Multivariate Multi-Polynomial Commitment and its Applications

Xiao Yang¹, Chengru Zhang², Mark Ryan¹, and Gao Meng¹

¹ University of Birmingham

Abstract. We introduce and formally define Multivariate Multi-Polynomial (MMP) commitment, a commitment scheme on multiple multivariate polynomials, and illustrate the concept with an efficient construction, which enjoys constant commitment size and logarithmic proof size. We further enhance our MMP scheme to achieve the zero-knowledge property.

Additionally, combined with a novel zero-knowledge range proof for Pedersen subvector commitment, we present a Zero-Knowledge Range Proof (ZKRP) for MMP commitment.

We present two sample applications. Firstly, our MMP commitment can be used for efficient aggregation of SNARK based on multivariate polynomial commitments. As a showcase, we apply MMP commitment to HyperPlonk and refer to this variant of HyperPlonk as aHyperPlonk. For k instances, each with circuit size n, the communication and verification complexity is reduced from $O(k \cdot \log n)$ to $O(\log k + \log n)$, while the prover complexity remains the same. Secondly, we propose a novel zeroknowledge proof for vehicle GPS traces based on ZKRP for MMP, which allows vehicle owners to prove if a vehicle has/hasn't passed through some location during a specific time interval.

Keywords: Polynomial Commitment · Zero-Knowledge Range Proof · SNARK.

1 Introduction

Polynomial commitment schemes allow a prover to commit to a polynomial and later convince the verifier of correct evaluations of the committed polynomial. It has recently been widely used to compile Interactive Oracle Proofs (IOPs) [7] into efficient cryptographic arguments [48,27,34,55,26,23]. The paradigm of polynomial commitment in cryptographic arguments follows a commit-and-prove structure, where the prover commits to a set of polynomials and later proves the correctness of the evaluation of these polynomials at random points. This scenario underscores the importance of constructing polynomial commitment schemes where the proof size and verification time are sublinear in the number of polynomials. Motivated by this, we propose a Multivariate Multi-Polynomial (MMP) commitment scheme that allows for the commitment of multiple multivariate polynomials with constant commitment size and logarithmic proof size and verification time, *w.r.t.* the number of polynomials. Our MMP scheme can be applied to SNARKs with multivariate polynomial commitment, e.g., Spartan [55] and HyperPlonk [26], to enable efficient aggregation of instances.

Apart from SNARK aggregation, we have also developed a novel application scenario for MMP commitment: a zero-knowledge proof system for vehicle GPS traces. In this context, a prover can demonstrate that a vehicle has/has not passed through a specific location without disclosing the full driving trajectory. The scheme essentially functions as a Zero-Knowledge Range Proof (ZKRP) for a zero-knowledge MMP commitment scheme. To accomplish this, we enhanced the MMP commitment scheme to be zero-knowledge and introduced a novel ZKRP for Pedersen subvector commitment [37].

The rest of the paper is organized as follows. We summarize our contribution and related work in Sections 1.1 and 1.2. Section 2 introduces notations and building blocks. In Section 3, we formally define MMP in Section 3.1 and describe the construction of the MMP commitment in Section 3.2. We analyze the security and evaluate the efficiency of the MMP commitment in Section 3.3 and 3.4. Section 3.5 provides an enhanced version of MMP commitment: a variant of our MMP commitment with zero-knowledge property. In Section 4, we propose a zero-knowledge range proof protocol for Pedersen subvector commitment and analyze its security and performance. Lastly, in Section 5, we demonstrate two sample applications.

1.1 Contribution

The contribution of this paper is fivefold.

- MMP Commitment: Definition and Construction. We formally define MMP commitment and present a specific construction. Our construction is essentially powered by a two-tiered commitment scheme: the first tier is PST commitment [51] for polynomials, and the second tier is AFGHO commitment [1] for PST commitments from the first tier, yielding a commitment of size O(1). Leveraging the homomorphic property of PST commitment and our adapted GIPA protocol, for n polynomials with m variables and maximum degree d, our scheme enjoys $O((m + n) \cdot \binom{m+d}{d})$ prover time (or $O((m + n) \cdot 2^m)$ for multilinear polynomials), $O(m + \log n)$ proof size, and $O(m + \log n)$ verifier time. Our work can be regarded as a natural extension of the Univariate Multi-Polynomial (UMP) commitment developed by Ambrona et al. in aPlonk [2]. Additionally, we improve the MMP protocol to be zero-knowledge at the cost of efficiency. As a side work, we proposed a hiding PST commitment in the batched setting, which is an nature extension of the hiding KZG commitment in [41].
- ZKRP for Pedersen Subvector Commitment. We propose a ZKRP for Pedersen subvector commitment. For n vectors of size l, the communication

and verification complexity are both $O(\log(n \cdot l))$. As a comparison, the only known ZKRP for vector commitment was presented in [46]. It was also designed for Pedersen subvector commitment and has constant proof size (only 3 group elements). But it only supports range proofs for vectors with small norms, and its verification complexity is $O(n \cdot l)$. Besides, our ZKRP scheme is also more flexible as it supports range proofs for any subset of vector elements.

- **SNARK Aggregation.** With a similar methodology to embed UMP in Plonk, our MMP commitment can be used to aggregate some recent SNARKs based on multilinear polynomials [55,26]. To showcase this capability, we introduce aHyperPlonk, a variant of HyperPlonk [26] that allows efficient aggregation of multiple instances. For k instances with n constraints each, our technique reduces the proof size and verification cost from $O(k \log n)$ to $O(\log k + \log n)$, while the prover complexity remains unchanged.
- Zero-Knowledge Proof for Vehicle GPS Traces. We introduce a novel application scenario of MMP commitments: a zero-knowledge proof for vehicle GPS traces. Specifically, we redefine the question of whether a vehicle has passed through a specific GPS coordinate P as a point-in-rectangles/point-out-of-rectangles problem, which can be transformed into determining whether the evaluations of MMP commitments at P are greater than zero. As a result, the zero-knowledge proof for vehicle GPS traces is represented as a zero-knowledge range proof for MMP commitment. It's worth noting that our scheme applies to any application scenario that can be abstracted as a point-in-rectangles/point-out-of-rectangles problem.
- Implementation. We provide a reference implementation of our MMP scheme and evaluate its performance. The evaluation results show that our MMP scheme is highly practical: for 1024 polynomials of degree 3, each with 3 variables, the proof generation algorithm takes 536.99ms, and the verification algorithm completes in 18.595ms.

1.2 Related Work

Polynomial Commitment. A polynomial commitment scheme (PCS) allows the prover to commit to a polynomial and later convince the verifier of evaluations of the committed polynomial on a given point. The notion of polynomial commitment schemes was first introduced by the seminal work of Kate et al. [40], where the authors also proposed a PCS construction (referred to as KZG commitment) for univariate polynomials. In a nutshell, for polynomials of degree $\leq d$, KZG commitment requires a trusted setup that produces a structured reference string $(g^{\tau^0}, g^{\tau^1}, \ldots, g^{\tau^d}, \tilde{g}^{\tau})$, where g and \tilde{g} are generators of two pairing friendly groups. A commitment to a polynomial $f(X) = \sum_{i=0}^{d} f_i X^i$ is computed as $c = g^{f(\tau)} = \prod_{i=0}^{d} (g^{\tau^i})^{f_i}$. The proof of evaluation at point x is $\pi = g^{q(\tau)} = \prod_{i=0}^{d} (g^{\tau^i})^{q_i}$, where the quotient polynomial $q(X) = \frac{f(X) - f(x)}{X - x}$. The verifier checks the proof by comparing if $e(\pi, \tilde{g}^{\tau-x}) = e(c/g^{f(x)}, \tilde{g})$. Later, various lines of research have been conducted to improve the functionality, security, and efficiency of PCS. We list below several representative works.

- Multivariate Polynomials. PST commitment [51] extended KZG commitment to a multivariate setting. Later, Zhang et al. proposed vSQL [59], whose underlying PCS is a variant of PST commitment that provides knowledge soundness under the KoE assumption. Both these PCSs have logarithmic (w.r.t. the size N of the polynomial) proof size and verification time.
- Transparent Setup. The first PCSs with transparent setup were introduced in [11,12], which have square root proof size and verifier. Hyrax [57] extended the ideas of [11,12] to multilinear polynomials. Bulletproofs [16] could also be used to construct a multilinear PCS with transparent setup, enjoying logarithmic proof size, but the verification time is linear. DARK [20] and Dory [45] provided transparent-setup PCSs for multivariate polynomials with logarithmic proof size and verification time, where the former is powered by groups of unknown order, while the latter works in a pairing-based setting. Dew [4] made use of groups of unknown order as well and further improved the efficiency of DARK, resulting in a PCS with constant proof size and logarithmic verifier.
- Strictly Linear Time Prover. In Brakedown [36], the authors made improvements to the prover's efficiency of previous polynomial commitments and proposed a multilinear PCS with strictly linear time prover³, thanks to their concretely efficient linear-time encodable linear code. Orion [58] reduced the proof size and verification time of [36] from $O(\sqrt{N})$ to $O(\log^2 N)$ by leveraging proof composition. Orion+ [26] made various optimizations and further shrank the proof size of Orion [58].
- Aggregation and Batching. There are also several works on aggregated or batched polynomial commitments and evaluation proofs. In [40], a variant of KZG commitment was also given, allowing for opening multiple evaluations on a single polynomial succinctly. For multiple evaluations in multiple polynomials,Plonk [34] and Marlin [27] batched pairing operations in their verification algorithms using random linear combinations, but the proof size is still linear in the number of polynomials. Subsequently, SHPlonk [9] addressed this issue and achieved constant proof size for multiple polynomials. In addition, Boomy [44] explored opening multiple evaluations on one multivariate polynomial in a batched manner.

aPlonk [2] introduced the notion of multi-polynomial commitment and provided an efficient construction. With a two-tiered commitment algorithm (first committing to the polynomials and then the commitments), aPlonk achieved constant commitment size w.r.t. the number of polynomials. [2] also deployed an IPA protocol [21], as well as a SNARK for meta verification, to achieve logarithmic proof size and verification time.

³ Note that the authors of [36] regard multiexponentiation (MSM) as a superlinear operation, since its complexity is $O(N \log |\mathbb{F}| / \log N)$ with Pippenger's algorithm [53], where for security, \mathbb{F} 's order should satisfy $\log |\mathbb{F}| = \omega(\log N)$.

- Zero-knowledge. In [60], the authors of built a zero-knowledge variant of the PCS in vSQL [59], with only logarithmic overheads on prover's side. Zeromorph is also a multilinear PCS with zero-knowledge property, but it is concretely more efficient than [60], as it avoids computing $O(\log N)$ pairings in the verification algorithm.

SNARK. Polynomial commitment is also an important building block in SNARKs. The general paradigm of constructing modern SNARKs is to combine Interactive Oracle Proof (IOP) [7], an information-theoretic object, with a functional commitment scheme. By plugging in a suitable polynomial commitment scheme, one can compile Polynomial-IOPs into SNARKs. For instance, Sonic [48], Marlin [27], and Plonk [34] are built upon univariate polynomial commitment schemes, while more recent ones like Spartan [55], HyperPlonk [26], and Testudo [23] are powered by multilinear polynomial commitment schemes.

A highly related line of research focuses on the composition of SNARKs [56,8,14,19,18,10,42,17], among which aPlonk [2] achieves efficient aggregation for Plonk instances without recursion overheads. The core idea behind aPlonk is a UMP commitment scheme, which, for n polynomials, produces a proof of size log n that can be verified in log n time. Although the UMP commitment scheme in [2] is only designed for Plonk, as we will show later, aPlonk's approach is an excellent starting point of a MMP commitment scheme that is suitable for these recent SNARKs based on multivariate polynomials.

ZKRP. ZKRP allows a prover to convince a verifier that a committed value lies in a given interval without revealing any other information. Zero-knowledge range proof was first introduced by [15], and has been studied across diverse applications like anonymous credential systems [22,6], confidential transactions [50], among others. Efficient zero-knowledge range proof can be derived from 1) square decomposition [13,47,38,30,29], 2) binary/n-ary decomposition [22,16,49,32,5,46], and 3) hash-chain [25]. We refer the readers to [28] for a comprehensive comparison between these schemes. Recently, Libert proposed a range proof with a short proof size (only three group elements) tailored for Pedersen subvector commitment in [46], attesting that the committed vector is of small norm. On the downside, the verification time of this scheme is linear in the number of bits being checked.

2 Building blocks

In this section, we briefly review the assumptions and models involved in this paper. We also introduce the definition of vector and polynomial commitment schemes, and give several constructions that underlie our MMP commitment scheme. Before that, we first list below the notations used in the rest of this paper.

Notation. We denote $\mathsf{PoK}\{(y), (x) : f(x) = y\}$ a proof of knowledge of secret value x for public value y such that f(x) = y holds.

We define a group generation algorithm GGen:

 $-(\mathbb{G}, p) \leftarrow \mathsf{GGen}(\lambda)$. On input security parameter λ , GGen generates cyclic group \mathbb{G} of prime order $p(|p| = \lambda)$.

as well as a bilinear group generation algorithm BGGen,

- $(e, \mathbb{G}, \mathbb{G}, \mathbb{G}_T, p) \leftarrow \mathsf{BGGen}(\lambda)$. On input security parameter λ , BGGen generates cyclic groups \mathbb{G}, \mathbb{G} , and \mathbb{G}_T of prime order p $(|p| = \lambda)$ and a bilinear mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

We use bold font \boldsymbol{x} to represent vectors. We denote

- $|\mathbf{v}|$ the number of elements of \mathbf{v} , e.g. $|\mathbf{v}| = n$,
- $(v_i)_{i \in [1,n]}$ the vector $(v_1, ..., v_n)$,
- $(v_{i...j})_{i \in [0,n],...,j \in [0,n']}$ the vector $(v_{0...0}, v_{0...1}, ..., v_{n...n'})$,
- $\boldsymbol{v}_{[:n']} = (v_1, ..., v_{n'}), \, \boldsymbol{v}_{[n':]} = (v_{n'+1}, ..., v_n),$
- \boldsymbol{v}_S the subvector of \boldsymbol{v} indexed by set $S \subseteq [1, n]$.

Let \mathbb{G}^n , \mathbb{Z}_p^n be *n*-dimensional vector spaces over \mathbb{G} , \mathbb{Z}_p , respectively. For $k \in \mathbb{Z}_p$, $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n$, and $\boldsymbol{g} \in \mathbb{G}^n$, $\tilde{\boldsymbol{g}} \in \widetilde{\mathbb{G}}^n$, we denote

 $\begin{aligned} &-\mathbf{k}^{n} = (k^{i})_{i \in [0, n-1]}, \, \mathbf{k}^{-n} = (k^{i})_{i \in [0, -n+1]}, \\ &-k \cdot \boldsymbol{a} = (k \cdot a_{1}, \dots, k \cdot a_{n}), \\ &-\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{i=1}^{n} a_{i} \cdot b_{i} \text{ the inner product of } \boldsymbol{a}, \boldsymbol{b}, \\ &-\langle \boldsymbol{a}, \boldsymbol{g} \rangle = \prod_{i=1}^{n} g_{i}^{a_{i}} \text{ the inner product of } \boldsymbol{a}, \boldsymbol{g}, \\ &-\langle \boldsymbol{g}, \widetilde{\boldsymbol{g}} \rangle = \prod_{i=1}^{n} e(g_{i}, \widetilde{g}_{i}) \text{ the inner product of } \boldsymbol{g}, \widetilde{\boldsymbol{g}}, \\ &-\boldsymbol{a} \circ \boldsymbol{b} = (a_{i} \cdot b_{i})_{i \in [1, n]} \text{ the Hadamard product of } \boldsymbol{a}, \boldsymbol{b}, \end{aligned}$

where a_i, b_i, g_i are the *i*-th element of a, b, g.

2.1 Hardness Assumption

Definition 1 ((m, n)-Discrete Logarithm Assumption). Let $(\mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T)$ be asymmetric bilinear groups of order p. For \mathbb{G} 's generator g and $\widetilde{\mathbb{G}}$'s generator $\widetilde{g}, m, n \in \mathbb{Z}$, given $(g, g^x, g^{x^2}, ..., g^{x^m})$ and $(\widetilde{g}, \widetilde{g}^x, ..., \widetilde{g}^{x^n})$, it is computationally infeasible to find $x \in \mathbb{Z}_p$.

2.2 Proof System

Let R(u, w) be an efficiently decidable binary relation. Language $L \subset \Sigma^*$ (a subset of finite strings) in the relation R is defined as

$$L = \{u | \exists w : R(u, w) = 1\}$$

We call w a witness for statement u.

A pair of interactive probabilistic polynomial-time algorithms $\langle \mathcal{P}, \mathcal{V} \rangle$ for relation R is defined as below:

 $-\langle \mathcal{P}(u,w), \mathcal{V}(u) \rangle$. Taking as input w and u, this pair of algorithms outputs 1 if \mathcal{V} accepts.

Definition 2 (Proof System). $(\mathcal{P}, \mathcal{V})$ is a proof system for relation R if it satisfies completeness and soundness.

- Completeness. $(\mathcal{P}, \mathcal{V})$ is complete if for any (u, w) such that R(u, w) = 1,

$$\Pr\left[\langle \mathcal{P}(u,w), \mathcal{V}(u) \rangle = 1\right] = 1$$

- Soundness. $(\mathcal{P}, \mathcal{V})$ is sound if for any $u \notin L$ and any cheating prover \mathcal{P}^* ,

$$\Pr\left[\langle \mathcal{P}^*(u), \mathcal{V}(u) \rangle = 1\right] = \operatorname{negl}$$

We further say $(\mathcal{P}, \mathcal{V})$ is an argument system if its soundness only holds for probabilistic polynomial-time \mathcal{P}^* .

Definition 3 (Argument of Knowledge). $(\mathcal{P}, \mathcal{V})$ is an argument of knowledge for relation R if it satisfies completeness and computational witness-extended emulation.

- Witness-Extended Emulation. $(\mathcal{P}, \mathcal{V})$ satisfies computational witnessextended emulation if for all deterministic polynomial-time \mathcal{P}^* , there exists an expected polynomial-time extractor \mathcal{B} such that for any probabilistic polynomial-time adversary \mathcal{A} ,

$$\Pr\left[\begin{array}{cc} (u,s) \leftarrow \mathcal{A}(\lambda);\\ tr \leftarrow \langle \mathcal{P}^*(u,s), \mathcal{V}(u) \rangle:\\ \mathcal{A}(tr) = 1 \end{array}\right] \approx \Pr\left[\begin{array}{cc} (u,s) \leftarrow \mathcal{A}(\lambda); (tr,w) \leftarrow \mathcal{B}^{\mathcal{O}}(u):\\ \mathcal{A}(tr) = 1\\ \land \quad (tr \text{ is accepting}) \rightarrow (u,w) \in R \end{array}\right]$$

where the transcript oracle $\mathcal{O} = \langle \mathcal{P}^*(u, s), \mathcal{V}(u) \rangle$, which allows to rewind \mathcal{P}^* to any point and continues with fresh randomness from \mathcal{V} .

Definition 4 (Public Coin). A proof (or argument) system $(\mathcal{P}, \mathcal{V})$ is public coin if the messages sent by the verifier are predetermined consecutive segments of its random tape.

Definition 5 (Special Honest-Verifier Zero-Knowledge (SHVZK)). A public coin proof (or argument) system $(\mathcal{P}, \mathcal{V})$ is SHVZK if there exists a probabilistic polynomial-time simulator S such that for any probabilistic polynomial-time adversary A,

$$\Pr \left[\begin{array}{c} (u, w, \eta) \leftarrow \mathcal{A}(\lambda); \\ tr \leftarrow \langle \mathcal{P}(u, w), \mathcal{V}(u; \eta) \rangle : \\ (u, w) \in R \land \mathcal{A}(tr) = 1 \end{array} \right] \approx \Pr \left[\begin{array}{c} (u, w, \eta) \leftarrow \mathcal{A}(\mathsf{srs}); \\ tr \leftarrow S(u, \eta) : \\ (u, w) \in R \land \mathcal{A}(tr) = 1 \end{array} \right]$$

where η is the public coin randomness used by \mathcal{V} .

Definition 6 (Non-interactive Zero-Knowledge (NIZK)). *NIZK is a zeroknowledge proof that requires no interaction between the prover and the verifier.*

A SHVZK can be transformed into a NIZK using the Fiat-Shamir heuristic, which replaces \mathcal{V} with a hash of all previous transcripts.

Construction 1 (Schnorr's Protocol [54]) Let $(\mathbb{G}, p) \leftarrow \mathsf{GGen}(\lambda)$ and $g \leftarrow R$ \mathbb{G} . Schnorr's protocol is a SHVZK for the discrete logarithm relation R such that

$$R = \mathsf{PoK} \left\{ \begin{array}{l} (h \in \mathbb{G}), (x \in \mathbb{Z}_p) : \\ h = g^x \end{array} \right\},$$

The protocol is defined as follows.

- $-\mathcal{P}$ picks $\varrho \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and sends $D = g^{\varrho}$ to \mathcal{V} .
- $\begin{array}{c} -\mathcal{V} \ picks \ \eta \stackrel{R}{\leftarrow} \mathbb{Z}_p \ and \ sends \ to \ \mathcal{P}. \\ -\mathcal{P} \ sends \ b = \varrho \eta x \ to \ \mathcal{P}. \end{array}$
- $-\mathcal{V}$ accepts if $g^b h^\eta = D$.

Construction 2 (Generalized Inner Product Argument (GIPA) [21]) Inner product argument is an Argument of Knowledge that allows for efficient verification of the inner product of vectors. [21] proposed a GIPA protocol for the following relation.

$$\mathsf{PoK} \left\{ \begin{array}{l} (n, I_{\boldsymbol{ab}}, \boldsymbol{c}, I_{\boldsymbol{ac}}, \boldsymbol{d}, I_{\boldsymbol{bd}}), (\boldsymbol{a}, \boldsymbol{b}) : \\ I_{\boldsymbol{ab}} = \langle \boldsymbol{a}, \boldsymbol{b} \rangle, I_{\boldsymbol{ac}} = \langle \boldsymbol{a}, \boldsymbol{c} \rangle, I_{\boldsymbol{bd}} = \langle \boldsymbol{b}, \boldsymbol{d} \rangle \end{array} \right\},$$

It is referred to as "generalized" because it supports any combinations of the following three types of inner product relations

$$\langle \cdot, \cdot \rangle : \begin{cases} \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F} & (\boldsymbol{x}, \boldsymbol{y}) \mapsto x_1 y_1 + \dots + x_n y_n \\ \mathbb{F}^n \times \mathbb{G}^n \to \mathbb{G} & (\boldsymbol{x}, \boldsymbol{g}) \mapsto g_1^{x_1} \dots g_n^{x_n} \\ \mathbb{G}^n \times \widetilde{\mathbb{G}}^n \to \mathbb{G}_T & (\boldsymbol{g}, \boldsymbol{h}) \mapsto e(g_1, h_1) \dots e(g_n, h_n) \end{cases}$$

where $\mathbf{x} = (x_1, \ldots, x_n), \ \mathbf{g} = (g_1, \ldots, g_n), \ etc.$

Looking ahead, our ZKRP protocol for Pedersen subvector commitment leverages the inner product arguments of Bulletproofs [16] as a black box, which is essentially the GIPA protocol in [21] with $a, b \in \mathbb{Z}_n^n$ and $c, d \in \mathbb{G}^n$.

To fit our application, we present an adapted GIPA for the following inner product relation.

$$\mathsf{PoK} \left\{ \begin{array}{l} (n, \boldsymbol{b}, I_{\boldsymbol{a}\boldsymbol{b}}, \boldsymbol{c}, I_{\boldsymbol{a}\boldsymbol{c}}, I_{\boldsymbol{d}\boldsymbol{c}}), (\boldsymbol{a}) :\\ I_{\boldsymbol{a}\boldsymbol{b}} = \langle \boldsymbol{a}, \boldsymbol{b} \rangle, I_{\boldsymbol{a}\boldsymbol{c}} = \langle \boldsymbol{a}, \boldsymbol{c} \rangle \end{array} \right\}.$$

The construction of the adapted GIPA scheme is defined in Figure 1. Note that even if a is not a secret, GIPA protocol can be applied to reduce communication complexity. As shown in the figure, the proof size is logarithmic to vector length n, but the verification time is linear w.r.t n due to Steps 5 and 6, which computes b_0 and c_0 recursively. Many optimizations for **b** or **c** with special arithmetic structure are proposed [31,21]. Taking \boldsymbol{b} as an example, the optimization techniques involved in this paper for achieving logarithmic verification complexity are summarized as follows.

- 1. Define $\hat{\eta}(X) = \prod_{i=1}^{\kappa} (\eta_i^{-1} + \eta_i X^{2^{\kappa-i}})$. Let $\hat{\eta}$ be the coefficient vector of polynomial $\hat{\eta}(X)$. As pointed in [2], b_0 can be rewritten as $\langle \hat{\eta}, b \rangle$. Based on this observation, we have the following conclusions.
 - (a) For $\boldsymbol{b} = (1, r^1, ..., r^{n-1})$ where $r \in \mathbb{Z}_p$ is a public parameter, $b_0 = \hat{\eta}(r)$ can be computed in logarithmic time by simply plugging r into $\hat{\eta}(X)$.
 - (b) For **b** = (g, g^α, ..., g^{αⁿ⁻¹}), b₀ = g^{ĝ(α)} is a KZG commitment on polynomial ĝ(X). In this case, P can initiate a KZG evaluation proof with V to prove that b₀ is indeed a PST commitment to ĝ(X). More specifically, P adds b₀ to π along with a KZG evaluation proof π_{kzg} on a random input ρ. Upon receiving b₀ and π_{kzg}, V computes ĝ(ρ) and accepts b₀ if only e(b₀/g^{ĝ(ρ)}, ğ) = e(π_{kzg}, g^α/g^ρ).
 - (c) For $\boldsymbol{b} = (g, g^{r \cdot \alpha}, ..., g^{(r \cdot \alpha)^{n-1}})$ where $r \in \mathbb{Z}_p$ is a public element, $b_0 = \hat{\eta}(r \cdot \alpha)$ is a KZG commitment on polynomial $\hat{\eta}(r \cdot X)$. The remainder is the same as in point 1.2.
- 2. For **b**: $\{\boldsymbol{b} \leftarrow (1), \boldsymbol{b} \leftarrow (\boldsymbol{b}, r_i \cdot \boldsymbol{b})\}_{i \in [0, \kappa 1]}$ where $r_i \in \mathbb{Z}_p$ is public elements, for $i \in [0, \kappa 1]$,

$$\boldsymbol{b}_{i} = \eta_{i}^{-1} \cdot \boldsymbol{b}_{[:2^{i}]} + \eta_{i} \cdot \boldsymbol{b}_{[2^{i}:]} = \eta_{i}^{-1} \cdot \boldsymbol{b}_{[:2^{i}]} + \eta_{i} \cdot (r_{i} \cdot \boldsymbol{b}_{[:2^{i}]}) = (\eta_{i}^{-1} + \eta_{i} \cdot r_{i}) \cdot \boldsymbol{b}_{[:2^{i}]},$$

which implies $b_0 = \prod_{i=0}^{\kappa} (\eta_i^{-1} + \eta_i \cdot r_i).$

2.3 Algebraic Group Model

In this paper, the security of the PST commitment scheme in the batched setting and our MMP protocol is analyzed in the Algebraic Group Model (AGM) [33], a security model that lies between the standard security model and Generic Group Model (GGM) in terms of the restrictions put on the adversary. In such a model, whenever an algebraic adversary \mathcal{A} outputs a group element in \mathbb{G} or $\widetilde{\mathbb{G}}$, it must also output the representation of the group element w.r.t the group elements it has received so far. Following the definition of [34], algebraic adversary \mathcal{A} in an SRS-based protocol is defined as below.

Given input srs = { $g = (g^{f_i(x)})_{i \in [1,n]}, \widetilde{g} = (\widetilde{g}^{\widetilde{f}_i(x)})_{i \in [1,n]}$ }, where $f_i, \widetilde{f}_i \in \mathbb{F}_{\langle d \in \mathbb{Z}}[X]$, when \mathcal{A} outputs a group element $E \in \mathbb{G}$, it also outputs its representation w.r.t g, which is, vector u such that $\langle u, g \rangle = E$. \mathcal{A} behaves in the same manner for $E \in \widetilde{\mathbb{G}}$. In this paper, we use $\mathsf{REP}_{\mathsf{srs}}^E$ to represent such representation.

Now, denote the group elements output by \mathcal{A} as $(g^{a_j})_{j\in[1,k]}, (\tilde{g}^{\tilde{b}_j})_{j\in[1,l]}$, and their representations respectively as $(\boldsymbol{u}^{[j]})_{j\in[1,k]}, (\boldsymbol{v}^{[j]})_{j\in[1,l]}$. Define $L_j(X) = \sum_{i=0}^n u_i^{[j]} f_i(X)$, and analogously, $R_j(X) = \sum_{i=0}^n v_i^{[j]} \tilde{f}_i(X)$. We set $\boldsymbol{a} = (a_j)_{j\in[1,k]}, \boldsymbol{b} = (b_j)_{j\in[1,l]}, \boldsymbol{L} = (L_j(X))_{j\in[1,k]}, \boldsymbol{R} = (R_j(X))_{j\in[1,l]}$.

We set $\boldsymbol{a} = (a_j)_{j \in [1,k]}$, $\boldsymbol{b} = (b_j)_{j \in [1,l]}$, $\boldsymbol{L} = (L_j(X))_{j \in [1,k]}$, $\boldsymbol{R} = (R_j(X))_{j \in [1,l]}$. For matrix M_1 and M_2 over \mathbb{F} , we define the real pairing check and ideal check as follows.

REAL PAIRING CHECK. $(\boldsymbol{a} \cdot M_1) \cdot (M_2 \cdot \boldsymbol{b}) = 0$

IDEAL CHECK. $(\boldsymbol{L} \cdot M_1) \cdot (M_2 \cdot \boldsymbol{R}) \equiv 0$

Regarding the real pairing check and ideal check, we have the following lemma.

 $\mathcal{P}(n=2^{\kappa},\eta_{\kappa}=\emptyset,\boldsymbol{a},\boldsymbol{b},I_{\boldsymbol{a}\boldsymbol{b}},\boldsymbol{c},I_{\boldsymbol{a}\boldsymbol{c}})$

1: if n = 1 then 2: Return $a_0 = \boldsymbol{a}$ 3: else $n \ge 2$ $n' = \frac{n}{2}, i = \log_2(n')$ 4:
$$\begin{split} & L_{ab}^{[i]} = \langle \boldsymbol{a}_{[:n']}, \boldsymbol{b}_{[n':]} \rangle, R_{ab}^{[i]} = \langle \boldsymbol{a}_{[n':]}, \boldsymbol{b}_{[:n']} \rangle \\ & L_{ac}^{[i]} = \langle \boldsymbol{a}_{[:n']}, \boldsymbol{c}_{[n':]} \rangle, R_{ac}^{[i]} = \langle \boldsymbol{a}_{[n':]}, \boldsymbol{c}_{[:n']} \rangle \end{split}$$
5:6:
$$\begin{split} L_{ac}^{i} &= \langle \boldsymbol{a}_{[:n']}, \boldsymbol{c}_{[n':]} \rangle, R_{ac}^{i} &= \langle \boldsymbol{a}_{[n':]}, \boldsymbol{c}_{[:n']} \rangle \\ \eta_{i} &= H(\eta_{i+1}, L_{ab}^{i}, R_{ab}^{i}, L_{ac}^{i}, R_{ac}^{i}) \\ \boldsymbol{a}' &= \eta_{i} \cdot \boldsymbol{a}_{[:n']} + \eta_{i}^{-1} \cdot \boldsymbol{a}_{[n':]} \\ \boldsymbol{b}' &= \eta_{i}^{-1} \cdot \boldsymbol{b}_{[:n']} + \eta_{i} \cdot \boldsymbol{b}_{[n':]} \\ \boldsymbol{c}' &= \eta_{i}^{-1} \cdot \boldsymbol{c}_{[:n']} + \eta_{i} \cdot \boldsymbol{c}_{[n':]} \\ I_{ab}' &= \eta_{i}^{2} \cdot L_{ab}^{i} + I_{ab} + \eta_{i}^{-2} \cdot R_{ab}^{i} \\ I_{ac}' &= \eta_{i}^{2} \cdot L_{ac}^{i} + I_{ac} + \eta_{i}^{-2} \cdot R_{ac}^{i} \\ \text{Recursively call } \mathcal{P}(n', \eta_{i}, \boldsymbol{a}', \boldsymbol{b}', I_{ab}', \boldsymbol{c}', I_{ac}') \end{split}$$
7:8: 9: 10: 11: 12:13:14: Output $\pi = (a_0, \{L^i_{ab}, R^i_{ab}, L^i_{ac}, R^i_{ac}\}_{i \in [\kappa - 1, 0]})$ $\mathcal{V}(n, \boldsymbol{b}, I_{\boldsymbol{ab}}, \boldsymbol{c}, \boldsymbol{I_{ac}}, \pi)$ 1: Parse $\pi = (a, \{L_{ab}^i, R_{ab}^i, L_{ac}^i, R_{ac}^i\}_{i \in [\kappa - 1, 0]})$ 2: $\eta_{\kappa} = \emptyset$ 3: for $i \in [\kappa - 1, 0]$ do 4: $\eta_i = H(\eta_{i+1}, L_{ab}^i, R_{ab}^i, L_{ac}^i, R_{ac}^i)$ 5: $\boldsymbol{b}_i = \eta_i^{-1} \cdot \boldsymbol{b}_{[:n']} + \eta_i \cdot \boldsymbol{b}_{[n':]}$ 6: $\boldsymbol{c}_i = \eta_i^{-1} \cdot \boldsymbol{c}_{[:n']} + \eta_i \cdot \boldsymbol{c}_{[n':]}$ 7: $\hat{\eta}(X) = \prod_{j=1}^{\kappa} (\eta_i^{-1} + \eta_i X^{2^{\kappa-i}})$ 8: $\langle a_0, b_0 \rangle \stackrel{?}{=} I_{ab} + \sum_{i=1}^{\kappa} (\eta_i^2 \cdot L_{ab}^i + \eta_i^{-2} R_{ab}^i)$ 9: $\langle a_0, c_0 \rangle \stackrel{?}{=} I_{\boldsymbol{ac}} + \sum_{i=1}^{\kappa} (\eta_i^2 \cdot L_{\boldsymbol{ac}}^i + \eta_i^{-2} R_{\boldsymbol{ac}}^i)$

Fig. 1: Construction of GIPA Protocol.

Lemma 1. Given an algebraic adversary \mathcal{A} with srs, the probability of any real pairing check passing pass but ideal check fails is negligible.

Please refer to [34] for the proof of the lemma.

2.4 Vector commitment

We assume that the reader is familiar with the notion of commitment schemes. A commitment scheme can be binding, or both binding and hiding. We refer to the latter as a hiding commitment scheme. In the rest of the paper, we highlight in red the parameters and operations used solely to achieve hiding property.

In general, various commitment schemes allow a prover to commit to a vector of elements and later open the commitment. However, different schemes in the literature may require the opening algorithm to achieve different functionalities,

and its behavior depends on the concrete scheme or the application scenario. For clarity, below we categorize the commitment schemes for vectors into three categories, according to the behavior of the opening algorithm:

- 1. **Commitment scheme.** Open the entire vector (as well as the randomness, if the commitment is hiding).
- 2. Vector commitment scheme [24]. Open the element at a specific index.
- 3. Subvector commitment scheme [43]. Open the elements at a specific subset of indices.

We give the algorithm definition of subvector commitment and the constructions of Pedersen commitment, AFGHO commitment, and (Hiding) Pedersen subvector commitment in Appendix B.

2.5 Multivariate Polynomial Commitment

A multivariate polynomial commitment scheme consists of four algorithms Setup, Commit, Prove, and Eval.

- srs \leftarrow Setup $(\lambda, m, \hat{d} = (\hat{d}_i)_{i \in [1,m]})$. On input a security parameter λ , the number of variables m, and a degree bound vector \hat{d} , Setup outputs structured reference string srs.
- $-c_{f} \leftarrow \text{Commit}(\text{srs}, f \in \mathbb{F}_{<\hat{d}}[X_1, ..., X_m].$ Commit computes a commitment c_f on polynomial f.
- $-\pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \boldsymbol{v}, f, c_f)$. Prove generates a proof π for evaluation of f on vector \boldsymbol{v} .
- $-1/0 \leftarrow \mathsf{Eval}(\mathsf{srs}, \boldsymbol{v}, c_f, e_{\boldsymbol{v}}, \pi)$. Eval checks if $e_{\boldsymbol{v}}$ is the evaluation of the polynomial committed in c_f on vector \boldsymbol{v} .

Please refer to Appendix C for the security properties of multivariate polynomial commitments.

(Hiding) PST commitment [51]. We review the construction of (hiding) PST commitment, and additionally, introduce a pair of commitment and verification algorithms in the batched setting, i.e., BatchProve and BatchEval in Figure 2. As mentioned, this hiding PST commitment in the batched setting can be regarded as an extension of the hiding KZG commitment scheme proposed in [41]. Please refer to Appendix D for the security proofs for PST commitment and hiding PST commitment in the batched setting.

3 Multivariate Multi-Polynomial Commitment

3.1 Algorithm Definition

Following the definition of [2] on univariate multi-polynomial commitment, we extend to include multivariate polynomials. Same as [2], we require verification to be performed on a succinct commitment to the polynomial evaluations, enjoying logarithmic proof size w.r.t the number of polynomials.

srs \leftarrow Setup $(\lambda, m, \hat{d} = (\hat{d}_j)_{j \in [1,m]})$ 1: $(e, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, p) \leftarrow \mathsf{BGGen}(\lambda)$ 2: $g \stackrel{R}{\leftarrow} \mathbb{G}, \, \widetilde{g} \stackrel{R}{\leftarrow} \widetilde{\mathbb{G}}$ $\begin{array}{l} 3: \ \forall j \in [1,m], \beta_j \xleftarrow{R} \mathbb{Z}_p, \ \delta \xleftarrow{R} \mathbb{Z}_p \\ 4: \ h := g^{\delta}, \widetilde{h} := \widetilde{g}^{\delta} \end{array}$ 5: srs := $(p, \boldsymbol{g} = (g^{\prod_{j=1}^{m} \beta_j^{d_j}})_{d_1 \in [0, \hat{d_1}], \dots, d_m \in [0, \hat{d_m}]}, \tilde{g}, (\tilde{g}^{\beta_j})_{j \in [1, m]}, \boldsymbol{h}, \tilde{\boldsymbol{h}})$ $c_f \leftarrow \mathsf{Commit}(\mathsf{srs}, f)$ 1: $f(\mathbf{X}) = \sum_{d_1 \in [0,\hat{d}_1]} f_{d_1...d_m} \prod_{j=1}^m \overline{X_j^{d_j}}$ 2: $\boldsymbol{w} = \{f_{d_1...d_m}\}_{d_1 \in [0, \hat{d}_1], ..., d_m \in [0, \hat{d}_m]}^{\cdots}$ 3: $\rho \stackrel{R}{\leftarrow} \mathbb{Z}_p$ 4: $c_f = \langle \boldsymbol{w}, \boldsymbol{g} \rangle + \langle \rho, h \rangle$ $\pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \boldsymbol{v}, f, c_f, \boldsymbol{\rho})$ 1: Compute $\{q_j(\mathbf{X})\}_{j \in [1,m]} : f(\mathbf{X}) - f(\mathbf{v}) = \sum_{j=1}^m q_j(\mathbf{X})(x_j - v_j)$ $\begin{aligned} & 2: \ \forall j \in [1,m], \mu_j \xleftarrow{R} \mathbb{Z}_p \\ & 3: \ \pi = \{g^{q_j(\beta)} \cdot h^{\mu_j}\}_{j \in [1,m]} \\ & 4: \ \theta = g^{\sum_{j=1}^m v_j \cdot \mu_j + \rho} / \prod_{j=1}^m g^{\beta_j \cdot \mu_j} \end{aligned}$ $1/0 \leftarrow \mathsf{Eval}(\mathsf{srs}, \boldsymbol{v}, c_f, e_{\boldsymbol{v}}, \pi, \boldsymbol{\theta})$ 1: Parse π as $\{Q_j\}_{j \in [1,m]}$ 2: $e(c_f \cdot g^{-e_v}, \widetilde{g}) = \prod_{i=1}^m e(Q_j, \widetilde{g}^{\beta_i}/g^{v_i}) \cdot \underline{e(\theta, \widetilde{h})}$ $\pi \leftarrow \mathsf{BatchProve}(\mathsf{srs}, v, (c_{f_i}, f_i, \rho_i)_{i \in [1,n]})$ 1: $r = H(\boldsymbol{v}, (c_{f_i})_{i \in [1,n]})$ 2: $\forall i \in [1, n], \{q_j^i(\boldsymbol{X})\}_{j \in [1, m]} : f_i(\boldsymbol{X}) - f_i(\boldsymbol{v}) = \sum_{j=1}^m q_j^i(\boldsymbol{X})(X_j - v_j)$ 3: $\forall j \in [1, m], q_j(\boldsymbol{X}) = (q_j^i(\boldsymbol{X}))_{i \in [1, n]}, \hat{q}_j(\boldsymbol{X}) = \langle q_j(\boldsymbol{X}), \boldsymbol{r}^n \rangle$ 4: $\forall j \in [1, m], \mu_j \stackrel{R}{\leftarrow} \mathbb{Z}_p$ 5: $\pi = \{g^{\hat{q}_j(\beta)} \cdot h^{\mu_j}\}_{j \in [1, m]}$ 6: $\theta = g^{\sum_{j=1}^m v_j \cdot \mu_j + \sum_{i=1}^n \rho_i \cdot r^{i-1}} / \prod_{j=1}^m g^{\beta_j \cdot \mu_j}$ $1/0 \leftarrow \mathsf{BatchEval}(\mathsf{srs}, \boldsymbol{v}, (c_{f_i}, e_{\boldsymbol{v}}^{[i]})_{i \in [1,n]}, \pi, \theta)$ 1: Phase π as $\{Q_j\}_{j \in [1,n]}$ 2: $r = H(\boldsymbol{v}, (c_{f_i})_{i \in [1,n]})$ 3: $c_{\hat{f}} = \langle (c_{f_i})_{i \in [1,n]}, \boldsymbol{r}^n \rangle$ 4: $e_{\hat{v}} = \langle (e_v^{[i]})_{i \in [1,n]}, \boldsymbol{r}^n \rangle$ 5: $1/0 \leftarrow \mathsf{Eval}(\mathsf{srs}, \boldsymbol{v}, c_{\hat{f}}, e_{\hat{v}}, \pi, \theta)$

Fig. 2: Construction of PST Commitment. For logarithmic proof size w.r.t.m, Eval can be checked using GIPA protocol.

Definition 7. A MMP commitment scheme consists of five algorithms: Setup, Commit, Commit_{Eval}, Prove, and Eval.

- srs \leftarrow Setup (λ, n, m, \hat{d}) . On input a security parameter λ , the number of polynomials n, the number of variables m per polynomial, and a degree bound vector \hat{d} of polynomials, outputs a structed reference string srs.
- $-c_{f} \leftarrow \text{Commit}(\text{srs}, f)$. Commit *outputs a commitment* c_{f} *on* f.
- $-c_{v} \leftarrow \text{Commit}_{\text{Eval}}(\text{srs}, v, f)$. Commit_{Eval} outputs a commitment c_{v} on evaluations f(v).
- $-\pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \boldsymbol{v}, \boldsymbol{f}, c_{\boldsymbol{f}})$. Prove generates a proof π for evaluations of \boldsymbol{f} on vector \boldsymbol{v} .
- $-1/0 \leftarrow \mathsf{Eval}(\mathsf{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi)$. Eval checks if $e_{\boldsymbol{v}}$ is the evaluation of the polynomial behind $c_{\boldsymbol{f}}$ on \boldsymbol{v} .

MMP commitment satisfies completeness, polynomial binding, evaluation binding, and optionally, knowledge soundness.

COMPLETENESS. For any *n m*-variate polynomials \boldsymbol{f} with degree $\leq \hat{d}$ and evaluation vectors $\boldsymbol{v} \in \mathbb{F}^m$,

$$\Pr \begin{bmatrix} \mathsf{srs} \leftarrow \mathsf{Setup}(\lambda, n, m, \hat{d}), \\ c_{\boldsymbol{f}} \leftarrow \mathsf{Commit}(\mathsf{srs}, \boldsymbol{f}), \\ c_{\boldsymbol{v}} \leftarrow \mathsf{Commit}_{\mathsf{Eval}}(\mathsf{srs}, \boldsymbol{v}, \boldsymbol{f}), \\ \pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \boldsymbol{v}, \boldsymbol{f}, c_{\boldsymbol{f}}) : \\ \mathsf{Eval}(\mathsf{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi) = 1 \end{bmatrix} = 1$$

POLYNOMIAL BINDING. A MMP commitment scheme is polynomial binding if for all PPT adversary \mathcal{A} ,

$$\Pr \begin{bmatrix} \mathsf{srs} \leftarrow \mathsf{Setup}(\lambda, m, n, \hat{d}), \\ (\boldsymbol{f}, \boldsymbol{f'}) \leftarrow \mathcal{A}(\mathsf{srs}) : \\ \mathsf{Commit}(\mathsf{srs}, \boldsymbol{f}) \neq \mathsf{Commit}(\mathsf{srs}, \boldsymbol{f'}) \end{bmatrix} = \mathsf{negl}$$

EVALUATION BINDING. A MMP commitment scheme is evaluation binding if for all PPT adversary \mathcal{A} ,

$$\Pr \begin{bmatrix} \mathsf{srs} \leftarrow \mathsf{Setup}(\lambda, n, m, \hat{d}), \\ (\boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi, c_{\boldsymbol{v}}', \pi') \leftarrow \mathcal{A}(\mathsf{srs}) : \\ \mathsf{Eval}(\mathsf{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi) = 1 \\ \wedge \qquad \mathsf{Eval}(\mathsf{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}', \pi') = 1 \end{bmatrix} = \mathsf{negl}$$

KNOWLEDGE SOUNDNESS. A MMP commitment scheme is knowledge sound if for all PPT adversary \mathcal{A} , there exists an extractor \mathcal{E} such that

$$\Pr \begin{bmatrix} \mathsf{srs} \leftarrow \mathsf{Setup}(\lambda, n, m, \hat{d}), \\ (\boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi) \leftarrow \mathcal{A}(\mathsf{srs}), \\ (\boldsymbol{f}, (e_{\boldsymbol{v}}^{[i]})_{i=1}^n) \leftarrow \mathcal{E}(\mathsf{srs}) : \\ c_{\boldsymbol{v}} = \mathsf{Commit}_{\mathsf{Eval}}(\mathsf{srs}, \boldsymbol{v}, \boldsymbol{f}) \\ \wedge \quad \mathsf{Eval}(\mathsf{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi) = 1 \\ \wedge \qquad (e_{\boldsymbol{v}}^{[i]})_{i=1}^n \neq \boldsymbol{f}(\boldsymbol{v}) \end{bmatrix} = \mathsf{negl}$$

3.2 Construction

Our MMP commitment scheme combines PST commitment, AFGHO commitment, Pedersen Commitment, and GIPA protocol such that

- the polynomial commitment c_f is a two-tier commitment where the first tier is a PST commitment, i.e., $\phi := (\mathsf{pst.Commit}(\mathsf{srs}_{\mathsf{pst}}, f_j))_{i \in [1,n]}$ and the second tier is a Pedersen commitment, i.e., $c_f = \mathsf{afg.Commit}(\mathsf{srs}_{\mathsf{afg}}, \phi)$.
- the evaluation commitment algorithm is a Pedersen commitment, i.e, $c_{\boldsymbol{v}} = \text{ped.Commit}(srs_{ped}, \boldsymbol{\varphi})$ where $\boldsymbol{\varphi} = (f_i(\boldsymbol{v}))_{i \in [1,n]}$ and
- the prover initiates a batched evaluation proof with polynomial commitments ϕ on v. But different from the BatchEval algorithms of PST commitments, the prover will compute the aggregated commitment $c_{\hat{f}}$ and aggregated evaluations $e_{\hat{v}}$ for the verifier, and proves that the aggregation is carried out correctly using GIPA protocol. By doing so, the communication complexity scale logarithmically w.r.t the number of polynomials.

Denote

- (ped.Setup, ped.Commit, ped.Open) as Pedersen commitment,
- (afg.Setup, afg.Commit, afg.Open) as AFGHO commitment,
- (pst.Setup, pst.Commit, pst.Prove, pst.Eval) as PST commitment, and
- (gipa.Prove, gipa.Verify) as GIPA.

The construction of the MMP commitment scheme is described in 3. Observe that

- 1. $\langle \hat{\boldsymbol{\eta}}, \boldsymbol{r}^n \rangle = \hat{\eta}(r),$
- 2. $\langle \hat{\boldsymbol{\eta}}, \boldsymbol{g}_{\mathsf{afg}} \rangle = g^{\hat{\eta}(\gamma)}$ or $\langle \hat{\boldsymbol{\eta}}, \boldsymbol{g}_{\mathsf{ped}} \rangle = g^{\hat{\eta}(\alpha)}$ is a KZG commitment to polynomial $\hat{\eta}(X)$ with structure reference string $\boldsymbol{g}_{\mathsf{afg}}$ or $\boldsymbol{g}_{\mathsf{ped}}$.

The verification complexity of both GIPA protocols in 3 is logarithmic to n by implementing the optimization methods mentioned in Section 2.2.

3.3 Security Analysis

Completeness follows from the construction of the scheme. We show below MMP commitment scheme is 1) polynomial binding if AFGHO is binding and PST commitment is polynomial binding, and 2) knowledge sound in AGM if Lemma 1 holds.

Theorem 1 (Polynomial Binding of MMP Commitment). Given an adversary \mathcal{A} against the polynomial binding property of the MMP commitment scheme, there must exist an adversary \mathcal{A}^* that either

- breaks the binding property of the underlying AFGHO commitment scheme, or
- breaks the polynomial binding property of the PST commitment scheme.

srs \leftarrow Setup (λ, m, n, \hat{d}) Combine pst.Setup, afg.Setup, and pst.Setup as follows: 1: $(e, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, p) \leftarrow \mathsf{BGGen}(\lambda)$ 2: $\alpha, \beta_1, ..., \beta_n, \gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p^n$ 3: $\operatorname{srs}_{ped} = (p, \boldsymbol{g}_{ped} = (g^{\alpha^{j-1}})_{j \in [1,n]})$ 4: $\operatorname{srs}_{pst} = (p, \boldsymbol{g}_{pst} = (g^{\prod_{i=1}^{m} \beta_i^{d_i}})_{d_1 \in [0, \hat{d}_1], \dots, d_m \in [0, \hat{d}_m]}, \widetilde{g}, (\widetilde{g}^{\beta_j})_{j \in [1, m]})$ 5: $\operatorname{srs}_{\operatorname{afg}} = (p, \boldsymbol{g}_{\operatorname{afg}} = (\widetilde{g}^{\gamma^{j-1}})_{j \in [1,n]})$ 6: $srs = (srs_{ped}, srs_{pst}, srs_{afg})$ $c_{f} \leftarrow \mathsf{Commit}(\mathsf{srs}, f)$ 1: $\phi \leftarrow \mathsf{pst.Commit}(\mathsf{srs}_{\mathsf{pst}}, f)$ \triangleright Shorthand for $\phi \leftarrow (\mathsf{pst.Commit}(\mathsf{srs}_{\mathsf{pst}}, f_i))_{i \in [1,n]}$ 2: $c_f \leftarrow afg.Commit(srs_{afg}, \phi)$ $\triangleright c_f = \langle \boldsymbol{\phi}, \boldsymbol{g}_{afg} \rangle$ $c_{\boldsymbol{v}} \leftarrow \mathsf{Commit}_{\mathsf{Eval}}(\mathsf{srs}, \boldsymbol{v}, \boldsymbol{f})$ 1: $\boldsymbol{\varphi} = (f_j(\boldsymbol{v}))_{j \in [1,n]}, c_{\boldsymbol{v}} = \text{ped.Commit}(\text{srs}_{\text{ped}}, \boldsymbol{\varphi})$ $\triangleright c_{\boldsymbol{v}} = \langle \boldsymbol{\varphi}, \boldsymbol{g}_{\text{ped}} \rangle$ $\pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \boldsymbol{f}, c_{\boldsymbol{f}}, \boldsymbol{v} \in \mathbb{Z}_p^m)$ 1: $c_{\boldsymbol{v}} \leftarrow \mathsf{Commit}_{\mathsf{Eval}}(\mathsf{srs}, \boldsymbol{v}, \boldsymbol{f})$ 2: $r = H(c_f, v, c_v)$ 3: $\hat{f} = \langle \boldsymbol{f}, \boldsymbol{r}^n \rangle, c_{\hat{f}} = \langle \boldsymbol{\phi}, \boldsymbol{r}^n \rangle, e_{\hat{v}} = \langle \boldsymbol{\varphi}, \boldsymbol{r}^n \rangle$ 4: $\pi_{pst} \leftarrow pst.Prove(srs, \hat{f}, c_{\hat{f}}, v)$ 5: $\pi_{gipa}^{[1]} \leftarrow gipa.Prove(n, \emptyset, \phi, g_{afg}, c_f, r^n, c_{\hat{f}})$ 6: $\pi_{gipa}^{[2]} \leftarrow gipa.Prove(n, \emptyset, \varphi, g_{ped}, c_v, r^n, e_{\hat{v}})$ 7: $\pi = (\pi_{pst}, \pi_{gipa}^{[1]}, \pi_{gipa}^{[2]}, c_{\hat{f}}, e_{\hat{v}})$ $1/0 \leftarrow \mathsf{Eval}(\mathsf{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi)$ 1: Parse $\pi = (\pi_{pst}, \pi_{gipa}^{[1]}, \overline{\pi_{gipa}^{[2]}}, c_{\hat{f}}, e_{\hat{v}})$ 2: $r = H(c_f, v, c_v)$ 3: $1/0 \leftarrow \mathsf{pst.Eval}(\mathsf{srs}_{\mathsf{pst}}, c_{\hat{f}}, e_{\hat{f}}, v, \pi_{\mathsf{pst}})$ $\begin{array}{l} 4: \ 1/0 \leftarrow \mathsf{gipa}.\mathsf{Verify}(n, \boldsymbol{g_{\mathsf{afg}}}, c_{\boldsymbol{f}}, \boldsymbol{r}^n, c_{\hat{f}}, \pi_{\mathsf{gipa}}^{[1]}) \\ 5: \ 1/0 \leftarrow \mathsf{gipa}.\mathsf{Verify}(n, \boldsymbol{g_{\mathsf{ped}}}, c_{\boldsymbol{v}}, \boldsymbol{r}^n, e_{\hat{f}}, \pi_{\mathsf{gipa}}^{[2]}) \end{array}$

Fig. 3: Construction of MMP Commitment Scheme

Proof. Denote $\phi = \mathsf{pst.Commit}(\mathsf{srs}, f)$ and $\phi' = \mathsf{pst.Commit}(\mathsf{srs}, f')$. Commit (srs, f') implies one of the following two cases.

Case 1. $\phi \neq \phi' \land \mathsf{afg.Commit}(\mathsf{srs}, \phi) = \mathsf{afg.Commit}(\mathsf{srs}, \phi'),$

Case 2. $\phi = \phi'$.

Case 1 breaks the binding property of AFGHO commitment scheme. Case 2 breaks the polynomial binding property of the PST commitment scheme.

Theorem 2 (Knowledge Soundness of MMP Commitment). The knowledge soundness of the MMP commitment scheme follows from the knowledge soundness of the PST commitment scheme (in the batched setting) and witnessextended emulation of the inner product argument.

Proof Sketch. The srs_{pst} is generated in the same way as in the PST knowledge soundness game (Appendix D) using trapdoor x. Upon receiving srs, \mathcal{A} outputs $\boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}$, and $\pi = (\pi_{\mathsf{pst}}, \pi_{\mathsf{gipa}}^{[1]}, \pi_{\mathsf{gipa}}^{[2]}, c_{\hat{f}}, e_{\hat{v}}, \boldsymbol{v})$, along with $\mathsf{REP}_{\mathsf{srs}}^{c_{\boldsymbol{f}}}, \mathsf{REP}_{\mathsf{srs}}^{c_{\boldsymbol{v}}}$, and $\mathsf{REP}_{\mathsf{srs}}^{\pi}$. With these, \mathcal{E} can extract \hat{f} from $\mathsf{REP}_{\mathsf{srs}}^{\pi}$. By the witness extended emulation of GIPA protocol, we know that $c_{\hat{f}} = \langle \boldsymbol{\varphi}, \boldsymbol{r}^n \rangle$, $\hat{f} = \langle \boldsymbol{f}, \boldsymbol{r}^n \rangle$, and $e_{\hat{v}} =$ $\langle \boldsymbol{\varphi}, \boldsymbol{r}^n \rangle$. \mathcal{E} rewinds the protocol with *n* different *r* and obtains $\boldsymbol{f} = (f_i)_{i \in [1,n]}$, $\boldsymbol{\phi} = (c_{f_i})_{i \in [1,n]}, \, \boldsymbol{\varphi} = (e_{\boldsymbol{v}}^{[i]})_{i \in [1,n]}.$ Finally, by the knowledge soundness of PST commitment in the batched setting, $\forall i \in [1, n], f_i(v) = e_v^{[i]}$ if Lemma 1 holds.

Efficiency and Implementation 3.4

We first summarize the theoretical time complexity of our MMP scheme below, where \mathbb{Z}_p is for field operations, \mathbb{G} , \mathbb{G} and \mathbb{G}_T are for group scalar multiplications in the corresponding group, and P is for pairing operation.

- $\begin{array}{l} \ \mathsf{Commit:} \ n \cdot \binom{m+d}{d} \mathbb{G} + nP \\ \ \mathsf{Prove:} \ (n \cdot \binom{m+d}{d} + m \cdot \binom{m+d}{d}) \mathbb{Z}_p + (n + m \cdot \binom{m+d}{d}) \mathbb{G} + n\widetilde{\mathbb{G}} + nP \\ \ \mathsf{Eval:} \ \log n\mathbb{Z}_p + \log n\mathbb{G} + m\widetilde{\mathbb{G}} + \log n\mathbb{G}_T + mP \end{array}$

The term $\binom{m+d}{d}$ in the costs of Commit and Prove is due to the maximum size of a polynomial of degree d with m variables, and in practice it could be much smaller than this theoretical upper bound.

Now we provide the benchmark results of our implementation. We implement our MMP commitment scheme in $\sim 1k$ lines of Rust code by utilizing the arkworks [3] library, where BN254 is chosen as the pairing-friendly elliptic curve. We evaluate the performance of our implementation on a PC equipped with an Intel i9-12900K CPU and 64GB of RAM. During the evaluation, we test against multiple sets of parameters, where n, the number of polynomials, ranges from 2^0 to 2^{11} , and the number of variables m and highest degree d are from 1 to 5. The benchmark results are given in Figure 4.

Since Setup is only a one-time procedure, we exclude its performance from Figure 4 and instead briefly list its performance here: for all tested parameters, Setup takes 10-20 ms to complete.

Finally, we report the (compressed) proof size. For BN254, $|\pi| = 32m +$ $960 \log n + 384$ bytes, and in general, $|\pi| = (4 + 2 \log n) |\mathbb{Z}_n| + (4 + m + 4 \log n) |\mathbb{G}|$ $2|\widetilde{\mathbb{G}}| + 2\log n|\mathbb{G}_T|.$



Fig. 4: Running time of Commit, Prove, and Eval (in milliseconds). The results are grouped in three columns according to the algorithm, and we illustrate the time complexity of these algorithms w.r.t. n, m, and d in the first, second and third row, respectively.

3.5 Zero-Knowledge MMP protocol

Our MMP commitment is neither hiding nor zero-knowledge. However, by substituting the PST commitment scheme with the hiding PST commitment scheme, we obtain hiding MMP as the hiding property of the first tier commitment scheme will be preserved in the MMP commitment. To enable zero-knowledge, we adopt the following measures.

- Encode the vector evaluations in the form of $(g^{f_i(v)})_{i \in [1,n]}$ and prove its consistency with c_v using the aggregation property of Pedersen Subvector commitment,
- Prove $(g^{f_i(v)})_{i \in [1,n]}$ are committed evaluations for c_f using batch techniques, and

srs \leftarrow Setup (λ, m, n, \hat{d}) 1: $(e, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, p) \leftarrow \mathsf{BGGen}(\lambda)$ 2: $\alpha, \beta_1, ..., \beta_m, \gamma, \delta \stackrel{R}{\leftarrow} \mathbb{Z}_p$ 3: $h = g^{\delta}, \widetilde{h} = \widetilde{g}^{\delta}$ 4: $\forall i \in [1, 2n], g_i = g^{\alpha^{i-1}}$ 5: $\forall i \in [1, n], \widetilde{g}_i = g^{\alpha^i}, h_i = h^{\alpha^i}$ 6: $\operatorname{srs}_{psv} = (p, g_{psv} = (g_i)_{i \in [1,n]}, (g_i)_{i \in [n+2,2n]}, (\widetilde{g}_i)_{i \in [1,n]}, h, (h_i)_{i \in [1,n]})$ 7: $\operatorname{srs}_{hpst} = (p, g_{hpst} = (g \prod_{i=1}^{m} \beta_i^{d_j})_{d_1 \in [0, \hat{d}_1], \dots, d_m \in [0, \hat{d}_m]}, \widetilde{g}, (\widetilde{g}^{\beta_j})_{j \in [1, m]}, \underline{h}, \widetilde{h})$ 8: $\operatorname{srs}_{\operatorname{afg}} = (p, \boldsymbol{g}_{\operatorname{afg}} = (\widetilde{g}^{\gamma^{i-1}})_{i \in [1,n]})$ 9: $srs = (srs_{pst}, srs_{afg}, srs_{psv})$ $c_f \leftarrow \mathsf{Commit}(\mathsf{srs}, f)$ ▷ Shorthand for $\phi = (\text{hpst.Commit}(\text{srs}_{\text{hpst}}, f_i; \rho_j))_{i \in [1,n]}$ 1: $\phi \leftarrow \text{hpst.Commit}(\text{srs}_{\text{hpst}}, f; \rho \xleftarrow{R}{\leftarrow} \mathbb{Z}_p^n)$ 2: $c_f \leftarrow \mathsf{afg.Commit}(\mathsf{srs}_{\mathsf{afg}}, \phi)$ $\triangleright c_f = \langle \phi, g_{afg} \rangle$ $c_{\boldsymbol{v}} \leftarrow \mathsf{Commit}_{\mathsf{Eval}}(\mathsf{srs}, \boldsymbol{v}, \boldsymbol{f})$ 1: $\varphi = (f_i(v))_{i \in [1,n]}$ 2: $c_v \leftarrow \mathsf{psv.Commit}(\mathsf{srs}, \varphi; \rho' \xleftarrow{R} \mathbb{Z}_p)$ $\triangleright c_{\boldsymbol{v}} = \langle \boldsymbol{\varphi}, \boldsymbol{g}_{\mathsf{psv}} \rangle \cdot \langle \boldsymbol{\varrho}', \boldsymbol{h} \rangle$ $\pi \leftarrow \mathsf{Prove}(\mathsf{srs}, f, c_f, v \in \mathbb{Z}_p^m, \rho = (\rho_i)_{i \in [1,n]})$ 1: $c_{\boldsymbol{v}} \leftarrow \mathsf{Commit}_{\mathsf{Eval}}(\mathsf{srs}, \boldsymbol{v}, \boldsymbol{f}; \underline{\varrho}' \xleftarrow{R}{\leftarrow} \mathbb{Z}_{p})$ 2: $r = H(c_f, v, c_v)$ 3: $c_{\hat{f}} = \langle \boldsymbol{\phi}, \boldsymbol{r}^n \rangle$ ▷ Aggregation of polynomial commitments 4: $\pi_{gipa} \leftarrow gipa.Prove(n, \emptyset, \phi, g_{afg}, c_f, r^n, c_{\hat{f}})$ ▷ Proof for aggregation of polynomial commitments 5: $\rho \stackrel{R}{\leftarrow} \mathbb{Z}_p$ 6: $c_{\hat{\boldsymbol{v}}} = \langle (g^{f_i(\boldsymbol{v})})_{i \in [1,n]}, \boldsymbol{r} \rangle \cdot \langle \varrho, \boldsymbol{h} \rangle = g^{\sum_{i=1}^n f_i(\boldsymbol{v}) \cdot r^{i-1}} \boldsymbol{h}^{\varrho}$ ▷ Aggregation of evaluation commitments 7: $c' = \prod_{i=1}^{n} \prod_{j=1, j \neq i}^{n} g_{n+1-i+j}^{f_j(v), r^{i-1}} \cdot \prod_{i=1}^{n} h_{n+1-i}^{e', r^{i-1}} / h_n^e$ $\begin{array}{l} \varrho_1, \varrho_2 \stackrel{R}{\leftarrow} \mathbb{Z}_p, D = g^{\varrho_1} h^{\varrho_2}, \eta = H(r, c_{\hat{v}}, c', D), b_1 = \varrho_1 - \eta(\sum_{i=1}^n f_i(\boldsymbol{v}) \cdot r^{i-1}), b_2 = \varrho_2 - \eta \cdot \varrho_1 \\ \pi_{\mathsf{agg}} = (c_{\hat{v}}, c', D, b_1, b_2) \\ & \triangleright \text{ Proof for aggregation of evaluation com} \end{array}$ $\pi_{\mathsf{agg}} = (c_{\hat{v}}, c', D, b_1, b_2) \qquad \triangleright \text{ Proof for aggregation of evaluation commitments} \\ 8: \forall i \in [1, n], \{q_j^i(\mathbf{X})\}_{j \in [1, m]} : f_i(\mathbf{X}) - f_i(\mathbf{v}) = \sum_{j=1}^m q_j^i(\mathbf{X})(X_j - v_j) \\ \forall i \in [1, n], \{q_j^i(\mathbf{X})\}_{j \in [1, m]} : f_i(\mathbf{X}) - f_i(\mathbf{v}) = \sum_{j=1}^m q_j^i(\mathbf{X})(X_j - v_j) \\ \forall i \in [1, n], \{q_j^i(\mathbf{X})\}_{j \in [1, m]} : f_i(\mathbf{X}) - f_i(\mathbf{v}) = \sum_{j=1}^m q_j^i(\mathbf{X})(X_j - v_j) \\ \forall i \in [1, n], \{q_j^i(\mathbf{X})\}_{j \in [1, m]} : f_i(\mathbf{X}) - f_i(\mathbf{v}) = \sum_{j=1}^m q_j^i(\mathbf{X})(X_j - v_j) \\ \forall i \in [1, n], \{q_j^i(\mathbf{X})\}_{j \in [1, m]} : f_i(\mathbf{X}) - f_i(\mathbf{v}) = \sum_{j=1}^m q_j^i(\mathbf{X})(X_j - v_j) \\ \forall i \in [1, n], \{q_j^i(\mathbf{X})\}_{j \in [1, m]} : f_i(\mathbf{X}) = \sum_{j=1}^m q_j^i(\mathbf{X})(X_j - v_j) \\ \forall i \in [1, n], \{q_j^i(\mathbf{X})\}_{j \in [1, m]} : f_i(\mathbf{X}) = \sum_{j=1}^m q_j^i(\mathbf{X})(X_j - v_j) \\ \forall i \in [1, n], \{q_j^i(\mathbf{X})\}_{j \in [1, m]} : f_i(\mathbf{X}) = \sum_{j=1}^m q_j^i(\mathbf{X})(X_j - v_j) \\ \forall i \in [1, n], \{q_j^i(\mathbf{X})\}_{j \in [1, m]} : f_i(\mathbf{X}) = \sum_{j=1}^m q_j^i(\mathbf{X})(X_j - v_j) \\ \forall i \in [1, n], \{q_j^i(\mathbf{X})\}_{j \in [1, m]} : f_i(\mathbf{X}) = \sum_{j=1}^m q_j^i(\mathbf{X})(X_j - v_j) \\ \forall i \in [1, n], \{q_j^i(\mathbf{X})\}_{j \in [1, m]} : f_i(\mathbf{X}) \in [1, m] \}$ $\forall j \in [1,m], \boldsymbol{q}_j(\boldsymbol{X}) = (q_i^i(\boldsymbol{X}))_{i \in [1,n]}, q_j(\boldsymbol{X}) = \langle \boldsymbol{q}_j(\boldsymbol{X}), \boldsymbol{r}^n \rangle$ $\begin{aligned} \forall j \in [1, m], \mu_j &\stackrel{R}{\leftarrow} \mathbb{Z}_p \\ \pi_{\mathsf{pst}} &= (g^{q_j(\beta)} \cdot h^{\cdot \mu_j})_{j \in [1, m]} \\ \theta &= g^{\sum_{j=1}^m v_j \cdot \mu_j + \sum_{i=1}^n \rho_i \cdot r^{i-1} - \varrho} \cdot \prod_{j=1}^m g^{-\beta_j \cdot \mu_j} \end{aligned}$ \triangleright Batched proof for correct evaluations $1/0 \leftarrow \mathsf{Eval}(\mathsf{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi)$ 1: gipa.Verify $(c_f, c_{\hat{f}}, g_{afg}, r^n, \pi_{gipa}) \stackrel{?}{=} 1$ 2: Parse $\pi_{agg} = (c_{\hat{v}}, c', D, b_1, b_2)$ 3: $\eta = H(r, c_{\hat{v}}, c', D), g^{b_1} h^{b_2} c_{\hat{v}}^{\eta} \stackrel{?}{=} D, e(c_v, \prod_{i=1}^n \widetilde{g}_{n+1-i}^{r^{i-1}}) \stackrel{?}{=} e(c', \widetilde{g}) \cdot e(c_{\hat{v}}, \widetilde{g}_n)$ 4: Parse π_{pst} as $(Q_j)_{j \in [1,m]}$ 5: $e(c_{\hat{f}}/c_{\hat{v}}, \widetilde{g}) \stackrel{?}{=} \prod_{i=1}^{m} e(Q_j, \widetilde{g}^{\beta_j - v_j}) \cdot e(\theta, \widetilde{h})$

Fig. 5: Construction of Zero-Knowledge MMP Commitment Scheme

- Randomize all commitments and proof elements.

Denote

- (psv.Setup, psv.Commit, psv.Prove, psv.Eval) as Pedersen subvector commitment
- (hpst.Setup, hpst.Commit, hpst.Prove, hpst.Eval) as hiding PST commitment.

A zero-knowledge MMP protocol is presented in 5, and we prove the security of our scheme in Appendix E.

4 Range proof for Pedersen Subvector Commitment

From a very high level, our range proof for Pedersen vector commitment works as follows. Given a Pedersen commitment $c_{\boldsymbol{v}} = \prod_{i=1}^{n} g_i^{v_i} g^{\boldsymbol{\omega}}$, the prover computes $c = \tilde{g}_l^{\sum_{i=1}^{n} z^{i+1} \cdot v_i}$ with challenge z and proves that c has the claimed form. With c, the prover can use the technique of batched range proof for Pedersen commitments in [16] to prove $\forall i \in [1, n], v_i \in [0, 2^l - 1]$.

More specifically, as described in [16], to prove $v_i \in [0, 2^l - 1]$ for $i \in [1, n]$, it is sufficient to demonstrate the knowledge of vectors \boldsymbol{a} and $\overline{\boldsymbol{a}}$ such that

$$a \circ \overline{a} = \mathbf{0}^{nl}$$

$$a - \overline{a} - \mathbf{1}^{nl} = \mathbf{0}^{nl}$$

$$\langle a, V_1 \rangle = v_1$$

$$\dots$$

$$\langle a, V_i \rangle = v_i$$

$$\dots$$

$$\langle a, V_n \rangle = v_n$$
(1)

where $V_i = (\underbrace{0, \dots, 0}_{(i-1)l}, \underbrace{2^0, 2^1, \dots, 2^{l-1}}_l, \underbrace{0, \dots, 0}_{(n-i)l}).$

Choose $y, z \stackrel{R}{\leftarrow} \mathbb{Z}_p$, the above equations are aggregated as below. With only negligible probability, Equation 1 holds while Equation 2 doesn't.

$$\langle \boldsymbol{l}, \boldsymbol{r} \rangle = \langle \boldsymbol{v}, z^2 \cdot \boldsymbol{z}^n \rangle + \zeta(y, z)$$
 (2)

where

$$oldsymbol{l} = oldsymbol{a} - z \cdot oldsymbol{1}^{nl},$$

 $oldsymbol{r} = oldsymbol{y}^{nl} \circ (\overline{oldsymbol{a}} + z \cdot oldsymbol{1}^{nl}) + \sum_{i=1}^{n} z^{i+1} \cdot V_i,$
 $\zeta(y, z) = (z - z^2) \cdot \langle oldsymbol{1}^{nl}, oldsymbol{y}^{nl}
angle - \sum_{i=1}^{n} z^{i+2} \cdot \langle oldsymbol{1}^{l}, oldsymbol{2}^{l}
angle.$

Based on equation 2, [16] introduced a batched range proof for n Pedersen commitments $\boldsymbol{c} = (g^{v_i})_{i \in [1,n]}$, which essentially checks

$$g^{\langle \boldsymbol{l},\boldsymbol{r}\rangle} \stackrel{?}{=} \langle \boldsymbol{c}, z^2 \cdot \boldsymbol{z} \rangle \cdot g^{\zeta(y,z)} = g^{\sum_{i=1}^n z^{i+1} \cdot v_i} \cdot g^{\zeta(y,z)}.$$

However, as the vector elements are encoded in different public parameters (i.e., g_1, \ldots, g_n), Pedersen subvector commitment cannot straightforwardly aggregate vectors as batched Pedersen commitments do (which is simply the inner product of c and $z^2 \cdot z$). To fill in this gap, we introduce AoK_{agg}, a proof for the correct aggregation of elements of Pedersen subvector commitment. With a Pedersen subvector commitment $c_{\boldsymbol{v}} = \langle \boldsymbol{v}, \boldsymbol{g} \rangle$, AoK_{agg} between \mathcal{P} and \mathcal{V} is defined as below.

– Upon receiving challenge z from \mathcal{V}, \mathcal{P} computes a commitment on aggregated elements of v_S such that $c = g^{\sum_{i \in S} z^{\sigma(i)+1} v_i}$ and a proof element

$$c' = \prod_{i \in S} \prod_{j=1, j \neq i}^{n} g_{n+1-i+j}^{z^{\sigma(i)+1} \cdot v_j}.$$

- \mathcal{V} checks the correctness of aggregation by 1) checking $e(c_{\boldsymbol{v}}, \prod_{i \in S} \widetilde{g}_{n+1-i}^{z^{\sigma(i)+1}}) \stackrel{?}{=} e(c', \widetilde{g}) \cdot e(c, \widetilde{g}_n)$, and 2) runs Schnorr's protocol on c with \mathcal{P} : \mathcal{P} picks $\varrho \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and sends $D = g^{\varrho}$ to \mathcal{V}

 - \mathcal{V} sends $\eta \stackrel{R}{\leftarrow} \mathbb{Z}_p$ to \mathcal{P} . \mathcal{P} sends $b = \varrho \eta \cdot (\sum_{i \in S} z^{\sigma(i)+1} \cdot v_i)$ to \mathcal{V} .
 - \mathcal{V} checks $g^b \cdot c^\eta \stackrel{?}{=} D$

PROOF SKETCH. This protocol satisfies completeness. The witness-extend emulation can be proved in the AGM model by rewinding the protocol with at most n' different z and extracting v from $\mathsf{REP}_{\mathsf{srs}}^{c'}$. $\mathsf{AoK}_{\mathsf{agg}}$ can be transformed into a zero-knowledge argument of knowledge by randomizing c and c' as what is done in Figure 6.

4.1Construction

Our ZKRP protocol for Pedersen subvector commitment is a combination of AoK_{agg} and the batched ZKRP for Pedersen commitments in [16]. With a Pedersen subvector commitment generated by

```
- srs \leftarrow psv.Setup(\lambda, n),
-c_{\boldsymbol{v}} \leftarrow \mathsf{psv.Commit}(\mathsf{srs}, \boldsymbol{v}).
```

Range proof for a Pedersen subvector commitment in asymmetric bilinear pairing group $(\mathbb{G}, \mathbb{G}, \mathbb{G}_T)$ with order p can be formally expressed as

$$\mathsf{PoK} \left\{ \begin{array}{l} (c_{\boldsymbol{v}}, \mathsf{srs}, S) \ (\boldsymbol{v}, \omega) : \\ c_{\boldsymbol{v}} = \langle \boldsymbol{v}, \boldsymbol{g} \rangle \cdot \langle \boldsymbol{\omega}, \boldsymbol{h} \rangle \\ \land \quad \forall i \in S, v_i \in [0, 2^l - 1] \end{array} \right\}$$

Construction of the range proof for Pedersen subvector commitment is shown in Figure 6. Again, we highlight that which is required only to achieve hiding in commitments and zero-knowledge in the protocols in red.

$$\begin{array}{ll} \frac{\mathcal{P}(\mathrm{sr.} v, v, c_{\mathrm{sr.}} S = \{i_{j}\}_{j \in [1,n']})}{\sigma : \forall i_{j} \in S, \sigma(i_{j}) = j} & \mathcal{V}(\mathrm{srs.} c_{\mathrm{sr.}} S = \{i_{j}\}_{j \in [1,n']}) \\ \overline{\sigma} : \forall i_{j} \in S, \sigma(i_{j}) = j \\ \overline{\sigma} \in \{i_{j}\}_{j \in [1,n']}) & \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = j) \\ \overline{\sigma} = \{i_{j}\}_{j \in [1,n']}) & \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = j) \\ \overline{\sigma} = \{i_{j}\}_{j \in [1,n']}) & \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = j) \\ \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = j) \\ \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = j) \\ \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = j) \\ \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = j) \\ \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = j) \\ \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = j) \\ \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = j) \\ \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = j) \\ \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = i) \\ \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = i) \\ \overline{\sigma} : \nabla (s_{j} \in S, \sigma(i_{j}) = i) \\ \overline{\sigma} : \overline{$$

Fig. 6: Range Proof for Pedersen Subvector Commitment. The proof size is linear to n'l due to the transfer of l_x and r_x . As suggested in [16], \mathcal{P} and \mathcal{V} can initiate an inner product argument to achieve logarithmic proof size. Please refer to [16,21] for details.

Security Analysis 4.2

The completeness holds by construction. The proof for witness extended emulation is the same as that of batched range proof for Pedersen commitments, please refer to [16, Appendix C]. The zero-knowledge simulator uses the trapdoor of the common reference string srs, i.e., α , and simulates the protocol as below:

- $\begin{aligned} &- \text{ Choose } z, \hat{v}, \gamma \xleftarrow{R} \mathbb{Z}_p \\ &- \text{ Let } c' = c_{\boldsymbol{v}}^{\sum_{i=1}^n \alpha^{n+1-i} \cdot z^{i+1}} \cdot g^{\alpha^n \cdot \hat{v}} \cdot h_n^{\gamma}, c = g^{\hat{v}} h^{\gamma}. \end{aligned}$
- Run Schnorr's protocol on simulated c and obtain b_1, b_2, D .
- Choose all other proof elements and challenges randomly from their domains except

$$T_{1} = (g^{t_{x}}h^{\tau_{x}}/(c \cdot \tilde{g}^{\zeta(y,z)} \cdot T_{2}^{x^{2}}))^{x^{-1}}$$

$$B = (J^{l_x+z} k'^{r_x-z \cdot y^{n'_l}} h^{\rho_x} / (A \cdot \prod_{i=1} k_{[(i-1) \cdot n, j \cdot n-1]}^{'z^{j-1} \cdot 2^l}))^{x^{-1}})$$

- Run inner-product argument with the simulated l_x and r_x .

4.3 **Optimization and Efficiency**

Verification Optimization. The verification complexity can be improved at the cost of communication efficiency. Instead of choosing J and K randomly from \mathbb{G}^n , we set $J = (J^{\theta_1^{i-1}})_{i \in [1,n']}$ and $K = (K^{\theta_2^{i-1}})_{i \in [1,n']}$ where $J, K \leftarrow$ $\mathbb{G}, \theta_1, \theta_2 \stackrel{R}{\leftarrow} \mathbb{Z}_p$. With the optimization methods mentioned in Section 2.2, \mathcal{V} can outsource the underlying inner product operations to the prover, reducing the verification complexity from O(nl) to $O(n) + O(\log(nl))$.

We compared our ZKRP protocol, both with and without verifier optimization, to the protocol presented in [44], in Table 1. Because [44] only provides a range proof for all committed elements, we limit our comparison to the case where n = n'. As shown in the table, [44] outperforms our construction in terms of proof size, while our optimized version is better in terms of verification time. Moreover, our scheme is also generic as 44 only supports proof for vectors with small norms.

Table 1: Comparison of performance between [44] and our ZKRP protocol

	Prover time	Proof size	Verifier time
[44]	$(2nl)\mathbb{G} + (n+1)\widetilde{\mathbb{G}}$	$3\mathbb{G}$	$(2nl+1)\mathbb{G} + (nl)\widetilde{\mathbb{G}} + 4P$
This paper	$(10nl+4log(nl)+2n+13)\mathbb{G}$	$(2\log(nl)+7)\mathbb{G}+7\mathbb{Z}_p$	$(6nl+2log(nl)+n+11)\mathbb{G}+3P$
This paper+ ${\cal V}$	$(10nl+4log(nl)+2n+13)\mathbb{G}$	$(2\log(nl) + 7)\mathbb{G} + (9 + 2\log(nl))\mathbb{Z}_p$	$(8log(nl) + n + 11)\mathbb{G} + 3P$

23

Applications $\mathbf{5}$

aHyperplonk 5.1

Our MMP scheme can also be applied to SNARK proof aggregation, producing a single aggregated proof for k instances with sublinear proof size and verification time in k. Unlike existing SNARK aggregation schemes like SnarkPack [35] solely for Groth16 [39] and aPlonk [2] solely for Plonk [34], our scheme is applicable to more recent SNARKs powered by multivariate polynomial commitments, such as HyperPlonk [26], Spartan [55], etc. The key idea behind our aggregation technique is similar to aPlonk [2]. When proving multiple instances, instead of running the proof generation algorithm and committing to polynomials for each instance individually, the prover generates a single commitment to all these polynomials using MMP. Later, upon a challenge from the verifier, the prover opens the committed polynomials at the challenge point in batches and produces a proof attesting to the validity of all evaluation results.

Here we briefly review HyperPlonk and showcase aHyperPlonk, the integration of our MMP scheme with HyperPlonk. For clarity, we consider a minimal (Hyper)Plonk arithmetization with 3 types of wires L, R, O (L, R for inputs and O for output) and 2 types of selectors A, M (A for addition and M for multiplication). The *i*-th constraint (gate) in the circuit enforces that $A_i(L_i + R_i) + M_i(L_i \cdot R_i) = O_i.$

In HyperPlonk, an indexer \mathcal{I} is responsible for preprocessing the circuit. Consider a circuit C with $n = 2^{\mu}$ constraints. I needs to generate the selector polynomials $q_0(\mathbf{X})$ and $q_1(\mathbf{X})$, which satisfy $q_0(\langle x \rangle_{\mu}) = A_x$ and $q_1(\langle x \rangle_{\mu}) = M_x$ for all $x \in [0, 2^{\mu})$. Here, $\langle x \rangle_{\mu} \in \{0, 1\}^{\mu}$ is the binary representation of x whose maximal bit-length is μ . Moreover, a wiring polynomial $\sigma(\mathbf{X})$ is also computed in order to ensure the equality of witnesses across multiple constraints.

Then, when generating a proof for witnesses satisfying \mathcal{C} , the prover derives a witness polynomial $w(\mathbf{X})$, such that for all $x \in [0, 2^{\mu}), w(\langle 0 \rangle_2, \langle x \rangle_{\mu}) =$ $L_x, w(\langle 1 \rangle_2, \langle x \rangle_\mu) = R_x$, and $w(\langle 2 \rangle_2, \langle x \rangle_\mu) = O_x$. We further partition w into 3 partial polynomials w_0, w_1 and w_2 , where $w_i \coloneqq w(\langle i \rangle_2, \mathbf{X})$.

Gate Identity. In order to convince the verifier, the prover needs to first compute the gate identity polynomial $f(\mathbf{X}) \coloneqq q_0(\mathbf{X})(w_0(\mathbf{X}) + w_1(\mathbf{X})) + q_1(\mathbf{X})(w_0(\mathbf{X}) + w_1(\mathbf{X}))$ $w_1(\mathbf{X})) - w_2(\mathbf{X})$ and then prove that $f(\langle x \rangle_{\mu}) = 0$ for all $x \in [0, 2^{\mu})$, i.e., the constraints hold for all witness assignments.

Wiring Identity. Furthermore, the prover should also prove the correctness of wiring, i.e., the equality of witness assignments across multiple constraints. To this end, the prover proves that $w(\langle x \rangle_{\mu+2}) = w(\hat{\sigma}(\langle x \rangle_{\mu+2}))$ for all $x \in [0, 2^{\mu+2})$.

Now we are ready to describe how to aggregate k HyperPlonk instances using our MMP commitment scheme. To achieve this, the prover \mathcal{P} follows the steps below.

- 1. Generate an MMP commitment $c_w := \mathsf{Commit}(ck, \{w_0^{[j]}, w_1^{[j]}, w_2^{[j]}\}_{i=0}^{k-1}).$
- 2. Get challenges β, γ , and δ from \mathcal{V} . 3. Compute $\hat{w}_i(\mathbf{X}) \coloneqq \sum_{j=0}^{k-1} \delta^j w_i^{[j]}(\mathbf{X})$ for all $i \in [0,2]$.

4. Compute the product polynomial $\tilde{v}(\mathbf{X})$ such that $\tilde{v}(\langle 2^{\mu}-1\rangle_{\mu},0)=1$ and for all $x \in [0, 2^{\mu})$,

$$\tilde{v}(0, \langle x \rangle_{\mu}) = \prod_{i=0}^{2} \frac{\hat{w}_{i}(\langle x \rangle_{\mu}) + \beta s_{\mathrm{id}}(\langle i \rangle_{2}, \langle x \rangle_{\mu}) + \gamma}{\hat{w}_{i}(\langle x \rangle_{\mu}) + \beta \sigma_{i}(\langle x \rangle_{\mu}) + \gamma}$$
$$\tilde{v}(1, \langle x \rangle_{\mu}) = \tilde{v}(\langle x \rangle_{\mu}, 0) \cdot \tilde{v}(\langle x \rangle_{\mu}, 1).$$

Define the partial polynomials of $\tilde{v}(\mathbf{X})$ as $\tilde{v}_0(\mathbf{X}) \coloneqq \tilde{v}(0, \mathbf{X}), \tilde{v}_1(\mathbf{X}) \coloneqq \tilde{v}(1, \mathbf{X})$. Note that given $\tilde{v}_0(\mathbf{X})$ and $\tilde{v}_1(\mathbf{X})$, one can derive $\tilde{v}(\mathbf{X}, b) \coloneqq (1 - X_0) \cdot \tilde{v}_0(X_1, \ldots, X_{\mu-1}, b) + X_0 \cdot \tilde{v}_1(X_1, \ldots, X_{\mu-1}, b)$ for $b \in \{0, 1\}$.

- 5. Generate an MMP commitment $c_v \coloneqq \mathsf{Commit}(ck, \{\tilde{v}_0, \tilde{v}_1\}).$
- 6. Get challenges $\alpha, \boldsymbol{\rho} = \{\rho_0, \dots, \rho_{\mu-1}\}$ from \mathcal{V} .
- 7. For the virtual polynomials below,

$$f_j(\mathbf{X}) \coloneqq q_0(\mathbf{X})(w_0^{[j]}(\mathbf{X}) + w_1^{[j]}(\mathbf{X})) + q_1(\mathbf{X})(w_0^{[j]}(\mathbf{X}) \cdot w_1^{[j]}(\mathbf{X})) - w_2^{[j]}(\mathbf{X})$$
$$Q_1(\mathbf{X}) \coloneqq \tilde{v}(1, \mathbf{X}) - \tilde{v}(\mathbf{X}, 0)\tilde{v}(\mathbf{X}, 1)$$
$$Q_2(\mathbf{X}) \coloneqq \prod_{i=0}^2 (\hat{w}_i(\mathbf{X}) + \beta s_{\mathrm{id}}(\langle i \rangle_2, \mathbf{X}) + \gamma) - \tilde{v}(0, \mathbf{X}) \prod_{i=0}^2 (\hat{w}_i(\mathbf{X}) + \beta \sigma_i(\mathbf{X}) + \gamma)$$

prove that $\sum_{x=0}^{2^{\mu}-1} F(\langle x \rangle_{\mu}) \cdot eq(\langle x \rangle_{\mu}, \boldsymbol{\rho}) = 0$, where $F(\mathbf{X}) = \sum_{j=0}^{k-1} \alpha^{j} f_{j}(\mathbf{X}) + \alpha^{k} Q_{1}(\mathbf{X}) + \alpha^{k+1} Q_{2}(\mathbf{X}), eq(\mathbf{X}, \boldsymbol{\rho}) = \prod_{i=0}^{\mu-1} (X_{i}\rho_{i} + (1 - X_{i})(1 - \rho_{i}))$. This is done iteratively for $i = \mu - 1$ down to 0:

(a) Generate a univariate polynomial commitment (e.g., using KZG) c_{ri} ≔ kzg.Commit(ck_{kzg}, r_i), where r_i(X) = ∑^{2ⁱ-1}_{b=0} F(⟨b⟩_i, X, ζ_{i+1},..., ζ_{μ-1}) · eq(⟨b⟩_i, X, ζ_{i+1},..., ζ_{μ-1}, ρ).
(b) Cat a shallow ma ζ from W

- (b) Get a challenge ζ_i from \mathcal{V} .
- 8. Run the univariate batch-opening algorithm for evaluations $\{r_i(0), r_i(1), r_i(\zeta_i)\}_{i=0}^{\mu-1}$.
- 9. Compute the committed evaluations and the proofs below, where Prove' is a wrapper of Prove that also returns the result of Commit_{Eval}:

$$\begin{split} \eta_{q,\sigma}, \pi_{q,\sigma} &\coloneqq \mathsf{Prove}'(ck, \{q_0, q_1, \sigma_0, \sigma_1, \sigma_2\}, c_{q,\sigma}, \{\zeta_i\}_{i=0}^{\mu-1}) \\ \eta_w, \pi_w &\coloneqq \mathsf{Prove}'(ck, \{w_0^{[j]}, w_1^{[j]}, w_2^{[j]}\}_{j=0}^{k-1}, c_w, \{\zeta_i\}_{i=0}^{\mu-1}) \\ \eta_v^{[0]}, \pi_v^{[0]} &\coloneqq \mathsf{Prove}'(ck, \{\tilde{v}_0, \tilde{v}_1\}, c_v, \{\zeta_i\}_{i=0}^{\mu-1}) \\ \eta_v^{[1]}, \pi_v^{[1]} &\coloneqq \mathsf{Prove}'(ck, \{\tilde{v}_0, \tilde{v}_1\}, c_v, \{\zeta_i\}_{i=1}^{\mu-1} \cup \{0\}) \\ \eta_v^{[2]}, \pi_v^{[2]} &\coloneqq \mathsf{Prove}'(ck, \{\tilde{v}_0, \tilde{v}_1\}, c_v, \{\zeta_i\}_{i=1}^{\mu-1} \cup \{1\}) \\ \eta_v^{[3]}, \pi_v^{[3]} &\coloneqq \mathsf{Prove}'(ck, \{\tilde{v}_0, \tilde{v}_1\}, c_v, \{1\}_{i=1}^{\mu-1} \cup \{0\}) \end{split}$$

10. Define the meta-verification relation $\mathcal{R}_{meta}(x, w)$, where

$$\begin{aligned} x &\coloneqq (\beta, \gamma, \delta, \alpha, \{\rho_i\}_{i=0}^{\mu-1}, \{\zeta_i\}_{i=0}^{\mu-1}, \eta_{q,\sigma}, \eta_w, \{\eta_v^{[i]}\}_{i=0}^3, e_F) \\ w &\coloneqq (e_{q_0}, e_{q_1}, e_{\sigma_0}, e_{\sigma_1}, e_{\sigma_2}, \{e_{w_0}^{[j]}, e_{w_1}^{[j]}, e_{w_2}^{[j]}\}_{j=0}^{k-1}, \{e_{v_0}^{[i]}, e_{v_1}^{[i]}\}_{i=0}^3) \end{aligned}$$

Generate a SNARK proof π_{meta} for \mathcal{R}_{meta} , attesting that

$$\begin{split} e_{\hat{w}_{0}} &\coloneqq \sum_{j=0}^{k-1} \delta^{j} e_{w_{0}}^{[j]}, e_{\hat{w}_{1}} \coloneqq \sum_{j=0}^{k-1} \delta^{j} e_{w_{1}}^{[j]}, e_{\hat{w}_{2}} \coloneqq \sum_{j=0}^{k-1} \delta^{j} e_{w_{2}}^{[j]}, id \coloneqq \sum_{i=0}^{\mu-1} 2^{i} \zeta_{i} \\ e_{f_{j}} &\coloneqq e_{q_{0}} (e_{w_{0}}^{[j]} + e_{w_{1}}^{[j]}) + e_{q_{1}} (e_{w_{0}}^{[j]} \cdot e_{w_{1}}^{[j]}) - e_{w_{2}}^{[j]} \\ e_{Q_{1}} &\coloneqq e_{v_{1}}^{[0]} - ((1 - \zeta_{0}) e_{v_{0}}^{[1]} + \zeta_{0} e_{v_{1}}^{[1]})((1 - \zeta_{0}) e_{v_{0}}^{[2]} + \zeta_{0} e_{v_{1}}^{[2]}) \\ e_{Q_{2}} &\coloneqq \prod_{i=0}^{2} (e_{\hat{w}_{i}} + \beta(i + 2^{2} \cdot id) + \gamma) - e_{v_{0}}^{[0]} \prod_{i=0}^{2} (e_{\hat{w}_{i}} + \beta e_{\sigma_{i}} + \gamma) \\ \eta_{q,\sigma} &= \text{Commit}_{\text{Eval}}(ck, \{e_{q_{0}}, e_{q_{1}}, e_{\sigma_{0}}, e_{\sigma_{1}}, e_{\sigma_{2}}\}) \\ \wedge \eta_{w} &= \text{Commit}_{\text{Eval}}(ck, \{e_{w_{0}}^{[j]}, e_{w_{1}}^{[j]}, e_{w_{2}}^{[j]}\}_{j=0}^{k-1}) \\ \wedge \eta_{v}^{[0]} &= \text{Commit}_{\text{Eval}}(ck, \{e_{v_{0}}^{[0]}, e_{v_{1}}^{[1]}\}) \\ \wedge \eta_{v}^{[1]} &= \text{Commit}_{\text{Eval}}(ck, \{e_{v_{0}}^{[0]}, e_{v_{1}}^{[1]}\}) \\ \wedge \eta_{v}^{[2]} &= \text{Commit}_{\text{Eval}}(ck, \{e_{v_{0}}^{[2]}, e_{v_{1}}^{[2]}\}) \\ \wedge \eta_{v}^{[3]} &= \text{Commit}_{\text{Eval}}(ck, \{e_{v_{0}}^{[2]}, e_{v_{1}}^{[2]}\}) \\ \wedge \eta_{v}^{[3]} &= \text{Commit}_{\text{Eval}}(ck, \{e_{v_{0}}^{[2]}, e_{v_{1}}^{[2]}\}) \\ \wedge e_{v_{1}}^{[3]} &= 1 \wedge e_{F} &= \sum_{j=0}^{k-1} \alpha^{j} e_{f_{j}} + \alpha^{k} e_{Q_{1}} + \alpha^{k+1} e_{Q_{2}} \end{split}$$

Note that we follow the meta-verification approach in [2] and require the prover to generate a SNARK proof π_{meta} for \mathcal{R}_{meta} , which is necessary for keeping the final proof size and verifier time sublinear, as otherwise the verifier needs to check all the k identities. Such a \mathcal{R}_{meta} costs O(k) constraints when encoded as an arithmetic circuit. By choosing an appropriate SNARK (e.g., HyperPlonk itself), the additional cost for proving this relation will be O(k), whereas the size and verification cost of π_{meta} will be $O(\log k)$.

In Table 2 we compare the performance of HyperPlonk and aHyperPlonk for multiple instances in terms of prover time, proof size, and verifier time.

Prover timeProof sizeVerifier timeHyperPlonkO(kn) $O(k \log n)$ $O(k \log n)$ aHyperPlonkO(kn) $O(\log k + \log n)$ $O(\log k + \log n)$

Table 2: Performance of HyperPlonk and a HyperPlonk for k instances, each with n constraints.

5.2 Zero-Knowledge proof for Vehicle GPS Driving Trace

Vehicle owners sometimes need to provide their driving records to verify their presence or absence at specific locations during certain times, raising significant

privacy concerns. To protect privacy in such scenarios, we propose a ZKP system for vehicle geolocation verification. As we will show later, it is essentially a zero-knowledge proof for MMP commitment. We assume the vehicle has secure hardware (a secure element such as e.g. TPM, TrustZone, SGX) and can securely compute functions of its GPS coordinates. The vehicle owners can then generate proofs of whether they have/have not passed through specific locations without disclosing additional information.

A straightforward idea is to use a cryptographic accumulator to store all GPS coordinate readings and then prove the membership/non-membership of a specific location. However, this approach is not practical due to the large number of GPS locations. Note that the accumulator not only has to record new GPS coordinate readings, but also all GPS coordinates within error ranges (typically $\pm 10 \text{ m to } \pm 30 \text{ m}$). Suppose we add a new GPS coordinate reading whenever the forth decimal changes (which occurs every $\approx 10 \text{ meters}$), e.g., (66.6666, 88.8888), (66.6667, 88.8889) if the errors is ± 10 . It is certainly not desirable to store all these GPS coordinates in the accumulator.



Fig. 7: Determining whether a driver passed near a given location can be reduced to a 'point-in-rectangles' problem.

Observing that, in general, the driveways are straight or nearly straight. Therefore, a driving trace can be represented with a sequence of straight lines. Considering the GPS errors, as shown in Fig. 7, the driving route can be transformed into a set of rectangles with a certain width (e.g. 20 m if GPS errors are 10 m). Therefore, to prove that if a vehicle has or has not passed through some GPS coordinate P, it is sufficient to prove that if P falls into one of the rectangles or not. We refer to the former "point-in-rectangles" and the latter "point-out-of-rectangles" problems.

Below we show that these "point-in-rectangles" and "point-out-of-rectangles" problems can be mathematically represented as evaluations of MMP commitments.

POINT-IN-RECTANGLE.



Suppose an arbitrary vertex $E_0 = (x_0, y_0)$ of a rectangular and its two adjacent vertices, $E_1 = (x_1, y_1)$ and $E_2 = (x_2, y_2)$, a point P = (x, y) is inside the rectangle if and only if

$$0 < E_0 P \cdot E_0 E_1 < E_0 E_1 \cdot E_0 E_1 \tag{3}$$

$$0 < E_0 P \cdot E_0 E_2 < E_0 E_2 \cdot E_0 E_2 \tag{4}$$

In these formulas, E_0P is the vector between E_0 and P, and the dot is the scalar product of two vectors. These constraints can be equivalently⁴ expressed as

$$(E_0 P \cdot E_0 E_1) (E_0 P \cdot E_0 E_1 - E_0 E_1 \cdot E_0 E_1) < 0$$
(5)

$$(E_0 P \cdot E_0 E_2) (E_0 P \cdot E_0 E_2 - E_0 E_2 \cdot E_0 E_2) < 0 \tag{6}$$

For some fixed rectangle, the left hand side of either Equation 5 or Equation 6 can be regarded as two polynomials, denoted by f_1 and f_2 , on variables x and y such that for $b = \{1, 2\}$,

$$f_b(x,y) = f_{00}^b + f_{01}^b x + f_{02}^b x^2 + f_{10}^b y + f_{11}^b xy + f_{20}^b y^2$$

where the coefficients f_{00}^b , f_{01}^b , f_{02}^b , f_{10}^b , f_{11}^b , and f_{20}^b are computed from $x_0, y_0 x_b$, y_b . Therefore, a point P = (x, y) is inside the rectangle if and only if $f_1(x, y) < 0 \land f_2(x, y) < 0$. Hence, P is outside the rectangle if and only if $f_1(x, y) > 0 \lor f_2(x, y) > 0$.

As such, the "point-in-rectangle" problem is transformed into a range proof for MMP commitment scheme. The same method can be applied to the following scenarios and more:

POINT-IN-M-DIMENSION-HYPER-RECTANGLE. For a m dimension Hyperrectangles, given an arbitrary vertex E_0 and its adjacent vertices $E_1 \dots E_m$, a point P is inside the hyper-rectangles if and only if

$$(E_0P \cdot E_0E_1) \cdot (E_0P \cdot E_0E_1 - E_0E_1 \cdot E_0E_1) < 0$$
...
$$(E_0P \cdot E_0E_m) \cdot (E_0P \cdot E_0E_m - E_0E_m \cdot E_0E_m) < 0.$$

 \wedge

⁴ It is straightforward to see that Formulas 3, 4 implies 5, 6. The converse also holds because if, for example, 5 does not imply 3, then we must have $E_0E_1 \cdot E_0E_1 < E_0P \cdot E_0E_1 < 0$. However, $E_0E_1 \cdot E_0E_1 = (E_0E_1)^2 \ge 0$, leading to a contradiction.

POINT-IN-RECTANGLES. For k rectangles, given an arbitrary vertex E_0^i for $i \in [1, k]$, and the adjacent vertices E_1^i, E_2^i for $i \in [1, n-1]$, a point P is inside one of theses rectangles if and only if $\exists i \in [1, k]$,

$$(E_0^i P \cdot E_0^i E_1^i) \cdot (E_0^i P \cdot E_0^i E_1^i - E_0^i E_1^i \cdot E_0^i E_1^i) < 0$$

$$\wedge (E_0^i P \cdot E_0^i E_2^i) \cdot (E_0^i P \cdot E_0^i E_2^i - E_0^i E_2^i \cdot E_0^i E_2^i) < 0.$$

POINT-OUT-OF-RECTANGLES. For k rectangles, given an arbitrary vertex E_0^i for $i \in [1, k]$, and the adjacent vertices E_1^i, E_2^i for $i \in [1, n-1]$, a point P is outside of these rectangles if and only if $\forall i \in [1, k]$,

$$(E_0^i P \cdot E_0^i E_1^i) \cdot (E_0^i P \cdot E_0^i E_1^i - E_0^i E_1^i \cdot E_0^i E_1^i) > 0$$

$$/ (E_0^i P \cdot E_0^i E_2^i) \cdot (E_0^i P \cdot E_0^i E_2^i - E_0^i E_2^i \cdot E_0^i E_2^i) > 0.$$

We construct a zero-knowledge proof system for vehicle GPS traces below:

- Once vehicle GPS has recorded n/2 GPS coordinates, it calculates the corresponding n polynomials $\mathbf{f} = (f_i)_{i \in [1,n]}$, and runs $c_{\mathbf{f}} \leftarrow \mathsf{ZKMMP}.\mathsf{Commit}(\mathbf{f})$.
- Vehicle secure element computes a signature σ on c_f with its embedded signing key.
- When the vehicle owner needs to prove whether the vehicle has or has not passed through a GPS location $\mathbf{v} = (v_1, v_2)$, they first execute $c_{\mathbf{v}} \leftarrow$ ZKMMP.Commit_{Eval} (\mathbf{f}, \mathbf{v}) , which generates a Pedersen subvector commitment on $\mathbf{f}(\mathbf{v})$. They then prove: 1) that $c_{\mathbf{v}}$ is of the claimed form using ZKMMP.prove, and 2) that $\mathbf{f}(\mathbf{v})$ falls within the claimed range using ZKRP for Pedersen subvector commitments. Specifically, if a vehicle has passed through (v_1, v_2) , then for some $k \in [1, \frac{n}{2}]$, $f_{2k-1}(v_1, v_2) \leq 0$ and $f_{2k}(v_1, v_2) \leq$ 0. A ZKRP protocol is run for $c_{\mathbf{v}}$ with index set $S = \{2k-1, 2k\}$. If a vehicle has not passed through (v_1, v_2) , then for all $k \in [1, \frac{n}{2}]$, $f_{2k-1} > 0$ or $f_{2k} > 0$. We define $I = \{b_k\}_{k \in [1, \frac{n}{2}]}$ such that $b_k = 0$ if $f_k > 0$, and $b_k = 1$ otherwise. A ZKRP protocol is run on $c_{\mathbf{v}}$ with index set $S = \{2(k-1) + b_k\}_{k \in [1, \frac{n}{2}]}$.

To complete the account, the following points are needed to address:

- ZKRP protocols only apply to integers, but GPS coordinates are floatingpoint numbers. We process GPS coordinates to four decimal places (matching the GPS reading error range), and then multiply by 10⁴, making all coordinates integers.
- The ZKRP protocol cannot prove that the evaluation is greater than or less than 0, but only if it falls within a finite interval. Fortunately, in our application scenario, the polynomial coefficients are generated from GPS coordinates, and the evaluation vectors are also GPS coordinates, both of which have upper and lower limits. Hence, the polynomial evaluation also has limits, e.g., $\pm L$. In this case, > 0 is treated as the interval [0, |L|] and < 0 is the interval $[\lambda |L|, \lambda]$ where $\lambda = |p|$ and p is the order of the underlying prime field.

\

Acknowledgments. The authors would like to thank David Oswald for the useful discussions on the application scenarios of MMP commitment.

References

- Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structurepreserving signatures and commitments to group elements. Journal of Cryptology 29(2), 363–421 (Apr 2016). https://doi.org/10.1007/s00145-014-9196-7
- Ambrona, M., Beunardeau, M., Schmitt, A.L., Toledo, R.R.: aPlonK : Aggregated PlonK from multi-polynomial commitment schemes. Cryptology ePrint Archive, Report 2022/1352 (2022), https://eprint.iacr.org/2022/1352
- 3. arkworks contributors: arkworks zksnark ecosystem (2022), https://arkworks.rs
- Arun, A., Ganesh, C., Lokam, S.V., Mopuri, T., Sridhar, S.: Dew: A transparent constant-sized polynomial commitment scheme. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part II. LNCS, vol. 13941, pp. 542–571. Springer, Heidelberg (May 2023). https://doi.org/10.1007/978-3-031-31371-4_19
- Attema, T., Lyubashevsky, V., Seiler, G.: Practical product proofs for lattice commitments. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 470–499. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1 17
- Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (Aug 2009). https://doi.org/10.1007/978-3-642-03356-8 7
- Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_2
- Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 111–120. ACM Press (Jun 2013). https://doi.org/10.1145/2488608.2488623
- Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081 (2020), https://eprint.iacr.org/2020/081
- Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Halo infinite: Proof-carrying data from additive polynomial commitments. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 649–680. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0 23
- Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_12
- Bootle, J., Groth, J.: Efficient batch zero-knowledge arguments for low degree polynomials. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 561–588. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/ 978-3-319-76581-5 19
- Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (May 2000). https://doi.org/10.1007/3-540-45539-6 31

- Bowe, S., Grigg, J., Hopwood, D.: Halo: Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021 (2019), https: //eprint.iacr.org/2019/1021
- Brickell, E.F., Chaum, D., Damgård, I., van de Graaf, J.: Gradual and verifiable release of a secret. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 156– 166. Springer, Heidelberg (Aug 1988). https://doi.org/10.1007/3-540-48184-2_11
- Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press (May 2018). https://doi.org/10.1109/SP.2018.00020
- Bünz, B., Chen, B.: Protostar: Generic efficient accumulation/folding for specialsound protocols. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part II. LNCS, vol. 14439, pp. 77–110. Springer, Heidelberg (Dec 2023). https://doi.org/10.1007/ 978-981-99-8724-5 3
- Bünz, B., Chiesa, A., Lin, W., Mishra, P., Spooner, N.: Proof-carrying data without succinct arguments. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 681–710. Springer, Heidelberg, Virtual Event (Aug 2021). https: //doi.org/10.1007/978-3-030-84242-0 24
- Bünz, B., Chiesa, A., Mishra, P., Spooner, N.: Recursive proof composition from accumulation schemes. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 1–18. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/ 978-3-030-64378-2 1
- Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 677–706. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45721-1_24
- Bünz, B., Maller, M., Mishra, P., Tyagi, N., Vesely, P.: Proofs for inner pairing products and applications. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part III. LNCS, vol. 13092, pp. 65–97. Springer, Heidelberg (Dec 2021). https: //doi.org/10.1007/978-3-030-92078-4_3
- 22. Camenisch, J., Chaabouni, R., shelat, a.: Efficient protocols for set membership and range proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (Dec 2008). https://doi.org/10.1007/ 978-3-540-89255-7 15
- Campanelli, M., Gailly, N., Gennaro, R., Jovanovic, P., Mihali, M., Thaler, J.: Testudo: Linear time prover SNARKs with constant size proofs and square root size universal setup. In: Longa, P., Ràfols, C. (eds.) LATINCRYPT 2021. LNCS, vol. 12912, pp. 331–351. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/ 978-3-031-44469-2 17
- Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Heidelberg (Feb / Mar 2013). https://doi.org/10.1007/978-3-642-36362-7_5
- Chalkias, K., Cohen, S., Lewi, K., Moezinia, F., Romailler, Y.: HashWires: Hyperefficient credential-based range proofs. PoPETs 2021(4), 76–95 (Oct 2021). https://doi.org/10.2478/popets-2021-0061
- Chen, B., Bünz, B., Boneh, D., Zhang, Z.: HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In: Hazay, C., Stam, M. (eds.) EURO-CRYPT 2023, Part II. LNCS, vol. 14005, pp. 499–530. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30617-4_17

- Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45721-1 26
- Christ, M., Baldimtsi, F., Chalkias, K.K., Maram, D., Roy, A., Wang, J.: Sok: Zero-knowledge range proofs. Cryptology ePrint Archive, Paper 2024/430 (2024), https://eprint.iacr.org/2024/430, https://eprint.iacr.org/2024/430
- Couteau, G., Goudarzi, D., Klooß, M., Reichle, M.: Sharp: Short relaxed range proofs. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 609–622. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560628
- Couteau, G., Klooß, M., Lin, H., Reichle, M.: Efficient range proofs with transparent setup from bounded integer commitments. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part III. LNCS, vol. 12698, pp. 247–277. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77883-5 9
- Daza, V., Ràfols, C., Zacharakis, A.: Updateable inner product argument with logarithmic verifier and applications. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 527–557. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45374-9 18
- Esgin, M.F., Steinfeld, R., Liu, J.K., Liu, D.: Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 115–146. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/ 978-3-030-26948-7 5
- 33. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/ 978-3-319-96881-0 2
- Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over lagrange-bases for occumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019), https://eprint.iacr.org/2019/953
- Gailly, N., Maller, M., Nitulescu, A.: SnarkPack: Practical SNARK aggregation. In: Eyal, I., Garay, J.A. (eds.) FC 2022. LNCS, vol. 13411, pp. 203–229. Springer, Heidelberg (May 2022). https://doi.org/10.1007/978-3-031-18283-9 10
- Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Lineartime and field-agnostic SNARKs for R1CS. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 193–226. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2 7
- 37. Gorbunov, S., Reyzin, L., Wee, H., Zhang, Z.: Pointproofs: Aggregating proofs for multiple vector commitments. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 2007–2023. ACM Press (Nov 2020). https://doi.org/10.1145/ 3372297.3417244
- Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 05. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (Jun 2005). https://doi.org/10.1007/11496137 32
- Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5 11
- Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8_11

- Kohrita, T., Towa, P.: Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. Cryptology ePrint Archive, Paper 2023/917 (2023), https://eprint.iacr.org/2023/917, https://eprint.iacr.org/2023/ 917
- Kothapalli, A., Setty, S., Tzialla, I.: Nova: Recursive zero-knowledge arguments from folding schemes. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 359–388. Springer, Heidelberg (Aug 2022). https://doi.org/ 10.1007/978-3-031-15985-5 13
- Lai, R.W.F., Malavolta, G.: Subvector commitments with application to succinct arguments. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 530–560. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/ 978-3-030-26948-7 19
- Lavaur, T., Lacan, J.: Boomy: Batch opening of multivariate polynomial commitment. Cryptology ePrint Archive, Paper 2023/1599 (2023), https://eprint.iacr.org/ 2023/1599, https://eprint.iacr.org/2023/1599
- Lee, J.: Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part II. LNCS, vol. 13043, pp. 1–34. Springer, Heidelberg (Nov 2021). https://doi.org/10. 1007/978-3-030-90453-1 1
- Libert, B.: Vector commitments with proofs of smallness: Short range proofs and more. In: Tang, Q., Teague, V. (eds.) PKC 2024, Part II. LNCS, vol. 14602, pp. 36– 67. Springer, Heidelberg (Apr 2024). https://doi.org/10.1007/978-3-031-57722-2_ 2
- Lipmaa, H.: On diophantine complexity and statistical zero-knowledge arguments. In: Laih, C.S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg (Nov / Dec 2003). https://doi.org/10.1007/978-3-540-40061-5_26
- Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2111–2128. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3339817
- Nguyen, K., Tang, H., Wang, H., Zeng, N.: New code-based privacypreserving cryptographic constructions. In: Galbraith, S.D., Moriai, S. (eds.) ASI-ACRYPT 2019, Part II. LNCS, vol. 11922, pp. 25–55. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34621-8 2
- Noether, S.: Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098 (2015), https://eprint.iacr.org/2015/1098
- Papamanthou, C., Shi, E., Tamassia, R.: Signatures of correct computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 222–242. Springer, Heidelberg (Mar 2013). https://doi.org/10.1007/978-3-642-36594-2_13
- Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1 9
- Pippenger, N.: On the evaluation of powers and related problems. In: 17th Annual Symposium on Foundations of Computer Science (sfcs 1976). pp. 258–263 (1976). https://doi.org/10.1109/SFCS.1976.21
- Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0_22
- 55. Setty, S.: Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol.

12172, pp. 704–737. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56877-1 $\ 25$

- Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 1–18. Springer, Heidelberg (Mar 2008). https://doi.org/10.1007/978-3-540-78524-8 1
- 57. Wahby, R.S., Tzialla, I., shelat, a., Thaler, J., Walfish, M.: Doubly-efficient zk-SNARKs without trusted setup. In: 2018 IEEE Symposium on Security and Privacy. pp. 926–943. IEEE Computer Society Press (May 2018). https://doi.org/10. 1109/SP.2018.00060
- Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 299–328. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/ 978-3-031-15985-5 11
- Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In: 2017 IEEE Symposium on Security and Privacy. pp. 863–880. IEEE Computer Society Press (May 2017). https://doi.org/10.1109/SP.2017.43
- Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: A zeroknowledge version of vSQL. Cryptology ePrint Archive, Report 2017/1146 (2017), https://eprint.iacr.org/2017/1146

A Security Proof for GIPA

COMPLETENESS.

$$\Pr\left[\begin{matrix} \pi \leftarrow \mathcal{P}(n=2^{\kappa},ts=\emptyset, \pmb{a}, \pmb{b}, I_{\pmb{ab}}, \pmb{c}, I_{\pmb{ac}}): \\ \mathcal{V}(n, I_{\pmb{ab}}, I_{\pmb{ac}}, \pi) = 1 \end{matrix} \right] = 1$$

WITNESS-EXTENDED EMULATION.

- $-\kappa = 0$, the extractor \mathcal{E}_0 returns a, b, c.
- $-\kappa = 1$, on challenge h_i for $i \in [1, 4]$, the extractor \mathcal{E}_1 rewinds the prover 4 times, and obtain four pairs of (a'_i, b'_i, c'_i) such that

$$\begin{aligned} \boldsymbol{b}_{i}^{\prime} &= h_{i}^{-1} \cdot \boldsymbol{b}_{[:n']} + h_{i} \cdot \boldsymbol{b}_{[n':]} \\ \langle \boldsymbol{a}_{i}^{\prime}, \boldsymbol{b}_{i}^{\prime} \rangle &= h_{i}^{2} \cdot R_{\boldsymbol{a}\boldsymbol{b}} + I_{\boldsymbol{a}\boldsymbol{b}} + h_{i}^{-2} \cdot L_{\boldsymbol{a}\boldsymbol{b}} \\ \boldsymbol{c}_{i}^{\prime} &= h_{i}^{-1} \cdot \boldsymbol{c}_{[:n']} + h_{i} \cdot \boldsymbol{c}_{[n':]} \\ \langle \boldsymbol{a}_{i}^{\prime}, \boldsymbol{c}_{i}^{\prime} \rangle &= h_{i}^{2} \cdot R_{\boldsymbol{a}\boldsymbol{c}} + I_{\boldsymbol{a}\boldsymbol{c}} + h_{i}^{-2} \cdot L_{\boldsymbol{a}\boldsymbol{c}} \end{aligned}$$

 \mathcal{E} returns

$$\boldsymbol{a} = \sum_{i=1}^{3} (v_i h_i^{-1} \boldsymbol{a}'_i, v_i h_i \boldsymbol{a}'_i).$$

We can see that I_{ab} have the expected form:

$$\begin{split} \langle \boldsymbol{a}, \boldsymbol{b} \rangle &= \sum_{i=1}^{3} v_{i} \langle h_{i}^{-1} \cdot \boldsymbol{a}_{i}', \boldsymbol{b}_{[:n']} \rangle + \sum_{i=1}^{3} v_{i} \langle h_{i} \cdot \boldsymbol{a}_{i}', \boldsymbol{b}_{[n':]} \rangle \\ &= \sum_{i=1}^{3} v_{i} \langle \boldsymbol{a}_{i}', h_{i}^{-1} \boldsymbol{b}_{[:n']} + h_{i} \cdot \boldsymbol{b}_{[:n']} \rangle \\ &= \sum_{i=1}^{3} v_{i} \langle \boldsymbol{a}_{i}', \boldsymbol{b}_{i}' \rangle \\ &= \sum_{i=1}^{3} v_{i} (h_{i}^{2} \cdot R_{\boldsymbol{ab}} + I_{\boldsymbol{ab}} + h_{i}^{-2} \cdot L_{\boldsymbol{ab}}) \\ &= \sum_{i=1}^{3} v_{i} h_{i}^{2} \cdot R_{\boldsymbol{ab}} + \sum_{i=1}^{3} v_{i} \cdot I_{\boldsymbol{ab}} + \sum_{i=1}^{3} v_{i} \cdot h_{i}^{-2} \cdot L_{\boldsymbol{ab}} \\ &= I_{\boldsymbol{ab}} \end{split}$$

Similarly for $\langle \boldsymbol{a}, \boldsymbol{c} \rangle = I_{\boldsymbol{ac}}$. - $\kappa \geq 2$, Recursively apply extractor $\mathcal{E}_1 \kappa$ times.

B More on Commitment Schemes

A subvector commitment scheme is a tuple of four algorithm (Setup, Commit, Open, Eval) such that

- srs \leftarrow Setup (λ, n) . On input security parameter λ and a vector length n, Setup outputs structured reference string srs.
- $-c_{v} \leftarrow \mathsf{Commit}(\mathsf{srs}, v)$. Commit computes a vector commitment c_{v} on v.
- $-\pi_S \leftarrow \mathsf{Prove}(\mathsf{srs}, c_v, S, v_S)$. Prove generates a proof π_S such that v_S are evaluations of c_v on positions S.
- $-1/0 \leftarrow \mathsf{Eval}(\mathsf{srs}, c_{\boldsymbol{v}}, S, \boldsymbol{v}_S, \pi_S)$. Eval checks if the evaluation of $c_{\boldsymbol{v}}$ on positions S are \boldsymbol{v}_S .

and satisfies position binding property and, optionally, hiding property. POSITION BINDING. For any PPT adversary \mathcal{A} ,

$$\Pr \begin{bmatrix} \operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, n), \\ (c_{\boldsymbol{v}}, S, \boldsymbol{v}_{S}, \boldsymbol{v}'_{S}, \pi, \pi') \leftarrow \mathcal{A}(\operatorname{srs}) : \\ \operatorname{Eval}(\operatorname{srs}, c_{\boldsymbol{v}}, S, \boldsymbol{v}_{S}, \pi_{S}) = 1 \\ \wedge & \operatorname{Eval}(\operatorname{srs}, c_{\boldsymbol{v}}, S, \boldsymbol{v}'_{S}, \pi'_{S}) = 1 \\ \wedge & \boldsymbol{v}_{S} \neq \boldsymbol{v}'_{S} \end{bmatrix} = 1.$$

HIDING. For any PPT adversary \mathcal{A} ,

$$\Pr \begin{bmatrix} \operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, n), & & \\ (v_1, v_2) \leftarrow \mathcal{A}(\operatorname{srs}), & & \\ b \xleftarrow{}{\leftarrow} \{0, 1\}, & & \\ c_{v_b} \leftarrow \operatorname{Commit}(\operatorname{srs}, v_b), & & \\ b' \leftarrow \mathcal{A}(\operatorname{srs}, c_{v_b}) : & & \\ & & b \neq b' \end{bmatrix} = \operatorname{negl} + \frac{1}{2}$$

The construction of Pedersen commitment [52], AFGHO commitment [1] (i.e., pairing variant of Pedersen commitment), and (hiding) Pedersen subvector commitment [37] are in Figure 8, 9, and 10.

$srs \gets Setup(\lambda, n)$		
1: $(\mathbb{G}, p) \leftarrow GGen(\lambda)$ 2: $g \stackrel{R}{\leftarrow} \mathbb{G}, \alpha, \stackrel{R}{\leftarrow} \mathbb{Z}_p$ 3: $srs = (p, \mathbf{g} = (g^{\alpha^{i-1}})_{i \in [1,n]})$		
$\overline{c_{oldsymbol{v}} \leftarrow Commit(srs, oldsymbol{v} \in \mathbb{Z}_p^n)}$		
$1: c_{\boldsymbol{v}} = \langle \boldsymbol{v}, \boldsymbol{g} \rangle$		
$\overline{1/0 \leftarrow Open(srs, c_{\bm{v}}, \bm{v})}$		
1: $c_{\boldsymbol{v}} \stackrel{?}{=} \langle \boldsymbol{v}, \boldsymbol{g} \rangle$		

Fig. 8: Construction of Pedersen Commitment.

$srs \gets Setup(\lambda, n)$
$1: (e, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, p) \leftarrow BGGen(\lambda)$ $2: \widetilde{g} \stackrel{R}{\leftarrow} \widetilde{\mathbb{G}}$
3: $\operatorname{srs} = (p, \widetilde{g} = (\widetilde{g}^{\gamma^{i-1}})_{i \in [1,n]})$
$c_{\boldsymbol{v}} \leftarrow Commit(srs, \boldsymbol{v} \in \mathbb{G}^n; \omega)$
1: $c_{\boldsymbol{v}} = \langle \boldsymbol{v}, \widetilde{\boldsymbol{g}} \rangle$
$\frac{1}{1/0 \leftarrow Open(srs, c_{\boldsymbol{v}}, \boldsymbol{v})}$
1: $c_{\boldsymbol{v}} \stackrel{?}{=} \langle \boldsymbol{v}, \widetilde{\boldsymbol{g}} \rangle$

35

Fig. 9: Construction of AFGHO Commitment.

C Security Definition of Multivariate Polynomial Commitment

A multivariate polynomial commitment scheme should satisfy completeness, polynomial binding, evaluation binding, and optionally, knowledge soundness, polynomial hiding.

COMPLETENESS. For any *m*-variate polynomial f with degree $\leq \hat{d}$, and evaluation vector $\boldsymbol{v} \in \mathbb{F}^m$,

$$\Pr \begin{bmatrix} \mathsf{srs} \leftarrow \mathsf{Setup}(\lambda, m, \hat{d}), \\ c_f \leftarrow \mathsf{Commit}(\mathsf{srs}, f), \\ \pi \leftarrow \mathsf{Prove}(\mathsf{srs}, \boldsymbol{v}, f, c_f) : \\ \mathsf{Eval}(\mathsf{srs}, \boldsymbol{v}, c_f, e_{\boldsymbol{v}}, \pi) = 1 \end{bmatrix} = 1$$

 $\begin{aligned} \frac{\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, n) \\ \hline 1: & (e, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, p) \leftarrow \operatorname{BGGen}(\lambda) \\ 2: & g \stackrel{R}{\leftarrow} \mathbb{G}, \widetilde{g} \stackrel{R}{\leftarrow} \widetilde{\mathbb{G}}, \\ 3: & \alpha, \delta \stackrel{R}{\leftarrow} \mathbb{Z}_p \\ 4: & h = g^{\delta} \\ 5: & \forall i \in [1, 2n], g_i = g^{\alpha^i - 1}, \\ 6: & \forall i \in [1, n], \widetilde{g}_i = \widetilde{g}^{\alpha^i}, h_i = h^{\alpha^i} \\ 7: & \operatorname{srs} = (p, g = (g_i)_{i \in [1, n]}, (g_i)_{i \in [n+2, 2n]}, \widetilde{g}, (\widetilde{g}_i,)_{i \in [1, n]}, h, (h_i)_{i \in [1, n]}) \\ \hline c_v \leftarrow \operatorname{Commit}(\operatorname{srs}, v = (v_i)_{i \in [1, n]} \\ \hline 1: & \omega \stackrel{R}{\leftarrow} \mathbb{Z}_p \\ 2: & c_v = \langle v, g \rangle \cdot \langle \omega, h \rangle \qquad \qquad \triangleright c_v = \prod_{i=1}^n g_i^{v_i} \cdot h^{\omega} \\ \hline \overline{\pi_S} \leftarrow \operatorname{Prove}(\operatorname{srs}, c_v, S, v, \omega) \\ \hline 1: & z = H(c_v, S, v_S) \\ 2: & \forall i \in [1, n], \pi_i = \prod_{j=1, j \neq i}^n g_{n+1-i+j}^{v_j} h_{n+1-i}^{\omega} \\ 3: & \pi_S = \prod_{i \in S} \pi_i^{z^{i-1}} \qquad \triangleright \pi_S = \prod_{i \in S} \prod_{j=1, j \neq i}^n g_{n+1-i+j}^{z^{i-1}, \omega_i} \cdot \prod_{i \in S} h_{n+1-i}^{z^{i-1}, \omega_i} \\ \hline 1 \leftarrow \operatorname{Eval}(\operatorname{srs}, c_v, S, v_S, \pi_S; z) \\ \hline 1: & z = H(c_v, S, v_S) \\ 2: & e(c_v, \widetilde{g}_{\sum_{i \in S} z^{i-1}}^{\sum i^{-1}}) \stackrel{?}{=} e(\pi_S, \widetilde{g}) \cdot e(g^{\sum_{i \in S} z^{i-1} \cdot v_i}, \widetilde{g}_n) \end{aligned}$

Fig. 10: Construction of Pedersen Subvector Commitment.

POLYNOMIAL BINDING. For all PPT adversary \mathcal{A} ,

$$\Pr \begin{bmatrix} \mathsf{srs} \leftarrow \mathsf{Setup}(\lambda, m, \dot{d}), \\ (f, f') \leftarrow \mathcal{A}(\mathsf{srs}) : \\ \mathsf{Commit}(\mathsf{srs}, f) = \mathsf{Commit}(\mathsf{srs}, f') \end{bmatrix} = \mathsf{negl}$$

EVALUATION BINDING. For all PPT adversary \mathcal{A} ,

$$\Pr \begin{bmatrix} \operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, \hat{d}), \\ (\boldsymbol{v}, c_f, e_{\boldsymbol{v}}, \pi, e'_{\boldsymbol{v}}, \pi') \leftarrow \mathcal{A}(\operatorname{srs}) : \\ \operatorname{Eval}(\operatorname{srs}, \boldsymbol{v}, c_f, e_{\boldsymbol{v}}, \pi) = 1 \\ \wedge \qquad \operatorname{Eval}(\operatorname{srs}, \boldsymbol{v}, c_f, e'_{\boldsymbol{v}}, \pi') = 1 \end{bmatrix} = \operatorname{negl}$$

KNOWLEDGE SOUNDNESS. For all PPT adversary \mathcal{A} , there exists an extractor \mathcal{E} such that

$$\Pr \begin{bmatrix} \operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, \hat{d}), \\ (\boldsymbol{v}, c_f, e_{\boldsymbol{v}}, \pi) \leftarrow \mathcal{A}(\operatorname{srs}), \\ f \leftarrow \mathcal{E}(\operatorname{srs}) : \\ \operatorname{Eval}(\operatorname{srs}, \boldsymbol{v}, c_f, e_{\boldsymbol{v}}, \pi) = 1 \\ \wedge \qquad e_{\boldsymbol{v}} \neq f(\boldsymbol{v}) \end{bmatrix} = \operatorname{negl}$$

Remark. If a polynomial commitment scheme is knowledge sound in AGM, it is also evaluation binding. Let \mathcal{A}_{KS} and \mathcal{A}_{EB} denote the adversary of the knowledge soundness game and evaluation binding game, respectively. \mathcal{A}_{KS} can selectively output whatever \mathcal{A}_{EB} outputs, and wins the knowledge soundness game with the same probability as that of \mathcal{A}_{EB} . Looking ahead, the same conclusion also applies to polynomial commitment in the batched setting and UMP/MMP commitment. We therefore omit the proof for the polynomial binding property in the rest of this paper.

POLYNOMIAL HIDING. For all PPT adversary \mathcal{A} ,

$$\Pr \begin{bmatrix} \operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, d), & & \\ (f_0, f_1) \leftarrow \mathcal{A}(\operatorname{srs}), & & \\ b \xleftarrow{}{\leftarrow} \{0, 1\}, & & \\ c_{f_b} \leftarrow \operatorname{Commit}(\operatorname{srs}, f_b), & & \\ b' \leftarrow \mathcal{A}(\operatorname{srs}, c_{f_b}) : & & \\ & & b \neq b' \end{bmatrix} = \operatorname{negl} + \frac{1}{2}$$

EVALUATION BINDING (BATCH). A polynomial commitment scheme in the batched setting is evaluation binding if for all PPT adversary \mathcal{A} ,

$$\Pr \begin{bmatrix} \operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, \hat{d}), \\ ((c_{f_i}, e_{\boldsymbol{v}}^{[i]}, e_{\boldsymbol{v}}^{[i]'})_{i \in [1,n]}, \boldsymbol{f}, \boldsymbol{v}, \pi, \pi') \leftarrow \mathcal{A}(\operatorname{srs}) : \\ \operatorname{BatchEval}(\operatorname{srs}, \boldsymbol{v}, (c_{f_i}, e_{\boldsymbol{v}}^{[i]})_{i \in [1,n]}, \pi) = 1 \\ \wedge \operatorname{BatchEval}(\operatorname{srs}, \boldsymbol{v}, (c_{f_i}, e_{\boldsymbol{v}}^{[i]'})_{i \in [1,n]}, \pi') = 1 \end{bmatrix} = \operatorname{negl}$$

KNOWLEDGE SOUNDNESS (BATCH). A polynomial commitment scheme in the batched setting is knowledge sound in AGM model if for all PPT adversary \mathcal{A} , there exists an extractor \mathcal{E} such that

$$\Pr \begin{bmatrix} \operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, \hat{d}), \\ (\boldsymbol{v}, (c_{f_i}, e_{\boldsymbol{v}}^{[i]})_{i \in [1,n]}, \pi) \leftarrow \mathcal{A}(\operatorname{srs}), \\ \boldsymbol{f} \leftarrow \mathcal{E}(\operatorname{srs}) : \\ \operatorname{BatchEval}(\operatorname{srs}, \boldsymbol{v}, (c_{f_i}, e_{\boldsymbol{v}}^{[i]})_{i \in [1,n]}, \pi) = 1 \\ \wedge \qquad (e_{\boldsymbol{v}}^{[i]})_{i \in [1,n]} \neq \boldsymbol{f}(\boldsymbol{v}) \end{bmatrix} = \operatorname{negl}$$

D Security Proof for PST Commitment

Theorem 3. The PST commitment scheme in the batched setting satisfies completeness, polynomial binding, knowledge soundness in AGM model. *Proof.* Completeness follows from the construction of the scheme. We show below PST commitment scheme in the batched setting is knowledge sound in AGM if Lemma 1 holds.

POLYNOMIAL BINDING. Suppose there exists an adversary \mathcal{A} against the polynomial binding property, we build an adversary \mathcal{A}^* against (q, 1)-discrete logarithm assumption, on input a discrete logarithm problem instance $(g, g^x, ..., g^{x^q}, \tilde{g}, \tilde{g}^x)$, \mathcal{A}^* computes

 $- \forall j \in [1, m], a_j, b_j \stackrel{R}{\leftarrow} \mathbb{Z}_p, \text{ define } \beta_j = a_j x + b_j, \\ - \text{ Compute } \boldsymbol{q} = (q^{\prod_{j=1}^m \beta_j^{d_j}}), \text{ for } j, \text{ for } j \in [0, 1]$

- Set srs =
$$(\boldsymbol{g}, \tilde{g}, (\tilde{g}^{\beta_i})_{i \in [1,m]})^{d_1 \in [0, \tilde{d}_1], \dots, d_m \in [0, \tilde{d}_m]}$$

and sends srs to \mathcal{A} .

Set $\boldsymbol{\beta} := (a_j x + b_j)_{j \in [1,m]}$. The difference of $f(\boldsymbol{\beta})$ and $f'(\boldsymbol{\beta})$ yields a univariate polynomial $f^{\Delta}(X)$ evaluated at x, i.e., $f^{\Delta}(x) = f(\boldsymbol{\beta}) - f'(\boldsymbol{\beta})$. Finally, \mathcal{A}^* solves $f^{\Delta}(X) = 0$ and returns the solution x, breaking the (q, 1)-discrete logarithm assumption.

KNOWLEDGE SOUNDNESS IN THE BATCHED SETTING. Given an algebraic adversary \mathcal{A} that wins the knowledge soundness game with non-negligible probability, we could break Lemma 1.

On input $(g, g^x, ..., g^{x^q}, \tilde{g}, \tilde{g}^x)$, \mathcal{E} initiates the knowledge soundness game with srs same as the polynomial binding game above.

Upon receiving srs, \mathcal{A} outputs \boldsymbol{v} , $(c_{f_i}, e_{\boldsymbol{v}}^{[i]})_{i \in [1,n]}$ and π , along with $\mathsf{REP}_{\mathsf{srs}}^{(c_{f_i})_{i \in [1,n]}}$ and $\mathsf{REP}_{\mathsf{srs}}^{\pi}$. With these, \mathcal{E} can output polynomials $\boldsymbol{f} = (f_1, ..., f_n)$ and $(\hat{q}_j(\beta))_{j \in [1,m]}$.

Assume that $\exists i \in [1, n], f_i(\boldsymbol{v}) \neq e_{\boldsymbol{v}}^{[i]}$. Let $\hat{f}(x) = \langle \boldsymbol{f}, \boldsymbol{r}^n \rangle$ and $e_{\hat{v}} = \langle \boldsymbol{\varphi}, \boldsymbol{r}^n \rangle$ where $r = H(\boldsymbol{v}, (c_{f_i})_{i \in [1, n]})$ The ideal checking is in the form of $\hat{f}(\boldsymbol{\beta}) - e_{\boldsymbol{v}} = \sum_{j=1}^{m} \hat{q}_j(\boldsymbol{\beta})(\beta_j - v_j)$, which implies $\hat{f}(\boldsymbol{v}) \neq e_{\hat{f}}$ with probability $\leq \frac{p-n}{p}$. Hence, the ideal check passes with negligible probability. If the real pairing check passes with non-negligible probability, \mathcal{E} breaks Lemma 1. Therefore, \mathcal{A} can only win the knowledge soundness game with negligible probability.

Theorem 4. The hiding PST commitment inherits the security properties of the PST commitment and additionally satisfies the polynomial hiding property.

Proof. The hiding PST commitment is correct and hiding by its construction. The proof for polynomial binding and knowledge soundness works the same as the PST commitment, except that \mathcal{A}^* has to pick $\delta \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and compute h and \tilde{h} accordingly.

Let $c_{\boldsymbol{v}} = \langle \boldsymbol{v}, \boldsymbol{g} \rangle$ and index set $S = (i_j)_{j \in [1,n']}$, to prove $\forall i \in S, \boldsymbol{v}_i \in [0, 2^l - 1]$. Define function $\sigma : \sigma(i_j) = j, \forall j \in [1, n']$, and modify the AoK_{agg} protocol as follows.

$$- c = \widetilde{g}_{n}^{\sum_{i \in S} z^{\sigma(i)+1} v_{i}} \cdot \widetilde{g}^{\gamma}$$

$$- c' = \prod_{i \in S} \prod_{j=1, j \neq i}^{n'} \widetilde{g}_{n+1-i+j}^{z^{\sigma(i)+1} \cdot v_{j}} \prod_{i \in S} g_{n+1-i}^{\sum_{i \in S} z^{\sigma(i)+1} \cdot w} / g_{1}^{\gamma}$$

$$- e(c_{\boldsymbol{v}}, \sum_{i \in S} \widetilde{g}_{n+1-i}^{z^{\sigma(i)+1}}) \stackrel{?}{=} e(c', \widetilde{g}) \cdot e(g_{1}, c).$$

39

E Security Proof for zero-knowledge MMP

Correctness. Correctness follows from the correctness of GIPA and the following equations.

$$\begin{split} e(c_{v},\prod_{i=1}^{n}g_{n+1-i}^{r^{i-1}}) &= e(\prod_{i=1}^{n}g_{i}^{f_{i}(v)}\cdot h^{e'},\prod_{i=1}^{n}\widetilde{g}_{n+1-i}^{r^{i-1}}) \\ &= e(\prod_{i=1}^{n}\prod_{j=1,j\neq i}^{n}g_{n+1-i+j}^{f_{j}(v)\cdot r^{i-1}}\cdot\prod_{i=1}^{n}h_{n+1-i}^{e'\cdot r^{i-1}},\widetilde{g})\cdot e(g_{n+1}^{\sum_{i=1}^{n}f_{i}(v)\cdot r^{i-1}},\widetilde{g}) \\ &= e(\prod_{i=1}^{n}\prod_{j=1,j\neq i}^{n}g_{n+1-i+j}^{f_{j}(v)\cdot r^{i-1}}\cdot\prod_{i=1}^{n}h_{n+1-i}^{e'\cdot r^{i-1}}\cdot h_{n}^{-e},\widetilde{g})\cdot e(g_{n+1}^{\sum_{i=1}^{n}f_{i}(v)\cdot r^{i-1}}\cdot h^{e},\widetilde{g}_{n}) \\ &= e(c',\widetilde{g})e(c_{\hat{v}},\widetilde{g}) \\ g^{b_{1}}h^{b_{2}}c_{\hat{v}}^{\eta} &= g^{e_{1}-\eta\cdot\sum_{i=1}^{n}f_{i}(v)\cdot r^{i-1}}h^{e_{2}-\eta\cdot e}\cdot (g_{n+1}^{\sum_{i=1}^{n}f_{i}(v)\cdot r^{i-1}}h^{e})^{\eta} \\ &= g^{e_{1}}\cdot h^{e_{2}} \\ &= D \\ e(c_{f}/c_{\hat{v}},\widetilde{g}) &= e(g_{n}^{\sum_{i=1}^{n}(f_{i}(\beta)-f_{i}(v))\cdot r^{i-1}}h^{\sum_{i=1}^{n}\rho_{i}\cdot r^{i-1}-e},\widetilde{h}) \\ &= \prod_{j=1}^{m}e(g^{q_{j}(\beta)},\widetilde{g}^{\beta_{j}-v_{j}})\cdot e(g_{n+1}^{\sum_{i=1}^{n}\rho_{i}\cdot r^{i-1}-e},\widetilde{h}) \\ &= \prod_{j=1}^{m}e(g^{q_{j}(\beta)}\cdot h^{\mu_{j}},\widetilde{g}^{\beta_{j}-v_{j}})\cdot e(g_{n+1}^{\sum_{i=1}^{n}\rho_{i}\cdot r^{i-1}-e},\widetilde{h}) \\ &= \prod_{j=1}^{m}e(Q_{j},\widetilde{g}^{\beta_{j}-v_{j}})\cdot e(\theta,\widetilde{h}) \end{split}$$

Polynomial Binding (Proof Sketch). The polynomial binding property follows from the binding property of AFGHO commitment and hiding PST commitment. The proof is essentially the same as that of MMP commitment.

Knowledge Soundness (Proof Sketch). srs_{pst} is generated in the same way as in the PST knowledge soundness game (Appendix D) using trapdoor x.

- The binding property of Pedersen commitment ensures a unique $e_v^{[i]}$ for $i \in [1, n]$ corresponding to c_v being extracted by algebraic logarithm adversary \mathcal{A} .
- Similar to the knowledge soundness proof for MMP, the knowledge soundness of PST commitment in the batched setting guarantees that $\forall i \in [1, n], f_i(v) = e_v^{[i]}$.

SHVZK. A zero-knowledge simulator uses the trapdoor of srs, i.e., α , $(\beta)_{i \in [1,m]}, \eta$, and simulates the protocols as below:

- Choose $\phi \stackrel{R}{\leftarrow} \mathbb{G}^n$ and $r \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and compute $c_f = \langle \phi, g_{\mathsf{pst}} \rangle$ and $c_{\hat{f}} = \langle \phi, r^n \rangle$, and runs the GIPA protocol on the simulated c_f and $c_{\hat{f}}$.
- Choose $c_{v} \stackrel{R}{\leftarrow} \mathbb{G}$ and $e_{\hat{v}}, \varrho \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$. Compute $c' = c_{v}^{\sum_{i=1}^{n} \alpha^{n+1-i} \cdot r^{i-1}} / (g_{n+1}^{e_{\hat{v}}} h_{n}^{\varrho})$ and $c_{\hat{v}} = g^{e_{\hat{v}}} h^{\varrho}$. Run Schnorr's protocol on simulated $c_{\hat{v}}$ and obtains b_{1}, b_{2}, D . $\forall j \in [1, m]$, choose $Q_{j} \stackrel{R}{\leftarrow} \mathbb{G}$. Compute $\theta = c_{\hat{f}} / (c_{\hat{v}} \cdot \prod_{j=1}^{m} Q_{j}^{(\beta_{j} v_{j})^{-1}})$.