# Multivariate Multi-Polynomial Commitment and its Applications 

Xiao Yang ${ }^{1}$, Chengru Zhang ${ }^{2}$, Mark Ryan ${ }^{1}$, and Gao Meng ${ }^{1}$<br>${ }^{1}$ University of Birmingham<br>yangxiao97531@gmail.com, m.d.ryan@bham.ac.uk, gxm311@student.bham.ac.uk<br>${ }^{2}$ The University of Hong Kong<br>u3008875@connect.hku.hk


#### 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 $48|27| 34|55| 26 \mid 23$. 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 $O\left((m+n) \cdot\binom{m+d}{d}\right)$ prover time (or $O\left((m+n) \cdot 2^{m}\right)$ 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 [22. 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 536.99 ms , and the verification algorithm completes in 18.595 ms .


### 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 $\left(g^{\tau^{0}}, g^{\tau^{1}}, \ldots, g^{\tau^{d}}, \widetilde{g}^{\tau}\right)$, where $g$ and $\widetilde{g}$ 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 $c=g^{f(\tau)}=\prod_{i=0}^{d}\left(g^{\tau^{i}}\right)^{f_{i}}$. The proof of evaluation at point $x$ is $\pi=g^{q(\tau)}=\prod_{i=0}^{d}\left(g^{\tau^{i}}\right)^{q_{i}}$, where the quotient polynomial $q(X)=\frac{f(X)-f(x)}{X-x}$. The verifier checks the proof by comparing if $e\left(\pi, \widetilde{g}^{\tau-x}\right)=e\left(c / g^{f(x)}, \widetilde{g}\right)$.

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|2] 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 prover ${ }^{3}$ 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 $O\left(\log ^{2} N\right)$ 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.

[^0]- 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. Zeroknowledge range proof was first introduced by [15], and has been studied across diverse applications like anonymous credential systems [226], 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 $\operatorname{PoK}\{(y),(x): f(x)=y\}$ 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:
$-(\mathbb{G}, p) \leftarrow \operatorname{GGen}(\lambda)$. On input security parameter $\lambda$, GGen generates cyclic group $\mathbb{G}$ of prime order $p(|p|=\lambda)$.
as well as a bilinear group generation algorithm BGGen,
$-\left(e, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_{T}, p\right) \leftarrow \operatorname{BGGen}(\lambda)$. On input security parameter $\lambda$, BGGen generates cyclic groups $\mathbb{G}, \widetilde{\mathbb{G}}$, and $\mathbb{G}_{T}$ of prime order $p(|p|=\lambda)$ and a bilinear mapping $e: \mathbb{G} \times \widetilde{\mathbb{G}} \rightarrow \mathbb{G}_{T}$.

We use bold font $\boldsymbol{x}$ to represent vectors. We denote
$-|\boldsymbol{v}|$ the number of elements of $\boldsymbol{v}$, e.g. $|\boldsymbol{v}|=n$,

- $\left(v_{i}\right)_{i \in[1, n]}$ the vector $\left(v_{1}, \ldots, v_{n}\right)$,
$-\left(v_{i \ldots j}\right)_{i \in[0, n], \ldots, j \in\left[0, n^{\prime}\right]}$ the vector $\left(v_{0 \ldots 0}, v_{0 \ldots 1}, \ldots, v_{n \ldots n^{\prime}}\right)$,
$-\boldsymbol{v}_{\left[: n^{\prime}\right]}=\left(v_{1}, \ldots, v_{n^{\prime}}\right), \boldsymbol{v}_{\left[n^{\prime}:\right]}=\left(v_{n^{\prime}+1}, \ldots, v_{n}\right)$,
- $\boldsymbol{v}_{S}$ 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 $\boldsymbol{g} \in \mathbb{G}^{n}, \widetilde{\boldsymbol{g}} \in \widetilde{\mathbb{G}}^{n}$, we denote
$-\mathbf{k}^{n}=\left(k^{i}\right)_{i \in[0, n-1]}, \mathbf{k}^{-n}=\left(k^{i}\right)_{i \in[0,-n+1]}$,
$-k \cdot \boldsymbol{a}=\left(k \cdot a_{1}, \ldots, k \cdot a_{n}\right)$,
$-\langle\boldsymbol{a}, \boldsymbol{b}\rangle=\sum_{i=1}^{n} a_{i} \cdot b_{i}$ the inner product of $\boldsymbol{a}, \boldsymbol{b}$,
$-\langle\boldsymbol{a}, \boldsymbol{g}\rangle=\prod_{i=1}^{n=1} g_{i}^{a_{i}}$ the inner product of $\boldsymbol{a}, \boldsymbol{g}$,
$-\langle\boldsymbol{g}, \widetilde{\boldsymbol{g}}\rangle=\prod_{i=1}^{n} e\left(g_{i}, \widetilde{g}_{i}\right)$ the inner product of $\boldsymbol{g}, \widetilde{\boldsymbol{g}}$,
$-\boldsymbol{a} \circ \boldsymbol{b}=\left(a_{i} \cdot b_{i}\right)_{i \in[1, n]}$ the Hadamard product of $\boldsymbol{a}, \boldsymbol{b}$,
where $a_{i}, b_{i}, g_{i}$ are the $i$-th element of $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}$.

### 2.1 Hardness Assumption

Definition 1 ( $(m, n)$-Discrete Logarithm Assumption). Let $\left(\mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_{T}\right)$ 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 $\left(g, g^{x}, g^{x^{2}}, ., \ldots, g^{x^{m}}\right)$ and $\left(\widetilde{g}, \widetilde{g}^{x}, \ldots, \widetilde{g}^{x^{n}}\right)$, 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

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

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 $R(u, w)=1$,

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

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

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

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 $\mathcal{A}$,

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

where the transcript oracle $\mathcal{O}=\left\langle\mathcal{P}^{*}(u, s), \mathcal{V}(u)\right\rangle$, 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 $S H V Z K$ if there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that for any probabilistic polynomialtime adversary $\mathcal{A}$,

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

where $\eta$ 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 $(\mathbb{G}, p) \leftarrow \operatorname{GGen}(\lambda)$ and $g \stackrel{R}{\leftarrow}$ $\mathbb{G}$. Schnorr's protocol is a SHVZK for the discrete logarithm relation $R$ such that

$$
R=\operatorname{PoK}\left\{\begin{aligned}
&(h \in \mathbb{G}),\left(x \in \mathbb{Z}_{p}\right): \\
& h=g^{x}
\end{aligned}\right\},
$$

The protocol is defined as follows.
$-\mathcal{P}$ picks $\varrho \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$ and sends $D=g^{\varrho}$ to $\mathcal{V}$.
$-\mathcal{V}$ picks $\eta \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$ and sends to $\mathcal{P}$.
$-\mathcal{P}$ sends $b=\varrho-\eta x$ to $\mathcal{P}$.
$-\mathcal{V}$ 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.

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

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

$$
\langle\cdot, \cdot\rangle:\left\{\begin{aligned}
\mathbb{F}^{n} \times \mathbb{F}^{n} \rightarrow \mathbb{F} & & (\boldsymbol{x}, \boldsymbol{y}) \mapsto x_{1} y_{1}+\cdots+x_{n} y_{n} \\
\mathbb{F}^{n} \times \mathbb{G}^{n} \rightarrow \mathbb{G} & & (\boldsymbol{x}, \boldsymbol{g}) \mapsto g_{1}^{x_{1}} \ldots g_{n}^{x_{n}} \\
\mathbb{G}^{n} \times \widetilde{\mathbb{G}}^{n} \rightarrow \mathbb{G}_{T} & & (\boldsymbol{g}, \boldsymbol{h}) \mapsto e\left(g_{1}, h_{1}\right) \ldots e\left(g_{n}, h_{n}\right)
\end{aligned}\right.
$$

where $\boldsymbol{x}=\left(x_{1}, \ldots, x_{n}\right), \boldsymbol{g}=\left(g_{1}, \ldots, g_{n}\right)$, 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 $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_{p}^{n}$ and $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{G}^{n}$.

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

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

The construction of the adapted GIPA scheme is defined in Figure 1. Note that even if $\boldsymbol{a}$ 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 $\boldsymbol{b}$ or $\boldsymbol{c}$ with special arithmetic structure are proposed 3121. Taking $\boldsymbol{b}$ as an example, the optimization techniques involved in this paper for achieving logarithmic verification complexity are summarized as follows.

1. Define $\hat{\eta}(X)=\prod_{i=1}^{\kappa}\left(\eta_{i}^{-1}+\eta_{i} X^{2^{\kappa-i}}\right)$. Let $\hat{\boldsymbol{\eta}}$ be the coefficient vector of polynomial $\hat{\eta}(X)$. As pointed in [2], $b_{0}$ can be rewritten as $\langle\hat{\boldsymbol{\eta}}, \boldsymbol{b}\rangle$. Based on this observation, we have the following conclusions.
(a) For $\boldsymbol{b}=\left(1, r^{1}, \ldots, r^{n-1}\right)$ 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 $\boldsymbol{b}=\left(g, g^{\alpha}, \ldots, g^{\alpha^{n-1}}\right), b_{0}=g^{\hat{\eta}(\alpha)}$ is a KZG commitment on polynomial $\hat{\eta}(X)$. In this case, $\mathcal{P}$ can initiate a KZG evaluation proof with $\mathcal{V}$ to prove that $b_{0}$ is indeed a PST commitment to $\hat{\eta}(X)$. More specifically, $\mathcal{P}$ adds $b_{0}$ to $\pi$ along with a KZG evaluation proof $\pi_{\mathrm{kzg}}$ on a random input $\rho$. Upon receiving $b_{0}$ and $\pi_{\mathrm{kzg}}, \mathcal{V}$ computes $\hat{\eta}(\rho)$ and accepts $b_{0}$ if only $e\left(b_{0} / g^{\hat{\eta}(\rho)}, \widetilde{g}\right)=e\left(\pi_{\mathrm{kzg}}, g^{\alpha} / g^{\rho}\right)$.
(c) For $\boldsymbol{b}=\left(g, g^{r \cdot \alpha}, \ldots, g^{(r \cdot \alpha)^{n-1}}\right)$ 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 $\boldsymbol{b}:\left\{\boldsymbol{b} \leftarrow(1), \boldsymbol{b} \leftarrow\left(\boldsymbol{b}, r_{i} \cdot \boldsymbol{b}\right)\right\}_{i \in[0, \kappa-1]}$ where $r_{i} \in \mathbb{Z}_{p}$ is public elements, for $i \in[0, \kappa-1]$,
$\boldsymbol{b}_{i}=\eta_{i}^{-1} \cdot \boldsymbol{b}_{\left[: 2^{i}\right]}+\eta_{i} \cdot \boldsymbol{b}_{\left[2^{i}:\right]}=\eta_{i}^{-1} \cdot \boldsymbol{b}_{\left[: 2^{i}\right]}+\eta_{i} \cdot\left(r_{i} \cdot \boldsymbol{b}_{\left[: 2^{i}\right]}\right)=\left(\eta_{i}^{-1}+\eta_{i} \cdot r_{i}\right) \cdot \boldsymbol{b}_{\left[: 2^{i}\right]}$,
which implies $b_{0}=\prod_{i=0}^{\kappa}\left(\eta_{i}^{-1}+\eta_{i} \cdot r_{i}\right)$.

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

Now, denote the group elements output by $\mathcal{A}$ as $\left(g^{a_{j}}\right)_{j \in[1, k]},\left(\widetilde{g}^{b_{j}}\right)_{j \in[1, l]}$, and their representations respectively as $\left(\boldsymbol{u}^{[j]}\right)_{j \in[1, k]},\left(\boldsymbol{v}^{[j]}\right)_{j \in[1, l]}$. Define $L_{j}(X)=$ $\sum_{i=0}^{n} u_{i}^{[j]} f_{i}(X)$, and analogously, $R_{j}(X)=\sum_{i=0}^{n} v_{i}^{[j]} \widetilde{f}_{i}(X)$.

We set $\boldsymbol{a}=\left(a_{j}\right)_{j \in[1, k]}, \boldsymbol{b}=\left(b_{j}\right)_{j \in[1, l]}, \boldsymbol{L}=\left(L_{j}(X)\right)_{j \in[1, k]}, \boldsymbol{R}=\left(R_{j}(X)\right)_{j \in[1, l]}$. For matrix $M_{1}$ and $M_{2}$ over $\mathbb{F}$, we define the real pairing check and ideal check as follows.

Real Pairing Check. $\left(\boldsymbol{a} \cdot M_{1}\right) \cdot\left(M_{2} \cdot \boldsymbol{b}\right)=0$
Ideal Check. $\left(\boldsymbol{L} \cdot M_{1}\right) \cdot\left(M_{2} \cdot \boldsymbol{R}\right) \equiv 0$
Regarding the real pairing check and ideal check, we have the following lemma.

```
\(\mathcal{P}\left(n=2^{\kappa}, \eta_{\kappa}=\emptyset, \boldsymbol{a}, \boldsymbol{b}, I_{a b}, \boldsymbol{c}, I_{a c}\right)\)
    if \(n=1\) then
        Return \(a_{0}=\boldsymbol{a}\)
    else \(n \geq 2\)
        \(n^{\prime}=\frac{n}{2}, i=\log _{2}\left(n^{\prime}\right)\)
        \(L_{a b}^{[i]}=\left\langle\boldsymbol{a}_{\left[: n^{\prime}\right]}, \boldsymbol{b}_{\left[n^{\prime}:\right]}\right\rangle, R_{a b}^{[i]}=\left\langle\boldsymbol{a}_{\left[n^{\prime}:\right]}, \boldsymbol{b}_{\left[: n^{\prime}\right]}\right\rangle\)
        \(L_{a c}^{[i]}=\left\langle\boldsymbol{a}_{\left[: n^{\prime}\right]}, \boldsymbol{c}_{\left[n^{\prime}:\right]}\right\rangle, R_{a c}^{[i]}=\left\langle\boldsymbol{a}_{\left[n^{\prime}:\right]}, \boldsymbol{c}_{\left[: n^{\prime}\right]}\right\rangle\)
        \(\eta_{i}=H\left(\eta_{i+1}, L_{a b}^{i}, R_{a b}^{i}, L_{a c}^{i}, R_{a c}^{i}\right)\)
        \(\boldsymbol{a}^{\prime}=\eta_{i} \cdot \boldsymbol{a}_{\left[: n^{\prime}\right]}+\eta_{i}^{-1} \cdot \boldsymbol{a}_{\left[n^{\prime}:\right]}\)
        \(\boldsymbol{b}^{\prime}=\eta_{i}^{-1} \cdot \boldsymbol{b}_{\left[: n^{\prime}\right]}+\eta_{i} \cdot \boldsymbol{b}_{\left[n^{\prime}:\right]}\)
        \(\boldsymbol{c}^{\prime}=\eta_{i}^{-1} \cdot \boldsymbol{c}_{\left[: n^{\prime}\right]}+\eta_{i} \cdot \boldsymbol{c}_{\left[n^{\prime}:\right]}\)
        \(I_{a b}^{\prime}=\eta_{i}^{2} \cdot L_{a b}^{i}+I_{a b}+\eta_{i}^{-2} \cdot R_{a b}^{i}\)
        \(I_{a c}^{\prime}=\eta_{i}^{2} \cdot L_{a c}^{i}+I_{a c}+\eta_{i}^{-2} \cdot R_{a c}^{i}\)
        Recursively call \(\mathcal{P}\left(n^{\prime}, \eta_{i}, \boldsymbol{a}^{\prime}, \boldsymbol{b}^{\prime}, I_{a b}^{\prime}, \boldsymbol{c}^{\prime}, I_{a c}^{\prime}\right)\)
    Output \(\pi=\left(a_{0},\left\{L_{a b}^{i}, R_{a b}^{i}, L_{a c}^{i}, R_{a c}^{i}\right\}_{i \in[\kappa-1,0]}\right)\)
```

$\mathcal{V}\left(n, \boldsymbol{b}, I_{a b}, \boldsymbol{c}, I_{a c}, \pi\right)$
Parse $\pi=\left(a,\left\{L_{a b}^{i}, R_{a b}^{i}, L_{a c}^{i}, R_{\boldsymbol{a c}}^{i}\right\}_{i \in[\kappa-1,0]}\right)$
$\eta_{\kappa}=\emptyset$
for $i \in[\kappa-1,0]$ do
$\eta_{i}=H\left(\eta_{i+1}, L_{a b}^{i}, R_{a b}^{i}, L_{a c}^{i}, R_{a c}^{i}\right)$
$\boldsymbol{b}_{i}=\eta_{i}^{-1} \cdot \boldsymbol{b}_{\left[: n^{\prime}\right]}+\eta_{i} \cdot \boldsymbol{b}_{\left[n^{\prime}:\right]}$
$\boldsymbol{c}_{i}=\eta_{i}^{-1} \cdot \boldsymbol{c}_{\left[: n^{\prime}\right]}+\eta_{i} \cdot \boldsymbol{c}_{\left[n^{\prime}:\right]}$
$\hat{\eta}(X)=\prod_{i=1}^{\kappa}\left(\eta_{i}^{-1}+\eta_{i} X^{2^{\kappa-i}}\right)$
$\left\langle a_{0}, b_{0}\right\rangle \stackrel{?}{=} I_{\boldsymbol{a} \boldsymbol{b}}+\sum_{i=1}^{\kappa}\left(\eta_{i}^{2} \cdot L_{\boldsymbol{a} \boldsymbol{b}}^{i}+\eta_{i}^{-2} R_{\boldsymbol{a} \boldsymbol{b}}^{i}\right)$
$\left\langle a_{0}, c_{0}\right\rangle \stackrel{?}{=} I_{a c}+\sum_{i=1}^{\kappa}\left(\eta_{i}^{2} \cdot L_{a c}^{i}+\eta_{i}^{-2} R_{a c}^{i}\right)$

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

Please refer to Appendix Cfor 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.

```
\(\operatorname{srs} \leftarrow \operatorname{Setup}\left(\lambda, m, \hat{\boldsymbol{d}}=\left(\hat{d}_{j}\right)_{j \in[1, m]}\right)\)
    \(1:\left(e, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_{T}, p\right) \leftarrow \operatorname{BGGen}(\lambda)\)
2: \(g \stackrel{R}{\leftarrow} \mathbb{G}, \widetilde{g} \stackrel{R}{\leftarrow}^{\mathbb{G}}\)
3: \(\forall j \in[1, m], \beta_{j} \stackrel{R}{\leftarrow} \mathbb{Z}_{p}, \delta \stackrel{R}{\llcorner } \mathbb{Z}_{p}\)
4: \(h:=g^{\delta}, \widetilde{h}:=\widetilde{g}^{\delta}\)
5: srs := \(\left(p, \boldsymbol{g}=\left(g^{\Pi_{j=1}^{m} \beta_{j}^{d_{j}}}\right)_{d_{1} \in\left[0, \hat{d}_{1}\right], \ldots, d_{m} \in\left[0, \hat{d}_{m}\right]}, \widetilde{g},\left(\widetilde{g}^{\beta_{j}}\right)_{j \in[1, m]}, h, \widetilde{h}\right)\)
```

$c_{f} \leftarrow$ Commit(srs, $f$ )
1: $f(\boldsymbol{X})=\sum_{d_{1} \in\left[0, \hat{d}_{1}\right]} f_{d_{1} \ldots d_{m}} \prod_{j=1}^{m} X_{j}^{d_{j}}$
2: $\boldsymbol{w}=\left\{f_{d_{1} \ldots d_{m}} \begin{array}{c}d_{m} \in\left[0, \hat{d}_{m}\right] \\ d_{1} \in\left[0, \hat{d}_{1}\right], \ldots, d_{m} \in\left[0, \hat{d}_{m}\right]\end{array}\right.$
3: $\rho \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$
4: $c_{f}=\langle\boldsymbol{w}, \boldsymbol{g}\rangle+\langle\rho, h\rangle$
$\pi \leftarrow \operatorname{Prove}\left(\operatorname{srs}, \boldsymbol{v}, f, c_{f}, \rho\right)$

1: Compute $\left\{q_{j}(\boldsymbol{X})\right\}_{j \in[1, m]}: f(\boldsymbol{X})-f(\boldsymbol{v})=\sum_{j=1}^{m} q_{j}(\boldsymbol{X})\left(x_{j}-v_{j}\right)$
2: $\forall j \in[1, m], \mu_{j} \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$
3: $\pi=\left\{g^{q_{j}(\boldsymbol{\beta})} \cdot h^{\mu_{j}}\right\}_{j \in[1, m]}$
4: $\theta=g^{\sum_{j=1}^{m} v_{j} \cdot \mu_{j}+\rho} / \prod_{j=1}^{m} g^{\beta_{j} \cdot \mu_{j}}$
$1 / 0 \leftarrow \operatorname{Eval}\left(\operatorname{srs}, \boldsymbol{v}, c_{f}, e_{\boldsymbol{v}}, \pi, \theta\right)$
1: Parse $\pi$ as $\left\{Q_{j}\right\}_{j \in[1, m]}$
2: $e\left(c_{f} \cdot g^{-e_{\boldsymbol{v}}}, \widetilde{g}\right)=\prod_{i=1}^{m} e\left(Q_{j}, \widetilde{g}^{\beta_{i}} / g^{v_{i}}\right) \cdot e(\theta, \widetilde{h})$
$\pi \leftarrow$ BatchProve(srs, $\left.\boldsymbol{v},\left(c_{f_{i}}, f_{i}, \rho_{i}\right)_{i \in[1, n]}\right)$
1: $r=H\left(\boldsymbol{v},\left(c_{f_{i}}\right)_{i \in[1, n]}\right)$
2: $\forall i \in[1, n],\left\{q_{j}^{i}(\boldsymbol{X})\right\}_{j \in[1, m]}: f_{i}(\boldsymbol{X})-f_{i}(\boldsymbol{v})=\sum_{j=1}^{m} q_{j}^{i}(\boldsymbol{X})\left(X_{j}-v_{j}\right)$
3: $\forall j \in[1, m], \boldsymbol{q}_{j}(\boldsymbol{X})=\left(q_{j}^{i}(\boldsymbol{X})\right)_{i \in[1, n]}, \hat{q}_{j}(\boldsymbol{X})=\left\langle\boldsymbol{q}_{j}(\boldsymbol{X}), \boldsymbol{r}^{n}\right\rangle$
4: $\forall j \in[1, m], \mu_{j} \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$
5: $\pi=\left\{g^{\hat{q}_{j}(\boldsymbol{\beta})} \cdot h^{\mu_{j}}\right\}_{j \in[1, m]}$
6: $\theta=g^{\sum_{j=1}^{m} v_{j} \cdot \mu_{j}+\sum_{i=1}^{n} \rho_{i} \cdot r^{i-1}} / \prod_{j=1}^{m} g^{\beta_{j} \cdot \mu_{j}}$
$1 / 0 \leftarrow \operatorname{BatchEval}\left(\operatorname{srs}, \boldsymbol{v},\left(c_{f_{i}}, e_{\boldsymbol{v}}^{[i]}\right)_{i \in[1, n]}, \pi, \theta\right)$
1: Phase $\pi$ as $\left\{Q_{j}\right\}_{j \in[1, n]}$
2: $r=H\left(\boldsymbol{v},\left(c_{f_{i}}\right)_{i \in[1, n]}\right)$
3: $c_{\hat{f}}=\left\langle\left(c_{f_{i}}\right)_{i \in[1, n]}, \boldsymbol{r}^{n}\right\rangle$
4: $e_{\hat{v}}=\left\langle\left(e_{v}^{[i]}\right)_{i \in[1, n]}, \boldsymbol{r}^{n}\right\rangle$
$5: 1 / 0 \leftarrow \operatorname{Eval}\left(\operatorname{srs}, \boldsymbol{v}, c_{\hat{f}}, e_{\hat{v}}, \pi, \theta\right)$

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 ${ }_{\text {Eval }}$, Prove, and Eval.
$-\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, n, m, \hat{d})$. On input a security parameter $\lambda$, the number of polynomials $n$, the number of variables $m$ per polynomial, and a degree bound vector $\hat{\boldsymbol{d}}$ of polynomials, outputs a structed reference string srs.
$-c_{\boldsymbol{f}} \leftarrow$ Commit(srs, $\left.\boldsymbol{f}\right)$. Commit outputs a commitment $c_{\boldsymbol{f}}$ on $\boldsymbol{f}$.
$-c_{\boldsymbol{v}} \leftarrow \operatorname{Commit}_{\text {Eval }}(\mathrm{srs}, \boldsymbol{v}, \boldsymbol{f})$. Commit ${ }_{\mathrm{Eval}}$ outputs a commitment $c_{\boldsymbol{v}}$ on evaluations $\boldsymbol{f}(\boldsymbol{v})$.
$-\pi \leftarrow \operatorname{Prove}\left(\operatorname{srs}, \boldsymbol{v}, \boldsymbol{f}, c_{\boldsymbol{f}}\right)$. Prove generates a proof $\pi$ for evaluations of $\boldsymbol{f}$ on vector $\boldsymbol{v}$.
$-1 / 0 \leftarrow \operatorname{Eval}\left(\mathrm{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi\right)$. 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 $\boldsymbol{v} \in \mathbb{F}^{m}$,

$$
\operatorname{Pr}\left[\begin{array}{l}
\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, n, m, \hat{d}), \\
c_{\boldsymbol{f}} \leftarrow \operatorname{Commit}^{\operatorname{Coms}, \boldsymbol{f}),} \\
c_{\boldsymbol{v}} \leftarrow \operatorname{Commit}_{\text {Eval }}(\operatorname{srs}, \boldsymbol{v}, \boldsymbol{f}), \\
\pi \leftarrow \operatorname{Prove}\left(\operatorname{srs}, \boldsymbol{v}, \boldsymbol{f}, c_{\boldsymbol{f}}\right): \\
\quad \operatorname{Eval}\left(\operatorname{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi\right)=1
\end{array}\right]=1
$$

Polynomial Binding. A MMP commitment scheme is polynomial binding if for all PPT adversary $\mathcal{A}$,

$$
\operatorname{Pr}\left[\begin{array}{l}
\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, n, \hat{d}) \\
\left(\boldsymbol{f}, \boldsymbol{f}^{\prime}\right) \leftarrow \mathcal{A}(\operatorname{srs}): \\
\quad \operatorname{Commit}(\operatorname{srs}, \boldsymbol{f}) \neq \operatorname{Commit}\left(\mathrm{srs}, \boldsymbol{f}^{\prime}\right)
\end{array}\right]=\mathrm{negl}
$$

Evaluation Binding. A MMP commitment scheme is evaluation binding if for all PPT adversary $\mathcal{A}$,

$$
\operatorname{Pr}\left[\begin{array}{l}
\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, n, m, \hat{d}), \\
\left(\boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi, c_{\boldsymbol{v}}^{\prime}, \pi^{\prime}\right) \leftarrow \mathcal{A}(\mathrm{srs}): \\
\operatorname{Eval}\left(\operatorname{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi\right)=1 \\
\wedge \\
\operatorname{Eval}\left(\operatorname{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}^{\prime}, \pi^{\prime}\right)=1
\end{array}\right]=\mathrm{negl}
$$

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

$$
\operatorname{Pr}\left[\begin{array}{l}
\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, n, m, \hat{d}) \\
\left(\boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi\right) \leftarrow \mathcal{A}(\text { srs }), \\
\left(\boldsymbol{f},\left(e_{\boldsymbol{v}}^{[i]}\right)_{i=1}^{n}\right) \leftarrow \mathcal{E}(\text { srs }): \\
c_{\boldsymbol{v}}=\operatorname{Commit} \\
\wedge \quad \\
\begin{array}{l}
\text { Eval } \\
\\
\wedge
\end{array} \\
\left(\operatorname{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi\right)=1 \\
\left(e_{\boldsymbol{v}}^{[i]}\right)_{i=1}^{n} \neq \boldsymbol{f}(\boldsymbol{v})
\end{array}\right]=\operatorname{fegl}
$$

### 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., $\phi:=\left(\text { pst.Commit }\left(\text { srs }_{\text {pst }}, f_{j}\right)\right)_{i \in[1, n]}$ and the second tier is a Pedersen commitment, i.e., $c_{f}=\operatorname{afg} . C o m m i t\left(s_{\mathrm{s}_{\mathrm{afg}}}, \boldsymbol{\phi}\right)$.
- the evaluation commitment algorithm is a Pedersen commitment, i.e, $c_{v}=$ ped.Commit( $\left.\left(\operatorname{sr}_{\text {ped }}, \boldsymbol{\varphi}\right)\right)$ where $\boldsymbol{\varphi}=\left(f_{i}(\boldsymbol{v})\right)_{i \in[1, n]}$ and
- the prover initiates a batched evaluation proof with polynomial commitments $\phi$ on $\boldsymbol{v}$. 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. $\left\langle\hat{\boldsymbol{\eta}}, \boldsymbol{r}^{n}\right\rangle=\hat{\eta}(r)$,
2. $\left\langle\hat{\boldsymbol{\eta}}, \boldsymbol{g}_{\text {afg }}\right\rangle=g^{\hat{\eta}(\gamma)}$ or $\left\langle\hat{\boldsymbol{\eta}}, \boldsymbol{g}_{\text {ped }}\right\rangle=g^{\hat{\eta}(\alpha)}$ is a KZG commitment to polynomial $\hat{\eta}(X)$ with structure reference string $\boldsymbol{g}_{\text {afg }}$ or $\boldsymbol{g}_{\text {ped }}$.

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.

```
\(\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, n, \hat{d})\)
Combine pst.Setup, afg.Setup, and pst.Setup as follows:
\(1:\left(e, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_{T}, p\right) \leftarrow \operatorname{BGGen}(\lambda)\)
2: \(\alpha, \beta_{1}, \ldots, \beta_{n}, \gamma \stackrel{R}{\leftarrow} \mathbb{Z}_{p}^{n}\)
3: \(\operatorname{srs}_{\text {ped }}=\left(p, \boldsymbol{g}_{\text {ped }}=\left(g^{\alpha^{j-1}}\right)_{j \in[1, n]}\right)\)
4: \(\mathrm{srs}_{\text {pst }}=\left(p, \boldsymbol{g}_{\text {pst }}=\left(g^{\prod_{i=1}^{m} \beta_{i}^{d_{i}}}\right)_{d_{1} \in\left[0, \hat{d}_{1}\right], \ldots, d_{m} \in\left[0, \hat{d}_{m}\right]}, \tilde{g},\left(\widetilde{g}^{\beta_{j}}\right)_{j \in[1, m]}\right)\)
    \(\operatorname{srs}_{\mathrm{afg}}=\left(p, \boldsymbol{g}_{\mathrm{afg}}=\left(\widetilde{g}^{\gamma^{j-1}}\right)_{j \in[1, n]}\right)\)
    \(6: \mathrm{srs}=\left(\mathrm{srs}_{\text {ped }}\right.\), srs \(\left._{\mathrm{pst}}, \mathrm{srs}_{\mathrm{afg}}\right)\)
```

$c_{\boldsymbol{f}} \leftarrow \operatorname{Commit}($ srs, $\boldsymbol{f})$
1: $\boldsymbol{\phi} \leftarrow$ pst.Commit $\left(\right.$ srs $\left._{\text {pst }}, \boldsymbol{f}\right) \quad \triangleright$ Shorthand for $\phi \leftarrow\left(\right.$ pst.Commit(srspst,$\left.\left.f_{j}\right)\right)_{i \in[1, n]}$
2: $c_{f} \leftarrow \operatorname{afg}$.Commit( srs $\left._{\mathrm{afg}}, \boldsymbol{\phi}\right)$
$\triangleright c_{f}=\left\langle\phi, g_{\mathrm{afg}}\right\rangle$
$c_{\boldsymbol{v}} \leftarrow \operatorname{Commit}_{\text {Eval }}($ srs, $\boldsymbol{v}, \boldsymbol{f})$
1: $\boldsymbol{\varphi}=\left(f_{j}(\boldsymbol{v})\right)_{j \in[1, n]}, c_{\boldsymbol{v}}=$ ped.Commit $\left(\operatorname{srs}_{\text {ped }}, \boldsymbol{\varphi}\right) \quad \triangleright c_{\boldsymbol{v}}=\left\langle\boldsymbol{\varphi}, \boldsymbol{g}_{\text {ped }}\right\rangle$
$\pi \leftarrow \operatorname{Prove}\left(\mathrm{srs}, \boldsymbol{f}, c_{\boldsymbol{f}}, \boldsymbol{v} \in \mathbb{Z}_{p}^{m}\right)$
1: $c_{\boldsymbol{v}} \leftarrow$ Commit $_{\text {Eval }}($ srs $, \boldsymbol{v}, \boldsymbol{f})$
2: $r=H\left(c_{\boldsymbol{f}}, \boldsymbol{v}, c_{\boldsymbol{v}}\right)$
3: $\hat{f}=\left\langle\boldsymbol{f}, \boldsymbol{r}^{n}\right\rangle, c_{\hat{f}}=\left\langle\boldsymbol{\phi}, \boldsymbol{r}^{n}\right\rangle, e_{\hat{v}}=\left\langle\boldsymbol{\varphi}, \boldsymbol{r}^{n}\right\rangle$
4: $\pi_{\text {pst }} \leftarrow$ pst.Prove $\left(\right.$ srs $\left., \hat{f}, c_{\hat{f}}, \boldsymbol{v}\right)$
5: $\pi_{\mathrm{gipa}}^{[1]} \leftarrow \operatorname{gipa} . \operatorname{Prove}\left(n, \emptyset, \phi, \boldsymbol{g}_{\mathrm{afg}}, c_{\boldsymbol{f}}, \boldsymbol{r}^{n}, c_{\hat{f}}\right)$
6: $\pi_{\text {gipa }}^{[2]} \leftarrow \operatorname{gipa.Prove}\left(n, \emptyset, \boldsymbol{\varphi}, \boldsymbol{g}_{\text {ped }}, c_{\boldsymbol{v}}, \boldsymbol{r}^{n}, e_{\hat{v}}\right)$
$7: \pi=\left(\pi_{\text {pst }}, \pi_{\text {gipa }}^{[1]}, \pi_{\text {gipa }}^{[2]}, c_{\hat{f}}, e_{\hat{v}}\right)$
$1 / 0 \leftarrow \operatorname{Eval}\left(\operatorname{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi\right)$
Parse $\pi=\left(\pi_{\text {pst }}, \pi_{\text {gipa }}^{[1]}, \pi_{\text {gipa }}^{[2]}, c_{\hat{f}}, e_{\hat{v}}\right)$
$r=H\left(c_{\boldsymbol{f}}, \boldsymbol{v}, c_{\boldsymbol{v}}\right)$
$1 / 0 \leftarrow$ pst.Eval(srspst $\left., c_{\hat{f}}, e_{\hat{f}}, \boldsymbol{v}, \pi_{\text {pst }}\right)$
$1 / 0 \leftarrow \operatorname{gipa.} \operatorname{Verify}\left(n, \boldsymbol{g}_{\text {afg }}, c_{\boldsymbol{f}}, \boldsymbol{r}^{n}, c_{\hat{f}}, \pi_{\text {gipa }}^{[1]}\right)$
$5: 1 / 0 \leftarrow \operatorname{gipa} . \operatorname{Verify}\left(n, \boldsymbol{g}_{\text {ped }}, c_{\boldsymbol{v}}, \boldsymbol{r}^{n}, e_{\hat{f}}, \pi_{\text {gipa }}^{[2]}\right)$

Fig. 3: Construction of MMP Commitment Scheme

Proof. Denote $\boldsymbol{\phi}=$ pst.Commit(srs, $\boldsymbol{f})$ and $\boldsymbol{\phi}^{\prime}=$ pst.Commit(srs, $\boldsymbol{f}^{\prime}$ ). Commit(srs, $\left.\boldsymbol{f}\right)=$ Commit(srs, $\boldsymbol{f}^{\prime}$ ) implies one of the following two cases.

Case 1. $\phi \neq \phi^{\prime} \wedge$ afg.Commit(srs, $\left.\phi\right)=\operatorname{afg} . C o m m i t\left(\right.$ srs, $\left.\phi^{\prime}\right)$,

Case 2. $\phi=\phi^{\prime}$.
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 ${ }_{\text {pst }}$ 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 $\pi=\left(\pi_{\text {pst }}, \pi_{\text {gipa }}^{[1]}, \pi_{\mathrm{gipa}}^{[2]}, c_{\hat{f}}, e_{\hat{v}}, \boldsymbol{v}\right)$, along with $\operatorname{REP}_{\text {sfs }}^{c_{f}}, \operatorname{REP}_{\text {srs }}^{c_{v}}$, and $\operatorname{REP}_{\text {srs }}^{\pi}$. With these, $\mathcal{E}$ can extract $\hat{f}$ from $\operatorname{REP}_{\text {srs }}^{\pi}$. By the witness extended emulation of GIPA protocol, we know that $c_{\hat{f}}=\left\langle\varphi, \boldsymbol{r}^{n}\right\rangle, \hat{f}=\left\langle\boldsymbol{f}, \boldsymbol{r}^{n}\right\rangle$, and $e_{\hat{v}}=$ $\left\langle\boldsymbol{\varphi}, \boldsymbol{r}^{n}\right\rangle . \mathcal{E}$ rewinds the protocol with $n$ different $r$ and obtains $\boldsymbol{f}=\left(f_{i}\right)_{i \in[1, n]}$, $\phi=\left(c_{f_{i}}\right)_{i \in[1, n]}, \boldsymbol{\varphi}=\left(e_{v}^{[i]}\right)_{i \in[1, n]}$. Finally, by the knowledge soundness of PST commitment in the batched setting, $\forall i \in[1, n], f_{i}(\boldsymbol{v})=e_{\boldsymbol{v}}^{[i]}$ if Lemma 1 holds.

### 3.4 Efficiency and Implementation

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.

- Commit: $n \cdot\binom{m+d}{d} \mathbb{G}+n P$
- Prove: $\left(n \cdot\binom{m+d}{d}+m \cdot\binom{m+d}{d}\right) \mathbb{Z}_{p}+\left(n+m \cdot\binom{m+d}{d}\right) \mathbb{G}+n \widetilde{\mathbb{G}}+n P$
- Eval: $\log n \mathbb{Z}_{p}+\log n \mathbb{G}+m \widetilde{\mathbb{G}}+\log n \mathbb{G}_{T}+m P$

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 $\sim 1 \mathrm{k}$ 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 $2^{1} 1$, 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 $10-20 \mathrm{~ms}$ to complete.

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


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 $\left(g^{f_{i}(\boldsymbol{v})}\right)_{i \in[1, n]}$ and prove its consistency with $c_{\boldsymbol{v}}$ using the aggregation property of Pedersen Subvector commitment,
- Prove $\left(g^{f_{i}(\boldsymbol{v})}\right)_{i \in[1, n]}$ are committed evaluations for $c_{\boldsymbol{f}}$ using batch techniques, and

```
\(\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, n, \hat{d})\)
    1: \(\left(e, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_{T}, p\right) \leftarrow \operatorname{BGGen}(\lambda)\)
    2: \(\alpha, \beta_{1}, \ldots, \beta_{m}, \gamma, \delta \stackrel{R}{\leftarrow} \mathbb{Z}_{p}\)
    3: \(h=g^{\delta}, \widetilde{h}=\widetilde{g}^{\delta}\)
    4: \(\forall i \in[1,2 n], g_{i}=g^{\alpha^{i-1}}\)
    5: \(\forall i \in[1, n], \widetilde{g}_{i}=g^{\alpha^{i}}, h_{i}=h^{\alpha^{i}}\)
    6: \(\operatorname{srs}_{\text {psv }}=\left(p, \boldsymbol{g}_{\text {psv }}=\left(g_{i}\right)_{i \in[1, n]},\left(g_{i}\right)_{i \in[n+2,2 n]},\left(\widetilde{g}_{i}\right)_{i \in[1, n]}, h,\left(h_{i}\right)_{i \in[1, n]}\right)\)
    7: srS \(_{\text {hpst }}=\left(p, \boldsymbol{g}_{\text {hpst }}=\left(g^{\prod_{i=1}^{m} \beta_{i}^{d_{j}}}\right)_{d_{1} \in\left[0, \hat{d}_{1}\right], \ldots, d_{m} \in\left[0, \hat{d}_{m}\right]}, \widetilde{g},\left(\widetilde{g}^{\beta_{j}}\right)_{j \in[1, m]}, h, \widetilde{h}\right)\)
8: \(\operatorname{srs}_{\text {afg }}=\left(p, \boldsymbol{g}_{\text {afg }}=\left(\widetilde{g}^{\gamma^{i-1}}\right)_{i \in[1, n]}\right)\)
9: \(\mathrm{srs}=\left(\mathrm{srs}_{\mathrm{pst}}\right.\), srs \(\left._{\mathrm{afg}}, \mathrm{srs}_{\mathrm{psv}}\right)\)
\(c_{f} \leftarrow \operatorname{Commit}(\) srs, \(\boldsymbol{f})\)
    1: \(\boldsymbol{\phi} \leftarrow\) hpst.Commit(srshpst, \(\left.\boldsymbol{f} ; \boldsymbol{\rho} \stackrel{R}{\leftarrow} \mathbb{Z}_{p}^{n}\right) \quad \triangleright\) Shorthand for \(\boldsymbol{\phi}=\left(\right.\) hpst.Commit( srshpst \(\left.\left.^{\prime}, f_{i} ; \rho_{j}\right)\right)_{i \in[1, n]}\)
2: \(c_{\boldsymbol{f}} \leftarrow \operatorname{afg}\).Commit \(\left(\mathrm{srs}_{\mathrm{afg}}, \phi\right)\)
    \(\triangleright c_{\boldsymbol{f}}=\left\langle\phi, \boldsymbol{g}_{\mathrm{afg}}\right\rangle\)
\(c_{\boldsymbol{v}} \leftarrow\) Commit \(_{\text {Eval }}(\) srs, \(\boldsymbol{v}, \boldsymbol{f})\)
    1: \(\boldsymbol{\varphi}=\left(f_{i}(\boldsymbol{v})\right)_{i \in[1, n]}\)
2: \(c_{\boldsymbol{v}} \leftarrow \operatorname{psv} . C o m m i t\left(\operatorname{srs}, \boldsymbol{\varphi} ; \varrho^{\prime} \stackrel{R}{\leftarrow} \mathbb{Z}_{p}\right) \quad \triangleright c_{\boldsymbol{v}}=\left\langle\boldsymbol{\varphi}, \boldsymbol{g}_{\mathrm{psv}}\right\rangle \cdot\left\langle\varrho^{\prime}, h\right\rangle\)
\(\pi \leftarrow \operatorname{Prove}\left(\right.\) srs, \(\left.\boldsymbol{f}, c_{\boldsymbol{f}}, \boldsymbol{v} \in \mathbb{Z}_{p}^{m}, \boldsymbol{\rho}=\left(\rho_{i}\right)_{i \in[1, n]}\right)\)
    1: \(c_{\boldsymbol{v}} \leftarrow \operatorname{Commit}_{\text {Eval }}\left(\operatorname{srs}, \boldsymbol{v}, \boldsymbol{f} ; \varrho^{\prime} \stackrel{R}{\leftarrow} \mathbb{Z}_{p}\right)\)
    2: \(r=H\left(c_{\boldsymbol{f}}, \boldsymbol{v}, c_{\boldsymbol{v}}\right)\)
3: \(c_{\hat{f}}=\left\langle\phi, \boldsymbol{r}^{n}\right\rangle \quad \triangleright\) Aggregation of polynomial commitments
4: \(\pi_{\text {gipa }} \leftarrow\) gipa.Prove \(\left(n, \emptyset, \phi, \boldsymbol{g}_{\mathrm{afg}}, c_{\boldsymbol{f}}, \boldsymbol{r}^{n}, c_{\hat{f}}\right) \quad \triangleright\) Proof for aggregation of polynomial commitments
5: \(\varrho \stackrel{R}{\leftarrow} \mathbb{Z}_{p}\)
6: \(c_{\hat{\boldsymbol{v}}}=\left\langle\left(g^{f_{i}(\boldsymbol{v})}\right)_{i \in[1, n]}, \boldsymbol{r}\right\rangle \cdot\langle\varrho, h\rangle=g^{\sum_{i=1}^{n} f_{i}(\boldsymbol{v}) \cdot r^{i-1}} h^{\varrho} \quad \triangleright\) Aggregation of evaluation commitments
7: \(c^{\prime}=\prod_{i=1}^{n} \prod_{j=1, j \neq i}^{n} g_{n+1-i+j}^{f_{j}(\boldsymbol{v}) \cdot r^{i-1}} \cdot \prod_{i=1}^{n} h_{n+1-i}^{\varrho^{\prime} \cdot r^{i-1}} / h_{n}^{\varrho}\)
    \(\varrho_{1}, \varrho_{2} \stackrel{R}{\leftarrow} \mathbb{Z}_{p}, D=g^{\varrho_{1}} h^{\varrho_{2}}, \eta=H\left(r, c_{\hat{v}}, c^{\prime}, D\right), b_{1}=\varrho_{1}-\eta\left(\sum_{i=1}^{n} f_{i}(\boldsymbol{v}) \cdot r^{i-1}\right), b_{2}=\varrho_{2}-\eta \cdot \varrho\)
    \(\pi_{\mathrm{agg}}=\left(c_{\hat{v}}, c^{\prime}, D, b_{1}, b_{2}\right) \quad \triangleright\) Proof for aggregation of evaluation commitments
8: \(\forall i \in[1, n],\left\{q_{j}^{i}(\boldsymbol{X})\right\}_{j \in[1, m]}: f_{i}(\boldsymbol{X})-f_{i}(\boldsymbol{v})=\sum_{j=1}^{m} q_{j}^{i}(\boldsymbol{X})\left(X_{j}-v_{j}\right)\)
    \(\forall j \in[1, m], \boldsymbol{q}_{j}(\boldsymbol{X})=\left(q_{j}^{i}(\boldsymbol{X})\right)_{i \in[1, n]}, q_{j}(\boldsymbol{X})=\left\langle\boldsymbol{q}_{j}(\boldsymbol{X}), \boldsymbol{r}^{n}\right\rangle\)
    \(\forall j \in[1, m], \mu_{j} \stackrel{R}{\leftarrow} \mathbb{Z}_{p}\)
    \(\pi_{\mathrm{pst}}=\left(g^{q_{j}(\boldsymbol{\beta})} \cdot h^{\mu_{j}}\right)_{j \in[1, m]}\)
    \(\theta=g^{\sum_{j=1}^{m} v_{j} \cdot \mu_{j}+\sum_{i=1}^{n} \rho_{i} \cdot r^{i-1}-\varrho} \cdot \prod_{j=1}^{m} g^{-\beta_{j} \cdot \mu_{j}} \quad \triangleright\) Batched proof for correct evaluations
\(1 / 0 \leftarrow \operatorname{Eval}\left(\operatorname{srs}, \boldsymbol{v}, c_{\boldsymbol{f}}, c_{\boldsymbol{v}}, \pi\right)\)
    : gipa.Verify \(\left(c_{\boldsymbol{f}}, c_{\hat{f}}, \boldsymbol{g}_{\text {afg }}, \boldsymbol{r}^{n}, \pi_{\text {gipa }}\right) \stackrel{?}{=} 1\)
    Parse \(\pi_{\text {agg }}=\left(c_{\hat{v}}, c^{\prime}, D, b_{1}, b_{2}\right)\)
    \(\eta=H\left(r, c_{\hat{v}}, c^{\prime}, D\right), g^{b_{1}} h^{b_{2}} c_{\hat{v}}^{\eta} \stackrel{?}{=} D, e\left(c_{\boldsymbol{v}}, \prod_{i=1}^{n} \widetilde{g}_{n+1-i}^{r^{i-1}}\right) \stackrel{?}{=} e\left(c^{\prime}, \widetilde{g}\right) \cdot e\left(c_{\hat{v}}, \widetilde{g}_{n}\right)\)
    Parse \(\pi_{\text {pst }}\) as \(\left(Q_{j}\right)_{j \in[1, m]}\)
    \(: e\left(c_{\hat{f}} / c_{\hat{v}}, \widetilde{g}\right) \stackrel{?}{=} \prod_{i=1}^{m} e\left(Q_{j}, \widetilde{g}^{\beta_{j}-v_{j}}\right) \cdot e(\theta, \widetilde{h})\)
```

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 $c_{\boldsymbol{v}}=\prod_{i=1}^{n} g_{i}^{v_{i}} g^{\omega}$, the prover computes $c=\widetilde{g}_{l}^{\sum_{i=1}^{n} z^{i+1} \cdot v_{i}}$ 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 $\forall i \in[1, n], v_{i} \in\left[0,2^{l}-1\right]$.

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

$$
\begin{align*}
\boldsymbol{a} \circ \overline{\boldsymbol{a}} & =\mathbf{0}^{n l} \\
\boldsymbol{a}-\overline{\boldsymbol{a}}-\mathbf{1}^{n l} & =\mathbf{0}^{n l} \\
\left\langle\boldsymbol{a}, V_{1}\right\rangle & =v_{1} \\
\ldots &  \tag{1}\\
\left\langle\boldsymbol{a}, V_{i}\right\rangle & =v_{i} \\
\ldots & \\
\left\langle\boldsymbol{a}, V_{n}\right\rangle & =v_{n}
\end{align*}
$$

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

$$
\begin{equation*}
\langle\boldsymbol{l}, \boldsymbol{r}\rangle=\left\langle\boldsymbol{v}, z^{2} \cdot \boldsymbol{z}^{n}\right\rangle+\zeta(y, z) \tag{2}
\end{equation*}
$$

where

$$
\begin{gathered}
\boldsymbol{l}=\boldsymbol{a}-z \cdot \mathbf{1}^{n l}, \\
\boldsymbol{r}=\boldsymbol{y}^{n l} \circ\left(\overline{\boldsymbol{a}}+z \cdot \mathbf{1}^{n l}\right)+\sum_{i=1}^{n} z^{i+1} \cdot V_{i}, \\
\zeta(y, z)=\left(z-z^{2}\right) \cdot\left\langle\mathbf{1}^{n l}, \boldsymbol{y}^{n l}\right\rangle-\sum_{i=1}^{n} z^{i+2} \cdot\left\langle\mathbf{1}^{l}, \mathbf{2}^{l}\right\rangle .
\end{gathered}
$$

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

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

However, as the vector elements are encoded in different public parameters (i.e., $\left.g_{1}, \ldots, g_{n}\right)$, Pedersen subvector commitment cannot straightforwardly aggregate vectors as batched Pedersen commitments do (which is simply the inner product of $\boldsymbol{c}$ and $\left.z^{2} \cdot \boldsymbol{z}\right)$. To fill in this gap, we introduce $\mathrm{AoK}_{\mathrm{agg}}$, 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 ${ }_{\text {agg }}$ 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 $\boldsymbol{v}_{S}$ such that $c=g^{\sum_{i \in S} z^{\sigma(i)+1} v_{i}}$ and a proof element

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

$-\mathcal{V}$ checks the correctness of aggregation by 1$)$ checking $e\left(c_{\boldsymbol{v}}, \prod_{i \in S} \widetilde{g}_{n+1-i}^{z^{\sigma(i)+1}}\right) \stackrel{?}{=}$ $e\left(c^{\prime}, \widetilde{g}\right) \cdot e\left(c, \widetilde{g}_{n}\right)$, 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 $b=\varrho-\eta \cdot\left(\sum_{i \in S} z^{\sigma(i)+1} \cdot v_{i}\right)$ 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 $n^{\prime}$ different $z$ and extracting $\boldsymbol{v}$ from $\mathrm{REP}_{\mathrm{srs}}^{c^{\prime}}$. Aok $\mathrm{a}_{\mathrm{agg}}$ can be transformed into a zero-knowledge argument of knowledge by randomizing $c$ and $c^{\prime}$ as what is done in Figure 6 .

### 4.1 Construction

Our ZKRP protocol for Pedersen subvector commitment is a combination of AoK ${ }_{\text {agg }}$ and the batched ZKRP for Pedersen commitments in [16. With a Pedersen subvector commitment generated by

$$
\begin{aligned}
& -\operatorname{srs} \leftarrow \operatorname{psv} . \operatorname{Setup}(\lambda, n) \\
& -c_{\boldsymbol{v}} \leftarrow \operatorname{psv} . \operatorname{Commit}(\operatorname{srs}, \boldsymbol{v})
\end{aligned}
$$

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

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

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.

$$
\begin{aligned}
& \mathcal{\mathcal { P } ( \text { srs } , \boldsymbol { v } , c _ { v } , S = \{ i _ { j } \} _ { j \in [ 1 , n ^ { \prime } ] } ) \quad \mathcal { V } ( \text { srs } , c _ { v } , S = \{ i _ { j } \} _ { j \in [ 1 , n ^ { \prime } ] } )} \\
& \sigma: \forall i_{j} \in S, \sigma\left(i_{j}\right)=j \\
& \boldsymbol{a} \in\{0,1\}^{n^{\prime} l} \text { s.t. } \forall j \in\left[1, n^{\prime}\right],\left\langle\boldsymbol{a}, \mathbf{0}^{(j-1) l}\left\|\boldsymbol{2}^{l}\right\| \mathbf{0}^{\left(n^{\prime}-j\right)}\right\rangle=v_{\sigma^{-1}(j)} \\
& \bar{a}=1^{n^{\prime}}-a \\
& \boldsymbol{J}, \boldsymbol{k} \stackrel{R}{\leftarrow} \mathbb{G}^{n^{\prime} l}
\end{aligned}
$$

$$
\begin{aligned}
& A=\boldsymbol{J}^{a} \boldsymbol{k}^{\bar{a}} h^{\rho_{1}}, B=\boldsymbol{J}^{\rho_{l}} \boldsymbol{K}^{\rho_{r}} h^{\rho_{2}} \\
& \xrightarrow{A, B} \\
& \longleftarrow y, z \quad y, z \stackrel{R}{\leftarrow} \mathbb{Z}_{p} \\
& l(X)=\boldsymbol{a}-z \cdot \mathbf{1}^{n^{\prime} l}+\rho_{l} \cdot X \\
& r(X)=y^{n^{\prime} l} \circ\left(\bar{a}+z \cdot \mathbf{1}^{n^{\prime} l}+\rho_{r} \cdot X\right) \\
& +\sum_{i=1}^{n^{\prime}} z^{i+1} \cdot\left(\mathbf{0}^{(i-1) l}\left\|\mathbf{2}^{l}\right\| \mathbf{0}^{\left(n^{\prime}-i\right) l}\right) \\
& t(X)=\langle l(X), r(X)\rangle:=t_{0}+t_{1} \cdot X+t_{2} \cdot X^{2} \\
& \tau_{1}, \tau_{2} \stackrel{R}{\stackrel{R}{\sim}} \mathbb{Z}_{p} \\
& T_{1}=g^{t_{1}} h^{\tau_{1}}, T_{2}=g^{t_{2}} h^{\tau_{2}} \\
& \gamma \stackrel{R}{\leftarrow} \mathbb{Z}_{p} \\
& c=g^{\sum_{i \in S}} z^{\sigma(i)+1} v_{i} . h^{\gamma} \\
& c^{\prime}=\prod_{i \in S} \prod_{j=1, j \neq i}^{n} 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} / h_{n}^{\gamma} \\
& \varrho_{1}, \varrho_{2} \stackrel{R}{\leftarrow} \mathbb{Z}_{p}, D=g^{\varrho_{1}} h^{\varrho_{2}} \\
& \xrightarrow{T_{1}, T_{2}, c, c^{\prime}, D} \\
& \longleftarrow \eta{ }^{R}{ }^{R} \mathbb{Z}_{p} \\
& b_{1}=\varrho_{1}-\eta \cdot\left(\sum_{i \in S} z^{\sigma(i)+1} \cdot v_{i}\right) \\
& b_{2}=\varrho_{2}-\eta \cdot \gamma \\
& \xrightarrow{b_{1}, b_{2}} e\left(c_{\boldsymbol{v}}, \prod_{i \in S} \widetilde{g}_{n+1-i}^{z^{\sigma(i)+1}}\right) \stackrel{?}{=} e\left(c^{\prime}, \widetilde{g}\right) \cdot e\left(c, \widetilde{g}_{n}\right) \\
& D \stackrel{?}{=} c^{\eta} g^{b_{1}} h^{b_{2}} \\
& \longleftarrow x{ }^{x} x \stackrel{R}{\leftarrow} \mathbb{Z}_{p} \\
& \boldsymbol{l}_{x}=l(x), \boldsymbol{r}_{x}=r(x), t_{x}=\left\langle\boldsymbol{l}_{x}, \boldsymbol{r}_{x}\right\rangle \\
& \rho_{x}=\rho_{1}+\rho_{2} \cdot x, \tau_{x}=\tau_{1} \cdot x+\tau_{2} \cdot x^{2}+\gamma \\
& \xrightarrow{\boldsymbol{l}_{x}, \boldsymbol{r}_{x}, t_{x}, \rho_{x}, \tau_{x}} \boldsymbol{K}^{\prime}:=\boldsymbol{K}^{y^{-n^{\prime} l}} \\
& \zeta(y, z)=\left(z-z^{2}\right) \cdot \underline{\left\langle\mathbf{1}^{n^{\prime} l}, \boldsymbol{y}^{n^{\prime} l}\right\rangle}-\sum_{j=1}^{n^{\prime}} z^{j+2} \cdot\left\langle\mathbf{1}^{l}, \mathbf{2}^{l}\right\rangle \\
& g^{t_{x}} h^{\tau_{x}} \stackrel{?}{=} c \cdot g^{\zeta(y, z)} \cdot T_{1}^{x} \cdot T_{2}^{x^{2}} \\
& P=A B^{x} \boldsymbol{J}^{-z} \underline{K}^{\prime z y^{n^{\prime} l}} \prod_{j=1}^{n^{\prime}} \boldsymbol{K}_{[(j-1) \cdot l, j \cdot l-1]}^{\prime \prime j-1} \cdot{ }^{\prime}{ }^{l} \\
& P / h^{\rho_{x}} \stackrel{?}{=} \underline{\boldsymbol{J}}^{\boldsymbol{l}_{x}} \boldsymbol{K}^{\prime \boldsymbol{r}_{x}} \\
& t_{x} \stackrel{?}{=}\left\langle\boldsymbol{l}_{x}, \boldsymbol{r}_{x}\right\rangle
\end{aligned}
$$

Fig. 6: Range Proof for Pedersen Subvector Commitment. The proof size is linear to $n^{\prime} l$ due to the transfer of $\boldsymbol{l}_{x}$ and $\boldsymbol{r}_{x}$. 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.

### 4.2 Security Analysis

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., $\alpha$, and simulates the protocol as below:

- Choose $z, \hat{v}, \gamma \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$
- Let $c^{\prime}=c^{\sum_{\boldsymbol{v}}^{n} \alpha^{n+1-i} \cdot z^{i+1}} \cdot g^{\alpha^{n} \cdot \hat{v}} \cdot h_{n}^{\gamma}, c=g^{\hat{v}} h^{\gamma}$.
- 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

$$
\left.\left.\begin{array}{c}
T_{1}=\left(g^{t_{x}} h^{\tau_{x}} /\left(c \cdot \widetilde{g}^{\zeta(y, z)} \cdot T_{2}^{x^{2}}\right)\right)^{x^{-1}} \\
B=\left(\boldsymbol{J}^{\boldsymbol{l}_{x}+z} \boldsymbol{k}^{\prime \boldsymbol{r}_{x}-z \cdot \boldsymbol{y}^{n^{\prime} \boldsymbol{l}}} h^{\rho_{x}} /\left(A \cdot \prod_{i=1}^{n^{\prime}} \boldsymbol{k}_{[(i-1) \cdot n, j \cdot n-1]}^{\prime} z^{j-1} \cdot \boldsymbol{2}^{l}\right.\right.
\end{array}\right)^{x^{-1}}\right), ~ \$
$$

- Run inner-product argument with the simulated $\boldsymbol{l}_{x}$ and $\boldsymbol{r}_{x}$.


### 4.3 Optimization and Efficiency

Verification Optimization. The verification complexity can be improved at the cost of communication efficiency. Instead of choosing $\boldsymbol{J}$ and $\boldsymbol{K}$ randomly from $\mathbb{G}^{n}$, we set $\boldsymbol{J}=\left(J^{\theta_{1}^{i-1}}\right)_{i \in\left[1, n^{\prime}\right]}$ and $\boldsymbol{K}=\left(K^{\theta_{2}^{i-1}}\right)_{i \in\left[1, n^{\prime}\right]}$ where $J, K \stackrel{R}{\leftarrow}$ $\mathbb{G}, \theta_{1}, \theta_{2} \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$. 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 $O(n l)$ to $O(n)+O(\log (n l))$.

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 $n=n^{\prime}$. 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 | $(2 n l) \mathbb{G}+(n+1) \widetilde{\mathbb{G}}$ | $3 \mathbb{G}$ | $(2 n l+1) \mathbb{G}+(n l) \widetilde{\mathbb{G}}+4 P$ |
| This paper | $(10 n l+4 \log (n l)+2 n+13) \mathbb{G}$ | $(2 \log (n l)+7) \mathbb{G}+7 \mathbb{Z}_{p}$ | $(6 n l+2 \log (n l)+n+11) \mathbb{G}+3 P$ |
| This paper $+\mathcal{V}$ | $(10 n l+4 \log (n l)+2 n+13) \mathbb{G}$ | $(2 \log (n l)+7) \mathbb{G}+(9+2 \log (n l)) \mathbb{Z}_{p}$ | $(8 \log (n l)+n+11) \mathbb{G}+3 P$ |

## 5 Applications

## 5.1 aHyperplonk

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 $A_{i}\left(L_{i}+R_{i}\right)+M_{i}\left(L_{i} \cdot R_{i}\right)=O_{i}$.

In HyperPlonk, an indexer $\mathcal{I}$ is responsible for preprocessing the circuit. Consider a circuit $\mathcal{C}$ with $n=2^{\mu}$ constraints. $\mathcal{I}$ needs to generate the selector polynomials $q_{0}(\mathbf{X})$ and $q_{1}(\mathbf{X})$, which satisfy $q_{0}\left(\langle x\rangle_{\mu}\right)=A_{x}$ and $q_{1}\left(\langle x\rangle_{\mu}\right)=M_{x}$ for all $x \in\left[0,2^{\mu}\right)$. Here, $\langle x\rangle_{\mu} \in\{0,1\}^{\mu}$ is the binary representation of $x$ whose maximal bit-length is $\mu$. 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 $x \in\left[0,2^{\mu}\right), w\left(\langle 0\rangle_{2},\langle x\rangle_{\mu}\right)=$ $L_{x}, w\left(\langle 1\rangle_{2},\langle x\rangle_{\mu}\right)=R_{x}$, and $w\left(\langle 2\rangle_{2},\langle x\rangle_{\mu}\right)=O_{x}$. We further partition $w$ into 3 partial polynomials $w_{0}, w_{1}$ and $w_{2}$, where $w_{i}:=w\left(\langle i\rangle_{2}, \mathbf{X}\right)$.
Gate Identity. In order to convince the verifier, the prover needs to first compute the gate identity polynomial $f(\mathbf{X}):=q_{0}(\mathbf{X})\left(w_{0}(\mathbf{X})+w_{1}(\mathbf{X})\right)+q_{1}(\mathbf{X})\left(w_{0}(\mathbf{X})\right.$. $\left.w_{1}(\mathbf{X})\right)-w_{2}(\mathbf{X})$ and then prove that $f\left(\langle x\rangle_{\mu}\right)=0$ for all $x \in\left[0,2^{\mu}\right)$, 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 $w\left(\langle x\rangle_{\mu+2}\right)=w\left(\hat{\sigma}\left(\langle x\rangle_{\mu+2}\right)\right)$ for all $x \in\left[0,2^{\mu+2}\right)$.

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 $c_{w}:=\operatorname{Commit}\left(c k,\left\{w_{0}^{[j]}, w_{1}^{[j]}, w_{2}^{[j]}\right\}_{j=0}^{k-1}\right)$.
2. Get challenges $\beta, \gamma$, and $\delta$ from $\mathcal{V}$.
3. Compute $\hat{w}_{i}(\mathbf{X}):=\sum_{j=0}^{k-1} \delta^{j} w_{i}^{[j]}(\mathbf{X})$ for all $i \in[0,2]$.
4. Compute the product polynomial $\tilde{v}(\mathbf{X})$ such that $\tilde{v}\left(\left\langle 2^{\mu}-1\right\rangle_{\mu}, 0\right)=1$ and for all $x \in\left[0,2^{\mu}\right)$,

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

Define the partial polynomials of $\tilde{v}(\mathbf{X})$ as $\tilde{v}_{0}(\mathbf{X}):=\tilde{v}(0, \mathbf{X}), \tilde{v}_{1}(\mathbf{X}):=\tilde{v}(1, \mathbf{X})$. Note that given $\tilde{v}_{0}(\mathbf{X})$ and $\tilde{v}_{1}(\mathbf{X})$, one can derive $\tilde{v}(\mathbf{X}, b):=\left(1-X_{0}\right)$. $\tilde{v}_{0}\left(X_{1}, \ldots, X_{\mu-1}, b\right)+X_{0} \cdot \tilde{v}_{1}\left(X_{1}, \ldots, X_{\mu-1}, b\right)$ for $b \in\{0,1\}$.
5. Generate an MMP commitment $c_{v}:=\operatorname{Commit}\left(c k,\left\{\tilde{v}_{0}, \tilde{v}_{1}\right\}\right)$.
6. Get challenges $\alpha, \boldsymbol{\rho}=\left\{\rho_{0}, \ldots, \rho_{\mu-1}\right\}$ from $\mathcal{V}$.
7. For the virtual polynomials below,

$$
\begin{aligned}
f_{j}(\mathbf{X}) & :=q_{0}(\mathbf{X})\left(w_{0}^{[j]}(\mathbf{X})+w_{1}^{[j]}(\mathbf{X})\right)+q_{1}(\mathbf{X})\left(w_{0}^{[j]}(\mathbf{X}) \cdot w_{1}^{[j]}(\mathbf{X})\right)-w_{2}^{[j]}(\mathbf{X}) \\
Q_{1}(\mathbf{X}) & :=\tilde{v}(1, \mathbf{X})-\tilde{v}(\mathbf{X}, 0) \tilde{v}(\mathbf{X}, 1) \\
Q_{2}(\mathbf{X}) & :=\prod_{i=0}^{2}\left(\hat{w}_{i}(\mathbf{X})+\beta s_{\mathrm{id}}\left(\langle i\rangle_{2}, \mathbf{X}\right)+\gamma\right)-\tilde{v}(0, \mathbf{X}) \prod_{i=0}^{2}\left(\hat{w}_{i}(\mathbf{X})+\beta \sigma_{i}(\mathbf{X})+\gamma\right)
\end{aligned}
$$

prove that $\sum_{x=0}^{2^{\mu}-1} F\left(\langle x\rangle_{\mu}\right) \cdot e q\left(\langle x\rangle_{\mu}, \boldsymbol{\rho}\right)=0$, where $F(\mathbf{X})=\sum_{j=0}^{k-1} \alpha^{j} f_{j}(\mathbf{X})+$ $\alpha^{k} Q_{1}(\mathbf{X})+\alpha^{k+1} Q_{2}(\mathbf{X}), e q(\mathbf{X}, \boldsymbol{\rho})=\prod_{i=0}^{\mu-1}\left(X_{i} \rho_{i}+\left(1-X_{i}\right)\left(1-\rho_{i}\right)\right)$. This is done iteratively for $i=\mu-1$ down to 0 :
(a) Generate a univariate polynomial commitment (e.g., using KZG) $c_{r_{i}}:=$ kzg.Commit $\left(c k_{\mathrm{kzg}}, r_{i}\right)$, where $r_{i}(X)=\sum_{b=0}^{2^{i}-1} F\left(\langle b\rangle_{i}, X, \zeta_{i+1}, \ldots, \zeta_{\mu-1}\right)$. $e q\left(\langle b\rangle_{i}, X, \zeta_{i+1}, \ldots, \zeta_{\mu-1}, \boldsymbol{\rho}\right)$.
(b) Get a challenge $\zeta_{i}$ from $\mathcal{V}$.
8. Run the univariate batch-opening algorithm for evaluations $\left\{r_{i}(0), r_{i}(1), r_{i}\left(\zeta_{i}\right)\right\}_{i=0}^{\mu-1}$.
9. Compute the committed evaluations and the proofs below, where Prove' is a wrapper of Prove that also returns the result of Commit Eval :

$$
\begin{aligned}
& \eta_{q, \sigma}, \pi_{q, \sigma}:=\operatorname{Prove}^{\prime}\left(c k,\left\{q_{0}, q_{1}, \sigma_{0}, \sigma_{1}, \sigma_{2}\right\}, c_{q, \sigma},\left\{\zeta_{i}\right\}_{i=0}^{\mu-1}\right) \\
& \eta_{w}, \pi_{w}:=\operatorname{Prove}^{\prime}\left(c k,\left\{w_{0}^{[j]}, w_{1}^{[j]}, w_{2}^{[j]}\right\}_{j=0}^{k-1}, c_{w},\left\{\zeta_{i}\right\}_{i=0}^{\mu-1}\right) \\
& \eta_{v}^{[0]}, \pi_{v}^{[0]}:=\operatorname{Prove}^{\prime}\left(c k,\left\{\tilde{v}_{0}, \tilde{v}_{1}\right\}, c_{v},\left\{\zeta_{i}\right\}_{i=0}^{\mu-1}\right) \\
& \eta_{v}^{[1]}, \pi_{v}^{[1]}:=\operatorname{Prove}^{\prime}\left(c k,\left\{\tilde{v}_{0}, \tilde{v}_{1}\right\}, c_{v},\left\{\zeta_{i}\right\}_{i=1}^{\mu-1} \cup\{0\}\right) \\
& \eta_{v}^{[2]}, \pi_{v}^{[2]}:=\operatorname{Prove}^{\prime}\left(c k,\left\{\tilde{v}_{0}, \tilde{v}_{1}\right\}, c_{v},\left\{\zeta_{i}\right\}_{i=1}^{\mu-1} \cup\{1\}\right) \\
& \eta_{v}^{[3]}, \pi_{v}^{[3]}:=\operatorname{Prove}^{\prime}\left(c k,\left\{\tilde{v}_{0}, \tilde{v}_{1}\right\}, c_{v},\{1\}_{i=1}^{\mu-1} \cup\{0\}\right)
\end{aligned}
$$

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

$$
\begin{aligned}
& x:=\left(\beta, \gamma, \delta, \alpha,\left\{\rho_{i}\right\}_{i=0}^{\mu-1},\left\{\zeta_{i}\right\}_{i=0}^{\mu-1}, \eta_{q, \sigma}, \eta_{w},\left\{\eta_{v}^{[i]}\right\}_{i=0}^{3}, e_{F}\right) \\
& w:=\left(e_{q_{0}}, e_{q_{1}}, e_{\sigma_{0}}, e_{\sigma_{1}}, e_{\sigma_{2}},\left\{e_{w_{0}}^{[j]}, e_{w_{1}}^{[j]}, e_{w_{2}}^{[j]}\right\}_{j=0}^{k-1},\left\{e_{v_{0}}^{[i]}, e_{v_{1}}^{[i]}\right\}_{i=0}^{3}\right)
\end{aligned}
$$

Generate a SNARK proof $\pi_{\text {meta }}$ for $\mathcal{R}_{\text {meta }}$, attesting that

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

Note that we follow the meta-verification approach in 2 and require the prover to generate a SNARK proof $\pi_{\text {meta }}$ for $\mathcal{R}_{\text {meta }}$, 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 $\mathcal{R}_{\text {meta }}$ 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 $\pi_{\text {meta }}$ 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.

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

|  | Prover time | Proof size | Verifier time |
| :---: | :---: | :---: | :---: |
| HyperPlonk | $O(k n)$ | $O(k \log n)$ | $O(k \log n)$ |
| aHyperPlonk | $O(k n)$ | $O(\log k+\log n)$ | $O(\log k+\log n)$ |

### 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 $\pm 10 \mathrm{~m}$ to $\pm 30 \mathrm{~m}$ ). Suppose we add a new GPS coordinate reading whenever the forth decimal changes (which occurs every $\approx 10$ meters), e.g., ( $66.6666,88.8888$ ), we also need to include $(66.6665,88.8888),(66.6667,88.8888),(66.6666,88.8887)$, ( $66.6667,88.8889$ ) if the errors is $\pm 10$. 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 $E_{0}=\left(x_{0}, y_{0}\right)$ of a rectangular and its two adjacent vertices, $E_{1}=\left(x_{1}, y_{1}\right)$ and $E_{2}=\left(x_{2}, y_{2}\right)$, a point $P=(x, y)$ is inside the rectangle if and only if

$$
\begin{align*}
& 0<E_{0} P \cdot E_{0} E_{1}<E_{0} E_{1} \cdot E_{0} E_{1}  \tag{3}\\
& 0<E_{0} P \cdot E_{0} E_{2}<E_{0} E_{2} \cdot E_{0} E_{2} \tag{4}
\end{align*}
$$

In these formulas, $E_{0} P$ is the vector between $E_{0}$ and $P$, and the dot is the scalar product of two vectors. These constraints can be equivalently ${ }^{4}$ expressed as

$$
\begin{align*}
& \left(E_{0} P \cdot E_{0} E_{1}\right)\left(E_{0} P \cdot E_{0} E_{1}-E_{0} E_{1} \cdot E_{0} E_{1}\right)<0  \tag{5}\\
& \left(E_{0} P \cdot E_{0} E_{2}\right)\left(E_{0} P \cdot E_{0} E_{2}-E_{0} E_{2} \cdot E_{0} E_{2}\right)<0 \tag{6}
\end{align*}
$$

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 $b=\{1,2\}$,

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

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 $f_{1}(x, y)<$ $0 \wedge f_{2}(x, y)<0$. Hence, $P$ is outside the rectangle if and only if $f_{1}(x, y)>$ $0 \vee f_{2}(x, y)>0$.

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 $E_{1} \ldots E_{m}$, a point $P$ is inside the hyper-rectangles if and only if

$$
\begin{aligned}
&\left(E_{0} P \cdot E_{0} E_{1}\right) \cdot\left(E_{0} P \cdot E_{0} E_{1}-E_{0} E_{1} \cdot E_{0} E_{1}\right)<0 \\
& \ldots \\
& \wedge\left(E_{0} P \cdot E_{0} E_{m}\right) \cdot\left(E_{0} P \cdot E_{0} E_{m}-E_{0} E_{m} \cdot E_{0} E_{m}\right)<0 .
\end{aligned}
$$

[^1]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 $\exists i \in[1, k]$,

$$
\begin{aligned}
& \left(E_{0}^{i} P \cdot E_{0}^{i} E_{1}^{i}\right) \cdot\left(E_{0}^{i} P \cdot E_{0}^{i} E_{1}^{i}-E_{0}^{i} E_{1}^{i} \cdot E_{0}^{i} E_{1}^{i}\right)<0 \\
\wedge & \left(E_{0}^{i} P \cdot E_{0}^{i} E_{2}^{i}\right) \cdot\left(E_{0}^{i} P \cdot E_{0}^{i} E_{2}^{i}-E_{0}^{i} E_{2}^{i} \cdot E_{0}^{i} E_{2}^{i}\right)<0
\end{aligned}
$$

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 $\forall i \in[1, k]$,

$$
\begin{array}{r}
\left(E_{0}^{i} P \cdot E_{0}^{i} E_{1}^{i}\right) \cdot\left(E_{0}^{i} P \cdot E_{0}^{i} E_{1}^{i}-E_{0}^{i} E_{1}^{i} \cdot E_{0}^{i} E_{1}^{i}\right)>0 \\
\vee\left(E_{0}^{i} P \cdot E_{0}^{i} E_{2}^{i}\right) \cdot\left(E_{0}^{i} P \cdot E_{0}^{i} E_{2}^{i}-E_{0}^{i} E_{2}^{i} \cdot E_{0}^{i} E_{2}^{i}\right)>0
\end{array}
$$

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

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 $10^{4}$, 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

1. 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
2. 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
4. 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
5. 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
6. 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
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
8. 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
9. 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
10. 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
11. 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
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
13. 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
14. 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
15. 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. 156166. Springer, Heidelberg (Aug 1988). https://doi.org/10.1007/3-540-48184-2_11
16. 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
17. 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
18. 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
19. 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
20. 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
21. 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
23. 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
24. 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
25. 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
26. 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.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 499-530. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30617-4_17
27. 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
28. 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
29. 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
30. 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
31. 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
32. 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
34. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019), https://eprint.iacr.org/2019/953
35. 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
36. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakē̄own: 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
38. 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
39. 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
40. 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
41. 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
42. 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
43. 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
44. 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
45. 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
46. 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. 3667. Springer, Heidelberg (Apr 2024). https://doi.org/10.1007/978-3-031-57722-2_ 2
47. 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
48. 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
49. Nguyen, K., Tang, H., Wang, H., Zeng, N.: New code-based privacypreserving cryptographic constructions. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part II. LNCS, vol. 11922, pp. 25-55. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34621-8_2
50. Noether, S.: Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098 (2015), https://eprint.iacr.org/2015/1098
51. 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
52. 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
53. 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
54. 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
56. 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 zkSNARKs 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
58. 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
59. 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
60. 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.

$$
\operatorname{Pr}\left[\begin{array}{r}
\pi \leftarrow \mathcal{P}\left(n=2^{\kappa}, t s=\emptyset, \boldsymbol{a}, \boldsymbol{b}, I_{\boldsymbol{a} \boldsymbol{b}}, \boldsymbol{c}, I_{\boldsymbol{a} \boldsymbol{c}}\right): \\
\mathcal{V}\left(n, I_{\boldsymbol{a} \boldsymbol{b}}, I_{\boldsymbol{a} \boldsymbol{c}}, \pi\right)=1
\end{array}\right]=1
$$

## 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 $\left(a_{i}^{\prime}, b_{i}^{\prime}, c_{i}^{\prime}\right)$ such that

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

$\mathcal{E}$ returns

$$
\boldsymbol{a}=\sum_{i=1}^{3}\left(v_{i} h_{i}^{-1} \boldsymbol{a}_{i}^{\prime}, v_{i} h_{i} \boldsymbol{a}_{i}^{\prime}\right)
$$

We can see that $I_{a b}$ have the expected form:

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

Similarly for $\langle\boldsymbol{a}, \boldsymbol{c}\rangle=I_{a c}$.
$-\kappa \geq 2$, 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
$-\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, n)$. On input security parameter $\lambda$ and a vector length $n$, Setup outputs structured reference string srs.
$-c_{\boldsymbol{v}} \leftarrow$ Commit(srs, $\boldsymbol{v}$ ). Commit computes a vector commitment $c_{\boldsymbol{v}}$ on $\boldsymbol{v}$.
$-\pi_{S} \leftarrow$ Prove(srs, $c_{\boldsymbol{v}}, S, \boldsymbol{v}_{S}$ ). Prove generates a proof $\pi_{S}$ such that $\boldsymbol{v}_{S}$ are evaluations of $c_{v}$ on positions $S$.
$-1 / 0 \leftarrow \mathrm{Eval}\left(\operatorname{srs}, c_{\boldsymbol{v}}, S, \boldsymbol{v}_{S}, \pi_{S}\right)$. 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 $\mathcal{A}$,

Hiding. For any PPT adversary $\mathcal{A}$,

$$
\operatorname{Pr}\left[\begin{array}{ll}
\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, n), & \\
\left(\boldsymbol{v}_{1}, \boldsymbol{v}_{2}\right) \leftarrow \mathcal{A}(\mathrm{srs}), & \\
b \stackrel{R}{\leftarrow}\{0,1\}, & \\
c_{\boldsymbol{v}_{b}} \leftarrow \operatorname{Commit}\left(\operatorname{srs}, \boldsymbol{v}_{b}\right), & \\
b^{\prime} \leftarrow \mathcal{A}\left(\mathrm{srs}, c_{v_{b}}\right): & b \neq b^{\prime}
\end{array}\right]=\mathrm{negl}+\frac{1}{2}
$$

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 $\leftarrow \operatorname{Setup}(\lambda, n)$ |
| :--- |
| $1:(\mathbb{G}, p) \leftarrow \operatorname{GGen}(\lambda)$ |
| $2: g \stackrel{R}{\leftarrow} \mathbb{G}, \alpha, R_{\leftarrow} \mathbb{Z}_{p}$ |
| $3:$ srs $=\left(p, \boldsymbol{g}=\left(g^{\alpha^{i-1}}\right)_{i \in[1, n]}\right)$ |
| $c_{\boldsymbol{v}} \leftarrow \operatorname{Commit}\left(\operatorname{srs}, \boldsymbol{v} \in \mathbb{Z}_{p}^{n}\right)$ |
| $1: c_{\boldsymbol{v}}=\langle\boldsymbol{v}, \boldsymbol{g}\rangle$ |
| $1 / 0 \leftarrow \operatorname{Open}\left(\operatorname{srs}, c_{\boldsymbol{v}}, \boldsymbol{v}\right)$ |
| $1: c_{\boldsymbol{v}} \stackrel{?}{=}\langle\boldsymbol{v}, \boldsymbol{g}\rangle$ |

Fig. 8: Construction of Pedersen Commitment.

$$
\begin{aligned}
& \text { srs } \leftarrow \operatorname{Setup}(\lambda, n) \\
& \hline 1:\left(e, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_{T}, p\right) \leftarrow \operatorname{BGGen}(\lambda) \\
& 2: \widetilde{g} \stackrel{R}{\leftarrow} \widetilde{\mathbb{G}} \\
& 3: \text { srs }=\left(p, \widetilde{\boldsymbol{g}}=\left(\widetilde{g}^{i^{i-1}}\right)_{i \in[1, n]}\right) \\
& \hline c_{\boldsymbol{v}} \leftarrow \operatorname{Commit}\left(\operatorname{srs}, \boldsymbol{v} \in \mathbb{G}^{n} ; \omega\right) \\
& \hline 1: c_{\boldsymbol{v}}=\langle\boldsymbol{v}, \widetilde{\boldsymbol{g}}\rangle \\
& \hline 1 / 0 \leftarrow \operatorname{Open}\left(\text { srs }, c_{\boldsymbol{v}}, \boldsymbol{v}\right) \\
& \hline 1: c_{\boldsymbol{v}} \stackrel{?}{=}\langle\boldsymbol{v}, \widetilde{\boldsymbol{g}}\rangle
\end{aligned}
$$

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 $\leq \hat{\boldsymbol{d}}$, and evaluation vector $\boldsymbol{v} \in \mathbb{F}^{m}$,

$$
\operatorname{Pr}\left[\right]=1
$$

```
\(\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, n)\)
\(1:\left(e, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_{T}, p\right) \leftarrow \operatorname{BGGen}(\lambda)\)
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: \(\forall i \in[1,2 n], g_{i}=g^{\alpha^{i-1}}\),
6: \(\forall i \in[1, n], \widetilde{g}_{i}=\widetilde{g}^{\alpha^{i}}, h_{i}=h^{\alpha^{i}}\)
7: srs \(=\left(p, g, \boldsymbol{g}=\left(g_{i}\right)_{i \in[1, n]},\left(g_{i}\right)_{i \in[n+2,2 n]}, \widetilde{g},\left(\widetilde{g}_{i},\right)_{i \in[1, n]}, h,\left(h_{i}\right)_{i \in[1, n]}\right)\)
```

$c_{\boldsymbol{v}} \leftarrow$ Commit(srs, $\boldsymbol{v}=\left(v_{i}\right)_{i \in[1, n]}$
1: $\omega \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$
$2: c_{\boldsymbol{v}}=\langle\boldsymbol{v}, \boldsymbol{g}\rangle \cdot\langle\omega, h\rangle \quad \triangleright c_{\boldsymbol{v}}=\prod_{i=1}^{n} g_{i}^{v_{i}} \cdot h^{\omega}$
$\pi_{S} \leftarrow \operatorname{Prove}\left(\mathrm{srs}, c_{\boldsymbol{v}}, S, \boldsymbol{v}, \omega\right)$
1: $z=H\left(c_{\boldsymbol{v}}, S, \boldsymbol{v}_{S}\right)$
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: $\pi_{S}=\prod_{i \in S} \pi_{i}^{z^{i-1}} \quad \triangleright \pi_{S}=\prod_{i \in S} \prod_{j=1, j \neq i}^{n} g_{n+1-i+j}^{z^{i-1} \cdot v_{j}} \cdot \prod_{i \in S} h_{n+1-i}^{z^{i-1} \cdot \omega}$
$1 \leftarrow \operatorname{Eval}\left(\mathrm{srs}, c_{\boldsymbol{v}}, S, \boldsymbol{v}_{S}, \pi_{S} ; z\right)$
1: $z=H\left(c_{\boldsymbol{v}}, S, \boldsymbol{v}_{S}\right)$
2: $e\left(c_{\boldsymbol{v}}, \widetilde{g}_{n+1-i}^{\sum_{i \in S} z^{i-1}}\right) \stackrel{?}{=} e\left(\pi_{S}, \widetilde{g}\right) \cdot e\left(g^{\sum_{i \in S} z^{i-1} \cdot v_{i}}, \widetilde{g}_{n}\right)$
Fig. 10: Construction of Pedersen Subvector Commitment.

Polynomial Binding. For all PPT adversary $\mathcal{A}$,

$$
\operatorname{Pr}\left[\begin{array}{l}
\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, \hat{d}) \\
\left(f, f^{\prime}\right) \leftarrow \mathcal{A}(\mathrm{srs}): \\
\quad \operatorname{Commit}(\mathrm{srs}, f)=\operatorname{Commit}\left(\mathrm{srs}, f^{\prime}\right)
\end{array}\right]=\mathrm{negl}
$$

Evaluation Binding. For all PPT adversary $\mathcal{A}$,

$$
\operatorname{Pr}\left[\begin{array}{l}
\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, \hat{d}), \\
\left(\boldsymbol{v}, c_{f}, e_{\boldsymbol{v}}, \pi, e_{\boldsymbol{v}}^{\prime}, \pi^{\prime}\right) \leftarrow \mathcal{A}(\text { srs }): \\
\operatorname{Eval}\left(\operatorname{srs}, \boldsymbol{v}, c_{f}, e_{\boldsymbol{v}}, \pi\right)=1 \\
\wedge \\
\operatorname{Eval}\left(\operatorname{srs}, \boldsymbol{v}, c_{f}, e_{\boldsymbol{v}}^{\prime}, \pi^{\prime}\right)=1
\end{array}\right]=\mathrm{negl}
$$

Knowledge Soundness. For all PPT adversary $\mathcal{A}$, there exists an extractor $\mathcal{E}$ such that

$$
\operatorname{Pr}\left[\begin{array}{lr}
\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, \hat{d}), & \\
\left(\boldsymbol{v}, c_{f}, e_{\boldsymbol{v}}, \pi\right) \leftarrow \mathcal{A}(\text { srs }), & \\
f \leftarrow \mathcal{E} \text { (srs) }: & \operatorname{Eval}\left(\text { srs }, \boldsymbol{v}, c_{f}, e_{\boldsymbol{v}}, \pi\right)=1 \\
& \quad e_{\boldsymbol{v}} \neq f(\boldsymbol{v})
\end{array}\right]=\mathrm{negl}
$$

Remark. If a polynomial commitment scheme is knowledge sound in AGM, it is also evaluation binding. Let $\mathcal{A}_{\mathrm{KS}}$ and $\mathcal{A}_{\mathrm{EB}}$ denote the adversary of the knowledge soundness game and evaluation binding game, respectively. $\mathcal{A}_{\text {KS }}$ can selectively output whatever $\mathcal{A}_{\text {EB }}$ outputs, and wins the knowledge soundness game with the same probability as that of $\mathcal{A}_{\mathrm{EB}}$. 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 $\mathcal{A}$,

$$
\operatorname{Pr}\left[\begin{array}{ll}
\text { srs } \leftarrow \operatorname{Setup}(\lambda, m, \hat{d}), & \\
\left(f_{0}, f_{1}\right) \leftarrow \mathcal{A}(\text { srs }), & \\
b \stackrel{R}{\leftarrow}\{0,1\}, & \\
c_{f_{b}} \leftarrow \operatorname{Commit}\left(\text { srs, } f_{b}\right), & \\
b^{\prime} \leftarrow \mathcal{A}\left(\text { srs }, c_{f_{b}}\right): & b \neq b^{\prime}
\end{array}\right]=\mathrm{negl}+\frac{1}{2}
$$

Evaluation Binding (Batch). A polynomial commitment scheme in the batched setting is evaluation binding if for all PPT adversary $\mathcal{A}$,

$$
\operatorname{Pr}\left[\begin{array}{l}
\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, \hat{d}), \\
\left(\left(c_{f_{i}}, e_{\boldsymbol{v}}^{[i]}, e_{\boldsymbol{v}}^{[i]^{\prime}}\right)_{i \in[1, n]}, \boldsymbol{f}, \boldsymbol{v}, \pi, \pi^{\prime}\right) \leftarrow \mathcal{A}(\mathrm{srs}): \\
\operatorname{BatchEval}\left(\operatorname{srs}, \boldsymbol{v},\left(c_{f_{i}}, e_{\boldsymbol{v}}^{[i]}\right)_{i \in[1, n]}, \pi\right)=1 \\
\wedge \operatorname{BatchEval}\left(\operatorname{srs}, \boldsymbol{v},\left(c_{f_{i}}, e_{\boldsymbol{v}}^{[i]^{\prime}}\right)_{i \in[1, n]}, \pi^{\prime}\right)=1
\end{array}\right]=\operatorname{negl}
$$

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

$$
\operatorname{Pr}\left[\begin{array}{l}
\operatorname{srs} \leftarrow \operatorname{Setup}(\lambda, m, \hat{d}), \\
\left(\boldsymbol{v},\left(c_{f_{i}}, e_{\boldsymbol{v}}^{[i]}\right)_{i \in[1, n]}, \pi\right) \leftarrow \mathcal{A}(\text { srs }), \\
\boldsymbol{f} \leftarrow \mathcal{E}(\text { srs }): \\
\left.\quad \text { BatchEval(srs, } \boldsymbol{v},\left(c_{f_{i}}, e_{\boldsymbol{v}}^{[i]}\right)_{i \in[1, n]}, \pi\right)=1 \\
\wedge \quad \\
\quad\left(e_{\boldsymbol{v}}^{[i]}\right)_{i \in[1, n]} \neq \boldsymbol{f}(\boldsymbol{v})
\end{array}\right]=\mathrm{negl}
$$

## 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 $(q, 1)$-discrte logarithm assumption, on input a discrete logarithm problem instance $\left(g, g^{x}, \ldots, g^{x^{q}}, \widetilde{g}, \widetilde{g}^{x}\right)$, $\mathcal{A}^{*}$ computes
$-\forall j \in[1, m], a_{j}, b_{j} \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$, define $\beta_{j}=a_{j} x+b_{j}$,

- Compute $\boldsymbol{g}=\left(g^{\prod_{j=1}^{m} \beta_{j}^{d_{j}}}\right)_{d_{1} \in\left[0, \hat{d_{1}}\right], \ldots, d_{m} \in\left[0, \hat{d_{m}}\right]}$,
- Set srs $=\left(\boldsymbol{g}, \tilde{g},\left(\widetilde{g}^{\beta_{i}}\right)_{i \in[1, m]}\right)$
and sends srs to $\mathcal{A}$.
Set $\boldsymbol{\beta}:=\left(a_{j} x+b_{j}\right)_{j \in[1, m]}$. The difference of $f(\boldsymbol{\beta})$ and $f^{\prime}(\boldsymbol{\beta})$ yields a univariate polynomial $f^{\triangle}(X)$ evaluated at $x$, i.e., $f^{\triangle}(x)=f(\boldsymbol{\beta})-f^{\prime}(\boldsymbol{\beta})$. Finally, $\mathcal{A}^{*}$ solves $f^{\triangle}(X)=0$ 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 $\left(g, g^{x}, \ldots, g^{x^{q}}, \widetilde{g}, \widetilde{g}^{x}\right), \mathcal{E}$ initiates the knowledge soundness game with srs same as the polynomial binding game above.

Upon receiving srs, $\mathcal{A}$ outputs $\boldsymbol{v},\left(c_{f_{i}}, e_{\boldsymbol{v}}^{[i]}\right)_{i \in[1, n]}$ and $\pi$, along with $\operatorname{REP}_{\text {srs }}^{\left(c_{f_{i}}\right)_{i \in[1, n]}}$ and $\operatorname{REP}_{\text {srs }}^{\pi}$. With these, $\mathcal{E}$ can output polynomials $\boldsymbol{f}=\left(f_{1}, \ldots, f_{n}\right)$ and $\left(\hat{q}_{j}(\beta)\right)_{j \in[1, m]}$.

Assume that $\exists i \in[1, n], f_{i}(\boldsymbol{v}) \neq e_{\boldsymbol{v}}^{[i]}$. Let $\hat{f}(x)=\left\langle\boldsymbol{f}, \boldsymbol{r}^{n}\right\rangle$ and $e_{\hat{v}}=\left\langle\boldsymbol{\varphi}, \boldsymbol{r}^{n}\right\rangle$ where $r=H\left(\boldsymbol{v},\left(c_{f_{i}}\right)_{i \in[1, n]}\right)$ The ideal checking is in the form of $\hat{f}(\boldsymbol{\beta})-e_{\boldsymbol{v}}=$ $\sum_{j=1}^{m} \hat{q}_{j}(\boldsymbol{\beta})\left(\beta_{j}-v_{j}\right)$, 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 $\widetilde{h}$ accordingly.

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

$$
\begin{aligned}
& -c=\widetilde{g}_{n}^{\sum_{i \in S} z^{\sigma(i)+1} v_{i}} \cdot \widetilde{g}^{\gamma} \\
& -c^{\prime}=\prod_{i \in S} \prod_{j=1, j \neq i}^{n^{\prime}} \widetilde{g}_{n+1-i+j}^{\sigma^{\sigma(i)+1} \cdot v_{j}} \prod_{i \in S} g_{n+1-i}^{\sum_{i \in S} z^{\sigma(i)+1} \cdot w} / g_{1}^{\gamma} \\
& -e\left(c_{\boldsymbol{v}}, \sum_{i \in S} \widetilde{g}_{n+1-i}^{z^{\sigma(i)+1}}\right) \stackrel{?}{=} e\left(c^{\prime}, \widetilde{g}\right) \cdot e\left(g_{1}, c\right) .
\end{aligned}
$$

## E Security Proof for zero-knowledge MMP

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

$$
\begin{aligned}
e\left(c_{\boldsymbol{v}}, \prod_{i=1}^{n} g_{n+1-i}^{r^{i-1}}\right) & =e\left(\prod_{i=1}^{n} g_{i}^{f_{i}(\boldsymbol{v})} \cdot h^{\varrho^{\prime}}, \prod_{i=1}^{n} \widetilde{g}_{n+1-i}^{r^{i-1}}\right) \\
& =e\left(\prod_{i=1}^{n} \prod_{j=1, j \neq i}^{n} g_{n+1-i+j}^{f_{j}\left(\boldsymbol{v} \cdot r^{i-1}\right.} \cdot \prod_{i=1}^{n} h_{n+1-i}^{\varrho^{\prime} \cdot r^{i-1}}, \widetilde{g}\right) \cdot e\left(g_{n+1}^{\sum_{i=1}^{n} f_{i}(\boldsymbol{v}) \cdot r^{i-1}}, \widetilde{g}\right) \\
& =e\left(\prod_{i=1}^{n} \prod_{j=1, j \neq i}^{n} g_{n+1-i+j}^{f_{j}\left(\boldsymbol{v} \cdot r^{i-1}\right.} \cdot \prod_{i=1}^{n} h_{n+1-i}^{\rho^{\prime} \cdot r^{i-1}} \cdot h_{n}^{-\varrho}, \widetilde{g}\right) \cdot e\left(g^{\sum_{i=1}^{n} f_{i}(\boldsymbol{v}) \cdot r^{i-1}} \cdot h^{\varrho}, \widetilde{g}_{n}\right) \\
& =e\left(c^{\prime}, \widetilde{g}\right) e\left(c_{\hat{v}}, \widetilde{g}\right) \\
g^{b_{1}} h^{b_{2}} c_{\hat{v}}^{\eta} & =g^{\varrho_{1}-\eta \cdot \sum_{i=1}^{n} f_{i}(\boldsymbol{v}) \cdot r^{i-1}} h^{\varrho_{2}-\eta \cdot \varrho} \cdot\left(g^{\sum_{i=1}^{n} f_{i}(\boldsymbol{v}) \cdot r^{i-1}} h^{\rho}\right)^{\eta} \\
& =g^{\varrho_{1}} \cdot h^{\varrho_{2}} \\
& =D \\
e\left(c_{\tilde{f}} / c_{\hat{v}}, \widetilde{g}\right) & =e\left(g^{\sum_{i=1}^{n}\left(f_{i}(\boldsymbol{\beta})-f_{i}(\boldsymbol{v})\right) \cdot r^{i-1}} h^{\sum_{i=1}^{n} \rho_{i} \cdot r^{i-1}-\varrho}, \widetilde{g}\right) \\
& =\prod_{j=1}^{m} e\left(g^{q_{j}(\boldsymbol{\beta})}, \widetilde{g}^{\beta_{j}-v_{j}}\right) \cdot e\left(g^{\sum_{i=1}^{n} \rho_{i} \cdot r^{i-1}-\varrho}, \widetilde{h}\right) \\
& =\prod_{j=1}^{m} e\left(g^{q_{j}(\boldsymbol{\beta})} \cdot h^{\mu_{j}}, \widetilde{g}^{\beta_{j}-v_{j}}\right) \cdot e\left(g^{\sum_{i=1}^{n} \rho_{i} \cdot r^{i-1}+\sum_{j=1}^{m}\left(v_{j}-\beta_{j}\right) \cdot \mu_{j}-\varrho}, \widetilde{h}\right) \\
& =\prod_{j=1}^{m} e\left(Q_{j}, \widetilde{g}^{\beta_{j}-v_{j}}\right) \cdot e(\theta, \widetilde{h})
\end{aligned}
$$

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 $e_{\boldsymbol{v}}^{[i]}$ for $i \in$ $[1, n]$ corresponding to $c_{\boldsymbol{v}}$ 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 $\forall i \in[1, n], f_{i}(\boldsymbol{v})=$ $e_{\boldsymbol{v}}^{[i]}$.
$S H V Z K$. A zero-knowledge simulator uses the trapdoor of srs, i.e., $\alpha,(\beta)_{i \in[1, m]}, \eta$, and simulates the protocols as below:
- Choose $\boldsymbol{\phi} \stackrel{R}{\leftarrow} \mathbb{G}^{n}$ and $r \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$ and compute $c_{\boldsymbol{f}}=\left\langle\boldsymbol{\phi}, \boldsymbol{g}_{\mathrm{pst}}\right\rangle$ and $c_{\hat{f}}=\left\langle\boldsymbol{\phi}, \boldsymbol{r}^{n}\right\rangle$, and runs the GIPA protocol on the simulated $c_{\boldsymbol{f}}$ and $c_{\hat{f}}$.
- Choose $c_{\boldsymbol{v}} \stackrel{R}{\leftarrow} \mathbb{G}$ and $e_{\hat{v}}, \varrho \stackrel{R}{\leftarrow} \mathbb{Z}_{p}$.
- Compute $c^{\prime}=c_{\boldsymbol{v}}^{\sum_{i=1}^{n} \alpha^{n+1-i} \cdot r^{i-1}} /\left(g_{n+1}^{e_{\hat{v}}} h_{n}^{\varrho}\right)$ 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 $\theta=c_{\hat{f}} /\left(c_{\hat{v}} \cdot \prod_{j=1}^{m} Q_{j}^{\left(\beta_{j}-v_{j}\right)^{-1}}\right)$.


[^0]:    ${ }^{3}$ 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 $\log |\mathbb{F}|=\omega(\log N)$.

[^1]:    ${ }^{4}$ 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 $E_{0} E_{1} \cdot E_{0} E_{1}<$ $E_{0} P \cdot E_{0} E_{1}<0$. However, $E_{0} E_{1} \cdot E_{0} E_{1}=\left(E_{0} E_{1}\right)^{2} \geq 0$, leading to a contradiction.

