\mathbb{G}$ denote the elliptic curve group. We regard $\mathbb{G}$ as an additive group with the infinity point denoted by $\mathcal{O}$. The scalar field of $\mathbb{G}$ is referred to as $\mathbb{F}$. We use $\mathbb{F}[X]$ to denote the set of polynomials over the scalar field $\mathbb{F}$, i.e., whose coefficients are in $\mathbb{F}$. For $n \in \mathbb{N}^+$, we denote $[n]$ as the set $\{1, \cdots, n\}$. Vectors are denoted as $\vec{f} := (f_1, \cdots, f_n) \in \mathbb{F}^n$. We use $(\vec{f}, \vec{g})$ to denote appending vector $\vec{g}$ to $\vec{f}$. Furthermore, the Hadamard product is denoted as $\vec{f} \circ \vec{g} := (f_1 \cdot g_1, \cdots, f_k \cdot g_k)$ and $\bigcirc_{i=1}^{k} \vec{f_i} := \vec{f_1} \circ \cdots \circ \vec{f_k}$. $a \leftarrow\!\!\$\ \mathbb{F}$ denotes random sampling a from the field $\mathbb{F}$.

For a nondeterministic polynomial time ($\mathcal{NP}$) relation $\mathcal{R}$, we define it over *public statement* x and *witness* w, i.e., $(x; w) \in \mathcal{R}$.

Modules and module homomorphisms. Let $\mathfrak{R}$ be any arbitrary ring. A module $\mathbb{M}$ over $\mathfrak{R}$ extends the notion of vector space over a field, where the scalars are elements in $\mathfrak{R}$. Accordingly, $\mathbb{M}$ has an identity element 1 and for all $a, b \in \mathfrak{R}$ and $X, Y \in \mathbb{M}$, (i) $a \cdot (X + Y) = a \cdot X + a \cdot Y$, (ii) $(a + b) \cdot X = a \cdot X + b \cdot X$, (iii) $(a \cdot b) \cdot X = a \cdot (b \cdot X)$, and (iv) $1 \cdot X = X$. The elliptic curve group $\mathbb{G}$ is an example of a module over its scalar field $\mathbb{F}$.

A $\mathfrak{R}$-module homomorphism $h : \mathbb{M} \to \mathbb{M}'$ between modules $\mathbb{M}$ and $\mathbb{M}'$ is a function that preserves the module structures. Precisely, for every $X, Y \in \mathbb{M}$ and $a \in \mathfrak{R}$, we have (i) $h(X + Y) = h(X) + h(Y)$ and (ii) $h(a \cdot X) = a \cdot h(X)$.

2.2 Sum-checks and multilinear extension

The famous sum-check protocol [31] and the concept multilinear extension [17] can be naturally extended to polynomials over modules [8]. In this paper, we focus on a special case of [8] where the module $\mathbb{M}$ is over a field $\mathbb{F}$. This ensures that challenges can be directly sampled from $\mathbb{F}$ since all $\mathbb{F}$ elements are invertible.

Lemma 1 (Sum-check over modules [8]). *Given a module $\mathbb{M}$ over $\mathbb{F}$, let $f \in \mathbb{M}^{\leq d}[X_1, \cdots, X_\mu]$ be a μ-variate nonzero polynomial with per-variable degree at most d. The following protocol for proving $s = \sum_{\vec{b} \in \{0,1\}^\mu} f(\vec{b})$ has soundness error $\frac{\mu d}{|\mathbb{F}|}$.*

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

- *Upon receiving the challenges $r_1, \cdots, r_{i-1}$ from the previous rounds, $\mathcal{P}$ sends the univariate polynomial*

$$f_i(X) := \sum_{\vec{b} \in \{0,1\}^{\mu-i}} f(r_1, \cdots, r_{i-1}, X, \vec{b}) \quad \in \mathbb{M}^{\leq d}[X].$$

- *$\mathcal{V}$ checks $f_i(0)+f_i(1) = f_{i-1}(r_{i-1})$ and sends a random challenge $r_i \leftarrow\!\!\$\ \mathbb{F}$ (define $f_0(r_0) := s$).*

2. $\mathcal{V}$ checks $f_\mu(r_\mu) = f(r_1, \cdots, r_\mu)$.

Specifically, f_i is sent in its evaluation form, i.e., $(d+1)$ distinct evaluations of f_i. The prover can also send one less evaluation, for instance, omitting $f_i(0)$. In such a scenario, the verifier calculates $f_i(0)$ with $f_i(0) := f_{i-1}(r_{i-1}) - f_i(1)$ and reconstructs $f_i(X)$ for the subsequent round.

In many applications, the final verification step is omitted, resulting in a protocol producing a claim that $f_\mu(r_\mu) = f(r_1, \cdots, r_\mu)$. The verifier will either be able to efficiently check $f(r_1, \cdots, r_\mu)$ independently or ask the prover to prove this claim is correct via a PCS. In our applications, the verifier can compute the Pedersen vector commitment (defined in Section 2.3) of $f(r_1, \cdots, r_\mu)$ on its own. Therefore, there is no need to apply an extra PCS like other SNARKs built on the PIOP paradigm.

Definition 1 (Multilinear extension over modules [8]). *For a module $\mathbb{M}$ over $\mathbb{F}$, given a function $f : \{0,1\}^\mu \to \mathbb{M}$, the multilinear extension (MLE) of f is defined as*

$$\tilde{f}(\vec{X}) := \sum_{\vec{b} \in \{0,1\}^\mu} f(\vec{b}) \cdot \mathsf{eq}(\vec{b}, \vec{X}) \quad \in \mathbb{M}^{\leq 1}[X_1, \cdots, X_\mu],$$

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

2.3 Pedersen vector commitment

Given a group $\mathbb{G}$ with scalar field $\mathbb{F}$, the Pedersen vector commitment scheme [32] is defined by the following KeyGen, Commit, and Open algorithms:

- $\mathsf{KeyGen}(1^\lambda)$: On input security parameter λ, sample $(G_1, \cdots, G_m) \leftarrow\!\!\$\ \mathbb{G}^m$ and $G \leftarrow\!\!\$\ \mathbb{G}$. Output commitment key $\mathsf{ck} := (G_1, \cdots, G_m, G)$.
- $\mathsf{Commit}(\mathsf{ck}; \vec{f}, r)$: On input commitment key ck, a message $\vec{f} \in \mathbb{F}^m$, and a randomness $r \in \mathbb{F}$, output the commitment $F := \sum_{i=1}^m f_i G_i + rG$.
- $\mathsf{Open}(\mathsf{ck}; F, \vec{f}, r)$: On input opening $(\vec{f}, r) \in \mathbb{F}^{m+1}$ and a commitment $F \in \mathbb{G}$, output 1 if $F = \sum_{i=1}^m f_i G_i + rG$, otherwise 0.

For brevity, we regard the randomness r as an extra dimension of the message and leave the r underlying the commitment implicit throughout this paper.

The Pedersen vector commitment scheme is *computationally binding* and *perfectly hiding* under the discrete logarithm assumption, which are formally defined in Appendix A.1.

2.4 Interactive proofs and Σ-protocols

For an $\mathcal{NP}$ relation $\mathcal{R} := \{(x; w)\}$, an interactive proof for $\mathcal{R}$ allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that a statement x admits a witness w. If $\mathcal{V}$ generates and sends all its messages uniformly at random and independently of the prover's messages, then the protocol is called a *public coin* protocol. Public coin protocols can be transformed into non-interactive forms through the Fiat-Shamir heuristic [22], which applies to all protocols presented in this paper. Concretely, we consider public-coin interactive proofs (arguments) consisting of three PPT algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ and define their security properties. Let $\mathcal{G}$ be a setup algorithm that generates public parameters on input 1^λ. The prover $\mathcal{P}$ and verifier $\mathcal{V}$ runs an interactive protocol on inputs s and t and produce a transcript denoted as $tr \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$. We further denote $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b$ depending on whether the verifier rejects with $b = 0$ or accepts with $b = 1$. Typically, the triple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ satisfies *perfect completeness* and *soundness* properties as

- $(\mathcal{G}, \mathcal{P}, \mathcal{V}) = 1$ for all $(x; w) \in \mathcal{R}$ (perfect completeness);
- $(\mathcal{G}, \mathcal{P}^*, \mathcal{V}) = 1$ with negligible probability for all $(x; w) \notin \mathcal{R}$ and every malicious $\mathcal{P}^*$ (soundness).

The standard Σ-protocol [18] is a 3-move interactive proof for proving the knowledge of a witness w with respect to a statement x. The key difference is that Σ-protocol provides *computational knowledge soundness* and *zero knowledge*, yielding Σ-protocol as a more practical Argument of Knowledge with versatile applications. Recently, Bootle et al. further generalized standard Σ-protocol from 3-move to multiple moves for better efficiency [7]. Therefore, the rest of this work refers to Σ-protocol as a generalized public-coin multiple-move protocol (e.g., $(2\mu + 1)$) with security definitions adopted from [1] in Appendix A.2. Typically, it is more common to require the Σ-protocol satisfying $(k_1, ..., k_\mu)$-*special soundness* for simplify security proofs [2, 7]. According to the conclusion given in [2], we can imply knowledge soundness from special soundness:

Lemma 2 ($(k_1, ..., k_\mu)$-special soundness implies knowledge soundness [2]). *Let $\mu, k_1, ..., k_\mu \in \mathbb{N}$, $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ be a $(k_1, ..., k_\mu)$-special sound $(2\mu + 1)$-move interactive protocol for relation $\mathcal{R}$, where $\mathcal{V}$ samples each challenge uniformly at random from $\mathbb{F}$. Then $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is knowledge sound with knowledge error*

$$\kappa \leq \frac{\sum_{i=1}^{\mu} k_i - 1}{|\mathbb{F}|}.$$

2.5 Compressed Σ-protocol theory

Compressed Σ-protocol theory provides a general framework to build (multiple-move) Σ-protocols for homomorphic relations with logarithmic communication cost [2–4]. Generally, the framework consists of two phases. Given k homomorphic instances with m-size witnesses, the prover first conducts an amortization to fold these k instances into one with a random linear combination. Then, by further adopting Bulletproofs compression, it only takes $O(\log m)$ communication cost to verify the folded instance (more details in Section 3.1). To deal with a non-linear (non-homomorphic) relationship, Attema et al. [2] transforms the relationship represented by an arithmetic circuit into a linear form using a *linearization* technique.

Besides the Discrete Logarithm assumption (designed with Pedersen vector commitments), the compressed Σ-protocol theory can be instantiated from different hardness assumptions such as Strong-RSA [2], and Module Short Integer Solution [4].

2.6 Reduction of knowledge

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two distinct relations. A reduction of knowledge protocol Π [25] enables a prover to demonstrate to a verifier on input x_1 to derive an output x_2, such that for anyone who has w_2 where $(x_2; w_2) \in \mathcal{R}_2$, it is possible to extract w_1 such that $(x_1; w_1) \in \mathcal{R}_1$. Let $\mathcal{G}$ be a setup algorithm that generates public parameters pp for Π and $(\mathsf{tr}, x_2; w_2) \leftarrow \langle \mathcal{P}(\mathsf{pp}, x_1; w_1), \mathcal{V}(\mathsf{pp}, x_1)\rangle$ be the execution of Π (tr is the transcript). The protocol Π should satisfy three properties: *perfect completeness*, *knowledge soundness* defined in Appendix A.3, and *public reducibility* defined as follows:

Definition 2 (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 $\mathcal{P}^*$*

$$\Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), (x_1; w_1^*) \leftarrow \mathcal{A}(\mathsf{pp}), \\ (\mathsf{tr}, x_2; w_2) \leftarrow \langle \mathcal{P}^*(\mathsf{pp}, x_1; w_1^*), \mathcal{V}(\mathsf{pp}, x_1)\rangle : f(\mathsf{pp}, x_1, \mathsf{tr}) = x_2 \end{array}\right] = 1.$$

When considering two (or more) reduction of knowledge protocols Π_1 and Π_2, they can be combined in two distinct manners as described in [25]: sequential composition $\Pi_1 \diamond \Pi_2$ and parallel composition $\Pi_1 \times \Pi_2$.

Theorem 1 (Sequential composition [25]). *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 [25]). *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 Compressed $\varSigma$-Protocol Theory from Sum-check

3.1 Basic $\varSigma$-protocol

$\varSigma$-protocol for homomorphic relations. Our starting point is the homomorphic relation $\mathcal{R}_{\mathsf{Hom}}$ in [3], which is defined as follows.

$$\mathcal{R}_{\mathsf{Hom}} := \left\{ \begin{matrix} (F \in \mathbb{G}, \vec{v} \in \mathbb{F}^m; \vec{f} \in \mathbb{F}^m) : \\ \sum_{i=1}^m f_i G_i = F, \quad h(f_i) = v_i \quad \forall i \in [m] \end{matrix} \right\}, \tag{1}$$

where F is the commitment of the witness $\vec{f}$ using the Pedersen vector commitment scheme and $h : \mathbb{F} \to \mathbb{F}$ is a *fixed* homomorphic function. In addition, $\vec{G}$ and h serve as public parameters in our protocols. For the sake of simplicity, we omit the public parameters in all relations discussed in this paper. The randomness is ignored for brevity since it can be regarded as an additional dimension of the message, as with [2, 3]. We assume that the evaluations $h(f_i)$ do not leak any information about the witness. This can be further ensured by revealing $v_i G \in \mathbb{G}$ instead of v_i, or adopting a hiding evaluation claim [37] (also discussed in Section 3.5). Meanwhile, all techniques are applicable for general *module homomorphisms* in Section 2.1, such as $h : \mathbb{F} \to \mathbb{G}$.

In a $\varSigma$-protocol, to prove k instances $(F_j, \vec{v}_j; \vec{f}_j) \in \mathcal{R}_{\mathsf{Hom}}$ for all $j \in [k]$, the prover initiates by sampling a masking vector $\vec{f}_0 \leftarrow\$ \mathbb{F}^m$ and sends $F_0 := \sum_{i=1}^m f_{0,i} G_i, \vec{v}_0 := \big(h(f_{0,1}), \cdots, h(f_{0,m})\big)$. Upon receiving a challenge r from the verifier, the prover responds by revealing the opening $\vec{f} := \sum_{j=0}^k r^j \vec{f}_j$. The verifier checks $\sum_{i=1}^m f_i G_i = \sum_{j=0}^k r^j F_i$ and $h(f_i) = \sum_{j=0}^k r^j v_{j,i}$ for all $i \in [m]$.

The above masking vector $\vec{f}_0$ and the corresponding $F_0, \vec{v}_0$ can be regarded as a masking instance $(F_0, \vec{v}_0; \vec{f}_0) \in \mathcal{R}_{\mathsf{Hom}}$. Therefore, the $\varSigma$-protocol is essentially the processes of folding $(k+1)$ instances with a random combination and checking whether the folded instance is in $\mathcal{R}_{\mathsf{Hom}}$. With this abstraction in mind, we reformulate the $\varSigma$-protocol $\varPi_{\mathsf{Hom}}$ for $\mathcal{R}_{\mathsf{Hom}}$ in Protocol 1.

Protocol 1 $\varPi_{\mathsf{Hom}}$: Prove $(F_j, \vec{v}_j; \vec{f}_j) \in \mathcal{R}_{\mathsf{Hom}}, \forall j \in [k]$.

```
$\mathcal{P}\big(\vec{G}, (F_j, \vec{v}_j, \vec{f}_j)_{j=1}^k\big), \mathcal{V}\big(\vec{G}, (F_j, \vec{v}_j)_{j=1}^k\big)$
1: $\mathcal{P}$: Sample $\vec{f}_0 \leftarrow\$ \mathbb{F}^m$. Compute $F_0, \vec{v}_0$ such that $(F_0, \vec{v}_0; \vec{f}_0) \in \mathcal{R}_{\mathsf{Hom}}$.
2: $\mathcal{P} \to \mathcal{V}$: $F_0, \vec{v}_0$.
3: $\mathcal{V}$: Set $F(X) := \sum_{j=0}^k X^j \cdot F_j$ and $\vec{v}(X) := \sum_{j=0}^k X^j \cdot \vec{v}_j$.
4: $\mathcal{P}$: Set $\vec{f}(X) := \sum_{j=0}^k X^j \cdot \vec{f}_j$.
5: $\mathcal{V} \to \mathcal{P}$: Sample $r \leftarrow\$ \mathbb{F}$.
6: $\mathcal{P} \to \mathcal{V}$: $\vec{f}(r)$.
7: $\mathcal{V}$: Check $\big(F(r), \vec{v}(r); \vec{f}(r)\big) \in \mathcal{R}_{\mathsf{Hom}}$.
```

Remarks. When considering $\vec{f}(X)$ as a polynomial, steps 3 to 5 reduce the original relations to an evaluation claim of $\vec{f}(r)$. To ensure $\vec{f}(r)$ at step 6 is com-

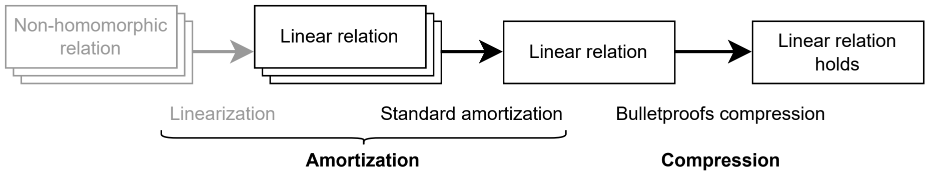


Fig. 1: The framework of compressed Σ-protocol theory [2]. The term "linearization" is grayed out since it is only required when handling non-homomorphic relations.

puted from the original witnesses, some approaches [13,33] employ an additional PCS. However, in Protocol 1, this extra step is omitted since the folded Pedersen vector commitment $F(r)$ essentially binds $\vec{f}(r)$ to the original witnesses. This contrasts with the approaches in [13, 33], which focus on the polynomial claim (h part) when designing PIOPs, whereas the Σ-protocols address *committed* relations, which contains both commitment and polynomial claims. Based on this observation, our design also avoids using PCS when applied to (zk)SNARKs.

Compressed Σ-protocol theory. The compressed Σ-protocol theory [2, 3] treats steps 3 and 4 in Protocol 1 — which fold $(k + 1)$ instances — as an *amortization* process. Additionally, the last two steps are substituted with *Bulletproofs compression* [9], effectively folding $\vec{f}(r)$ in to an $\mathbb{F}$ element with $O(\log m)$ communication cost. We summarize the framework in Figure 1 and conclude the compressed Π_{Hom} as three major phases: (i) the prover computes and sends the public statement of a masking instance to the verifier, (ii) the prover interacts with the verifier to fold multiple instances into one (i.e., the *standard amortization* in [2,3]), and (iii) the verifier efficiently checks the folded instance with *Bulletproofs compression* [9]. Given that the folded witness leaks no information about the original ones, we do not need to ensure the zero-knowledge property in the last phase.

When dealing with homomorphic relations, phase (ii) does not introduce any communication overhead (only the prover's messages are counted). Accordingly, most of the existing work [2, 3] focuses on optimizing phase (iii) to improve efficiency. However, when h is a high-degree polynomial, folding multiple instances directly incurs cross terms, which significantly increases the overhead of the protocol as described in Section 1. Even though it is possible to convert the high-degree relation into an arithmetic circuit relation with *linearization* [2], the transformation itself incurs a non-trivial cost. More importantly, the cost of the linearization scales linearly with the number of instances involved since the prover is required to send at least a public input (e.g., the commitment of the composed vector in [2]) for each linear relation. Additionally, even after efficiently folding multiple polynomial instances into a single instance without linearization, we cannot directly apply the Bulletproofs compression during phase (iii). This is because h is a polynomial, whereas Bulletproofs is primarily designed for inner-product relations (i.e., quadratic relations as noted in [38]).

3.2 Technique overview: encoding k polynomial relations

We observe that the processes of amortization in phase (ii) and compression in phase (iii) are essentially performing the same operation: folding a long witness into a short one while maintaining a verifiable relation (i.e., reducing from $\mathbb{F}^{k\times m}$ to $\mathbb{F}^m$ in amortization, and $\mathbb{F}^m$ to $\mathbb{F}$ for compression). To illustrate, consider a simple example with a per-element commitment scheme, i.e., $F_{j,i} = f_{j,i}G$ for all $j \in [k]$ and $i \in [m]$. In this case, both phases can be executed without incurring any prover's message for homomorphic relations. Consequently, the compression technique in Bulletproofs can also be applied for amortizing quadratic relations. Nonetheless, this approach is not general enough as it still requires transformations for high-degree polynomial relations.

In this paper, we introduce a new technique for both amortization and compression. To address the challenge posed by high-degree polynomials in both scenarios, we employ a pair of sum-check protocols. First, we defined a more generalized polynomial relation $\mathcal{R}_{\mathsf{Poly}}$ as follows.

$$\mathcal{R}_{\mathsf{Poly}} := \left\{ \begin{array}{c} (F \in \mathbb{G}, \vec{v} \in \mathbb{F}^m; \vec{f} \in \mathbb{F}^m) : \\ \sum_{i=0}^{m} f_i G_i = F, \quad h(f_i) = v_i \;\; \forall i \in [m] \end{array} \right\}, \tag{2}$$

where $h : \mathbb{F} \to \mathbb{F}$ is a high-degree polynomial (also generalized to $h : \mathbb{F} \to \mathbb{G}$).

When dealing with k instances $(F_j, \vec{v}_j, h_j; \vec{f}_j) \in \mathcal{R}_{\mathsf{Poly}}$ for all $j \in [k]$, we employ the sum-check protocol [31] to enhance efficiency. This requires us to encode all instances into a polynomial form. We first write the commitment claims $\sum_{i=0}^{m} f_{j,i}G_i = F_j$ for all $j \in [k]$ into a matrix form:

$$\begin{bmatrix} f_{1,1},\, f_{1,2},\, \cdots,\, f_{1,m} \\ \vdots \quad\; \vdots \quad\; \ddots \quad \cdots \\ f_{k,1},\, f_{k,2},\, \cdots,\, f_{k,m} \end{bmatrix} \cdot \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_m \end{bmatrix} = \begin{bmatrix} F_1 \\ \vdots \\ F_k \end{bmatrix}. \tag{3}$$

Consider the left matrix of $f_{j,i}$ as a function $f : \{0,1\}^{\log k} \times \{0,1\}^{\log m} \to \mathbb{F}$. This allows us to reference any element $f_{j,i}$ using two $\log k$-bit and $\log m$-bit identifiers. Similarly, we can regard $(G_1, \cdots, G_m)$ as a function $G : \{0,1\}^{\log m} \to \mathbb{G}$ and $(F_1, \cdots, F_k)$ as $F : \{0,1\}^{\log k} \to \mathbb{G}$. To prove the equation in Equation (3), it suffices to show that the following equation holds:

$$\sum_{\vec{y} \in \{0,1\}^{\log m}} f(\vec{x}, \vec{y}) \cdot G(\vec{y}) = F(\vec{x}), \quad \forall \vec{x} \in \{0,1\}^{\log k}. \tag{4}$$

Note that f, G, F are functions, not polynomials. Therefore, to enable the sum-check protocol, we conduct MLEs on f, G, F, which derive the polynomials $\tilde{f} : \mathbb{F}^{\log k} \times \mathbb{F}^{\log m} \to \mathbb{F}$, $\tilde{G} : \mathbb{F}^{\log m} \to \mathbb{G}$, and $\tilde{F} : \mathbb{F}^{\log k} \to \mathbb{G}$.

$$\sum_{\vec{y} \in \{0,1\}^{\log m}} \tilde{f}(\vec{x}, \vec{y}) \cdot \tilde{G}(\vec{y}) = \tilde{F}(\vec{x}), \quad \forall \vec{x} \in \{0,1\}^{\log k}. \tag{5}$$

Similarly, the polynomial claim can be encoded using the same approach. Specifically, $v_{j,i}$'s are encoded as a polynomial $\tilde{v} : \mathbb{F}^{\log k} \times \mathbb{F}^{\log m} \to \mathbb{F}$, such that

$$h\big(\tilde{f}(\vec{x}, \vec{y})\big) = \tilde{v}(\vec{x}, \vec{y}), \quad \forall \vec{x} \in \{0,1\}^{\log k}, \vec{y} \in \{0,1\}^{\log m}. \tag{6}$$

Therefore, to prove $(F_j, \vec{v}_j, h_j; \vec{f}_j) \in \mathcal{R}_{\mathsf{Poly}}$ for all $j \in [k]$, we can employ two sum-checks — one on $\vec{x}$ and another on $\vec{y}$ — to show Equation (5) and Equation (6) hold. Different from a traditional sum-check protocol, which requires the verifier to evaluate $\tilde{f}(\vec{r}_x, \vec{r}_y)$ ($\vec{r}_x, \vec{r}_y$ are challenges in the sum-checks), the final output claim of $\tilde{f}(\vec{r}_x, \vec{r}_y)$ effectively serves as opening in a Σ-protocol, akin to $\vec{f}(r)$ in step 6 of Protocol 1 (more explanations in Section 3.4 and Section 3.5).

3.3 Amortization with sum-check

Building sum-check polynomials. Recall Equation (5) and Equation (6), which are in the form of $g(\vec{x}) = 0$ for all $\vec{x} \in \{0,1\}^{\log k}$. The sum-check protocol enables us to prove $\sum_{\vec{x} \in \{0,1\}^{\log k}} g(\vec{x}) = 0$. However, this does not imply $g(\vec{x}) = 0$ for all $\vec{x} \in \{0,1\}^{\log k}$ since $g(\vec{x})$'s may cancel out each other. To address this issue, we adopt a prior idea [12, 33] to combine each term with random challenges $\vec{\alpha}$ and prove $\sum_{\vec{x} \in \{0,1\}^{\log k}} \mathsf{eq}(\vec{\alpha}, \vec{x}) \cdot g(\vec{x}) = 0$. Accordingly, we can safely use the sum-check protocol to prove the claim.

Upon receiving challenges $\vec{\alpha} \leftarrow\!\!\$\ \mathbb{F}^{\log k}$ and $\vec{\beta} \leftarrow\!\!\$\ \mathbb{F}^{\log m}$ from the verifier, the prover and verifier define two polynomials, $g_1 : \mathbb{F}^{\log k} \to \mathbb{G}$, $g_2 : \mathbb{F}^{\log k} \to \mathbb{F}$ as

$$\begin{aligned} g_1(\vec{X}) &:= \mathsf{eq}(\vec{\alpha}, \vec{X}) \cdot \Big(\tilde{F}(\vec{X}) - \sum_{\vec{y} \in \{0,1\}^{\log m}} \tilde{f}(\vec{X}, \vec{y}) \cdot \tilde{G}(\vec{y})\Big), \\ g_2(\vec{X}) &:= \mathsf{eq}(\vec{\alpha}, \vec{X}) \cdot \sum_{\vec{y} \in \{0,1\}^{\log m}} \Big(\mathsf{eq}(\vec{\beta}, \vec{y}) \cdot \Big(h\big(\tilde{f}(\vec{X}, \vec{y})\big) - \tilde{v}(\vec{X}, \vec{y})\Big)\Big). \end{aligned} \tag{7}$$

Accordingly, we construct two sum-check claims: $\sum_{\vec{x} \in \{0,1\}^{\log k}} g_1(\vec{x}) = \mathcal{O}$ and $\sum_{\vec{x} \in \{0,1\}^{\log k}} g_2(\vec{x}) = 0$.

Folding multiple instances with sum-check. To prove the two sum-check claims, we can concurrently execute two sum-check protocols on $\vec{x}$. This results in $2 \log k$ and $(d+1) \log k$ communication overhead in $\mathbb{G}$ and $\mathbb{F}$ respectively, where $d := \deg(h)$ is the degree of h. Alternatively, one can compose the two claims into one and run a single sum-check. However, this requires lifting Equation (6) to $\mathbb{G}$, leading to a communication cost of $(d+1) \log k$ in $\mathbb{G}$, with all computations being performed within $\mathbb{G}$.

To achieve better efficiency, we observe that Equation (5) is linear with respect to $\vec{x}$. This allows the verifier to derive the sum-check output directly without actually running the protocol (similar to the amortization on linear/homomorphic relations [2,3]). For ease of understanding, we first present our protocol with two sum-checks, and then describe the optimized version.

The amortization protocol $\mathit{\Pi}_{\mathsf{Amor}}$ reduces k instances of $\mathcal{R}_{\mathsf{Poly}}$ to one amortized polynomial relation $\mathcal{R}_{\mathsf{AmorPoly}}$, which is defined as follows:

$$\mathcal{R}_{\mathsf{AmorPoly}} := \left\{ \begin{array}{c} (F \in \mathbb{G}, s \in \mathbb{F}, \vec{v} \in \mathbb{F}^m, \vec{\beta} \in \mathbb{F}^{\log m}; \vec{f} \in \mathbb{F}^m) : \\ \sum_{i=1}^m f_i G_i = F, \sum_{i=1}^m \mathsf{eq}\big(\vec{\beta}, \mathsf{Bits}(i)\big)\big(h(f_i) - v_i\big) = s \end{array} \right\}, \tag{8}$$

where $\mathsf{Bits}(i)$ denotes the binary representation of i. We formally describe $\mathit{\Pi}_{\mathsf{Amor}}$ in Protocol 2. The blue parts are omitted in the optimized version.

Protocol 2 $\mathit{\Pi}_{\mathsf{Amor}}$: Reduce $(\mathcal{R}_{\mathsf{Poly}})^k$ to $\mathcal{R}_{\mathsf{AmorPoly}}$.

```
P(G⃗, (F_j, v⃗_j, f⃗_j)_{j=1}^k), V(G⃗, (F_j, v⃗_j)_{j=1}^k)
1: V → P: α⃗ ←$ F^{log k}, β⃗ ←$ F^{log m}.
2: P and V: Compute f̃(X⃗, Y⃗) from f⃗_j's and set g_1(X⃗), g_2(X⃗) as with Equation
   (7). Then engage in two sum-checks in parallel with the same challenges for
   two claims
```

$$\sum_{\vec{x} \in \{0,1\}^{\log k}} g_1(\vec{x}) = \mathcal{O}, \qquad \sum_{\vec{x} \in \{0,1\}^{\log k}} g_2(\vec{x}) = 0. \tag{9}$$

```
   The protocol reduces to two evaluation claims g_1(r⃗_x) = F_g and g_2(r⃗_x) = s_g,
   where r⃗_x ←$ F^{log k} are the challenges of the sum-check.
3: P → V: F, s which are defined as
```

$$\begin{aligned} F &:= \sum_{\vec{y} \in \{0,1\}^{\log m}} \tilde{f}(\vec{r}_x, \vec{y}) \cdot \tilde{G}(\vec{y}), \\ s &:= \sum_{\vec{y} \in \{0,1\}^{\log m}} \Big(\mathsf{eq}(\vec{\beta}, \vec{y}) \cdot \Big(h\big(\tilde{f}(\vec{r}_x, \vec{y})\big) - \tilde{v}(\vec{r}_x, \vec{y}) \Big) \Big). \end{aligned} \tag{10}$$

```
4: V: Compute e_x := eq(α⃗, r⃗_x) and F_x := F̃(r⃗_x). Check the following equations
   // set F := F_x in the optimized version
```

$$F_g = e_x \cdot (F_x - F), \qquad s_g = e_x \cdot s. \tag{11}$$

```
5: P: Output f⃗ := (f̃(r⃗_x, y⃗))_{y⃗ ∈ {0,1}^{log m}}.
6: V: Output F, s, v⃗ := (ṽ(r⃗_x, y⃗))_{y⃗ ∈ {0,1}^{log m}}, β⃗.
```

Efficiency. Let $d := \deg(h)$ be the degree of the polynomial h. The sum-checks require $2 \log k$ and $(d+1)\log k$ prover's messages in $\mathbb{G}$ and $\mathbb{F}$, respectively. In addition, at step 3, the prover needs to send $F \in \mathbb{G}$ and $s \in \mathbb{F}$. Regarding the time complexity, the sum-checks incur $O(kd \log^2 d)$ computational cost for the prover and $O(d \log k)$ for the verifier. At step 4, computing F_x takes $O(k)$ $\mathbb{G}$-operations and e_x tasks $O(\log k)$ $\mathbb{F}$-operations.

Theorem 3. *$\mathit{\Pi}_{\mathsf{Amor}}$ is a reduction of knowledge from $(\mathcal{R}_{\mathsf{Poly}})^k$ to $\mathcal{R}_{\mathsf{AmorPoly}}$.*

Proof Sketch. *The completeness is trivial to prove. For the knowledge soundness, we first build an extractor for special-soundness that can compute vectors $\vec{f}_j, j \in [k]$ from k transcripts with different challenges by querying to $\mathcal{P}^*$. Further guaranteed by the soundness of sum-check protocols, the extractor has an overwhelming probability to ensure the extracted $\vec{f}_j$'s satisfying Equation (9). By uniqueness of MLE and Schwartz-Zippel lemma, either $\vec{f}_j$ is a valid witness for $\mathcal{R}_{\mathsf{poly}}$ for all $j \in [k]$ or the extractor discovers a non-trivial DL relation between $\vec{G}$. Then knowledge soundness is implied according to Lemma 2. For zero knowledge, $\mathit{\Pi}_{\mathsf{Amor}}$ satisfies the special HVZK property, if the input instances include at least one masking instance. The general idea is first to randomly sample witness $\vec{f}_j$ for all $j \in [k]$. Then assuming the k-th instance is the masking instance, the simulator can elaborately choose proper F_k, v_k conditioning on Equation (9). After executing the remaining protocol on the chosen masking instance, the simulator finally obtains an indistinguishable transcript. The formal security proofs of Theorem 3 are presented in Appendix A.4.*

Optimization. Observe that the Pedersen vector commitment scheme is a homomorphic relation, which allows $\mathcal{V}$ to directly output F by setting $F := \tilde{F}(\vec{r}_x)$. The equation $F = \sum_{i=1}^{m} f_i G_i$ still holds on the folded $\vec{f}$. Consequently, we can apply the traditional amortization technique [2,3] directly without the sum-check on g_1. This approach avoids sending any $\mathbb{G}$ element. The public reducibility and completeness of the optimized protocol are straightforwardly maintained. Regarding the knowledge soundness, denote $\rho_j := \mathsf{eq}\big(\vec{\alpha}, \mathsf{Bits}(j)\big)$. We have

$$F = \sum_{i=1}^{m} f_i G_i \quad \Longrightarrow \quad \sum_{j=1}^{k} \rho_j F_j = \sum_{i=1}^{m} \Big(\sum_{j=1}^{k} \rho_j f_{j,i} \Big) G_i = \sum_{j=1}^{k} \rho_j \sum_{i=1}^{m} f_{j,i} G_i. \quad (12)$$

Since $\vec{\alpha}$ are uniformly chosen, $F_j = \sum_{i=1}^{m} f_{j,i} G_i$ for all $j \in [k]$ holds with overwhelming probability.

Efficiency. Without the sum-check on g_1, $\mathit{\Pi}_{\mathsf{Amor}}$ requires only $(d+1)\log k$ prover's messages in $\mathbb{F}$. Besides, the prover sends one $s \in \mathbb{F}$ at step 3. Regarding the time complexity, the sum-check process involves $O(kd \log^2 d)$ and $O(d \log k)$ $\mathbb{F}$-operations for the prover and verifier, respectively. At step 4, computing F_x requires $O(k)$ $\mathbb{G}$-operations (can be reduced to $O(\log k)$ when the verifier can preprocess $\tilde{F}(\vec{X})$) and e_x requires $O(\log k)$ $\mathbb{F}$-operations.

3.4 Compression with sum-check

Recall $\mathcal{R}_{\mathsf{AmorPoly}}$ in Equation (8). By conducting MLEs on $\vec{f}, \vec{G}, \vec{v}$, we can rewrite the claims in $\mathcal{R}_{\mathsf{AmorPoly}}$ as

$$\sum_{\vec{y} \in \{0,1\}^{\log m}} \tilde{f}(\vec{y}) \cdot \tilde{G}(\vec{y}) = F, \qquad \sum_{\vec{y} \in \{0,1\}^{\log m}} \mathsf{eq}(\vec{\beta}, \vec{y}) \cdot \big(h\big(\tilde{f}(\vec{y}) - \tilde{v}(\vec{y})\big)\big) = s. \quad (13)$$

Given that both claims take the form of summations, it is possible to employ two sum-check protocols in parallel to prove the two claims. However, to enhance efficiency, we lift the claim of $s \in \mathbb{F}$ to $\mathbb{G}$ and compose the two claims into one. As a result, this approach only requires a single sum-check protocol.

Specifically, we define two polynomials $p_1 : \mathbb{F}^{\log m} \to \mathbb{G}$ and $p_2 : \mathbb{F}^{\log m} \to \mathbb{F}$:

$$p_1(\vec{Y}) := \tilde{f}(\vec{Y}) \cdot \tilde{G}(\vec{Y}), \quad p_2(\vec{Y}) := \mathsf{eq}(\vec{\beta}, \vec{Y}) \cdot \Big(h\big(\tilde{f}(\vec{Y})\big) - \tilde{v}(\vec{Y})\Big). \tag{14}$$

Compose p_1, p_2 as $p(\vec{y}) := p_1(\vec{y}) + p_2(\vec{y}) \cdot H$, where $H \leftarrow\!\$\ \mathbb{G}$ is an additional generator sampled from the verifier with an unknown discrete logarithm relative with $G_1, \cdots, G_m$. The prover and verifier can engage in a sum-check protocol to show $\sum_{\vec{y} \in \{0,1\}^{\log m}} p(\vec{y}) = F + s \cdot H$.

The compression protocol $\mathit{\Pi}_{\mathsf{Comp}}$ is formally described in Protocol 3, which verifies a $\mathcal{R}_{\mathsf{AmorPoly}}$ instance with logarithmic communication overhead.

Protocol 3 $\mathit{\Pi}_{\mathsf{Comp}}$: Check $\mathcal{R}_{\mathsf{AmorPoly}}$ with compression.

$\mathcal{P}(\vec{G}, F, s, \vec{v}, \vec{\beta}, \vec{f}), \mathcal{V}(\vec{G}, F, s, \vec{v}, \vec{\beta})$

1: $\mathcal{V} \to \mathcal{P}$: $H \leftarrow\!\$\ \mathbb{G}$.

2: $\mathcal{P}$ and $\mathcal{V}$: Set $p(\vec{y}) := p_1(\vec{y}) + p_2(\vec{y}) \cdot H$ where $p_1(\vec{Y}), p_2(\vec{Y})$ are defined in Equation (14) and engage in a sum-check for the claim

$$\sum_{\vec{y} \in \{0,1\}^{\log m}} p(\vec{y}) = F + s \cdot H. \tag{15}$$

The protocol reduces to two evaluation claims $p_1(\vec{r}_y) = F_p$ and $p_2(\vec{r}_y) = s_p$, where $\vec{r}_y \leftarrow\!\$\ \mathbb{F}^{\log m}$ are the challenges of the sum-check.

3: $\mathcal{P} \to \mathcal{V}$: $f := \tilde{f}(\vec{r}_y)$.

4: $\mathcal{V}$: Compute $e_y := \mathsf{eq}(\vec{\beta}, \vec{r}_y)$, $G := \tilde{G}(\vec{r}_y)$, and $v := \tilde{v}(\vec{r}_y)$. Check the following equation

$$F_p + s_p \cdot H = f \cdot G + e_y \cdot \big(h(f) - v\big) \cdot H. \tag{16}$$

Remarks. Our approach is different from PIOP-based schemes [13,33], which require that the verifier perform an additional check of $\tilde{f}(\vec{r}_y)$ with a PCS. In our design, this step is omitted because the commitment claim is already included within the sum-check protocol in Equation (15). Thus, the Pedersen vector commitment F binds $\tilde{f}(\vec{r}_y)$ at step 3 with the original witness, and $\tilde{f}(\vec{r}_y)$ itself serves as its own proof to ensure it is derived from the original witness. Consequently, our design does not require an additional query to $\tilde{f}$ within $\mathit{\Pi}_{\mathsf{Comp}}$, analogous to how $\vec{f}(r)$ functions at step 6 in Protocol 1.

Efficiency and optimization. Let $d := \deg(h)$ denote the degree of the polynomial h. The sum-check requires $(d+1)\log m$ prover's messages in $\mathbb{G}$. Additionally, the prover needs to send one $f \in \mathbb{F}$ at step 3. Regarding the time complexity, a straightforward implementation of the sum-check requires $O(md \log^2 d)$ $\mathbb{G}$-operations for the prover. However, the prover can calculate the sums over p_1 and p_2 separately and combine the results with a single scalar multiplication. Accordingly, the cost is reduced to $O(m)$ $\mathbb{G}$-operations and $O(md \log^2 d)$ $\mathbb{F}$-operations. The verifier's cost in the sum-check verifier remains unchanged, which is $O(d \log m)$ $\mathbb{G}$-operations. At step 4, computing e_y, G, v takes $O(\log m)$ $\mathbb{F}$-operations, $O(m)$ $\mathbb{G}$-operations (can be reduced to $O(\log m)$ through preprocessing to derive $\tilde{G}(\vec{Y})$), and $O(m)$ $\mathbb{F}$-operations, respectively.

Theorem 4. *Π_{Comp} satisfies completeness and knowledge soundness.*

Proof Sketch. *The completeness is trivial to prove. For the knowledge soundness, an extractor for special soundnss can compute a vector $\vec{f}$ from m transcripts with different challenges by querying to $\mathcal{P}^*$. Further guaranteed by the soundness of sum-check protocols, the extractor has an overwhelming probability to ensure the extracted $\vec{f}$ satisfying Equation (15). Given that H is uniformly sampled from the verifier, by the uniqueness of MLE and Schwartz-Zippel lemma, either $\vec{f}$ is a valid witness for $\mathcal{R}_{\mathsf{AmorPoly}}$ or the extractor discovers a non-trivial DL relation between $\vec{G}$ and H. Then knowledge soundness is implied according to Lemma 2. The formal security proofs of Theorem 4 are presented in Appendix A.5.*

3.5 Putting everything together

With Π_{Amor} and Π_{Comp}, we can derive a compressed protocol Π_{Poly} for multiple $\mathcal{R}_{\mathsf{Poly}}$ instances based on a *standard* Σ-protocol (Protocol 1). We first describe Π_{Poly} in Protocol 4 and then demonstrate how to further improve efficiency.

Protocol 4 Π_{Poly}: Check k instances of $\mathcal{R}_{\mathsf{Poly}}$ with compression.

```
$\mathcal{P}(\vec{G}, (F_j, \vec{v}_j, \vec{f}_j)_{j=1}^{k}), \mathcal{V}(\vec{G}, (F_j, \vec{v}_j)_{j=1}^{k})$
1: $\mathcal{P}$: Sample $\vec{f}_0 \leftarrow\!\!\$\ \mathbb{F}^m$. Compute $F_0, \vec{v}_0$ such that $(F_0, \vec{v}_0; \vec{f}_0) \in \mathcal{R}_{\mathsf{Poly}}$.
2: $\mathcal{P} \to \mathcal{V}$: $F_0, \vec{v}_0$.
   // we slightly abuse $\Pi_{\mathsf{Amor}}$ here to handle $(k+1)$ instances
3: $\mathcal{P}$ and $\mathcal{V}$: Engage in $\Pi_{\mathsf{Amor}}$, which reduces to $(F, s, \vec{v}, \vec{\beta}; \vec{f}) \in \mathcal{R}_{\mathsf{AmorPoly}}$.
4: $\mathcal{P}$ and $\mathcal{V}$: Engage in $\Pi_{\mathsf{Comp}}$ for $(F, s, \vec{v}, \vec{\beta}; \vec{f})$.
```

We summarize the architecture of our design in Figure 2. The amortization process is represented by steps ① and ②. We adopt the traditional amortization technique [2,3] for step ①, while step ② is executed using a sum-check protocol. For the compression phases ③ and ④, we employ a single sum-check protocol for both steps.

In comparison, the work of Attema et al. [2,3] applies traditional amortization to steps ① and ②, and employs Bulletproofs for the compression phases ③ and

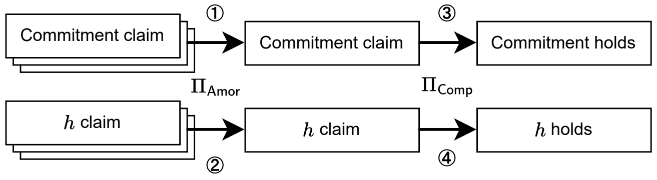


Fig. 2: The architecture of our design. $\mathit{\Pi}_{\mathsf{Amor}}$ utilizes a traditional amortization in [2,3] for step ① and a sum-check for step ②. In $\mathit{\Pi}_{\mathsf{Comp}}$, the sum-check protocol is applied to both steps ③ and ④.

④. Within PIOP-based frameworks [13, 33] — which focus on a single instance and thus do not require ① and ② — the PIOP component is tailored for step ④, whereas step ③ relies on a PCS.

Theorem 5. *$\mathit{\Pi}_{\mathsf{Poly}}$ satisfies completeness, knowledge soundness, and special HVZK.*

Proof. Based on Theorem 3 and Theorem 4, $\mathit{\Pi}_{\mathsf{Poly}}$ satisfies completeness and knowledge soundness. For the special HVZK, the simulator invokes the simulator of $\mathit{\Pi}_{\mathsf{Amor}}$ to simulate the transcripts at step 3 and $(F, s, \vec{v}, \vec{\beta}, \vec{f})$. It then continues to simulate the transcript for step 4 by executing $\mathit{\Pi}_{\mathsf{Comp}}$ with the previously simulated $(F, s, \vec{v}, \vec{\beta}, \vec{f})$. Since $(F, s, \vec{v}, \vec{\beta}; \vec{f}) \in \mathcal{R}_{\mathsf{AmorPoly}}$, $\mathcal{V}$ accepts. Additionally, the simulated transcript is indistinguishable from a real one. □

Efficiency. Let $d := \deg(h)$ be the degree of the polynomial h. The first step requires $O(m)$ $\mathbb{G}$-operations and $O(dm)$ $\mathbb{F}$-operations. The second step incurs 1 $\mathbb{G}$-message and m $\mathbb{F}$-messages.

The cost of the last two steps includes: $(d+1)\log m$ $\mathbb{G}$-messages and $((d+1)\log k + 4)$ $\mathbb{F}$-messages; proving time with $O(m)$ $\mathbb{G}$-operations and $O((m+k)d\log^2 d)$ $\mathbb{F}$-operations; and verification time with $O(d\log m + m + k)$ $\mathbb{G}$-operations and $O(d\log k + m)$ $\mathbb{F}$-operations.

Optimization. The cost of the first two steps is inherited from a standard $\mathit{\Sigma}$-protocol (Protocol 1). We demonstrate that this cost can be further mitigated by employing zero-knowledge adaptations of $\mathit{\Pi}_{\mathsf{Amor}}$ and $\mathit{\Pi}_{\mathsf{Comp}}$. Recall the two protocols, which leak the witness information during the sum-checks and the evaluation claims. To address this, we can replace them with zero-knowledge sum-checks [35] and claims with the hiding property [37]. We provide a brief overview of each approach here.

Zero-knowledge sum-check [35]. To prove the claim $\sum_{\vec{x}\in\{0,1\}^{\log\ell}} f(\vec{x}) = s_f$ where $d := \deg(f)$, the prover samples $a_0, a_{i,j} \leftarrow\$ \mathbb{F}$ for each $i \in [\log\ell]$ and $j \in [d]$. Subsequently, the prover computes a polynomial $g(\vec{x}) = a_0 + \sum_{i=1}^{\log\ell} g_i(x_i)$, where $g_i(x_i) = \sum_{j=1}^{d} a_{i,j} x_i^j$. The prover sends $s_g := \sum_{\vec{x}\in\{0,1\}^{\log\ell}} g(\vec{x})$ and engages a sum-check on $s_f + \rho\cdot s_g = \sum_{\vec{x}\in\{0,1\}^{\log\ell}} \big(f(\vec{x}) + \rho\cdot g(\vec{x})\big)$ with the verifier. In the final round, the prover sends $g(\vec{r})$. This whole process entails an additional cost of $O(d\log\ell)$ for the prover.

Hiding evaluation claim [37]. For μ evaluation claims $s = \tilde{f}(\vec{r})$ on the MLE of $\vec{f}$, the prover samples a vector $\vec{f'} \leftarrow\$ \mathbb{F}^{\mu}$ and computes $v'_i := h(f'_i)$ for all $i \in [\mu]$. The prover then sends $F' := \sum_{i=1}^{\mu} f'_i H_i$ and $\vec{v}'$ to the verifier, where $H_1, \cdots, H_\mu \in \mathbb{G}$ are generators whose discrete logarithm relations with $G_1, \cdots, G_m$ are unknown. The prover regards $\vec{f}^* := (\vec{f}, \vec{f'})$ as the new witness. Consequently, the evaluation claim on $\vec{f}^*$ become $\tilde{f}^*(\vec{r}, \vec{r'}) = s + \tilde{f}'(\vec{r'})$, which is uniformly distributed in $\mathbb{F}$. Since μ is a constant in our protocol, the hiding evaluation claims only incur a constant additional cost.

Efficiency. Instead of sending $F_0, \vec{v}_0$ during step 2 of Protocol 4, the prover sends $s_g \in \mathbb{F}$ for the zero-knowledge sum-check at the first round, along with $F' \in \mathbb{G}, \vec{v}^* \in \mathbb{F}^2$ for the hiding claims (noting that Π_{Poly} only has two evaluation claims). After step 3, the prover additionally sends one $\mathbb{G}$ element for the second zero-knowledge sum-check. The overall asymptotic computational cost of the protocol remains unchanged.

4 Extensions

In this section, we describe some variants of our techniques to accommodate a broader range of scenarios. Note that these variants are capable of functioning simultaneously, for instance, handing multiple polynomial claims over inner-products. While it is straightforward to support the function h that maps to $\mathbb{G}$, we focus on these variants where h maps to $\mathbb{F}$.

4.1 Polynomial over inner-products

We first consider a scenario where the polynomial claim is on multiple inner-products of $\vec{f}$. Denote the inner-product of two vectors $\vec{a}, \vec{f} \in \mathbb{F}^m$ as $\langle \vec{a}, \vec{f} \rangle = \sum_{i=1}^{m} a_i f_i$. The polynomial claim is defined as $h(\langle \vec{a}_1, \vec{f} \rangle, \cdots, \langle \vec{a}_t, \vec{f} \rangle) = v$ for some public $v \in \mathbb{F}, \vec{a}_1, \cdots, \vec{a}_t \in \mathbb{F}^m$. Accordingly, the relation $\mathcal{R}_{\mathsf{PolyIP}}$ is defined in Equation (17).

$$\mathcal{R}_{\mathsf{PolyIP}} := \left\{ \begin{array}{l} (F \in \mathbb{G}, v \in \mathbb{F}, \vec{a}_1, \cdots, \vec{a}_t \in \mathbb{F}^m; \vec{f} \in \mathbb{F}^m) : \\ \sum_{i=1}^{m} f_i G_i = F, h\big(\langle \vec{a}_1, \vec{f} \rangle, \cdots, \langle \vec{a}_t, \vec{f} \rangle\big) = v \end{array} \right\}. \tag{17}$$

Different from [2,3], we cannot perform an inner-product operation on the polynomial claims within $\mathcal{R}_{\mathsf{Poly}}$ to obtain $\mathcal{R}_{\mathsf{PolyIP}}$ since h is non-homomorphic.

For k instances $(F_j, v_j, \vec{a}_{j,1}, \cdots, \vec{a}_{j,t}; \vec{f}_j) \in \mathcal{R}_{\mathsf{PolyIP}}, \forall j \in [k]$, after MLE, the polynomial claims are equivalent to the following claim for all $\vec{x} \in \{0,1\}^{\log k}$:

$$h\Big(\sum_{\vec{y} \in \{0,1\}^{\log m}} \tilde{a}_1(\vec{x}, \vec{y}) \cdot \tilde{f}(\vec{x}, \vec{y}), \cdots, \sum_{\vec{y} \in \{0,1\}^{\log m}} \tilde{a}_t(\vec{x}, \vec{y}) \cdot \tilde{f}(\vec{x}, \vec{y}) \Big) = \tilde{v}(\vec{x}). \tag{18}$$

Therefore, in Π_{Amor}, we replace $g_2(\vec{X})$ in Equation (7) with

$$g_2(\vec{X}) := \mathsf{eq}(\vec{\alpha}, \vec{X}) \cdot \Big(\tilde{v}(\vec{X}) - h\Big(\sum_{\vec{y} \in \{0,1\}^{\log m}} \tilde{a}_1(\vec{X}, \vec{y}) \cdot \tilde{f}(\vec{X}, \vec{y}), \cdots , \sum_{\vec{y} \in \{0,1\}^{\log m}} \tilde{a}_t(\vec{X}, \vec{y}) \cdot \tilde{f}(\vec{X}, \vec{y})\Big)\Big). \tag{19}$$

Accordingly, we have t claims in Equation (10) for each inner-product

$$s_i := \sum_{\vec{y} \in \{0,1\}^{\log m}} \tilde{a}_i(\vec{r}_x, \vec{y}) \cdot \tilde{f}(\vec{r}_x, \vec{y}), \quad \forall i \in [t], \tag{20}$$

and the second check in Equation (11) becomes

$$s_g = e_x \cdot \big(v_x - h(s_1, \cdots , s_t)\big), \tag{21}$$

where $v_x := \tilde{v}(\vec{r}_x)$ is computed by the verifier. The verifier additionally outputs $v := \tilde{v}(\vec{r}_x), \vec{a}_1 := \big(\tilde{a}_1(\vec{r}_x, \vec{y})\big)_{\vec{y} \in \{0,1\}^{\log m}}, \cdots , \vec{a}_t := \big(\tilde{a}_t(\vec{r}_x, \vec{y})\big)_{\vec{y} \in \{0,1\}^{\log m}}$ ($\vec{\beta}, \vec{v}$ are not required).

In Π_{Comp}, the verifier can compose the t claims in Equation (20) with a random challenge $\gamma \leftarrow\!\!\$\ \mathbb{F}$ as $s := \sum_{i=1}^{t} \gamma^i s_i = \sum_{\vec{y} \in \{0,1\}^{\log m}} (\sum_{i=1}^{t} \gamma^i \cdot \tilde{a}_i(\vec{y})) \cdot \tilde{f}(\vec{y})$. Accordingly, we adjust $p_2(\vec{Y})$ in Equation (14) as

$$p_2(\vec{Y}) := \big(\sum_{i=1}^{t} \gamma^i \cdot \tilde{a}_i(\vec{Y})\big) \cdot \tilde{f}(\vec{Y}), \tag{22}$$

and the final check in Equation (16) becomes

$$F_p + s_p \cdot H = f \cdot G + (\sum_{i=1}^{t} \gamma^i a_i) \cdot f \cdot H, \tag{23}$$

where $a_i := \tilde{a}_i(\vec{r}_y)$ is computed by the verifier for each $i \in [t]$.

Given that Equation (20) yields t separate claims, the cost will be escalated by a factor of t. Nonetheless, as t is generally a predefined constant, the overall asymptotic cost remains the same.

4.2 Relation with multiple polynomials

We consider the scenario in which a prover aims to prove multiple polynomial claims on one witness. The relation $\mathcal{R}_{\mathsf{MultPoly}}$ is defined in Equation (24).

$$\mathcal{R}_{\mathsf{MultPoly}} := \left\{ \begin{array}{c} (F \in \mathbb{G}, \vec{v} \in \mathbb{F}^m; \vec{f} \in \mathbb{F}^m) : \\ \sum_{i=1}^{m} f_i G_i = F, h_1(f_i) = v_{1,i}, \cdots , h_s(f_i) = v_{s,i}, \forall i \in [m] \end{array} \right\}. \tag{24}$$

In this case, the prover can efficiently combine s polynomials with a challenge $\zeta \leftarrow\!\!\$\ \mathbb{F}$ from the verifier, reducing to $h(f_i) = v_i$ for all $i \in [m]$, where $h(X) := \sum_{j=1}^{s} \zeta^j h_j(X)$ and $v_i := \sum_{j=1}^{s} \zeta^j v_{j,i}$. This resultant relation falls in $\mathcal{R}_{\mathsf{Poly}}$.

This process is a reduction of knowledge from $\mathcal{R}_{\mathsf{MultPoly}}$ to $\mathcal{R}_{\mathsf{Poly}}$. The public reducibility and completeness trivially hold. For the knowledge soundness, the witness $\vec{f}$ in $\mathcal{R}_{\mathsf{Poly}}$ is exactly the extracted witness. Since ζ is uniformly sampled from $\mathbb{F}$ and $h(f_i) = v_i$ holds for each $i \in [m]$, with overwhelming probability, $h_j(f_i) = v_{j,i}$ holds for all $i \in [m]$ and $j \in [k]$.

4.3 Relations with different polynomials

We further consider another relation $\mathcal{R}_{\mathsf{DiffPoly}}$ where the polynomial h is specified as part of the public statement:

$$\mathcal{R}_{\mathsf{DiffPoly}} := \left\{ \begin{array}{c} (F \in \mathbb{G}, \vec{v} \in \mathbb{F}^m, h : \mathbb{F} \to \mathbb{F}; \vec{f} \in \mathbb{F}^m) : \\ \sum_{i=1}^m f_i G_i = F, \quad h(f_i) = v_i \quad \forall i \in [m] \end{array} \right\}. \tag{25}$$

Consequently, for k instances $(F_j, s_j, \vec{v}_j, h_j; \vec{f}_j)_{j=1}^k$, the polynomial h_j varies across the instances.

To utilize $\mathit{\Pi}_{\mathsf{Amor}}$, it is also necessary to perform an MLE on the polynomials h_j. This can be achieved by regarding h_j's as elements within a polynomial ring. Thus, Definition 1 is directly applicable to h_j's (the MLE over rings is also given in [14]). For the sake of clarity, we describe the MLE over polynomials in Definition 3.

Definition 3. ***(Multilinear extension over polynomials).*** *Given μ-many d-degree polynomial $h_i[\vec{Y}] : \mathbb{F}^m \to \mathbb{F}$ for all $i \in [\mu]$, the MLE of h_i's is defined as*

$$\tilde{h}(\vec{Y}, \vec{X}) := \sum_{i=1}^{\mu} h_i(\vec{Y}) \cdot \mathsf{eq}(\mathsf{Bits}(i), \vec{X}) \quad \in \mathbb{F}^{\leq d}[X_1, \cdots, X_{\log \mu}, Y_1, \cdots, Y_m].$$

The individual degree of $Y_1, \cdots, Y_m$ remains d as we are merely summing up each h_i, while the individual degree of $X_1, \cdots, X_{\log \mu}$ is (at most) 1 as we conduct an MLE. The existence and uniqueness of MLE are preserved [8, 14]. In the context of sum-check protocols for $\tilde{h}$-related claims, the challenges are sampled from the field $\mathbb{F}$ (instead of the polynomial ring), ensuring that all elements are invertible, which in turn maintains the validity of the sum-check process.

Accordingly, g_2 in Equation (7) and p_2 in Equation (14) become

$$\begin{aligned} g_2(\vec{X}) &:= \mathsf{eq}(\vec{\alpha}, \vec{X}) \cdot \sum_{\vec{y} \in \{0,1\}^{\log m}} \Big(\mathsf{eq}(\vec{\beta}, \vec{y}) \cdot \Big(\tilde{h}\big(\vec{y}, \tilde{f}(\vec{X}, \vec{y})\big) - \tilde{v}(\vec{X}, \vec{y}) \Big) \Big), \\ p_2(\vec{Y}) &:= \mathsf{eq}(\vec{\beta}, \vec{Y}) \cdot \Big(\tilde{h}\big(\vec{Y}, \tilde{f}(\vec{Y})\big) - \tilde{v}(\vec{Y}) \Big). \end{aligned} \tag{26}$$

The remaining steps are the same.

5 Arithmetic Circuit Satisfiability

Our protocol can easily support arithmetic circuit relations, paving the way for the construction of (zk)SNARKs. In this section, we demonstrate how to prove the Customizable Constraint Systems (CCS) [34] relation, which is a generalized arithmetization technique supporting Rank-1 Constraint System (R1CS) [24], Plonkish [23], and Algebraic Intermediate Representation (AIR) circuits [5].

Our construction follows the commit and prove paradigm, i.e., the prover commits to the witness and subsequently proves that it satisfies the required committed relation. We begin by revisiting the CCS relation in [34]. Given the following public integers: k (the number of constraints), m (the extended witness size), and ℓ (the public input size), the public parameters include: (i) t matrices $M_1, \cdots, M_t \in \mathbb{F}^{k\times m}$; (ii) m_s multisets $\mathbb{S}_1, \cdots, \mathbb{S}_{m_s} \subset [t]$; and (iii) m_s scalars $c_1, \cdots, c_{m_s} \in \mathbb{F}$. The committed CCS relation $\mathcal{R}_{\mathsf{CCS}}$ is defined as

$$\mathcal{R}_{\mathsf{CCS}} := \left\{ \begin{array}{c} (F \in \mathbb{G}, \vec{x} \in \mathbb{F}^{\ell}; \vec{w} \in \mathbb{F}^{m'}) : \sum_{i=1}^{m'} w_i G_i = F, \\ \vec{z} := (\vec{w}, 1, \vec{x}) \in \mathbb{F}^m, \quad \sum_{i=1}^{m_s} c_i \cdot \bigcirc_{j\in\mathbb{S}_i} (M_j \cdot \vec{z}) = \vec{0} \end{array} \right\}, \tag{27}$$

where $m' := m - \ell - 1$ and $\vec{0}$ is a k-size zero vector.

Furthermore, we demonstrate the application of our technique to prove $\mathcal{R}_{\mathsf{CCS}}$ relations. Consider each row of M_j as a vector $\vec{m}_{j,u}$ for all $u \in [k]$. The $\sum_{i=1}^{m_s} c_i \cdot \bigcirc_{j\in\mathbb{S}_i} (M_j \cdot \vec{z}) = \vec{0}$ part in Equation (27) can be reformulated as follows:

$$\sum_{i=1}^{m_s} c_i \cdot \prod_{j\in\mathbb{S}_i} \langle \vec{m}_{j,u}, \vec{z} \rangle = 0, \quad \forall u \in [k], \tag{28}$$

where $\langle \vec{a} \cdot \vec{b} \rangle = \sum_{i=1}^{m} a_i b_i$. For each row, we define a new relation $\mathcal{R}_{\mathsf{rCCS}}$:

$$\mathcal{R}_{\mathsf{rCCS}} := \left\{ \begin{array}{c} (Z \in \mathbb{G}, \vec{m}_1, \cdots, \vec{m}_t \in \mathbb{F}^m; \vec{z} \in \mathbb{F}^m) : \\ \sum_{i=1}^{m} z_i G_i = Z, \quad \sum_{i=1}^{m_s} c_i \cdot \prod_{j\in\mathbb{S}_i} \langle \vec{m}_j, \vec{z} \rangle = 0 \end{array} \right\}, \tag{29}$$

where $G_{m'+1}, \cdots, G_m$ are randomly sampled $\mathbb{G}$ elements from the verifier with unknown discrete logarithm relations with $G_1, \cdots, G_{m'}$. Therefore, $\mathcal{R}_{\mathsf{CCS}}$ is essentially the case of simultaneously satisfying k-many $\mathcal{R}_{\mathsf{rCCS}}$ relations

$$\mathcal{R}_{\mathsf{CCS}} := \left\{ \begin{array}{c} (F \in \mathbb{G}, \vec{x} \in \mathbb{F}^{\ell}; \vec{w} \in \mathbb{F}^{m'}) : Z := F + G_{m'+1} + \sum_{i=1}^{\ell} x_i G_{m'+1+i}, \\ \vec{z} := (\vec{w}, 1, \vec{x}), \quad (Z, \vec{m}_{1,u}, \cdots, \vec{m}_{t,u}; \vec{z}) \in \mathcal{R}_{\mathsf{rCCS}} \quad \forall u \in [k] \end{array} \right\}.$$

The binding property of the commitment scheme ensures that the extracted $\vec{z}$ from the $\mathcal{R}_{\mathsf{rCCS}}$ extractor must satisfy $\sum_{i=1}^{m'} z_i G_i = F$.

Since $\mathcal{R}_{\mathsf{rCCS}}$ is a special case of $\mathcal{R}_{\mathsf{PolyIP}}$ where $h\big(\langle \vec{m}_{1,u}, \vec{z} \rangle, \cdots, \langle \vec{m}_{t,u}, \vec{z} \rangle\big) := \sum_{i=1}^{m_s} c_i \cdot \prod_{j\in\mathbb{S}_i} \langle \vec{m}_{j,u}, \vec{z} \rangle$ and $v := 0$, we can utilize $\mathit{\Pi}_{\mathsf{Amor}}$ to amortize k relations and $\mathit{\Pi}_{\mathsf{Comp}}$ to prove the amortized relation. The details of the protocol $\mathit{\Pi}_{\mathsf{CCS}}$ are described in Protocol 5.

Remarks. Given that all k instances share the same witness $\vec{z}$, it is unnecessary to fold $\vec{z}$ during $\mathit{\Pi}_{\mathsf{Amor}}$. Consequently, $\mathit{\Pi}_{\mathsf{Amor}}$ is essentially the process

Protocol 5 $\mathit{\Pi}_{\mathsf{CCS}}$: Check $\mathcal{R}_{\mathsf{CCS}}$ with compression.

```
P(G⃗, F, x⃗, w⃗), V(G⃗, F, x⃗)
1: $\mathcal{V} \to \mathcal{P}$: $G_{m'+1}, \cdots, G_m \leftarrow\$ \mathbb{G}$.
2: $\mathcal{P}$: Set $\vec{z} := (\vec{w}, 1, \vec{x})$.
3: $\mathcal{V}$: Set $Z := F + G_{m'+1} + \sum_{i=1}^{\ell+1} x_i G_{m'+1+i}$. Obtain $\vec{m}_{i,u}$ for all $i \in [t]$ and
   $u \in [k]$ from the public input.
4: $\mathcal{P}$ and $\mathcal{V}$: Engage in $\mathit{\Pi}_{\mathsf{Poly}}$ to prove $k$ instances in $\mathcal{R}_{\mathsf{rCCS}}$ with the same $\vec{z}$.
```

of folding k sets of vectors $(\vec{m}_{1,u}, \cdots, \vec{m}_{t,u})$. The output witness of $\mathit{\Pi}_{\mathsf{Amor}}$ is $\vec{z}$. Besides, as discussed in Section 3.5, since a single sum-check is utilized to prove both the commitment and CCS claims in $\mathit{\Pi}_{\mathsf{Comp}}$, there is no additional need for a PCS to validate $\tilde{z}$ in [13, 33]. This is attributed to the fact that $\mathit{\Pi}_{\mathsf{Comp}}$, when proving the commitment claim, already binds the evaluation claim to the original witness.

6 Applications

We demonstrate the application of Σ-Check to construct compressed Σ-protocols for various applications. It is important to note that in these contexts, the witness for the masking instance should be uniformly sampled from $\mathbb{F}$. Thus, the masking instance should belong to $\mathcal{R}_{\mathsf{Poly}}$ relation instead of the specific ones.

6.1 Binary proofs

A binary proof serves as a crucial component for numerous cryptographic applications, enabling a prover to demonstrate possession of a binary witness. The relation is defined as follows:

$$\mathcal{R}_{\mathsf{Bin}} := \left\{ \begin{array}{c} (F \in \mathbb{G}; \vec{f} \in \mathbb{F}^m) : \\ \sum_{i=1}^m f_i G_i = F, \quad (1 - f_i) f_i = 0 \quad \forall i \in [m] \end{array} \right\}. \tag{30}$$

This relation is a specific case of $\mathcal{R}_{\mathsf{Poly}}$, where $h(X) := (1 - X)X$ and $v_i := 0$ for all $i \in [m]$. Consequently, we can directly apply $\mathit{\Pi}_{\mathsf{Poly}}$ to prove k instances of $\mathcal{R}_{\mathsf{Bin}}$ relation.

6.2 Range proofs

Range Proofs have been extensively adopted in many real-world applications such as confidential transactions. In a range proof, a prover aims to demonstrate that a witness v falls within a public range, say $[0, \cdots, 2^m - 1]$. The corresponding relation is defined as follows:

$$\mathcal{R}_{\mathsf{Range}} := \left\{ \begin{array}{c} (V \in \mathbb{G}; v \in \mathbb{F}) : v \cdot G = F, \\ \sum_{i=1}^m 2^{i-1} \cdot f_i = v, \quad (1 - f_i) f_i = 0 \quad \forall i \in [m] \end{array} \right\}, \tag{31}$$

where $\vec{f}$ can be regarded as the binary representation of v. We rewrite $\mathcal{R}_{\mathsf{Range}}$ relation by regarding the $\vec{f}$ as the witness and V as a public value:

$$\mathcal{R}_{\mathsf{Range}} := \left\{ \begin{array}{c} (V \in \mathbb{G}; \vec{f} \in \mathbb{F}^m): \\ (\sum_{i=1}^{m} 2^{i-1} \cdot f_i) \cdot G = V, \quad (1-f_i)f_i = 0 \quad \forall i \in [m] \end{array} \right\}. \tag{32}$$

Denote $\vec{a} := (1, 2, \cdots, 2^{m-1})$. We have a relation under two different polynomials: one in the form of an inner-product $h_1\big(\langle \vec{a}, \vec{X} \rangle\big) = \langle \vec{a}, \vec{X} \rangle \cdot G$, and the other in the form $h_2 = (1-X)X$. Moreover, since the binary representation of a value is unique, it implies that V is uniquely bound to $\vec{f}$ in $\mathcal{R}_{\mathsf{Range}}$. Accordingly, by regarding $\mathcal{R}_{\mathsf{Range}}$ as a binding commitment relation, we can apply Lemma 3 on $\mathcal{R}_{\mathsf{Range}}$. This enables us to execute Π_{Poly} for $\mathcal{R}_{\mathsf{MultPoly}}$ (one polynomial is over an inner-product) without an additional commitment claim on $\vec{f}$.

6.3 Partial knowledge proofs

The partial knowledge proof (k-out-of-m proof) is a basic anonymous technique that has been extensively applied in ring signatures and anonymous cryptocurrencies. This method enables a prover to demonstrate the knowledge of k openings within a set of m public commitments. The relation is defined as follows:

$$\mathcal{R}_{\mathsf{k/m}} := \left\{ \begin{array}{c} (\vec{P} \in \mathbb{G}^m, k \in [m]; \mathbb{S} \in [m], \vec{s} \in \mathbb{F}^m): \\ |\mathbb{S}| = k; \quad P_i = s_i G, \forall i \in \mathbb{S}; \quad s_i = 0, \forall i \notin \mathbb{S} \end{array} \right\}. \tag{33}$$

We introduce an additional vector $\vec{b} \in \{0,1\}^m$ where $b_i = 1$ if $i \in \mathbb{S}$ and $b_i = 0$ otherwise. The last two equations in $\mathcal{R}_{\mathsf{k/m}}$ can be equivalently expressed as $b_i P_i = s_i G$ for all $i \in [m]$. Thus, given a random challenge $\gamma \leftarrow\$ \mathbb{F}$ from the verifier, we can rewrite $\mathcal{R}_{\mathsf{k/m}}$ as follows:

$$\mathcal{R}_{\mathsf{k/m}} := \left\{ \begin{array}{c} (\vec{P} \in \mathbb{G}^m, k \in [m], \gamma \in \mathbb{F}; \vec{f}, \vec{b} \in \mathbb{F}^m): \sum_{i=1}^{m} b_i = k, \\ \sum_{i=1}^{m} \gamma^{i-1} b_i P_i = \sum_{i=1}^{m} \gamma^{i-1} s_i G, \quad (1-b_i)b_i = 0 \quad \forall i \in [m]. \end{array} \right\}, \tag{34}$$

Denote $\vec{G}_\gamma := (G, \gamma G, \cdots, \gamma^{m-1} G)$ and $\vec{P}_\gamma := (P_1, \gamma P_2, \cdots, \gamma^{m-1} P_m)$. $\mathcal{R}_{\mathsf{k/m}}$ has three different polynomials, two of them take the form of inner-products, $h_1\big(\langle \vec{1}, \vec{X} \rangle\big) = \langle \vec{1}, \vec{X} \rangle$ and $h_2\big(\langle \vec{G}_\gamma, \vec{X} \rangle, \langle \vec{P}_\gamma, \vec{Y} \rangle\big) = \langle \vec{P}_\gamma, \vec{X} \rangle - \langle \vec{G}_\gamma, \vec{Y} \rangle$; while the third polynomial is given by $h_3(X) = (1-X)X$. It is apparent that $\mathcal{R}_{\mathsf{k/m}}$ is a binding relation, and thus, Lemma 3 holds on $\mathcal{R}_{\mathsf{k/m}}$.

When dealing with multiple instances, the prover first computes $\vec{b}_j$ for each instance. The verifier then sends a random challenge $\gamma \leftarrow\$ \mathbb{F}$ and locally computes $\vec{G}_\gamma$ and $\vec{P}_{\gamma,j}$'s for each instance. Finally, they engage in the Π_{Poly} for $\mathcal{R}_{\mathsf{MultPoly}}$ (with two polynomials over inner-products).

7 Σ-Check from other assumptions

Till now, we have implemented our protocols using Pedersen vector commitment based on the DL assumption. In this section, we will extend our techniques to other assumptions, including the Strong-RSA and GDLR assumptions.

7.1 Strong-RSA assumption

To generalize our techniques to the Strong-RSA assumption, we adopt a similar approach in [2] by replacing the Pedersen vector commitment to an integer PCS [10, 19]. The commitment space is a group $\mathbb{G}$ of unknown order (with an upper bound $\mathcal{B}$), which only requires two group elements $G, H \in \mathbb{G}$ that generate the same subgroup of $\mathbb{G}$.

Let $\mathbb{Z}$ denote the set of all integers, and $\mathbb{Z}_p$ represent integers in $[-\frac{p-1}{2}, \frac{p-1}{2}]$. Specifically, for a vector $\vec{x} \in \mathbb{Z}_p^n$ such that $||\vec{x}||_\infty \leq \frac{p-1}{2}$ and a random $\gamma \in \mathbb{Z}$ chosen uniformly from $[0, \mathcal{B} \cdot 2^\lambda]$, where λ is the security parameter, the commitment scheme is defined as follows:

$$F = G^{\sum_{i=1}^n x_i q^{i-1}} H^\gamma, \tag{35}$$

where q is a "large enough" integer, e.g., $\frac{q}{2} > (\frac{p+1}{2})^{\log k + \log m}$ in our design. Intuitively, since $\vec{x} \mapsto \sum_{i=1}^n x_i q^{i-1}$ is injective, and the prover does not know the order of the group $\mathbb{G}$, this commitment scheme is binding (following from the root assumption [10, 19]). Additionally, since the distribution of H^γ is statistically close to the uniform distribution of the subgroup, the scheme statistically hides $\vec{x}$.

When applying the Strong-RSA assumption to our protocols, we only need to replace all Pedersen commitment parts with the commitment in Equation (35). Specifically, we update the original relations $\mathcal{R}_{\mathsf{Poly}}$ and $\mathcal{R}_{\mathsf{AmorPoly}}$ to $\mathcal{R}_{\mathsf{Poly}}^{\mathsf{RSA}}$ and $\mathcal{R}_{\mathsf{AmorPoly}}^{\mathsf{RSA}}$ as follows:

$$\mathcal{R}_{\mathsf{Poly}}^{\mathsf{RSA}} := \left\{ \begin{array}{c} (F \in \mathbb{G}, \vec{v} \in \mathbb{Z}^m; \vec{f} \in \mathbb{Z}_p^m, \gamma \in \mathbb{Z}) : \\ G^{\sum_{i=1}^m f_i q^{i-1}} H^\gamma = F, \quad h(f_i) = v_i \quad \forall i \in [m], \quad \|\vec{f}\|_\infty \leq \frac{p-1}{2} \end{array} \right\},$$

$$\mathcal{R}_{\mathsf{AmorPoly}}^{\mathsf{RSA}} := \left\{ \begin{array}{c} (F \in \mathbb{G}, s, b \in \mathbb{Z}, \vec{v} \in \mathbb{Z}^m, \vec{\beta} \in \mathbb{Z}^{\log m}; \vec{f} \in \mathbb{Z}_p^m, \gamma \in \mathbb{Z}) : \\ G^{\sum_{i=1}^m f_i q^{i-1}} H^\gamma = F, \sum_{i=1}^m \mathsf{eq}\big(\vec{\beta}, \mathsf{Bits}(i)\big)\big(h(f_i) - v_i\big) = s, \|\vec{f}\|_\infty \leq b \end{array} \right\}.$$

Since a norm constraint is included in the relation, the sum-check verifier should also conduct a norm check in each round. Specifically, in Lemma 1, the verifier additionally checks $\|f_i\|_\infty \leq \frac{p+1}{2} \cdot \|f_{i-1}\|_\infty < \frac{q}{2}$ in each iteration, where $\|f_i\|_\infty$ is the infinity norm of f_i's coefficients (sum-check challenges are sampled from $\big[-\frac{p-1}{2}, \frac{p-1}{2}\big]$).

In the optimized version of $\mathit{\Pi}_{\mathsf{Amor}}$ (Protocol 2), since all commitment-related parts (i.e., the blue parts) are removed, we only need to include $\tilde{\gamma}(\vec{r}_x)$ and the norm bound b in the output as follows, where $\tilde{\gamma}(\vec{X})$ is the MLE of $(\gamma_j)_{i=1}^k$:

$$\tilde{\gamma}(\vec{X}) := \sum_{j=1}^k \mathsf{eq}(\mathsf{Bits}(j), \vec{X}) \cdot \gamma_j, \qquad b := \Big(\frac{p+1}{2}\Big)^{\log k}.$$

In $\mathit{\Pi}_{\mathsf{Comp}}$ (Protocol 3), both the prover and verifier set $\tilde{q}(\vec{Y}) := \sum_{i=1}^m \mathsf{eq}(\mathsf{Bits}(i), \vec{Y}) \cdot q^{i-1}$. Additionally, we define the polynomial p_1 as $p_1(\vec{Y}) := \tilde{f}(\vec{Y}) \cdot \tilde{q}(\vec{Y})$ and use a random challenge $r \leftarrow\!\!\$\ \mathbb{Z}$ to combine $p(\vec{y}) := p_1(\vec{y}) + r \cdot p_2(\vec{y})$ in step 2. Accordingly, the sum-check claim of Equation (15) becomes

$$G^{\sum_{\vec{y}\in\{0,1\}^{\log m}} p(\vec{y})} H^{\gamma} = F \cdot G^{r \cdot s}.$$

We denote the resulting evaluation claim on p_1 as $G^{p_1(\vec{r}_y)} = F_p$. The final verification in Equation (16) becomes

$$F_p \cdot G^{r \cdot s_p} = G^{f + r \cdot e_y(h(f)-v)} H^{\gamma}, \qquad |f| < b \cdot \left(\frac{p+1}{2}\right)^{\log m} < \frac{q}{2}.$$

We formally design the amortization and compression protocols based on the Strong-RSA assumption, with detailed information available in Appendix B.1.

7.2 GIPA-based commitments

In addition to the DL assumption and the Strong-RSA assumption, our protocols can be applied to any GIPA-based commitment schemes [11].

GIPA is a generalization of inner-product commitment schemes, which supports Pedersen vector commitment (based on DL) and LMR commitment [28] (based on GDLR). Let $\mathsf{CM}(\vec{\mathsf{ck}}, \vec{f})$ denote a commitment scheme based on GIPA such that $\mathsf{CM}(\vec{\mathsf{ck}}, \vec{f}) = \sum_{i=1}^{m} \mathsf{CM}(\mathsf{ck}_i, f_i)$, where $\vec{\mathsf{ck}}$ is an m-size commitment key and $\vec{f}$ is an m-size message. For Pedersen vector commitment and LMR commitment, the $\mathsf{CM}(\mathsf{ck}_i, f_i)$ is defined as follows:

$$\begin{aligned} \text{Pedersen:} \quad & \mathsf{CM} : \mathbb{G} \times \mathbb{F} \to \mathbb{G}, & \mathsf{CM}(\mathsf{ck}_i, f_i) = f_i \cdot \mathsf{ck}_i; \\ \text{LMR:} \quad & \mathsf{CM} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T, & \mathsf{CM}(\mathsf{ck}_i, f_i) = e(\mathsf{ck}_i, f_i), \end{aligned}$$

where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing mapping.

We can apply arbitrary GIPA-based commitments to our protocols by replacing the commitment part using $\mathsf{CM}(\vec{\mathsf{ck}}, \vec{f})$. Specifically, we modify relation $\mathcal{R}_{\mathsf{Poly}}$ and $\mathcal{R}_{\mathsf{AmorPoly}}$ to $\mathcal{R}^{\mathsf{GIPA}}_{\mathsf{Poly}}$ and $\mathcal{R}^{\mathsf{GIPA}}_{\mathsf{AmorPoly}}$ as follows:

$$\begin{aligned} \mathcal{R}^{\mathsf{GIPA}}_{\mathsf{Poly}} &:= \left\{ (F, \vec{v}; \vec{f}) : \mathsf{CM}(\vec{\mathsf{ck}}, \vec{f}) = F, \quad h(f_i) = v_i \ \ \forall i \in [m] \right\}, \\ \mathcal{R}^{\mathsf{GIPA}}_{\mathsf{AmorPoly}} &:= \left\{ (F, s, \vec{v}, \vec{\beta}; \vec{f}) : \mathsf{CM}(\vec{\mathsf{ck}}, \vec{f}) = F, \sum_{i=1}^{m} \mathsf{eq}\big(\vec{\beta}, \mathsf{Bits}(i)\big)\big(h(f_i) - v_i\big) = s \right\}. \end{aligned}$$

Since it is straightforward to generalize the Pedersen-related parts in our protocols to a GIPA-based commitment, we leave the details in Appendix B.2.

8 Evaluations

8.1 Theoretical performance

We summarize the theoretical efficiency of $\mathit{\Pi}_{\mathsf{Poly}}$ in Table 2. The "non-optimized" version refers to a direct application of our techniques to a Σ-protocol, i.e.,

Table 2: Efficiency of Π_{Poly}. $\mathbb{G}$ and $\mathbb{F}$ in the context of proof size denote the number of elements in $\mathbb{G}$ and $\mathbb{F}$, respectively. Regarding the computation time for the prover and verifier, $\mathbb{G}$-op and $\mathbb{F}$-op refer to the number of operations performed in the groups $\mathbb{G}$ and $\mathbb{F}$, respectively.

	Non-optimized	**Optimized**
Proof size	$((d+1)\log m+1)$ $\mathbb{G}$ $((d+1)\log k+m+4)$ $\mathbb{F}$	$((d+1)\log m+2)$ $\mathbb{G}$ $((d+1)\log k+7)$ $\mathbb{F}$
Prover time	$O(m)$ $\mathbb{G}$-op, $O((m+k)d\log^2 d)$ $\mathbb{F}$-op	
Verifier time	$O(d\log m+m+k)$ $\mathbb{G}$-op, $O(d\log k+m)$ $\mathbb{F}$-op	

Table 3: Performance between Σ-Check and [2]. [2]-Amortization and [2]-Compression represent the Amortization and Compression phase in [2], respectively. Ours-Amortization and Ours-Compression represent the Amortization and Compression phase in our work, respectively. $|C|$ represents the size of the arithmetic circuit in [2], which is of $O(kd)$.

	Proof Size	**Prover time**	**Verifier time**
[2]-Amortization	$3k+|C|$ $\mathbb{F}$ k $\mathbb{G}$	$O(kd\log d)$ $\mathbb{F}$-op $O(kd)$ $\mathbb{G}$-op	$O(k)$ $\mathbb{F}$-op
[2]-Compression	2 $\mathbb{F}$ $2\log(m+kd)-2$ $\mathbb{G}$	$O((m+kd)\log(m+kd)$ $\mathbb{G}$-op $O((m+kd)\log(m+kd)$ $\mathbb{F}$-op	$O(m+kd)$ $\mathbb{F}$-op $O(m+kd)$ $\mathbb{G}$-op
Ours-Amortization	$(d+1)\log k$ $\mathbb{G}$	$O(kd\log^2 d)$ $\mathbb{F}$-op	$O(d\log k)$ $\mathbb{F}$-op $O(k)$ $\mathbb{G}$-op
Ours-Compression	$(d+1)\log m$ $\mathbb{G}$	$O(md\log^2 d)$ $\mathbb{F}$-op $O(m)$ $\mathbb{G}$-op	$O(m)$ $\mathbb{F}$-op $O(d\log m+m)$ $\mathbb{G}$-op

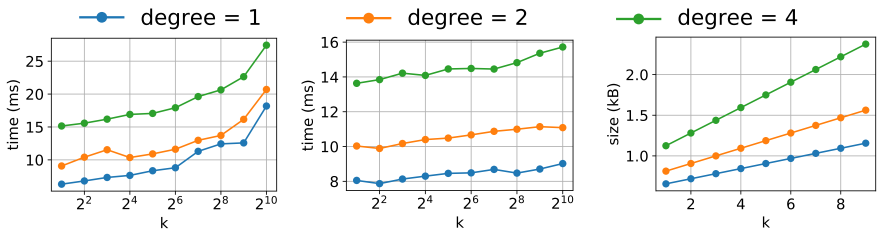
the efficiency of Protocol 4. On the other hand, the "optimized" version is the optimized Π_{Poly} in Section 3.5. Notably, in the optimized version, the proof size remains logarithmically to k and m, surpassing existing approaches that incur costs linear to k with linearization. Additionally, the computationally intensive $\mathbb{G}$-operations in verification can further be reduced to a logarithmic scale with preprocessing, as discussed in Section 3.3 and Section 3.4.

We also conducted a theoretical analysis and compared Σ-Check with [2]. Our study presents a performance evaluation of [2] in handling k instances, where each instance comprises a witness of size m, and h represents a function with a maximum degree of d. The outcomes are detailed in the table Table 3.

8.2 Implementation and experiments

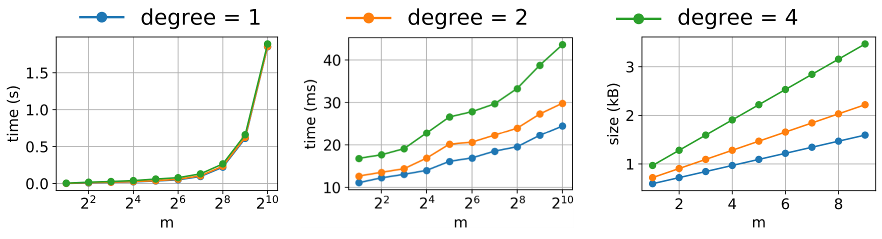
We implement Σ-Check in RUST,[3] with the core building block as a sum-check protocol for polynomials with coefficients in group elements. This project is built on the Arkworks [16] with adoptions of the finite field, elliptic curve, and

[3] `https://github.com/QMorning/Compressed-Sigma-Protocol-from-Sumcheck`.



(a) Prover time (b) Verifier time (c) Proof size

Fig. 3: Performance vs. instance number k



(a) Prover time (b) Verifier time (c) Proof size

Fig. 4: Performance vs. witness length m

multilinear polynomial libraries. It supports different elliptic curves and enables non-interactive protocol by utilizing the Merlin transcript [39].

To evaluate the experimental performance, we benchmark Σ-Check on a MacBook Pro (macOS Catalina, 2.064 GHz eight cores, M1, 16G memory). The commitment scheme is instantiated with the curve secp256k1, and the multi-threading feature is disabled. In the experiment, we consider two main factors, (i) the number of instances k (related to Π_{Amor}) and (ii) the size of a witness m (related to Π_{Comp}). We measure the performance of our protocol by three criteria, including the prover time, verifier time, and proof size. The experiment results are presented in Figure 3 and Figure 4. Specifically, for analyzing the first factor, we scale k from 2^1 to 2^{10} and fix $m = 2$. Figure 3 depicts the prover time, verifier time, and proof size increasing on k, indicating the asymptotic performance of the amortization phase. For the second factor, we scale m from 2^1 to 2^{10} and fix $k = 2$. Figure 4 indicates the asymptotic performance of the compression phase. Cases with various degrees are marked with different colors in the figures above. Note that the prover time in Figure 4a is dominated by $O(m)$ group operations in Π_{Comp}, resulting in a significantly greater concrete time compared to that shown in Figure 3a. Additionally, the prover's performance across different degrees exhibits minimal variation, as the $O(m)$ group operations are not influenced by the degree of the polynomial. For the proof size in 3c and 4c, the field and group elements are computed with sizes of 256 bits and 512 bits, respectively.

In conclusion, the experimental results keep consistent with the theoretical performance, demonstrating that Σ-Check can efficiently process multiple high-degree polynomial relations. In our future work, we will implement the protocol for more applications, including proof gadgets for binary proofs, partial knowledge proofs, and general-purpose argument systems with arithmetic circuits.

Acknowledgement

This research has received partial support from HK RGC GRF under Grants PolyU 15205624, 15202123, 15207522, and 15216721, NSFC Youth 62302418, and Ethereum Foundation ESP FY24-1497. We thank Kurt Pan for coming up with the Σ-Check name.

References

1. Attema, T.: Compressed Σ-protocol Theory. Ph.D. thesis, Leiden University (2023)
2. Attema, T., Cramer, R.: Compressed Σ-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics. In: Proc. of the Annual International Cryptology Conference (CRYPTO). pp. 513–543. Springer (2020), `https://doi.org/10.1007/978-3-030-56877-1_18`
3. Attema, T., Cramer, R., Fehr, S.: Compressing Proofs of k-out-of-n Partial Knowledge. In: Proc. of the Annual International Cryptology Conference (CRYPTO). pp. 65–91. Springer (2021), `https://doi.org/10.1007/978-3-030-84259-8_3`
4. Attema, T., Cramer, R., Kohl, L.: A Compressed Σ-Protocol Theory for Lattices. In: Proc. of the Annual International Cryptology Conference (CRYPTO). pp. 549–579. Springer (2021), `https://doi.org/10.1007/978-3-030-84245-1_19`
5. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable Zero Knowledge with no Trusted Setup. In: Proc. of the Annual International Cryptology Conference (CRYPTO). pp. 701–732. Springer (2019), `https://doi.org/10.1007/978-3-030-26954-8_23`
6. Boneh, D., Chen, B.: LatticeFold: A Lattice-based Folding Scheme and its Applications to Succinct Proof Systems. Cryptology ePrint Archive, Paper 2024/257 (2024), `https://eprint.iacr.org/2024/257`
7. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting. In: Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). pp. 327–357. Springer (2016), `https://doi.org/10.1007/978-3-662-49896-5_12`
8. Bootle, J., Chiesa, A., Sotiraki, K.: Sumcheck Arguments and Their Applications. In: Proc. of the Annual International Cryptology Conference (CRYPTO). pp. 742–773. Springer (2021), `https://doi.org/10.1007/978-3-030-84242-0_26`
9. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short Proofs for Confidential Transactions and More. In: Proc. of the IEEE Symposium on Security and Privacy (S&P). pp. 315–334. IEEE (2018), `https://doi.org/10.1109/SP.2018.00020`
10. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Proc. of the International Conference on the Theory and Application of Cryptology and Information Security (EUROCRYPT). pp. 677–706. Springer (2020), `https://doi.org/10.1007/978-3-030-45721-1_24`
11. Bünz, B., Maller, M., Mishra, P., Tyagi, N., Vesely, P.: Proofs for Inner Pairing Products and Applications. In: Proc. of the Annual International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT). pp. 65–97. Springer (2021), `https://doi.org/10.1007/978-3-030-92078-4_3`
12. Campanelli, M., Fiore, D., Querol, A.: LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs. In: Proc. of the ACM Conference on Computer & Communications Security (CCS). pp. 2075–2092. ACM (2019), `https://doi.org/10.1145/3319535.3339820`
13. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: Hyperplonk: Plonk with Linear-Time Prover and High-Degree Custom Gates. In: Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). pp. 499–530. Springer (2023), `https://doi.org/10.1007/978-3-031-30617-4_17`

14. Chen, S., Cheon, J.H., Kim, D., Park, D.: Verifiable Computing for Approximate Computation. Cryptology ePrint Archive, Paper 2019/762 (2019), https://eprint.iacr.org/2019/762
15. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: Preprocessing zksnarks with universal and updatable srs. In: Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39. pp. 738–768. Springer (2020), https://doi.org/10.1007/978-3-030-45721-1_26
16. arkworks contributors: arkworks zksnark ecosystem (2022), https://arkworks.rs
17. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical Verified Computation with Streaming Interactive Proofs. In: Proc. of the Innovations in Theoretical Computer Science Conference (ITCS). pp. 90–112 (2012), https://doi.org/10.1145/2090236.2090245
18. Cramer, R.: Modular Design of Secure yet Practical Cryptographic Protocols. Ph. D.-thesis, CWI and U. of Amsterdam **2** (1996)
19. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Proc. of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT). pp. 125–142. Springer (2002), https://doi.org/10.1007/3-540-36178-2_8
20. Daza, V., Ràfols, C., Zacharakis, A.: Updateable Inner Product Argument with Logarithmic Verifier and Applications. In: Proc. of the International Conference on Public-Key Cryptography (PKC). pp. 527–557. Springer (2020), https://doi.org/10.1007/978-3-030-45374-9_18
21. Dutta, M., Ganesh, C., Jawalkar, N.: Succinct Verification of Compressed Sigma Protocols in the Updatable SRS Setting. In: Proc. of the International Conference on Public-Key Cryptography (PKC). pp. 305–336. Springer (2024), https://doi.org/10.1007/978-3-031-57722-2_10
22. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). pp. 186–194. Springer (1986), https://doi.org/10.1007/3-540-47721-7_12
23. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive Arguments of Knowledge. Cryptology ePrint Archive, Paper 2019/953 (2019), https://eprint.iacr.org/2019/953
24. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic Span Programs and Succinct NIZKs without PCPs. In: Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). pp. 626–645. Springer (2013), https://doi.org/10.1007/978-3-642-38348-9_37
25. Kothapalli, A., Parno, B.: Algebraic Reductions of Knowledge. In: Proc. of the Annual International Cryptology Conference (CRYPTO). pp. 669–701. Springer (2023), https://doi.org/10.1007/978-3-031-38551-3_21
26. Kothapalli, A., Setty, S.: CycleFold: Folding-Scheme-based Recursive Arguments over a Cycle of Elliptic Eurves. Cryptology ePrint Archive, Paper 2023/1192 (2023), https://eprint.iacr.org/2023/1192
27. Kothapalli, A., Setty, S.: HyperNova: Recursive Arguments for Customizable Constraint Systems. Cryptology ePrint Archive, Paper 2023/573 (2023), https://eprint.iacr.org/2023/573
28. Lai, R.W., Malavolta, G., Ronge, V.: Succinct Arguments for Bilinear Group Arithmetic: Practical Structure-Preserving Cryptography. In: Proc. of the ACM Con-

ference on Computer & Communications Security (CCS). pp. 2057–2074. ACM (2019), https://doi.org/10.1145/3319535.3354262
29. Lipmaa, H., Siim, J., Zajac, M.: Counting vampires: from univariate sumcheck to updatable zk-snark. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 249–278. Springer (2022), https://doi.org/10.1007/978-3-031-22966-4_9
30. Liu, X., Gao, S., Zheng, T., Guo, Y., Xiao, B.: SnarkFold: Efficient Proof Aggregation from Incrementally Verifiable Computation and Applications. Cryptology ePrint Archive, Paper 2023/1946 (2023), https://eprint.iacr.org/2023/1946
31. Lund, C., Fortnow, L., Karloff, H., Nisan, N.: Algebraic Methods for Interactive Proof Systems. Journal of the ACM (JACM) **39**(4), 859–868 (1992), https://doi.org/10.1145/146585.146605
32. Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Proc. of the Annual International Cryptology Conference (CRYPTO). pp. 129–140. Springer (1991), https://doi.org/10.1007/3-540-46766-1_9
33. Setty, S.: Spartan: Efficient and General-Purpose zkSNARKs without Trusted Setup. In: Proc. of the Annual International Cryptology Conference (CRYPTO). pp. 704–737. Springer (2020), https://doi.org/10.1007/978-3-030-56877-1_25
34. Setty, S., Thaler, J., Wahby, R.: Customizable Constraint Systems for Succinct Arguments. Cryptology ePrint Archive, Paper 2023/552 (2023), https://eprint.iacr.org/2023/552
35. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation. In: Proc. of the Annual International Cryptology Conference (CRYPTO). pp. 733–764. Springer (2019), https://doi.org/10.1007/978-3-030-26954-8_24
36. Zhang, M., Chen, Y., Yao, C., Wang, Z.: 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). pp. 208–242. Springer (2023), https://doi.org/10.1007/978-981-99-8724-5_7
37. Zheng, T., Gao, S., Guo, Y., Xiao, B.: KiloNova: Non-Uniform PCD with Zero-Knowledge Property from Generic Folding Schemes. Cryptology ePrint Archive, Paper 2023/1579 (2023), https://eprint.iacr.org/2023/1579
38. Zheng, T., Gao, S., Song, Y., Xiao, B.: Leaking Arbitrarily Many Secrets: Any-out-of-Many Proofs and Applications to RingCT Protocols. In: Proc. of the IEEE Symposium on Security and Privacy (S&P). pp. 2533–2550. IEEE (2023), https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.10179292
39. zkcrypto: Merlin transcript. https://merlin.cool/ (2022)

A Detailed Definitions and Proofs

A.1 Definitions for Pedersen Commitments

Definition 4 (Computationally binding). *A commitment scheme is computationally binding if for all Probabilistic Polynomial Time (PPT) adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{c}\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda), (\vec{f_0}, \vec{f_1}, r_0, r_1) \leftarrow \mathcal{A}(\mathsf{ck}) : \\ \mathsf{Commit}(\mathsf{ck}; \vec{f_0}, r_0) = \mathsf{Commit}(\mathsf{ck}; \vec{f_1}, r_1), (\vec{f_0}, r_0) \neq (\vec{f_1}, r_1)\end{array}\right] \approx 0.$$

Definition 5 (Perfectly hiding). *A commitment scheme is perfectly hiding if for all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c}\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda), (\vec{f_0}, \vec{f_1}) \leftarrow \mathcal{A}(\mathsf{ck}), b \leftarrow\!\!\$\ \{0,1\}, \\ F \leftarrow \mathsf{Commit}(\mathsf{ck}; \vec{f_b}) : \mathcal{A}(F) = b.\end{array}\right] = \frac{1}{2}.$$

A.2 Definitions for $\varSigma$-Protocol

Definition 6 (Perfect completeness). *A $\varSigma$-protocol $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}$ provides perfect completeness if for all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), (x; w) \leftarrow \mathcal{A}(\mathsf{pp}) : \langle \mathcal{P}(\mathsf{pp}, x, w), \mathcal{V}(\mathsf{pp}, x)\rangle = 1 \vee (x; w) \notin \mathcal{R}\right] = 1.$$

Definition 7 (Computational Knowledge Soundness). *A $\varSigma$-protocol $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}$ provides soundness with soundness error σ if for all deterministic polynomial time $\mathcal{P}^*$ with success probability ϵ, there exists an expected polynomial time extractor $\mathcal{E}$ such that for all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), (x; w) \leftarrow \mathcal{A}(\mathsf{pp}), \langle \mathcal{P}(\ maths fpp, x, w), \mathcal{V}(\mathsf{pp}, x)\rangle = 1, \\ w \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{pp}, x) : \wedge (x; w) \notin \mathcal{R}\end{array}\right] \geq \frac{\epsilon - \kappa(|x|)}{\mathsf{poly}(|x|)}.$$

where $\kappa(|x|)$ is negligible soundness error dependent on the statement length $|x|$.

For defining special soundness property, we first present a denotation of the tree of transcript.

Definition 8 (Tree of transcript). *Let $\mu \in \mathbb{N}$ and $(k_1, ..., k_\mu) \in \mathbb{N}^\mu$. A $(k_1, ..., k_\mu)$-tree of transcripts constitutes a set of $\prod_{i=1}^{\mu} k_i$ transcripts of a tree-like structure. The edges within this tree represent the challenges of the verifier, while the vertices are the messages from the prover, which can be empty. Each node at depth i has exactly k_i child nodes, corresponding to k_i distinct challenges. Every transcript is uniquely represented by one path from the root node to a leaf node.*

Definition 9 ($(k_1, ..., k_\mu)$-special soundness). *$\varPi$ provides $(k_1, ..., k_\mu)$-special soundness, if there is an effective PPT extraction algorithm $\mathcal{E}$ that is capable of extracting the witness w given x and any $(k_1, \cdots, k_\mu)$-tree of accepting transcripts T [2] (defined above). Specifically, for all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), (x, T) \leftarrow \mathcal{A}(\mathsf{pp}), w \leftarrow \mathcal{E}(\mathsf{pp}, x, T) : (x; w) \in \mathcal{R}\right] \approx 1.$$

$(k_1, ..., k_\mu)$-special soundness is a generalization to the standard notion of special soundness. For example, a typical $\varSigma$-protocol given in [18] is a special type of the 3-move interactive proof satisfying k-special soundness, where $\mathcal{P}$ sends an initial message a, $\mathcal{V}$ issues with a random challenge $r \leftarrow\!\!\$\ \mathbb{F}$, and finally, $\mathcal{P}$ provides with a response z. Compared to knowledge soundness property, special soundness is typically easier to prove for an interactive proof. Although special soundness should be regarded as a weaker notion of knowledge soundness, [2] proves that $(k_1, ..., k_\mu)$-special-soundness tightly implies knowledge soundness as long as $K = \prod_{i=1}^{\mu} k_i$ is constant. Therefore, the protocols in this work are all arguments of knowledge under the DL assumption from the results of [4].

Definition 10 (Special honest verifier zero-knowledge). *A Σ-protocol $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}$ provides special Honest Verifier Zero-Knowledge (special HVZK) if there exists an efficient PPT simulator $\mathcal{S}$ capable of generating an accepting transcript that is indistinguishable from real transcripts. Specifically, for all PPT adversaries $\mathcal{A}_1, \mathcal{A}_2$*

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), (x, w, \rho) \leftarrow \mathcal{A}_2(\mathsf{pp}), tr \leftarrow \langle \mathcal{P}(\mathsf{pp}, x, w), \mathcal{V}(\mathsf{pp}, x; \rho)\rangle : \\ (x; w) \in \mathcal{R} \wedge \mathcal{A}_1(tr) = 1\end{array}\right]$$
$$\approx \Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), (x, w, \rho) \leftarrow \mathcal{A}_2(\mathsf{pp}), tr \leftarrow \mathsf{S}(\mathsf{pp}, x, \rho) : \\ (x; w) \in \mathcal{R} \wedge \mathcal{A}_1(tr) = 1\end{array}\right].$$

A.3 Definitions for RoK

Definition 11 (Perfect completeness). *Π has perfect completeness if for all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), (x_1; w_1) \leftarrow \mathcal{A}(\mathsf{pp}), (x_1; w_1) \in \mathcal{R}_1, \\ (\mathsf{tr}, x_2, w_2) \leftarrow \langle \mathcal{P}(\mathsf{pp}, x_1, w_1), \mathcal{V}(\mathsf{pp}, x_1)\rangle : (x_2; w_2) \in \mathcal{R}_2\end{array}\right] = 1.$$

Definition 12 (Knowledge soundness). *Π has knowledge soundness if for every expected polynomial-time adversary $\mathcal{A}$ and malicious prover $\mathcal{P}^*$, there is an expected polynomial-time extractor $\mathcal{E}$ (*tr *is omitted)*

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), (x_1, w_1^*) \leftarrow \mathcal{A}(\mathsf{pp}), \langle \mathcal{P}^*(\mathsf{pp}, x_1, w_1^*), \mathcal{V}(\mathsf{pp}, x_1)\rangle \in \mathcal{R}_2 : \\ \big(x_1; \mathcal{E}(\mathsf{pp}, x_1, w_1^*)\big) \in \mathcal{R}_1\end{array}\right] \approx 1.$$

A.4 Proof of Theorem 3

Proof. Public reducibility. Given k public statements $(F_j, \vec{v}_j)$ for all $j \in [k]$ and a transcript that includes $\vec{\beta}, \vec{r}_x, F, s$, one can compute $\vec{v} := \big(\tilde{v}(\vec{r}_x, \vec{y})\big)_{\vec{y} \in \{0,1\}^{\log m}}$. If the final checks pass, $(F, s, \vec{v}, \vec{\beta})$ is an output.

Completness. Given k input instances $(F_j, \vec{v}_j; \vec{f}_j) \in \mathcal{R}_{\mathsf{Poly}}$ for all $j \in [k]$ that are maliciously chosen by the adversary, $\mathcal{P}$ and $\mathcal{V}$ honestly run the protocol. The sum-check protocols will not abort due to the completeness of sum-check. Additionally, $F = \sum_{i=1}^m f_i G_i$ and $s = \sum_{i=1}^m \mathsf{eq}\big(\vec{\beta}, \mathsf{Bits}(i)\big)\big(h(f_i) - v_i\big)$ hold based on the definitions of F and s in Equation (10) and the definitions of g_1 and g_2. Therefore, $(F, s, \vec{v}, \vec{\beta}; \vec{f}) \in \mathcal{R}_{\mathsf{AmorPoly}}$.

Knowledge soundness. Given k input public statements $(F_j, \vec{v}_j)$ for all $j \in [k]$ that are maliciously chosen by the adversary and a malicious prover $\mathcal{P}^*$, the extractor $\mathcal{E}$ runs Π_{Amor} with $\mathcal{P}^*$ as an honest $\mathcal{V}$ with different $\vec{r}_x^{(i)}$'s for all $i \in [k]$. If $\mathcal{V}$ rejects, $\mathcal{E}$ aborts. Otherwise, denote the output as $(F^{(i)}, s^{(i)}, \vec{v}^{(i)}, \vec{\beta}; \vec{f}^{(i)})$ for all $i \in [k]$. Since the matrix $\big(\mathsf{eq}(\vec{0}, \vec{r}_x^{(i)}), \cdots, \mathsf{eq}(\vec{1}, \vec{r}_x^{(i)})\big)_{i=1}^k$ is invertible with overwhelming probability, $\mathcal{E}$ can extract $\vec{f}_j$'s for all $j \in [k]$ by solving the following equations

$$\sum_{j=1}^k \mathsf{eq}(\mathsf{Bits}(j), \vec{r}_x^{(i)}) \cdot \vec{f}_j = \vec{f}^{(i)}, \quad \forall i \in [k]. \tag{36}$$

Now we argue that if $\mathcal{E}$ does not abort and $(F, s, \vec{v}, \vec{\beta}; \vec{f}) \in \mathcal{R}_{\mathsf{AmorPoly}}$, then $(F_j, \vec{v}_j; \vec{f}_j) \in \mathcal{R}_{\mathsf{Poly}}$ for all $j \in [k]$. Conduct a MLE on the extracted $\vec{f}_j$'s to derive $\tilde{f}(\vec{X}, \vec{Y})$. Since the checks on Equation (11) pass, by the soundness of the sum-check, Equation (9) holds with overwhelming probability over the sum-check challenges $\vec{r}_x^{(i)}$'s. Define $\hat{g}_1$ and $\hat{g}_2$ as

$$\begin{aligned} \hat{g}_1(\vec{X}) &:= \tilde{F}(\vec{X}) - \sum_{\vec{y} \in \{0,1\}^{\log m}} \tilde{f}(\vec{X}, \vec{y}) \cdot \tilde{G}(\vec{y}), \\ \hat{g}_2(\vec{X}, \vec{Y}) &:= h\big(\tilde{f}(\vec{X}, \vec{Y})\big) - \tilde{v}(\vec{X}, \vec{Y}). \end{aligned} \tag{37}$$

By the uniqueness of MLE, we have $\hat{g}_1(\vec{\alpha}) = \sum_{\vec{x} \in \{0,1\}^{\log k}} g_1(\vec{x})$ and $\hat{g}_2(\vec{\alpha}, \vec{\beta}) = \sum_{\vec{x} \in \{0,1\}^{\log k}, \vec{y} \in \{0,1\}^{\log m}} g_2(\vec{x}, \vec{y})$. Based on the definitions of g_1 and g_2, we can rewrite Equation (9) as

$$\sum_{\vec{x} \in \{0,1\}^{\log k}} g_1(\vec{x}) = \hat{g}_1(\vec{\alpha}) = \mathcal{O}, \quad \sum_{\substack{\vec{x} \in \{0,1\}^{\log k}, \\ \vec{y} \in \{0,1\}^{\log m}}} g_2(\vec{x}, \vec{y}) = \hat{g}_2(\vec{\alpha}, \vec{\beta}) = 0. \tag{38}$$

Since $\vec{\alpha}, \vec{\beta}$ are uniformly chosen from the challenge space, by the Schwartz-Zippel lemma, we have $\hat{g}_1(\vec{X}) = \mathcal{O}$ and $\hat{g}_2(\vec{X}, \vec{Y}) = 0$ with overwhelming probability. Accordingly, we have

$$\begin{aligned} \tilde{F}(\vec{X}) = \sum_{\vec{y} \in \{0,1\}^{\log m}} \tilde{f}(\vec{X}, \vec{y}) \cdot \tilde{G}(\vec{y}) \quad &\Longrightarrow \quad F_j = \sum_{i=1}^{m} f_{j,i} G_i, \quad \forall j \in [k], \\ h\big(\tilde{f}(\vec{X}, \vec{Y})\big) = \tilde{v}(\vec{X}, \vec{Y}) \quad &\Longrightarrow \quad h(f_{j,i}) = v_{j,i}, \quad \forall i \in [m], j \in [k]. \end{aligned} \tag{39}$$

Thus, $(F_j, \vec{v}_j; \vec{f}_j) \in \mathcal{R}_{\mathsf{Poly}}$ for all $j \in [k]$. □

Denote the verification version of Π_{Amor} as Π_{AmorV}, which modifies the last two steps of the protocol by having $\mathcal{P}$ send $\vec{f}$ and $\mathcal{V}$ check $(F, s, \vec{v}, \vec{\beta}; \vec{f}) \in \mathcal{R}_{\mathsf{AmorPoly}}$. We have the following theory.

Theorem 6. *Π_{Amor} satisfies the special HVZK property, if the input instances include at least one masking instance (i.e., the public statement of the masking instance can be simulated).*

The proof follows the same logic as the result in [35], except we simulate with $\mathcal{R}_{\mathsf{Poly}}$ instances instead of polynomials.

Proof. We build the simulator $\mathcal{S}$ that compute a satisfying instance $(F, s, \vec{v}, \vec{\beta}; \vec{f}) \in \mathcal{R}_{\mathsf{AmorPoly}}$ as follows. Without loss of generality, we regard $(F_k, \vec{v}_k; \vec{f}_k)$ as the masking instance.

- Receive $\vec{\alpha}, \vec{\beta}$ from $\mathcal{V}$.
- Sample $\vec{f}_j \leftarrow\$ \mathbb{F}^m$ for all $j \in [k]$.

– Compute $\vec{v}_k \in \mathbb{F}^m$ based on the following equation. If there are multiple solutions, choose one at random.

$$\sum_{\vec{y}\in\{0,1\}^{\log m}} \Big(\mathsf{eq}(\vec{\beta},\vec{y}) \cdot \Big(h(\tilde{f}(\vec{\alpha},\vec{y})) - \tilde{v}(\vec{\alpha},\vec{y})\Big)\Big) = 0.$$

– Compute F_k based on the following equation.

$$\tilde{F}(\vec{\alpha}) - \sum_{\vec{y}\in\{0,1\}^{\log m}} \tilde{f}(\vec{\alpha},\vec{y}) \cdot \tilde{G}(\vec{y}) = \mathcal{O}\,.$$

– Engage in two sum-checks in parallel with $\mathcal{V}$ to show Equation (9) holds.
– Compute F, s based on Equation (10) and step 6.
– Output the simulated public statement (F_k, v_k), and the simulated transcript including $(\vec{\alpha}, \vec{\beta}, F, s, \vec{f})$ and the interactions of the sum-check.

Next, we argue $\mathcal{V}$ does not abort and accepts $(F, s, \vec{v}, \vec{\beta}; \vec{f}) \in \mathcal{R}_{\mathsf{AmorPoly}}$. Since Equation (A.4) and Equation (A.4) holds, by the uniqueness of MLE, Equation (9) holds. Therefore, based on the completeness of sum-check, $\mathcal{V}$ does not abort at step 2, and the check on Equation (11) passes. By the definitions of F, s in Equation (10), $\sum_{i=1}^{m} f_i G_i = F$ and $\sum_{i=1}^{m} \mathsf{eq}\big(\vec{\beta}, \mathsf{Bits}(i)\big)\big(h(f_i) - v_i\big) = s$ hold. Thus, $(F, s, \vec{v}, \vec{\beta}; \vec{f}) \in \mathcal{R}_{\mathsf{AmorPoly}}$.

Finally, the distributions of challenges are identical to the real ones. The simulated $F_0, v_0, F, s, \vec{f}$ are indistinguishable from the real ones. □

A.5 Proof of Theorem 4

Proof. Completeness. Given a maliciously chosen input instance $(F, s, \vec{v}, \vec{\beta}; \vec{f}) \in \mathcal{R}_{\mathsf{AmorPoly}}$, $\mathcal{P}$ and $\mathcal{V}$ honestly run the protocol. The sum-check protocols will not abort due to the completeness of sum-check. Additionally, $F_p = \sum_{i=1}^{m} f_i G_i$ and $s_p = \sum_{i=1}^{m} \mathsf{eq}\big(\vec{\beta}, \mathsf{Bits}(i)\big)\big(h(f_i) - v_i\big)$ hold based on the definitions of F_p and s_p in Equation (13) and the definitions of p_1 and p_2 in Equation (14). Accordingly, Equation (16) holds, and the check at step 4 passes.

Knowledge soundness. Given a maliciously chosen public statement $(F, s, \vec{v}, \vec{\beta})$ and a malicious prover $\mathcal{P}^*$. The extractor $\mathcal{E}$ runs Π_{Comp} with $\mathcal{P}^*$ as an honest $\mathcal{V}$ with different $\vec{r}_y^{(i)}$'s for all $i \in [m]$ and the same H. If $\mathcal{V}$ rejects, $\mathcal{E}$ aborts. Otherwise, denote $F_p^{(i)} := p_1(\vec{r}_y^{(i)}), s_p^{(i)} := p_2(\vec{r}_y^{(i)})$ at step 2, $f^{(i)} := \tilde{f}(\vec{r}^{(i)})$ at step 3, and $e_y^{(i)} := \mathsf{eq}(\vec{\beta}, \vec{r}_y^{(i)}), G^{(i)} := \tilde{G}(\vec{r}_y^{(i)}), v^{(i)} := \tilde{v}(\vec{r}_y^{(i)})$ at step 4. Since the matrix $\big(\mathsf{eq}(\vec{0}, \vec{r}_y^{(i)}), \cdots, \mathsf{eq}(\vec{1}, \vec{r}_y^{(i)})\big)_{i=1}^{m}$ is invertible with overwhelming probability, $\mathcal{E}$ can extract $\vec{f} = (f_1, ...f_m)$ by solving the following equations

$$\sum_{j=1}^{m} \mathsf{eq}(\mathsf{Bits}(j), \vec{r}_y^{(i)}) \cdot f_j = f^{(i)}, \quad \forall i \in [m]. \tag{40}$$

Now we argue that if $\mathcal{E}$ does not abort, then $(F, s, \vec{v}, \vec{\beta}; \vec{f}) \in \mathcal{R}_{\mathsf{AmorPoly}}$. First, observe that $\mathcal{R}_{\mathsf{AmorPoly}}$ is a binding relation, then f at step 3 must be evaluated based on $\tilde{f}$. This is formally proved in Lemma 3. Therefore, the whole sum-check protocol is correct. Based on the soundness of the sum-check, Equation (15) holds with overwhelming probability over the sum-check challenges $\vec{r}_y^{(i)}$'s. Furthermore, recall the definition of p. Given that H is uniformly sampled from the verifier, by the Schwartz-Zippel lemma, it follows with an overwhelming probability that:

$$\sum_{\vec{y}\in\{0,1\}^{\log m}} p_1(\vec{y}) = F, \qquad \sum_{\vec{y}\in\{0,1\}^{\log m}} p_2(\vec{y}) = s. \tag{41}$$

By the definitions of p_1 and p_2 in Equation (14), we have

$$\begin{aligned} F &= \sum_{\vec{y}\in\{0,1\}^{\log m}} \tilde{f}(\vec{y})\cdot\tilde{G}(\vec{y}) = \sum_{i=1}^{m} f_i G_i, \\ s &= \sum_{\vec{y}\in\{0,1\}^{\log m}} \mathsf{eq}(\vec{\beta},\vec{y})\cdot\big(h\big(\tilde{f}(\vec{y})-\tilde{v}(\vec{y})\big)\big) = \sum_{i=1}^{m} \mathsf{eq}\big(\vec{\beta},\mathsf{Bits}(i)\big)\big(h(f_i)-v_i\big). \end{aligned} \tag{42}$$

Therefore, $(F, s, \vec{v}, \vec{\beta}; \vec{f}) \in \mathcal{R}_{\mathsf{AmorPoly}}$. □

We present a lemma to ensure the binding of the evaluation claim within a sum-check argument, which is similar to Theorem 3 in [8], but we focus on the binding property. A sum-check friendly commitment satisfies the following relation:

$$\mathcal{R}_{\mathsf{SC}} := \Big\{(S\in\mathbb{G}; \vec{f}\in\mathbb{F}^m) : \sum_{\vec{y}\in\{0,1\}^{\log m}} p\big(\tilde{f}(\vec{y}),\tilde{G}(\vec{y})\big) = S\Big\}, \tag{43}$$

where $\vec{G}\in\mathbb{G}^m$ is the commitment key ($\tilde{G}$ is the MLE on $\vec{G}$) and $p:\mathbb{F}\times\mathbb{G}\to\mathbb{G}$ is a polynomial. Assuming that $\mathcal{R}_{\mathsf{SC}}$ is a binding relation, let us denote a sum-check argument for $\mathcal{R}_{\mathsf{SC}}$ as Π_{SC} (similar to Π_{Comp} in Protocol 3, except the polynomial p is in a generalized form). We have the following lemma.

Lemma 3. *If the verifier accepts in Π_{SC}, then the final evaluation claim (analogous to f at step 3 in Protocol 3) must be evaluated based on the MLE of the original witness.*

Proof. Regard S as a binding commitment of the witness $\vec{f}$ under the commitment key $\vec{G}$. Denote the sum-check challenges as $\vec{r}$ and the polynomials sent during the sum-check protocol as p_i for all $i\in[\log m]$. Based on the definition of p in Equation (43), we have

$$p_i(Y) = \sum_{\vec{y}\in\{0,1\}^{\log m - i}} p\big(\tilde{f}(r_1,\cdots,r_{i-1},Y,\vec{y}),\tilde{G}(r_1,\cdots,r_{i-1},Y,\vec{y})\big). \tag{44}$$

Let $\vec{G}[r_1,\cdots,r_{i-1}]\in\mathbb{G}^{m/2^{i-1}}$ be the coefficients of $\tilde{G}(r_1,\cdots,r_{i-1},Y_i,\cdots,Y_{\log m})$ and $\vec{f}[r_1,\cdots,r_{i-1}]\in\mathbb{F}^{m/2^{i-1}}$ be the coefficients of $\tilde{f}(r_1,\cdots,r_{i-1},Y_i,\cdots,Y_{\log m})$. In each round, the verifier's check implies that

– If $i > 1$, then $p_i(0) + p_i(1) = p_{i-1}(r_{i-1})$. Thus, $\vec{f}[r_1, \cdots, r_{i-1}]$ is an opening to $p_{i-1}(r_{i-1})$ under commitment key $\vec{G}[r_1, \cdots, r_{i-1}]$.
– If $i = 1$, then $p_1(0)+p_1(1) = S$. Thus, $\vec{f}$ is an opening to S under commitment key $\vec{G}$.

Therefore, based on the binding property of the relation, the evaluation claim must be evaluated based on an MLE of the witness $\vec{f}$, that is, $\tilde{f}(\vec{r})$. □

B Protocols under Different Assumptions

We highlight the differences between DL-based protocols in red.

B.1 Integer polynomial commitment

The amortization protocol $\Pi^{\mathsf{RSA}}_{\mathsf{Amor}}$ and compression protocol $\Pi^{\mathsf{RSA}}_{\mathsf{Comp}}$ are depicted in Protocol 6 and Protocol 7, respectively.

Protocol 6 $\Pi^{\mathsf{RSA}}_{\mathsf{Amor}}$: Reduce $(\mathcal{R}^{\mathsf{RSA}}_{\mathsf{Poly}})^k$ to $\mathcal{R}^{\mathsf{RSA}}_{\mathsf{AmorPoly}}$.

$\mathcal{P}\big(G, H, (F_j, \vec{v}_j, \vec{f}_j, \gamma_j)_{j=1}^k\big), \mathcal{V}\left(G, H, (F_j, \vec{v}_j)_{j=1}^k\right)$

1: $\mathcal{V} \to \mathcal{P}$: $\vec{\alpha} \leftarrow_\$ \mathbb{Z}^{\log k}, \vec{\beta} \leftarrow_\$ \mathbb{Z}^{\log m}$.

2: $\mathcal{P}$ and $\mathcal{V}$: Compute $\tilde{f}(\vec{X}, \vec{Y})$ with $\vec{f}_j$'s and set $g_2(\vec{X})$. Then engage in a bounded sum-check for the claim

$$\sum_{\vec{x}\in\{0,1\}^{\log k}} g_2(\vec{x}) = 0. \tag{45}$$

The protocol reduces to an evaluation claim $g_2(\vec{r}_x) = s_g$, where $\vec{r}_x \leftarrow_\$ \mathbb{Z}^{\log k}$ are the challenges of the sum-check.

3: $\mathcal{P} \to \mathcal{V}$: s which are defined as

$$s := \sum_{\vec{y}\in\{0,1\}^{\log m}} \Big(\mathsf{eq}(\vec{\beta}, \vec{y}) \cdot \Big(h\big(\tilde{f}(\vec{r}_x, \vec{y})\big) - \tilde{v}(\vec{r}_x, \vec{y})\Big)\Big). \tag{46}$$

4: $\mathcal{V}$: Compute $e_x := \mathsf{eq}(\vec{\alpha}, \vec{r}_x)$ and $F := \tilde{F}(\vec{r}_x)$. Check the following equation

$$s_g = e_x \cdot s. \tag{47}$$

5: $\mathcal{P}$: Output $\vec{f} := \big(\tilde{f}(\vec{r}_x, \vec{y})\big)_{\vec{y}\in\{0,1\}^{\log m}}, \gamma := \tilde{\gamma}(\vec{r}_x)$.

6: $\mathcal{V}$: Output $F, s, b := (\frac{p+1}{2})^{\log k}, \vec{v} := \big(\tilde{v}(\vec{r}_x, \vec{y})\big)_{\vec{y}\in\{0,1\}^{\log m}}, \vec{\beta}$.

Protocol 7 $\mathit{\Pi}^{\mathsf{RSA}}_{\mathsf{Comp}}$: Check $\mathcal{R}^{\mathsf{RSA}}_{\mathsf{AmorPoly}}$ with compression.

$\mathcal{P}(G, H, F, s, b, \vec{v}, \vec{\beta}, \vec{f}, \gamma), \mathcal{V}(G, H, F, s, b, \vec{v}, \vec{\beta})$

```
1: $\mathcal{V} \to \mathcal{P}$: $r \leftarrow\!\!\$\ \mathbb{Z}_p$.
2: $\mathcal{P}$ and $\mathcal{V}$: Set $\tilde{q}(\vec{Y}) := \sum_{i=1}^{m} \mathsf{eq}(\mathsf{Bits}(i), \vec{Y}) \cdot q^{i-1}$, $p(\vec{y}) := p_1(\vec{y}) + r \cdot p_2(\vec{y})$
   where $p_1(\vec{Y}) := \tilde{f}(\vec{Y}) \cdot \tilde{q}(\vec{Y})$ and $p_2(\vec{Y})$ is defined in Equation (14). Engage
   in a sum-check for the claim
       $$G^{\sum_{\vec{y} \in \{0,1\}^{\log m}} p(\vec{y})} H^{\gamma} = F \cdot G^{r \cdot s}. \quad (48)$$
   The protocol reduces to two evaluation claims $G^{p_1(\vec{r}_y)} = F_p$ and $p_2(\vec{r}_y) = s_p$,
   where $\vec{r}_y \leftarrow\!\!\$\ \mathbb{Z}_p^{\log m}$ are the challenges of the sum-check.
3: $\mathcal{P} \to \mathcal{V}$: $f := \tilde{f}(\vec{r}_y)$.
4: $\mathcal{V}$: Compute $e_y := \mathsf{eq}(\vec{\beta}, \vec{r}_y)$ and $v := \tilde{v}(\vec{r}_y)$. Check the following equation
       $$F_p \cdot G^{r \cdot s_p} = G^{f + r \cdot e_y (h(f) - v)} H^{\gamma}, \quad |f| < b \cdot \left(\frac{p+1}{2}\right)^{\log m} < \frac{q}{2}. \quad (49)$$
```

B.2 GIPA-based commitment

Let $\mathbb{M}_k$ and $\mathbb{M}_m$ denote the modules of the commitment key space and message space, respectively, which share a same scalar ring $\mathfrak{R}$. The amortization protocol $\mathit{\Pi}^{\mathsf{GIPA}}_{\mathsf{Amor}}$ and compression protocol $\mathit{\Pi}^{\mathsf{GIPA}}_{\mathsf{Comp}}$ are depicted in Protocol 8 and Protocol 9, respectively.

Protocol 8 $\Pi^{\mathsf{GIPA}}_{\mathsf{Amor}}$: Reduce $(\mathcal{R}^{\mathsf{GIPA}}_{\mathsf{Poly}})^k$ to $\mathcal{R}^{\mathsf{GIPA}}_{\mathsf{AmorPoly}}$.

$\mathcal{P}\big(\vec{\mathsf{ck}}, (F_j, \vec{v}_j, \vec{f}_j)_{j=1}^k\big), \mathcal{V}\left(\vec{\mathsf{ck}}, (F_j, \vec{v}_j)_{j=1}^k\right)$

1: $\mathcal{V} \to \mathcal{P}$: $\vec{\alpha} \leftarrow\!\!\$\; \mathfrak{R}^{\log k}, \vec{\beta} \leftarrow\!\!\$\; \mathfrak{R}^{\log m}$.

2: $\mathcal{P}$ and $\mathcal{V}$: Compute $\tilde{f}(\vec{X}, \vec{Y})$ as with $\vec{f}_j$'s and set $g_2(\vec{X})$ as with Equation (7). Then engage in a sum-check with the same challenges for the claim

$$\sum_{\vec{x} \in \{0,1\}^{\log k}} g_2(\vec{x}) = 0. \tag{50}$$

The protocol reduces to an evaluation claim $g_2(\vec{r}_x) = s_g$, where $\vec{r}_x \leftarrow\!\!\$\; \mathfrak{R}^{\log k}$ are the challenges of the sum-check.

3: $\mathcal{P} \to \mathcal{V}$: s which are defined as

$$s := \sum_{\vec{y} \in \{0,1\}^{\log m}} \left(\mathsf{eq}(\vec{\beta}, \vec{y}) \cdot \left(h\big(\tilde{f}(\vec{r}_x, \vec{y})\big) - \tilde{v}(\vec{r}_x, \vec{y})\right)\right). \tag{51}$$

4: $\mathcal{V}$: Compute $e_x := \mathsf{eq}(\vec{\alpha}, \vec{r}_x)$ and $F := \tilde{F}(\vec{r}_x)$. Check the following equations

$$s_g = e_x \cdot s. \tag{52}$$

5: $\mathcal{P}$: Output $\vec{f} := \left(\tilde{f}(\vec{r}_x, \vec{y})\right)_{\vec{y} \in \{0,1\}^{\log m}}$.

6: $\mathcal{V}$: Output $F, s, \vec{v} := \left(\tilde{v}(\vec{r}_x, \vec{y})\right)_{\vec{y} \in \{0,1\}^{\log m}}, \vec{\beta}$.

Protocol 9 $\Pi^{\mathsf{GIPA}}_{\mathsf{Comp}}$: Check $\mathcal{R}^{\mathsf{GIPA}}_{\mathsf{AmorPoly}}$ with compression.

$\mathcal{P}(\vec{\mathsf{ck}}, F, s, \vec{v}, \vec{\beta}, \vec{f}), \mathcal{V}(\vec{\mathsf{ck}}, F, s, \vec{v}, \vec{\beta})$

1: $\mathcal{V} \to \mathcal{P}$: $\mathsf{ck}_H \leftarrow\!\!\$\; \mathbb{M}_k$.

2: $\mathcal{P}$ and $\mathcal{V}$: Set $p(\vec{y}) := p_1(\vec{y}) + \mathsf{CM}(\mathsf{ck}_H, p_2(\vec{y}))$ where $p_1(\vec{y}) := \mathsf{CM}(\tilde{\mathsf{ck}}(\vec{y}), \tilde{f}(\vec{y}))$ and $p_2(\vec{Y})$ is defined in Equation (14). Engage in a sum-check for the claim

$$\sum_{\vec{y} \in \{0,1\}^{\log m}} p(\vec{y}) = F + \mathsf{CM}(\mathsf{ck}_H, s). \tag{53}$$

The protocol reduces to two evaluation claims $p_1(\vec{r}_y) = F_p$ and $p_2(\vec{r}_y) = s_p$, where $\vec{r}_y \leftarrow\!\!\$\; \mathfrak{R}^{\log m}$ are the challenges of the sum-check.

3: $\mathcal{P} \to \mathcal{V}$: $f := \tilde{f}(\vec{r}_y)$.

4: $\mathcal{V}$: Compute $e_y := \mathsf{eq}(\vec{\beta}, \vec{r}_y)$, $\mathsf{ck} := \tilde{\mathsf{ck}}(\vec{r}_y)$, and $v := \tilde{v}(\vec{r}_y)$. Check the following equation

$$F_p + \mathsf{CM}(\mathsf{ck}_H, s_p) = \mathsf{CM}\big(\mathsf{ck}, f\big) + \mathsf{CM}\big(\mathsf{ck}_H, e_y \cdot (h(f) - v)\big). \tag{54}$$