A Succinct Range Proof for Polynomial-based Vector Commitment

Rui Gao
Jiangsu Cryptographic Technology Engineering Research Center, Nanjing University of Posts and Telecommunications; Zhejiang Lab
Nanjing, China
2021040405@njupt.edu.cn

Zhiguo Wan*
Zhejiang Lab
Hangzhou, China
wanzhiguo@zhejianglab.com

Yuncong Hu*
Department of Computer Science and Engineering, Shanghai Jiao Tong University
Shanghai, China
huyuncong@sjtu.edu.cn

Huaqun Wang*
Jiangsu Cryptographic Technology Engineering Research Center, Nanjing University of Posts and Telecommunications
Nanjing, China
whq@njupt.edu.cn

ABSTRACT

A range proof serves as a protocol for the prover to prove to the verifier that a committed number lies in a specified range, such as $[0, 2^n)$, without disclosing the actual value. Range proofs find extensive application in various domains. However, the efficiency of many existing schemes diminishes significantly when confronted with batch proofs encompassing multiple elements.

To improve the scalability and efficiency, we propose MissileProof, a vector range proof scheme, proving that every element in the committed vector is within $[0, 2^n)$. We first reduce this argument to a bi-to-univariate SumCheck problem and a bivariate polynomial ZeroTest problem. Then generalizing the idea of univariate SumCheck PIOP, we design a bi-to-univariate SumCheck PIOP. By introducing a random polynomial, we construct the bivariate polynomial ZeroTest using a univariate polynomial ZeroTest and a univariate polynomial SumCheck PIOP. Finally, combining the PIOP for vector range proof, a KZG-based polynomial commitment scheme and the Fiat-Shamir transformation, we get a zero-knowledge succinct non-interactive vector range proof.

Compared with existing schemes, our scheme has the optimal proof size ($O(1)$), the optimal commitment length ($O(1)$), and the optimal verification time ($O(1)$), at the expense of slightly sacrificing proof time ($O(l \log l \cdot n \log n)$ operations on the prime field for FFT and $O(ln)$ group exponentiations in $\mathbb{G}$). Moreover, we implemented an anti-money-laundering stateless blockchain based on the MissileProof. The gas consumption of the verification smart contract is reduced by 85%.

CCS CONCEPTS

• **Security and privacy → Information-theoretic techniques**.

KEYWORDS

Range proofs, zero knowledge, polynomial interactive oracle proofs, succinct arguments, sumcheck protocol

ACM Reference Format:
Rui Gao, Zhiguo Wan*, Yuncong Hu*, and Huaqun Wang*. 2024. A Succinct Range Proof for Polynomial-based Vector Commitment. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3670324

1 INTRODUCTION

A zero-knowledge proof (ZKP) is a protocol that allows the prover to convince the verifier, that he knows a secret witness satisfying the certain relation without revealing the witness itself. A range proof is a type of ZKPs that allows a prover to convince a verifier that for a commitment C, he knows the committed value v, and v is in a certain range $[0, 2^n)$. Range proofs serve as the core building block in numerous applications, such as anonymous credentials [1], e-voting [2], e-cash [3], electronic auctions [4][5][6], and cryptocurrencies Monero[1], Beam[2], Grin[3]. In decentralized anonymous payment system such as Monero and ZCash [7], range proofs are used to prove that the sender's balance is greater than the transferred amount.

The existing range proof schemes mostly focus on proving the commitment to a single element, which hampers scalability. Therefore, we introduce MissileProof, a range proof for a vector commitments to improve the efficiency of range proofs for large-scale data.

[1]https://web.getmonero.org/resources/moneropedia/BulletProofs.html
[2]https://github.com/BeamMW/beam
[3]https://grin.mw

* Corresponding authors.

This work was supported in part by the National Natural Science Foundation of China under Grants (Nos. U23B2002, 62272238, 62272425), the Key Research Project of Zhejiang Laboratory under Grant 2022PD0AC01, and the Major Basic Research Program of the Shandong Provincial Natural Science Foundation under Grant ZR2020ZD01.

Vector commitment schemes have been used in many scenarios such as stateless cryptocurrency [8] and account-based blockchain [7]. For example, in the stateless blockchain, the miners maintains a vector commitment to all users' balances and do not need to store all blocks of the entire chain. They can quickly verify the transactions solely relying on the balance vector commitment. It is of great practical value to design a range proof for vector commitment.

There are three most important properties of zero-knowledge proofs: the proof size, prover complexity and verifier complexity. Range proofs are widely applied in the blockchain, where the storage resources of blockchain and the computational resources of smart contracts are both highly expensive. In contrast, the generation process of zero-knowledge proofs can be performed off-chain at a relatively low cost. Therefore, we designed a vector range proof scheme, MissileProof with the **smallest proof size, shortest verification time and acceptable proving time**. As an application demonstration, we designed an anti-money-laundering regulatable stateless blockchain system based on MissileProof.

Problem. In a formalized form, common range proof schemes prove the problem instance statement $\mathbb{x}$ (a commitment C) and the witness $\mathbb{w}$ (the secret value v) has the following relation: ($\mathbb{G}$ is the field of the commitment, $\mathbb{F}$ is a prime field.)

$$\{(C \in \mathbb{G}; v \in \mathbb{F}) : C = \mathsf{Commit}(v) \wedge v \in [0, 2^n)\}$$

When the prover needs to perform range proofs on a vector $\mathbf{v}$ of l elements, i.e. to prove the following relation:

$$\{(\mathrm{C} \in \mathbb{G}^l; \mathbf{v} \in \mathbb{F}^l) : \forall i \in [0, l), C_i = \mathsf{Commit}(v_i) \wedge v_i \in [0, 2^n)\}$$

Though some works [9][10] introduce batch proofs and batch verification schemes to reduce the proof size and the verification time, they cannot decrease the commitment length since the original commitments $\mathrm{C} \in \mathbb{G}^l$ of length l has to be sent to the verifier, which results in great communication pressure.

Essentially, we argue that the core obstacle of the scalability is that these schemes only deal with the commitment scheme to a single element and the prover has to transfer a set of commitments to the elements as the statement. Moreover, the proof size, proving time and verification time are also unsatisfactory.

Therefore, we propose a new vector range proof scheme, MissileProof, proving that every element in a committed vector $\mathbf{v}$ of length l lies in a range $[0, 2^n)$. Informally, it proves the relation $\mathcal{R}_{\mathrm{VCRP}}$ as follows (VC refers to a vector commitment scheme):

$$\{(C \in \mathbb{G}; \mathbf{v} \in \mathbb{F}^l) : C = \mathsf{VC.Commit}(\mathbf{v}) \wedge \forall i \in [0, l), v_i \in [0, 2^n)\}$$

As shown in Table 1, compared with existing schemes, our scheme has the optimal proof size ($O(1)$), the optimal commitment length ($O(1)$), and the optimal verification time ($O(1)$), at the expense of slightly sacrificing proof time ($O(l \log l \cdot n \log n)$ operations on the prime field for FFT and $O(ln)$ group exponentiations in $\mathbb{G}$).

1.1 Our approach and results

In this section we give a high-level overview about how MissileProof presents a non-interactive argument of knowledge for vector range proof as shown in Fig 1.

First we introduce a reduction for the vector range proof. Because the binary decomposition of every integer is unique, then if for a vector $\mathbf{v} = \{v_0, ..., v_{l-1}\} \in \mathbb{F}^l$, every element $v_i \in \mathbf{v}, i \in [0, l)$ is in a range $[0, 2^n)$, then the following condition holds:

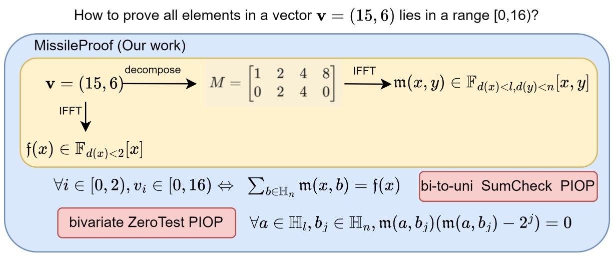

How to prove all elements in a vector $\mathbf{v} = (15, 6)$ lies in a range [0,16)?
MissileProof (Our work)
$\mathbf{v} = (15, 6)$
decompose
$M = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 0 & 2 & 4 & 0 \end{bmatrix}$
IFFT
$\mathfrak{m}(x, y) \in \mathbb{F}_{d(x)<l, d(y)<n}[x, y]$
IFFT
$\mathfrak{f}(x) \in \mathbb{F}_{d(x)<2}[x]$
$\forall i \in [0, 2), v_i \in [0, 16) \Leftrightarrow \sum_{b \in \mathbb{H}_n} \mathfrak{m}(x, b) = \mathfrak{f}(x)$
bi-to-uni SumCheck PIOP
bivariate ZeroTest PIOP
$\forall a \in \mathbb{H}_l, b_j \in \mathbb{H}_n, \mathfrak{m}(a, b_j)(\mathfrak{m}(a, b_j) - 2^j) = 0$

Figure 1: The intuition of our approach. The yellow block shows the reduction process. The blue block shows the new equivalent relations after reduction. The red block shows the core techniques that used to prove the relations given in the blue blocks.

$$\exists M \in \mathbb{F}^{l \times n}, v_i = \sum_{j=0}^{n-1} M_{i,j} \wedge M_{i,j} \in \{0, 2^j\}$$

One can obtain the matrix M using binary decomposition. Conversely, the condition above ensures that the largest possible value in the vector $\mathbf{v}$ is $\sum_{j=0}^{n-1} \cdot 2^j = 2^n - 1$.

Take $l = 2, n = 4$ and $\mathbf{v} = (15, 6)$ as an example: there exists a unique matrix $M = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 0 & 2 & 4 & 0 \end{bmatrix}$ such that the condition holds.

Let $\mathbb{H}_l$ and $\mathbb{H}_n$ be two multiplicative subgroups of order l and n over the field $\mathbb{F}$ and let $\mathfrak{f}(X) \in \mathbb{F}_{d(X)<l}[X]$ be a polynomial of degree at most $l-1$ over $\mathbb{F}$ that extends $\mathbf{v}$ in the sense that for all $a_i \in \mathbb{H}_l, \mathfrak{f}(a_i) = v_i$. Similarly, we can extend the decomposition matrix M to a bivariate polynomial $\mathfrak{m}(X, Y) \in \mathbb{F}_{d(X)<l, d(Y)<n}[X, Y]$, s.t. for all $a_i \in \mathbb{H}_l$ and $b_j \in \mathbb{H}_n, \mathfrak{m}(a_i, b_j) = M_{i,j}$. Considering the relationship between the vector $\mathbf{v}$ and matrix M, the polynomials $\mathfrak{f}$ and $\mathfrak{m}$ should satisfy the following conditions:

- *Condition 1.* Bi-to-univariate polynomial SumCheck: $\sum_{b \in \mathbb{H}_n} \mathfrak{m}(X, b) = \mathfrak{f}(X)$.
- *Condition 2.* Bivariate polynomial ZeroTest: $\forall a \in \mathbb{H}_l, b_j \in \mathbb{H}_n, \mathfrak{m}(a, b_j)(\mathfrak{m}(a, b_j) - 2^j) = 0$.

These two conditions cannot be succinctly verified since they involves operations like "$\forall$" and "$\sum$". So we need to reduce these relations to new relations without "$\forall$" or "$\sum$". Moreover, both conditions involve bivariate polynomials, which makes proof even more difficult. Our core theoretical contribution lies in designing proofs for the two conditions.

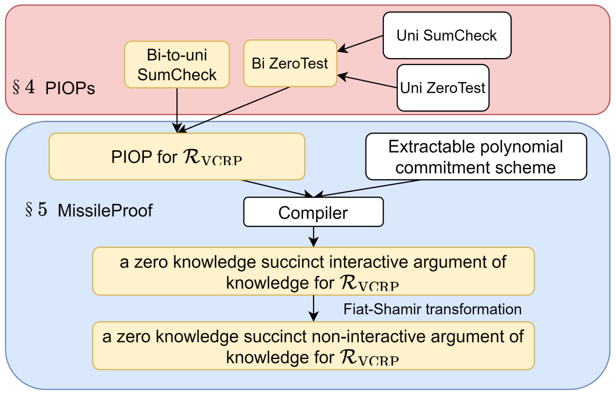
To efficiently prove the bi-to-univariate polynomial SumCheck (condition 1), we propose Lemma 4.1, that for any bivariate polynomial $\mathfrak{m}(X, Y) \in \mathbb{F}_{d(X) \leq l-1, d(Y) \leq n-1}[X, Y]$, $\sum_{b \in \mathbb{H}_n} \mathfrak{m}(X, b) = \mathfrak{f}(X)$ if and only if there exists a bivariate polynomial $\mathfrak{u}(X, Y) \in \mathbb{F}_{d(X)<l, d(Y)<n-1}[X, Y]$, such that $\mathfrak{m}(X, Y) = \frac{\mathfrak{f}(X)}{n} + Y \cdot \mathfrak{u}(X, Y)$. Via this lemma, the verifier can efficiently check the condition 1 by checking the equality at a random point using Schwartz-Zippel lemma 3.1.

For condition 2, the bivariate polynomial ZeroTest, we introduce a bivariate random polynomial $\mathfrak{r}(R, Y)$ and reduce condition 2 to a combination of a univariate polynomial SumCheck relation and a univariate polynomial ZeroTest relation.

Let $\mathfrak{p}(Y)$ be a polynomial such that $\forall b_j \in \mathbb{H}_n, \mathfrak{p}(b_j) = 2^j$. Let $\mathfrak{M}(X, b) = \mathfrak{m}(X, b)(\mathfrak{m}(X, b) - \mathfrak{p}(b))$. Let $\mathfrak{r}(R, Y)$ be a bivariate random polynomial, which satisfies that for all $b \in \mathbb{H}_n, \mathfrak{r}(R, b)$ are n linearly independent polynomials. Then we have:

$$\forall a \in \mathbb{H}_l, b \in \mathbb{H}_n, \mathfrak{M}(a, b) = 0 \Leftrightarrow$$
$$\forall a \in \mathbb{H}_l, \mathfrak{M}^*(R, a) = \sum_{b \in \mathbb{H}_n} \mathfrak{M}(a, b)\mathfrak{r}(R, b) = 0$$

Completeness of the second check is straightforward. Soundness follows from the fact that if any evaluation of $\mathfrak{M}$ does not equal 0, the combined polynomial will not equal 0 over $\mathbb{H}_l$ with high probability. Then the proof for the condition 2 can be reduced to two phases: a univariate polynomial ZeroTest and a univariate polynomial SumCheck. Concretely, $\mathcal{V}$ randomly selects $\tau_r, \tau_x \xleftarrow{\$} \mathbb{F}^2$, and sends τ_r, τ_x to $\mathcal{P}$. In the phase 1, $\mathcal{P}$ then proves that $\forall a \in \mathbb{H}_l, \mathfrak{M}^*(\tau_r, a) = 0$. In the phase 2, $\mathcal{P}$ proves that $\mathfrak{M}^*(\tau_r, \tau_x) = \sum_{b \in \mathbb{H}_n} \mathfrak{M}(\tau_x, b)\mathfrak{r}(\tau_r, b)$ using the univariate polynomial SumCheck protocol.



§ 4 PIOPs
Bi-to-uni SumCheck
Bi ZeroTest
Uni SumCheck
Uni ZeroTest
PIOP for $\mathcal{R}_{\text{VCRP}}$
Extractable polynomial commitment scheme
§ 5 MissileProof
Compiler
a zero knowledge succinct interactive argument of knowledge for $\mathcal{R}_{\text{VCRP}}$
Fiat-Shamir transformation
a zero knowledge succinct non-interactive argument of knowledge for $\mathcal{R}_{\text{VCRP}}$

Figure 2: Our methodology for constructing a zero-knowledge succinct non-interactive argument of knowledge for vector range proof. Things in the yellow blocks are our main contribution and core innovation. Things in the white blocks are implemented using the existing techniques.

As shown in Fig 2, following the intuition given above, we construct four PIOPs in section 4. In section 5, we reduce the vector range relation to a bi-to-univariate polynomial SumCheck problem and a bivariate polynomial ZeroTest problem and propose a PIOP for it. Then, we compile the public-coin PIOP with a KZG-based polynomial commitment scheme and use Fiat-Shamir transformation to get a succinct non-interactive argument of knowledge for vector range proof.

1.2 Our contributions

- **MissileProof**. This paper presents MissileProof, a zero-knowledge succinct non-interactive argument of knowledge for range proof of a polynomial-based vector commitment. Our scheme has the optimal proof size ($O(1)$), the optimal commitment length ($O(1)$), and the optimal verification time ($O(1)$), at the expense of slightly sacrificing proof time.
- **PIOP tools**. As the core building block, we propose a new bi-to-univariate SumCheck PIOP and a bivariate ZeroTest PIOP which can be seen as an independent interest. Then we construct a PIOP for vector range proof based on them.
- **Experimental efficiency**. Experiments show that for proving each element of a secret vector of 16384 elements lies in a range $[0, 2^{64})$, MissileProof costs 0.03125 Kb for commitment length, 0.8125 Kb for proof size, 421 s for prover computation, 0.0087 s for verification.
- **Application demonstration**. We designed an anti-money-laundering regulatable stateless blockchain based on MissileProof. Administrators can use smart contracts to verify whether all users on the entire chain have overspending behavior and do the basic audit of on-chain transactions. Compared to the state-of-the-art TurboPlonk, we reduced gas costs by 85%.

Paper outline. Section 2 introduces some related works. Section 3 presents the notations and related knowledge used in this paper. Section 4 introduces four PIOP components. Section 5 presents a PIOP for $\mathcal{R}_{\text{VCRP}}$ and construct a non-interactive succinct argument of knowledge for it. Section 6 discusses some more general application scenarios of our work. Section 7 demonstrates the efficiency of MissileProof through experiments and gives an application. Finally, Section 8 concludes our work.

2 RELATED WORK

Range proofs for single element. For range proofs, there are two high-level approaches for construction: square decomposition and k-ary decomposition. The square decomposition range proof was first proposed by of Boudot et al. [11]. It reduces the original statement that a secret value v is in range $[a, b]$ to two sub-statement: $v - a$ and $b - v$ are non-negative. This work employs a fact that any positive integer can be decomposed into four squares and construct a constant size proof. The advantage of this scheme is that the proof time and verification time are constant, regardless of the size of the range. However, although there have been many subsequent works CLKR [12], Sharp [13] to optimize it, this approach still has the following shortcomings: 1. it requires RSA groups or class groups with a discriminant that is difficult to factorize, resulting in very large element sizes and poor performance in practical applications. Sharp implements it in an elliptic curve at the expense of additional overhead. 2. this solution is difficult to prove in batches.

Another approach is k-ary decomposition that is more widely used and has better efficiency. To prove that a secret value v lies in range $[0, k^n)$, it is equivalent to prove that there exists a k-ary decomposition vector $(v_0, ..., v_{n-1})$ such that each bit $v_i \in [0, k)$ and $v = \sum_{j=0}^{n-1} v_i \cdot k^i$. Based on binary decomposition and inner product argument (IPA), Bünz et al. [10] proposed their schemes for proving the secret value in a Pedersen commitment lies in a range $[0, 2^n)$. Though the proof size is $O(\log n)$, its verification time is $O(n)$. Daza et al. [9] introduced the structured reference string to reduce the verification complexity of IPA, such that their scheme has both $O(\log n)$ communication and verification complexities. Although there have been subsequent works [14] to further accelerate the range proof by improving the IPA protocol, the asymptotic complexities has not changed essentially.

Table 1: Comparison among different range proof schemes for multiple elements.

Protocols	Commitment length	Proof size	Prover complexity	Verifier complexity	Setup
Sharp [13]	$O(1)\mathbb{G}$	$O(l)\mathbb{G} + O(l)\mathbb{F}$	$O(l)E + O(l)M$	$O(l)E + O(l)M$	transparent
BulletProof [10]	$O(l)\mathbb{G}$	$O(\log l + \log n)\mathbb{G}+$ $O(1)\mathbb{F}$	$O(ln)E+$ $O(ln)M$	$O(ln)E+$ $O(ln)M$	transparent
Daza et al. [9]	$O(l)\mathbb{G}_1$	$O(\log l + \log n)\mathbb{G}_1+$ $O(\log l + \log n)\mathbb{F}$	$O(ln)E_1+$ $O(ln)M$	$O(l + \log n)E_1 + O(\log n)E_2+$ $O(\log n)M + O(\log n)P$	private
TurboPlonk [15]	$O(1)\mathbb{G}_1$	$O(n)\mathbb{G}_1+$ $O(n)\mathbb{F}$	$O(ln)E_1+$ $O(ln \log l)M$	$O(n)E_1 + O(1)E_2+$ $O(n)M + O(1)P$	private
Plonkup [16]	$O(1)\mathbb{G}_1$	$O(1)\mathbb{G}_1+$ $O(1)\mathbb{F}$	$O(l + 2^n)E_1+$ $O((l + 2^n) \cdot 2^n \log(l + 2^n))M$	$O(1)E_1 + O(1)E_2+$ $O(1)M + O(1)P$	private
This work	$O(1)\mathbb{G}_1$	$O(1)\mathbb{G}_1+$ $O(1)\mathbb{F}$	$O(ln)E_1+$ $O(l \log l \cdot n \log n)M$	$O(1)E_1 + O(1)E_2+$ $O(1)M + O(1)P$	private

Notes: l is the vector length and n is the max bit length of elements ($v \in [0, 2^n)$). $\mathbb{G}$ is a cyclic group element. ($\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$) is a bilinear group elements. and $\mathbb{F}$ means prime field elements. E means group exponentiations in $\mathbb{G}$; E_1 means group exponentiations in $\mathbb{G}_1$; E_2 means group exponentiations in $\mathbb{G}_2$; M means field multiplications; P means pairings; transparent means no trusted setup; private means the setup is generated by a trusted institution, moreover, the structured reference string of the setup algorithm is universally updatable. Concrete cost is shown in Table 3.

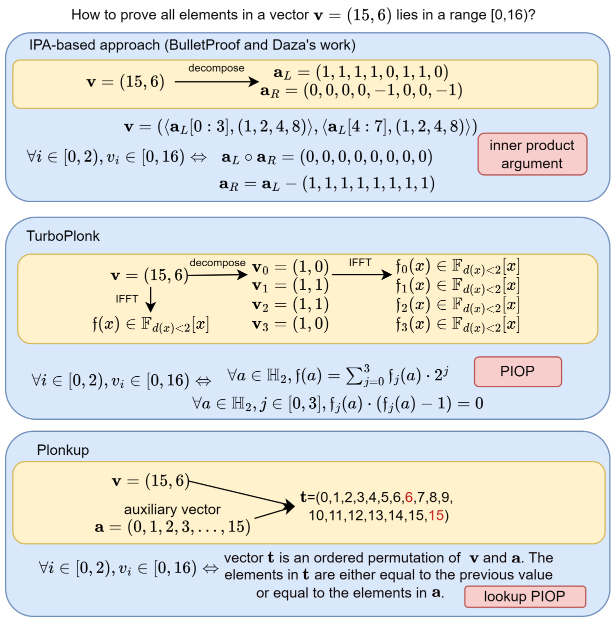

How to prove all elements in a vector $\mathbf{v} = (15, 6)$ lies in a range [0,16)?
IPA-based approach (BulletProof and Daza's work)
$\mathbf{v} = (15, 6)$
decompose
$\mathbf{a}_L = (1, 1, 1, 1, 0, 1, 1, 0)$
$\mathbf{a}_R = (0, 0, 0, 0, -1, 0, 0, -1)$
$\mathbf{v} = (\langle \mathbf{a}_L[0:3], (1, 2, 4, 8)\rangle, \langle \mathbf{a}_L[4:7], (1, 2, 4, 8)\rangle)$
$\forall i \in [0, 2), v_i \in [0, 16) \Leftrightarrow$ $\mathbf{a}_L \circ \mathbf{a}_R = (0, 0, 0, 0, 0, 0, 0, 0)$
$\mathbf{a}_R = \mathbf{a}_L - (1, 1, 1, 1, 1, 1, 1, 1)$
inner product argument
TurboPlonk
$\mathbf{v} = (15, 6)$
decompose
$\mathbf{v}_0 = (1, 0)$
$\mathbf{v}_1 = (1, 1)$
$\mathbf{v}_2 = (1, 1)$
$\mathbf{v}_3 = (1, 0)$
IFFT
$\mathfrak{f}_0(x) \in \mathbb{F}_{d(x)<2}[x]$
$\mathfrak{f}_1(x) \in \mathbb{F}_{d(x)<2}[x]$
$\mathfrak{f}_2(x) \in \mathbb{F}_{d(x)<2}[x]$
$\mathfrak{f}_3(x) \in \mathbb{F}_{d(x)<2}[x]$
IFFT
$\mathfrak{f}(x) \in \mathbb{F}_{d(x)<2}[x]$
$\forall i \in [0, 2), v_i \in [0, 16) \Leftrightarrow$ $\forall a \in \mathbb{H}_2, \mathfrak{f}(a) = \sum_{j=0}^{3} \mathfrak{f}_j(a) \cdot 2^j$
$\forall a \in \mathbb{H}_2, j \in [0, 3], \mathfrak{f}_j(a) \cdot (\mathfrak{f}_j(a) - 1) = 0$
PIOP
Plonkup
$\mathbf{v} = (15, 6)$
auxiliary vector
$\mathbf{a} = (0, 1, 2, 3, \ldots, 15)$
$\mathbf{t}$=(0,1,2,3,4,5,6,6,7,8,9, 10,11,12,13,14,15,15)
$\forall i \in [0, 2), v_i \in [0, 16) \Leftrightarrow$ vector $\mathbf{t}$ is an ordered permutation of $\mathbf{v}$ and $\mathbf{a}$. The elements in $\mathbf{t}$ are either equal to the previous value or equal to the elements in $\mathbf{a}$.
lookup PIOP

Figure 3: Comparison of intuitions among other range proof schemes. Note: "$\langle , \rangle$" denotes inner product and "$\circ$" denotes Hadamard product. The yellow blocks show the reduction preprocesses. The blue blocks show the new equivalent relations after reduction. The red blocks show the core techniques that used to prove the relations given in the blue blocks.

Range proofs for multiple elements. Sharp [13] is the only vector range proof scheme based on square decomposition. As shown in Table 1, its advantage is smaller proof complexity, but both its proof size and verifier complexity are linearly related to the vector length, which hinders scalability.

As for the k-ary decomposition way, we roughly divide techniques for constructing batch range proofs into two categories: IPA based schemes and PIOP based schemes. For each scheme, we provide a toy example of its intuition in Fig 3, and Table 1 shows their complexities.

IPA based range proofs can be batch proved and verified. The prover splits each element in the secret vector $\mathbf{v} \in \mathbb{F}^l$ into a binary vector and concatenates l binary vectors to form a long binary vector of length ln. Then, using IPA, the prover proves the relationship between this string and the commitment vector. Compared with repeatedly generating a single proof l times, its main advantage is reducing the proof size from $O(l \cdot \log n)$ to $O(\log l + \log n)$. However, this level of optimization still fails to meet our requirements, and transmitting l commitments has already incurred a communication complexity of $O(l)$.

The PIOP based argument schemes extend the secret vector $\mathbf{v} \in \mathbb{F}^l$ to a polynomial $\mathfrak{f}(X)$ over the field $\mathbb{F}$ of degree less than l, such that for all $a \in \mathbb{H}_l$, $\mathfrak{f}(a) = v_i$, where $\mathbb{H}_l$ is a multiplicative subgroup of the field $\mathbb{F}$. Then the prover proves that for all $a \in \mathbb{H}_l$, $\mathfrak{f}(a) \in [0, 2^n)$. TurboPlonk [15] first splits each element v_i into a binary vector $(v_{i,0}, ..., v_{i,n-1})$, where $i \in [0, l)$. Subsequently, for $j \in [0, n)$, it extends n vectors $(v_{0,j}, ..., v_{l-1,j})$ to n univariate polynomials $\mathfrak{f}_j(X)$ of degree less than l over $\mathbb{F}$, such that $\forall a_i \in \mathbb{H}_l, \mathfrak{f}_j(a_i) = v_{i,j}$. TurboPlonk then designs a PIOP to prove that for all $a_i \in \mathbb{H}_l$ and $j \in [0, n), \mathfrak{f}_j(a_i) \in \{0, 1\}$ and $\mathfrak{f}(a_i) = \sum_{j=0}^{n-1} \mathfrak{f}_j(a_i) \cdot 2^j$. (TurboPlonk here is a customized version for range proof but not the original generic SNARK.) Its advantage lies in short proving time. However, as a part of the proof, it needs to send the commitments to all polynomials $\mathfrak{f}_j, j \in [0, n)$, resulting in a proof size of $O(n)$. Accordingly, its verification complexity is also $O(n)$. As shown in Fig 3, Plonkup designs a scheme based on the lookup argument PIOP. It introduces an auxiliary vector $\mathbf{a} = (0, 1, ..., 2^n - 1) \in \mathbb{F}^{2^n}$. The prover then concatenates the vectors $\mathbf{a}$ and $\mathbf{b}$ and sorts it to get a long vector $\mathbf{t} \in \mathbb{F}^{2^n+l}$. Then he proves that the elements in $\mathbf{t}$ are either equal to the previous element or equal to the elements in the auxiliary vector. Plonkup performs well when l is large and n is small since its proving complexity is $O(l + 2^n)$. But by the same token, it is very inefficient in dealing with the situation where n is large.

Polynomial interactive oracle proof (PIOP). Here we introduce a general paradigm to design a non-interactive argument protocol which our work follows.

PIOP [17] is a type of interactive information-theoretic proof system. In a PIOP, the prover sends the oracles to multi-variate polynomials as messages, and the verifier can query these polynomials at arbitrary points. The statement can also consist of oracles to polynomials which the verifier can query.

One can obtain a succinct interactive argument of knowledge by compiling a PIOP [17] with a cryptographic primitive called a polynomial commitment scheme [18][19]. In a nutshell, the compiler replaces each oracle and associated evaluation query in the PIOP with a polynomial commitment scheme, and transform the entire protocol into a succinct argument [20]. Then this interactive argument can be turned into a non-interactive one via the Fiat-Shamir transformation [21].

In summary, one can obtain a succinct argument via the following three-step design process.

- *PIOP.* Design a public-coin PIOP for a certain relation.
- *Compile.* Run a compiler to replace oracles in the PIOP with polynomial commitment schemes to obtain a public-coin, interactive succinct argument.
- *Fiat-Shamir transform.* Remove the interaction via Fiat-Shamir transformation to get a non-interactive succinct argument. Concretely, it replaces the random challenge sent from the verifier with a hash function output.

Vector commitment. Vector commitments (VC) [22][23][8] are commitments to a vector $\mathbf{v}$. Polynomial-based vector commitment (PVC) [8] is a type of vector commitment schemes which encodes the vector $\mathbf{v}$ to a univariate polynomial $\mathfrak{f}(X) \in \mathbb{F}_{d(X)<l}[X]$, such that for all $a_i \in \mathbb{H}_l, \mathfrak{f}(a_i) = v_i$. Then the PVC computes $C_{\mathfrak{f}}$, the polynomial commitment to $\mathfrak{f}$ as the output vector commitment (the full definition of the polynomial-based vector commitment scheme and the polynomial commitment scheme can be seen in definition 3.3 and definition 3.6).

Although the MissileProof proposed in this paper can be applied to prove all PVC schemes, in the experimental part and the complexity analysis part, we choose to implement the range proof for the aggregatable subvector commitment (aSVC) [8] scheme. aSVC is a vector commitment scheme based on the KZG [18] polynomial commitment scheme. aSVC can aggregate multiple proofs into a single, small subvector proof and is used in the stateless cryptocurrencies.

3 PRELIMINARIES

We use $\mathbb{F}$ to denote a finite field (a prime field $\mathbb{F}_p$ for a large prime p) and λ to denote the security parameter. A univariate polynomial $\mathfrak{f}$ of degree $< l$ over the field $\mathbb{F}$ is denoted as $\mathfrak{f}(X) = \sum_{i=0}^{l-1} f_i X^i \in \mathbb{F}_{d(X)<l}[X]$. Correspondingly, a bivariate polynomial is denoted as $\mathfrak{m}(X,Y) = \sum_{i=0,j=0}^{l-1,n-1} m_{i,j} X^i Y^j \in \mathbb{F}_{d(X)<l,d(Y)<n}[X,Y]$. We assume that n and l are powers of 2 and divide $p-1$. Let l denote the vector length and n denote the max bit length of elements ($v \in [0, 2^n)$) Let ω_n be the n-th primitive root of unity in $\mathbb{F}$ and let $\mathbb{H}_n = [\omega_n^j]_{j\in[0,n)}$. Let ω_l be the l-th primitive root of unity in $\mathbb{F}$ and let $\mathbb{H}_l = [\omega_l^i]_{i\in[0,l)}$. $\mathbb{H}_n$ and $\mathbb{H}_l$ are two multiplicative subgroups of $\mathbb{F}$. "$\Leftrightarrow$" means "if and only if". A negligible function is denoted as $\mathsf{negl}(\lambda)$ and "probabilistic polynomial-time" is abbreviated as PPT. The prover is denoted as $\mathcal{P}$ and the verifier is denoted as $\mathcal{V}$.

Moreover, we introduce two special polynomials:

- Vanishing polynomial. $z_{\mathbb{H}_l}(X) = \prod_{a\in\mathbb{H}_l}(X-a) = X^l - 1$.
- Bivariate random polynomial. $\mathfrak{r}(R,Y)$ is a polynomial such that all $b \in \mathbb{H}_n, \mathfrak{r}(R,b)$ are n linearly independent polynomials. Concretely, we choose $\mathfrak{r}(R,Y) = \frac{R^n - Y^n}{R-Y}$ as an instance of bivariate random polynomial (This random polynomial is proposed in Marlin [24]. Its advantage is that it can be efficiently evaluated in $O(1)$ operations).

Lemma 3.1. ***(Schwartz-Zippel Lemma).*** *Let $\mathfrak{f} \in \mathbb{F}[X_1,...,X_k]$ be a non-zero multivariate polynomial of total degree d over field $\mathbb{F}$. Randomly choose $r_1,...,r_k \xleftarrow{\$} \mathbb{F}$, then $\Pr[\mathfrak{f}(r_1,...,r_k)=0] \le \frac{d}{|\mathbb{F}|}$.*

3.1 Succinct interactive arguments of knowledge

A relation $\mathcal{R}$ is a set of pairs $(\mathbb{x}, \mathbb{w})$, where the $\mathbb{x}$ is the problem instance statement and the $\mathbb{w}$ is the witness. A pair of PPT interactive algorithms is denoted as $\langle\mathcal{P},\mathcal{V}\rangle$ and Setup is an algorithm that takes as input the security parameter λ and outputs public parameters pp.

Definition 3.1. ***Public-coin succinct interactive argument of knowledge [25]*** *. A protocol between a pair of* PPT *algorithms $\langle\mathcal{P},\mathcal{V}\rangle$ is called a public-coin succinct interactive argument of knowledge for a relation $\mathcal{R}$ if:*

- **Completeness**. For any problem instance $\mathbb{x} \in \mathcal{R}$, there exists a witness $\mathbb{w}$ such that for all $r \in \{0,1\}^*$, $\Pr\{\langle\mathcal{P}(\mathsf{pp},\mathbb{w}), \mathcal{V}(\mathsf{pp},r)\rangle(\mathbb{x}) = 1\} \ge 1 - \mathsf{negl}(\lambda)$.
- **Soundness**. For any non-satisfiable problem instance $\mathbb{x}$, any PPT prover $\mathcal{P}^*$, and for all $\mathbb{w}, r \in \{0,1\}^*$, $\Pr\{\langle\mathcal{P}^*(\mathsf{pp},\mathbb{w}), \mathcal{V}(\mathsf{pp},r)\rangle(\mathbb{x}) = 1\} \le \mathsf{negl}(\lambda)$.
- **Knowledge soundness**. For any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that for any problem instance $\mathbb{x}$ and for all $\mathbb{w}, r \in \{0,1\}^*$, if $\Pr\{\langle\mathcal{A}(\mathsf{pp},\mathbb{w}),\mathcal{V}(\mathsf{pp},r)\rangle(\mathbb{x}) = 1\} \ge \mathsf{negl}(\lambda)$, then $\Pr\{\mathsf{Sat}_{\mathcal{R}}(\mathbb{x},\mathbb{w}') = 1 | \mathbb{w}' \leftarrow \mathcal{E}^{\mathcal{A}}(\mathsf{pp},\mathbb{x})\} \ge \mathsf{negl}(\lambda)$.
- **Succinctness**. The total communication between $\mathcal{P}$ and $\mathcal{V}$ is sub-linear in the size of the NP statement $\mathbb{x} \in \mathcal{R}$.
- **Public coin**. $\mathcal{V}$'s messages are chosen uniformly at random.

Definition 3.2. ***(Zero-knowledge).*** *An interactive argument* (Setup, $\mathcal{P}, \mathcal{V}$) *for $\mathcal{R}$ is computational zero-knowledge if for every* PPT *interactive machine $\mathcal{V}^*$, there exists a* PPT *algorithm $\mathcal{S}$ called the simulator, running in time polynomial in the length of its first input such that for every problem instance $\mathbb{x} \in \mathcal{R}$, $\mathsf{Sat}_{\mathcal{R}}(\mathbb{x},\mathbb{w}) = 1$, and $z \in \{0,1\}^*$, the following holds when the distinguishing gap is considered as a function of $|\mathbb{x}|$:*

$$\mathsf{View}(\langle\mathcal{P}(\mathbb{w}),\mathcal{V}^*(z)\rangle(\mathbb{x})) \approx_c \mathcal{S}(\mathbb{x},z)$$

where $\mathsf{View}(\langle\mathcal{P}(\mathbb{w}),\mathcal{V}^*(z)\rangle(\mathbb{x}))$ *denotes the distribution of the transcript of interaction between $\mathcal{P}$ and $\mathcal{V}^*$, and $\approx_c$ denotes that the two quantities are computationally indistinguishable.*

3.2 Polynomial commitment schemes for bivariate polynomials

In a list of arguments or returned tuples, variables before the semicolon are public and variables after are secret; semicolon is omitted if there is no secret information.

Definition 3.3. *[26][17][18]* ***Polynomial commitment scheme for bivariate polynomials****. A polynomial commitment scheme for bivariate polynomials is a tuple of four protocols* PC = (Setup, Commit, Open, Eval)*:*

- pp $\leftarrow$ Setup(1^λ, $\mathbf{D}$): takes as input D (the max degree of bivariate polynomials $\mathbb{F}[X, Y]$, $\mathbf{D} = (D_x, D_y)$); produces public parameters pp.
- $(C; S) \leftarrow$ Commit(pp, $\mathfrak{m}$): takes as input a bivariate polynomial $\mathfrak{m} \in \mathbb{F}_{d(X)<D_x, d(Y)<D_y}[X, Y]$; produces a public commitment C and a secret opening hint S.
- $b \leftarrow$ Open(pp, $C, \mathfrak{m}, S$): $\mathcal{V}$ verifies the opening of commitment C to the bivariate polynomial $\mathfrak{m} \in \mathbb{F}[X, Y]$ with the opening hint S; outputs $b \in \{0, 1\}$.
- $b \leftarrow$ Eval(pp, $C, (\tau_x, \tau_y), v; \mathfrak{m}, S$) is an interactive public-coin protocol between a PPT prover $\mathcal{P}$ and verifier $\mathcal{V}$. Both $\mathcal{V}$ and $\mathcal{P}$ hold a commitment C, a specified coordinate (τ_x, τ_y) and a scalar $v \in \mathbb{F}$. $\mathcal{P}$ additionally knows a bivariate polynomial $\mathfrak{m} \in \mathbb{F}[X, Y]$ and its secret opening hint S. $\mathcal{P}$ attempts to convince $\mathcal{V}$ that $\mathfrak{m}(\tau_x, \tau_y) = v$. At the end of the protocol, $\mathcal{V}$ outputs $b \in \{0, 1\}$.

Definition 3.4. ***Polynomial commitment scheme properties.*** *A tuple of four protocols* PC = (Setup, Commit, Open, Eval) *is a secure extractable polynomial commitment scheme for bivariate polynomials over a finite field* $\mathbb{F}$ *if the following conditions hold:* ***Hiding, Binding, Knowledge soundness, Zero knowledge*** *(See appendix C for detailed definitions).*

Enforcing a different degree bound to the polynomial. Let us first take the univariate polynomial commitment scheme as an example to discuss how to limit the degree bound of a committed polynomial. The polynomial commitment scheme naturally enforces a bound D on the degrees of the polynomials $\mathfrak{f}(X) \in \mathbb{F}_{d(X)\le D}[X]$. To enforce a new degree bound d to the polynomial ($d < D$), the sender needs to commit not only to $\mathfrak{f}(X) \in \mathbb{F}_{d(X)\le d}[X]$, but also to a "shifted polynomials" $\mathfrak{f}'(X) = X^{D-d}\mathfrak{f}(X)$. For a randomly chosen point $\tau \xleftarrow{\$} \mathbb{F}$, $\mathcal{V}$ checks that $\mathfrak{f}'(\tau) \overset{?}{=} \tau^{D-d}\mathfrak{f}(\tau)$.

Similarly, we can also enforce different degree bounds for bivariate polynomials. Furthermore, Marlin [24] introduced an improved scheme for setting new degree bounds to reduce the computational overhead of the prover. This will not be elaborated further here.

3.3 PIOP compilation

Definition 3.5. ***Polynomial interactive oracle proof (PIOP)*** *A PIOP is a public-coin interactive proof for a polynomial oracle relation* $\mathcal{R} = (\mathbb{x}; \mathbb{w})$*, which is an oracle relation in that* $\mathbb{x}$ *can contain oracles to polynomials over some field* $\mathbb{F}$*. The oracles specify the number of variables and the degree in each variable. These oracles can be queried at arbitrary points to evaluate the polynomial at these points. The actual polynomials corresponding to the oracles are contained in the* pp *and the* $\mathbb{x}$*. We denote an oracle to a polynomial* $\mathfrak{f}$ *by* $\mathfrak{f}^O$*. In every protocol message, the* $\mathcal{P}$ *sends multi-variate polynomial oracles.* $\mathcal{V}$ *in every round sends a random challenge.*

PIOP compilation transforms the interactive oracle proof into an interactive argument of knowledge (without oracles) Π. The compilation replaces the oracles with polynomial commitments. Every query by $\mathcal{V}$ is replaced with an invocation of the Eval protocol at the query point τ. The compiled verifier accepts if the PIOP verifier accepts and if the output of all Eval invocations is 1. If Π is public-coin, it can further be compiled into a non-interactive argument of knowledge using the Fiat-Shamir transform.

Theorem 3.1. ***(PIOP Compilation [17][24][27])****. If the polynomial commitment scheme* PC *has witness-extended emulation, and if the t-round Polynomial IOP for* $\mathcal{R}$ *has negligible knowledge error, then the output of the PIOP compilation* Π*, is a secure (non-oracle) argument of knowledge for* $\mathcal{R}$*. The compilation also preserves zero knowledge.*

Batching. Batch openings of polynomial commitments can significantly reduce the prover time, verifier time, and proof size. Specifically, the proof size is influenced solely by the number of oracles and a single batch opening.

3.4 Polynomial based vector commitment

Definition 3.6. ***Polynomial based vector commitment scheme****. We adapt and extend the definitions from aSVC [8]. A polynomial based vector commitment scheme is a tuple of four protocols* VC = (Setup, Commit, Open, Eval)*:*

- pp $\leftarrow$ Setup($1^\lambda, l$): takes as input l (the max length of the vector length); produces public parameters pp.
- $(C_\mathfrak{f}; S) \leftarrow$ Commit(pp, $\mathbf{v}$): takes as input a vector $\mathbf{v}$, then extends it to a univariate polynomial $\mathfrak{f} \in \mathbb{F}_{d(X)<l}[X]$, s.t. $\forall a_i \in \mathbb{H}_l, \mathfrak{f}(a_i) = v_i$; Computes the polynomial commitment to $\mathfrak{f}$: $(C_\mathfrak{f}; S) \leftarrow$ PC.Commit(pp, $\mathfrak{f}$) and outputs $C_\mathfrak{f}$ as the vector commitment and S as a secret opening.
- $b \leftarrow$ Open(pp, $C_\mathfrak{f}, \mathfrak{f}, S$): verifies the opening of commitment $C_\mathfrak{f}$ to the vector $\mathbf{v}$ with the opening hint S; outputs $b \in \{0, 1\}$.
- $b \leftarrow$ Eval(pp, $C_\mathfrak{f}, i \in [0, l), v_i; \mathfrak{m}, S$) is an interactive public-coin protocol between a PPT prover $\mathcal{P}$ and verifier $\mathcal{V}$. $\mathcal{P}$ proves that v_i is the i-th element of $\mathbf{v}$ using the PC.Eval protocol. At the end of the protocol, $\mathcal{V}$ outputs $b \in \{0, 1\}$.

Remark. Since the output of polynomial based vector commitment is a polynomial commitment to the polynomial $\mathfrak{f}$ that extends $\mathbf{v}$, the vector commitment can be seen as an oracle $\mathfrak{f}^O$ to $\mathfrak{f}$ in the PIOP paradigm.

3.5 Fact and lemmas

Here we present some mathematical lemmas which will be used to construct the PIOP toolbox in the next section.

Fact 3.1. ***(Summation)****. Recall that a multiplicative group* $\mathbb{H}_n = \{\omega_n^0, \omega_n^1, ..., \omega_n^{n-1}\}$*, then*

$$\sum_{a\in\mathbb{H}_n} a^k = \begin{cases} 0 & if \quad k \in \{1, 2, ..., n-1\} \\ n & if \quad k = 0 \end{cases}$$

Proof: if $k = 0$, $\sum_{a\in\mathbb{H}_n} a^k = \sum_{a\in\mathbb{H}_n} 1 = n$. If $k \in \{1, 2, ..., n-1\}$, according to the formula for summing geometric series, $\sum_{a\in\mathbb{H}_n} a^k = \sum_{j=0}^{n-1}(\omega_n^j)^k = \frac{\omega_n^{k*n}-1}{\omega_n^k-1} = 0$. □

Lemma 3.2. ***(Univariate SumCheck).*** *[28][24] For any univariate polynomial $\mathfrak{f}(X) = \sum_{i=0}^{D-1} f_i X^i \in \mathbb{F}_{d(X)<D}[X]$, we have:*
$\sum_{a\in\mathbb{H}_l} \mathfrak{f}(a) = v$, if and only if, $\exists \mathfrak{q}(X) \in \mathbb{F}[X], \mathfrak{u}(X) \in \mathbb{F}_{d(X)<l-1}[X]$, $\mathfrak{f}(X) = \frac{v}{l} + X\cdot\mathfrak{u}(X) + \mathfrak{q}(X)z_{\mathbb{H}_l}(X)$.

The proof of this lemma can be seen in section A.1.

Specially, if the degree of $\mathfrak{f}(X)$ equals to l, this lemma can be simplified to: $\sum_{a\in\mathbb{H}_l}\mathfrak{f}(a) = v \Leftrightarrow \exists\mathfrak{u}(X)\in\mathbb{F}_{d(X)<l-1}[X] \wedge \mathfrak{f}(X) = \frac{v}{l} + X\cdot\mathfrak{u}(X)$.

Lemma 3.3. ***(Univariate ZeroTest).*** *For a multiplicative subgroup $\mathbb{H}_l$ and a polynomial $\mathfrak{f}(X) = \sum_{i=0}^{d} f_i X^i \in \mathbb{F}_{d(X)=d}[X]$, $\mathfrak{f}(X)$ is identically zero on $\mathbb{H}_l$ if and only if $\mathfrak{f}(X)$ is divisible by the vanishing polynomial $z_{\mathbb{H}_l}(X)$, i.e.*

$$\forall a\in\mathbb{H}_l, \mathfrak{f}(a) = 0 \Leftrightarrow \exists\mathfrak{e}(X)\in\mathbb{F}[X], \mathfrak{e}(X)\cdot z_{\mathbb{H}_l}(X) = \mathfrak{f}(X)$$

Proof: $\Rightarrow$: Since all the elements in $\mathbb{H}_l$ must be the root of $\mathfrak{f}(X)$, so $\mathfrak{f}(X)$ is divisible by all polynomials $(X-a), a\in\mathbb{H}_l$. Therefore $\mathfrak{f}(X)$ is divisible by the vanishing polynomial $z_{\mathbb{H}_l}(X)$.
$\Leftarrow$: Obviously, $\mathfrak{f}(a) = \mathfrak{e}(a)\cdot z_{\mathbb{H}_l}(a) = \mathfrak{e}(a)\cdot 0 = 0$. □

4 PIOP TOOLBOX

In this section, we describe PIOPs for the relations including univariate SumCheck, bi-to-univariate SumCheck, univariate ZeroTest and bivariate ZeroTest. Among these PIOPs, the PIOPs for bi-to-univariate SumCheck and bivariate ZeroTest are first designed and are our core contributions.

4.1 Univariate SumCheck PIOP

Here we describe a PIOP for the univariate SumCheck relation proving that a function $\mathfrak{f}(X)$ satisfying that $\sum_{a\in\mathbb{H}_l}\mathfrak{f}(a) = v$. This PIOP is constructed based on Lemma 3.2.

Definition 4.1. ***(Univariate SumCheck relation)*** *The relation $\mathcal{R}_{UniSum}$ is the set of all tuples $(\mathbb{x};\mathbb{w}) = (\mathfrak{f}^{\mathcal{O}}, v; \mathfrak{f}(X)\in\mathbb{F}_{d(X)<D}[X])$ where $\sum_{a\in\mathbb{H}_l}\mathfrak{f}(a) = v$.*

Theorem 4.1. *There exists a PIOP for $\mathcal{R}_{UniSum}$, which is perfectly complete and has knowledge soundness error $\delta_{UniSum} = D/|\mathbb{F}|$.*

Proof.
PIOP Construction.

- $\mathcal{P}$ computes the polynomials $\mathfrak{q}(X), \mathfrak{u}(X)\in\mathbb{F}_{d(X)<l-1}[X]$, such that $\mathfrak{f}(X) = \frac{v}{l} + X\cdot\mathfrak{u}(X) + \mathfrak{q}(X)z_{\mathbb{H}_l}(X)$. $\mathcal{P}$ sends the oracles $\mathfrak{u}^{\mathcal{O}}$ and $\mathfrak{q}^{\mathcal{O}}$ to $\mathcal{V}$.
- $\mathcal{V}$ checks that $\mathfrak{u}(X)\in\mathbb{F}_{d(X)<l-1}[X]$ and checks the equation at a random point $\tau_x \overset{\$}{\leftarrow}\mathbb{F}$: $\mathcal{V}$ queries oracles to get $\mu_{\mathfrak{u}} = \mathfrak{u}(\tau_x)$, $\mu_{\mathfrak{q}} = \mathfrak{q}(\tau_x)$, $\mu_{\mathfrak{f}} = \mathfrak{f}(\tau_x)$, and checks that $\mu_{\mathfrak{f}} \overset{?}{=} \frac{v}{l} + \tau_x\cdot\mu_{\mathfrak{u}} + \mu_{\mathfrak{q}} z_{\mathbb{H}_l}(\tau_x)$.
- **Completeness and Knowledge soundness**. As shown in Lemma 3.2, $\sum_{a\in\mathbb{H}_l}\mathfrak{f}(a) = v$, if and only if, $\exists\mathfrak{q}(X)\in\mathbb{F}[X], \mathfrak{u}(X)\in\mathbb{F}_{d(X)<l-1}[X], \mathfrak{f}(X) = \frac{v}{l}+X\cdot\mathfrak{u}(X)+\mathfrak{q}(X)z_{\mathbb{H}_l}(X)$.
So the PIOP for $\mathcal{R}_{BUSum}$ is perfectly complete and the soundness error is the maximum degree over the field size, which is at most $\frac{D}{|\mathbb{F}|}$. □
- **Complexities**.
  - round complexity: 2-round.
  - prover complexity: $O(D)$.
  - proof size: 2 oracles.
  - verifier complexity: query oracles 3 times.

4.2 Bi-to-univariate SumCheck PIOP

Here we first propose a new lemma and then design a PIOP for the bi-to-univariate SumCheck relation proving that a function $\mathfrak{m}(X,Y)$ satisfying that $\sum_{b\in\mathbb{H}_n}\mathfrak{m}(X,b) = \mathfrak{f}(X)$.

Lemma 4.1. ***(Bi-to-univariate SumCheck).*** *For any bivariate polynomial $\mathfrak{m}(X,Y) = \sum_{i=0}^{l-1}\sum_{j=0}^{n-1} m_{i,j}X^iY^j \in \mathbb{F}_{d(X)<l,d(Y)<n}[X,Y]$, we have:*
$\sum_{b\in\mathbb{H}_n}\mathfrak{m}(X,b) = \mathfrak{f}(X)$, if and only if, $\exists\mathfrak{u}(X,Y)\in\mathbb{F}_{d(X)<l,d(Y)<n-1}[X,Y]$, $\mathfrak{m}(X,Y) = \frac{\mathfrak{f}(X)}{n} + Y\cdot\mathfrak{u}(X,Y)$.

Proof:
$\Rightarrow$: By Fact 3.1, we have:

$$\begin{aligned}\sum_{b\in\mathbb{H}_n}\mathfrak{m}(X,b) &= \sum_{b\in\mathbb{H}_n}\sum_{i=0}^{l-1}\sum_{j=0}^{n-1} m_{i,j}X^ib^j \\ &= \sum_{i=0}^{l-1}\sum_{j=0}^{n-1} m_{i,j}X^i\sum_{b\in\mathbb{H}_n}b^j \\ &= \sum_{i=0}^{l-1}\sum_{j=1}^{n-1} m_{i,j}X^i\sum_{b\in\mathbb{H}_n}b^j + \sum_{i=0}^{i=l-1} m_{i,0}X^i\sum_{b\in\mathbb{H}_n}b^0 \\ &= \sum_{i=0}^{l-1}\sum_{j=1}^{n-1} m_{i,j}X^i\cdot 0 + \sum_{i=0}^{i=l-1} m_{i,0}X^i\cdot n \\ &= n\cdot\sum_{i=0}^{i=l-1} m_{i,0}X^i.\end{aligned}$$

Thus, $\mathfrak{f}(X) = \sum_{b\in\mathbb{H}_n}\mathfrak{m}(X,b) = n\sum_{i=0}^{i=l-1}m_{i,0}X^i$.

Now, set $\mathfrak{u}(X,Y) = \frac{\mathfrak{m}(X,Y)-\frac{1}{n}\mathfrak{f}(X)}{Y} = \sum_{i=0}^{l-1}\sum_{j=1}^{n-1}m_{i,j}X^iY^{j-1} \in \mathbb{F}_{d(X)<l,d(Y)<n-1}[X,Y]$. This choice of $\mathfrak{u}(X,Y)$ ensures that $\frac{\mathfrak{f}(X)}{n} + Y\cdot\mathfrak{u}(X,Y) = \mathfrak{m}(X,Y)$.

$\Leftarrow$: Since $\mathfrak{u}(X,Y)$ is a bivariate polynomial in $\mathbb{F}_{d(X)<l,d(Y)<n-1}[X,Y]$, we can represent it as $\mathfrak{u}(X,Y) = \sum_{i=0}^{l-1}\sum_{j=0}^{n-2}u_{i,j}X^iY^j$. Then, using the given expression $\mathfrak{m}(X,Y) = \frac{\mathfrak{f}(X)}{n} + Y\cdot\mathfrak{u}(X,Y)$ and the Fact 3.1, we have:

$$\begin{aligned}\sum_{b\in\mathbb{H}_n}\mathfrak{m}(X,b) &= \sum_{b\in\mathbb{H}_n}\left(\frac{\mathfrak{f}(X)}{n} + b\cdot\mathfrak{u}(X,b)\right) \\ &= \mathfrak{f}(X) + \sum_{i=0}^{l-1}\sum_{j=0}^{n-2}u_{i,j}X^i\sum_{b\in\mathbb{H}_n}b^{j+1} \\ &= \mathfrak{f}(X).\end{aligned}$$

This completes the proof. □

Definition 4.2. ***(Bi-to-univariate SumCheck relation)*** *The relation $\mathcal{R}_{BUSum}$ is the set of all tuples* $(\mathbb{x}; \mathbb{w}) = (\mathfrak{m}^{\mathcal{O}}, \mathfrak{f}^{\mathcal{O}}; \mathfrak{m}(X,Y) \in \mathbb{F}_{d(X)<l, d(Y)<n}[X,Y], \mathfrak{f}(X) \in \mathbb{F}_{d(X)<l}[X], \sum_{b\in\mathbb{H}_n} \mathfrak{m}(X,b) = \mathfrak{f}(X))$. *(note that in this PIOP, we constrained the degree of* $\mathfrak{m}(X,Y)$*.)*

Theorem 4.2. *There exists a PIOP for $\mathcal{R}_{BUSum}$, which is perfectly complete and has knowledge soundness error $\delta_{BUSum} = \frac{l+n}{|\mathbb{F}|}$.*

Proof.
PIOP Construction.

- $\mathcal{P}$ computes the bivariate polynomial $\mathfrak{u}(X,Y) = \frac{\mathfrak{m}(X,Y) - \mathfrak{f}(X)/n}{Y} \in \mathbb{F}_{d(X)<l, d(Y)<n-1}[X,Y]$. $\mathcal{P}$ sends the oracle $\mathfrak{u}^{\mathcal{O}}$ to $\mathcal{V}$.
- $\mathcal{V}$ checks the equation at a random point $\tau_x, \tau_y \xleftarrow{\$} \mathbb{F}$: $\mathcal{V}$ queries $\mu_m = \mathfrak{m}(\tau_x, \tau_y)$, $\mu_\mathfrak{f} = \mathfrak{f}(\tau_x)$, $\mu_\mathfrak{u} = \mathfrak{u}(\tau_x, \tau_y)$, and checks that $\mu_\mathfrak{m} \overset{?}{=} \frac{\mu_\mathfrak{f}}{n} + \tau_y \cdot \mu_\mathfrak{u}$. Moreover, $\mathcal{V}$ checks that $\mathfrak{u}(X,Y) \in \mathbb{F}_{d(X)<l, d(Y)<n-1}[X,Y]$.
- **Completeness and Knowledge soundness**. As shown in Lemma 4.1, $\sum_{b\in\mathbb{H}_n} \mathfrak{m}(X,b) = \mathfrak{f}(X)$, if and only if $\exists \mathfrak{u}(X,Y) \in \mathbb{F}_{d(X)<l, d(Y)<n-1}[X,Y], \mathfrak{m}(X,Y) = \frac{\mathfrak{f}(X)}{n} + Y \cdot \mathfrak{u}(X,Y)$. So the PIOP for $\mathcal{R}_{BUSum}$ is perfectly complete and the soundness error is the total degree over the field size, which is at most $\frac{l+n}{|\mathbb{F}|}$. □
- **Complexities**.
  - round complexity: 2-round.
  - prover complexity: $O(ln)$.
  - proof size: 1 oracle.
  - verifier complexity: query oracles 3 times.

4.3 Univariate ZeroTest PIOP

Here we describe a PIOP proving that a univariate polynomial evaluates to zero everywhere on a subgroup $\mathbb{H}_l$. The PIOP leverages the ZeroTest Lemma 3.3.

Definition 4.3. ***(Univariate ZeroTest relation)*** *The relation $\mathcal{R}_{UniZT}$ is the set of all tuples* $(\mathbb{x}; \mathbb{w}) = (\mathfrak{f}^{\mathcal{O}}; \mathfrak{f}(X) \in \mathbb{F}_{d(X)<D}[X])$ *where for all* $a \in \mathbb{H}_l, \mathfrak{f}(a) = 0$.

Theorem 4.3. *There exists a PIOP for $\mathcal{R}_{UniZT}$, which is perfectly complete and has knowledge soundness error $\delta_{UniZT} = \frac{D}{|\mathbb{F}|}$.*

Proof.
PIOP Construction.

- $\mathcal{P}$ computes the univariate polynomial $\mathfrak{q}(X)$, such that $\mathfrak{f}(X) = \mathfrak{q}(X) \cdot z_{\mathbb{H}_l}(X)$. $\mathcal{P}$ sends the oracle $\mathfrak{q}^{\mathcal{O}}$ to $\mathcal{V}$.
- $\mathcal{V}$ checks the equation at a random point $\tau_x \xleftarrow{\$} \mathbb{F}$: $\mathcal{V}$ queries oracles to get $\mu_\mathfrak{q} = \mathfrak{q}(\tau_x)$ and $\mu_\mathfrak{f} = \mathfrak{f}(\tau_x)$ and checks that $\mu_\mathfrak{f} \overset{?}{=} \mu_\mathfrak{q} z_{\mathbb{H}_l}(\tau_x)$.
- **Completeness and Knowledge soundness**. As shown in Lemma 3.3, $\forall a \in \mathbb{H}_l, \mathfrak{f}(a) = 0 \Leftrightarrow \exists \mathfrak{e}(X) \in \mathbb{F}[X], \mathfrak{e}(X) \cdot z_{\mathbb{H}_l}(X) = \mathfrak{f}(X)$, so the PIOP for $\mathcal{R}_{BUSum}$ is perfectly complete. The soundness error is the maximum degree over the field size, which is at most $\frac{D}{|\mathbb{F}|}$. □
- **Complexities**.
  - round complexity: 2-round.
  - prover complexity: $O(D)$.
  - proof size: 1 oracle.
  - verifier complexity: query oracles 2 times.

4.4 Bivariate ZeroTest PIOP

Here we describe a PIOP proving that a bivariate polynomial evaluates to zero everywhere on a set $\{(a,b)\}_{a\in\mathbb{H}_l, b\in\mathbb{H}_n}$. The PIOP leverages the univariate polynomial ZeroTest PIOP and the univariate SumCheck PIOP introduced in section 4.1 and the ZeroTest PIOP in section 4.3.

Definition 4.4. ***(Bivariate ZeroTest relation)***. *The relation $\mathcal{R}_{BiZT}$ is the set of all tuples* $(\mathbb{x}; \mathbb{w}) = (\mathfrak{M}^{\mathcal{O}}; \mathfrak{M}(X,Y) \in \mathbb{F}_{d(X)<D_x, d(Y)<D_y}[X,Y])$, *where for all* $a \in \mathbb{H}_l$ *and all* $b \in \mathbb{H}_n, \mathfrak{M}(a,b) = 0$.

Theorem 4.4. *There exists a PIOP for $\mathcal{R}_{BiZT}$, which is perfectly complete and has knowledge soundness error $\delta_{BiZT} = \frac{D_x + D_y}{|\mathbb{F}|}$.*

Proof.
Let $\mathfrak{r}(R,Y)$ be a prescribed polynomial such that all $\mathfrak{r}(R,b), b \in \mathbb{H}_n$ are n linearly independent polynomials. Denote $\mathfrak{M}^*(R,X) = \sum_{b\in\mathbb{H}_n} \mathfrak{M}(X,b)\mathfrak{r}(R,b)$. The ZeroTest relation is equivalent to the below equations:

- Equation 1: $\forall a \in \mathbb{H}_l, \mathfrak{M}^*(R,a) = 0$.
- Equation 2: $\mathfrak{M}^*(R,X) = \sum_{b\in\mathbb{H}_n} \mathfrak{M}(X,b)\mathfrak{r}(R,b)$.

The PIOP for $\mathcal{R}_{BiZT}$ is essentially a combination of the PIOP for $\mathcal{R}_{UniZT}$ and the PIOP for $\mathcal{R}_{UniSum}$.
PIOP Construction:
Phase 1: Following the PIOP for $\mathcal{R}_{UniZT}$, $\mathcal{P}$ proves that $\forall a \in \mathbb{H}_l, \mathfrak{M}^*(r,a) = 0$.

- $\mathcal{V}$ chooses a random value $\tau_r \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ computes the univariate polynomial $\mathfrak{M}^*(\tau_r, X) = \sum_{b\in\mathbb{H}_n} \mathfrak{M}(X,b)\mathfrak{r}(\tau_r,b)$. $\mathcal{P}$ computes the polynomial $\mathfrak{e}(X)$, such that $\mathfrak{e}(X) \cdot z_{\mathbb{H}_l}(X) = \mathfrak{M}^*(\tau_r, X)$. $\mathcal{P}$ sends the oracle $\mathfrak{e}^{\mathcal{O}}$ to $\mathcal{V}$.
- $\mathcal{V}$ chooses a random value $\tau_x \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ computes the univariate polynomial $\mu_{\mathfrak{M}^*} = \mathfrak{M}^*(\tau_r, \tau_x)$ and sends $\mu_{\mathfrak{M}^*}$ to $\mathcal{V}$.
- $\mathcal{V}$ checks the equation at τ_x: $\mathcal{V}$ queries oracle $\mathfrak{e}^{\mathcal{O}}$ to get $\mu_\mathfrak{e} = \mathfrak{e}(\tau_x)$ and checks that $\mu_{\mathfrak{M}^*} \overset{?}{=} \mu_\mathfrak{e} z_{\mathbb{H}_l}(\tau_x)$.

Phase 2: Following the PIOP for $\mathcal{R}_{UniSum}$, $\mathcal{P}$ proves that the purported $\mu_{\mathfrak{M}^*} = \sum_{b\in\mathbb{H}_n} \mathfrak{M}(\tau_x, b)\mathfrak{r}(\tau_r, b)$.

- $\mathcal{P}$ computes the univariate polynomial $\mathfrak{q}(Y), \mathfrak{u}(Y) \in \mathbb{F}_{d(Y)<n-1}[Y]$, such that $\mathfrak{M}(\tau_x, Y)\mathfrak{r}(\tau_r, Y) = \frac{\mu_{\mathfrak{M}^*}}{n} + Y \cdot \mathfrak{u}(Y) + \mathfrak{q}(Y) z_{\mathbb{H}_n}(Y)$. $\mathcal{P}$ sends the oracles $\mathfrak{u}^{\mathcal{O}}$ and $\mathfrak{q}^{\mathcal{O}}$ to $\mathcal{V}$.
- $\mathcal{V}$ checks that $\mathfrak{u}(Y) \in \mathbb{F}_{d(Y)<n-1}[Y]$. Then $\mathcal{V}$ checks the equation at a random point $\tau_y \xleftarrow{\$} \mathbb{F}$: $\mathcal{V}$ queries oracles to get $\mu_\mathfrak{u} = \mathfrak{u}(\tau_y)$, $\mu_\mathfrak{q} = \mathfrak{q}(\tau_y)$, $\mu_\mathfrak{m} = \mathfrak{f}(\tau_x, \tau_y)$, and checks that $\mathfrak{M}(\tau_x, \tau_y)\mathfrak{r}(\tau_r, \tau_y) \overset{?}{=} \frac{\mu_{\mathfrak{M}^*}}{n} + \tau_y \cdot \mu_\mathfrak{u} + \mu_\mathfrak{q} z_{\mathbb{H}_n}(\tau_y)$.
- **Completeness and Knowledge soundness**. Follows the PIOPs introduced in section 4.3 and 4.1, the PIOP for $\mathcal{R}_{BiZT}$ is perfectly complete and the soundness error is the total degree over the field size, which is at most $O(\frac{D_x + D_y}{|\mathbb{F}|})$. □
- **Complexities**.
  - round complexity: 5-round (In practice, phase 1 and phase 2 can run in parallel).

  - prover complexity: $O(D_x D_y)$.
  - proof size: 3 oracles and one field element.
  - verifier complexity: query oracles 4 times.

5 MISSILEPROOF: A SUCCINCT NON-INTERACTIVE ARGUMENT FOR VECTOR COMMITMENT RANGE PROOF

In this section we first define a relation $\mathcal{R}_{\text{VCRP}}$ of vector range proof, then reduce it to an equivalent polynomial oracle relation $\mathcal{R}_{\text{VCRP}_{\text{PIOP}}}$. In subsection 5.1, we construct a PIOP for $\mathcal{R}_{\text{VCRP}_{\text{PIOP}}}$ and then compile it with a KZG-based polynomial commitment scheme to get an interactive argument of knowledge. Finally, we turn it into a succinct non-interactive argument via the Fiat-Shamir transformation.

Here we give the definition for the vector range proof relation $\mathcal{R}_{\text{VCRP}}$.

Definition 5.1. ***Relation*** $\mathcal{R}_{VCRP}$. *The relation* $\mathcal{R}_{VCRP}$ *is the set of all pairs:* $(\mathbb{x}, \mathbb{w}) = (C \in \mathbb{G}; \mathbf{v} \in \mathbb{F}^l)$, *where*

$$\{(C \in \mathbb{G}; \mathbf{v} \in \mathbb{F}^l) : \forall i \in [0, l), C = \text{VC.Commit}(\mathbf{v}) \wedge v_i \in [0, 2^n)\}$$

Note that a polynomial-based vector commitment to a secret vector $\mathbf{v}$ is a polynomial commitment to a univariate polynomial $\mathfrak{f}(X) \in \mathbb{F}_{d(X)<l}[X]$ that extends $\mathbf{v}$ over $\mathbb{F}$. Thus the vector commitment C can be seen as an oracle $\mathfrak{f}^O$ to the polynomial $\mathfrak{f}(X)$. Recall that

$$\exists M \in \mathbb{F}^{l \times n}, v_i = \sum_{j=0}^{n-1} M_{i,j} \wedge M_{i,j} \in \{0, 2^j\}$$

Then $\mathcal{R}_{\text{VCRP}}$ can be equivalently reduced to $\mathcal{R}_{\text{VCRP}_{\text{PIOP}}}$, which is defined below.

Definition 5.2. ***Relation*** $\mathcal{R}_{VCRP_{PIOP}}$. *Let* $\mathfrak{p}(Y) \in \mathbb{F}_{d(Y)<n}[Y]$ *be a univariate polynomial satisfying that for all* $b_j \in \mathbb{H}_n, \mathfrak{p}(b_j) = 2^j$. *The relation* $\mathcal{R}_{VCRP_{PIOP}}$ *is the set of all pairs:* $(\mathbb{x}, \mathbb{w}) = (\mathfrak{f}^O, \mathfrak{p}^O; \mathfrak{f} \in \mathbb{F}_{d(X)<l}[X])$, *where exists a bivariate polynomial* $\mathfrak{m}(X, Y) \in \mathbb{F}_{d(X)<l,d(Y)<n}[X, Y]$, *s.t.*

$$\sum_{b \in \mathbb{H}_n} \mathfrak{m}(X, b) = \mathfrak{f}(X)$$
$$\wedge \forall a \in \mathbb{H}_l, b \in \mathbb{H}_n, \mathfrak{m}(a, b)(\mathfrak{m}(a, b) - \mathfrak{p}(b)) = 0$$

The following theorem summarizes part of our result in this section.

Theorem 5.1. *Given secure polynomial commitment schemes for bivariate and univariate polynomials, there exists a public-coin succinct interactive argument of knowledge for* $\mathcal{R}_{VCRP}$ *where security holds under the assumptions needed for the polynomial commitment schemes. When use the KZG-based polynomial commitment scheme [24][26], the complexities are as follows:*

- *soundness error:* $O(\frac{l+n}{|\mathbb{F}|})$.
- *round complexity:* 6.
- *prover complexity:* $O(l \log l \cdot n \log n)$ *field operations and* $O(ln)$ *group exponentiations in the bilinear pairing group* $\mathbb{G}_1$.
- *proof size:* $O(1)$.
- *verifier complexity:* $O(1)$.
- *size of an updatable strctured reference string (SRS):* $O(ln)$.

Remark: The polynomial commitment scheme mentioned in the theorem 5.1 can be arbitrarily replaced with other existing schemes to obtain different properties.

To prove the theorem 5.1, we first provide a construction of a public-coin PIOP for $\mathcal{R}_{\text{VCRP}}$. Then compile the PIOP and the KZG-based polynomial commitment scheme into a succinct interactive argument of knowledge.

Finally, we turn it into a non-interactive argument of knowledge using Fiat-Shamir transform and then analyze its costs and security.

5.1 Public-coin PIOP for $\mathcal{R}_{\text{VCRP}_{\text{PIOP}}}$

In this section, we give a public-coin polynomial IOP for $\mathcal{R}_{\text{VCRP}_{\text{PIOP}}}$. The whole protocol is shown in Fig 6.

PIOP construction:

Phase 1: $\mathcal{P}$ generates decomposition bivariate polynomial $\mathfrak{m}(X, Y)$.

- For all $i \in [0, l)$, split v_i into a binary vector $(v_{i,0}, ..., v_{i,n-1})$, s.t. $v_i = \sum_{j=0}^{n-1} v_{i,j} \cdot 2^j$. Then generate the matrix $M \in \mathbb{F}^{l \times n}, M_{i,j} = v_{i,j} \cdot 2^j$. Then extend the matrix M to a bivariate polynomial $\mathfrak{m}(X, Y) \in \mathbb{F}_{d(X)<l,d(Y)<n}[X, Y]$, s.t. $\forall a_i \in \mathbb{H}_l, b_j \in \mathbb{H}_n, \mathfrak{m}(a_i, b_j) = M_{i,j}$. $\mathcal{P}$ then sends oracle $\mathfrak{m}^O$ to $\mathcal{V}$.

$\mathcal{P}$ is left to convince $\mathcal{V}$ that the following two conditions hold:

1. Bi-to-univariate polynomial SumCheck: $\sum_{b \in \mathbb{H}_n} \mathfrak{m}(X, b) = \mathfrak{f}(X)$.
2. Bivariate polynomial ZeroTest: $\forall a \in \mathbb{H}_l, b \in \mathbb{H}_n, \mathfrak{m}(a, b)(\mathfrak{m}(a, b) - \mathfrak{p}(b)) = 0$.

Phase 2: $\mathcal{P}$ proves that $\sum_{b \in \mathbb{H}_n} \mathfrak{m}(X, b) = \mathfrak{f}(X)$.

- In order to convince $\mathcal{V}$ of the first condition (bi-to-univariate polynomial SumCheck), $\mathcal{P}$ and $\mathcal{V}$ run a bi-to-univariate polynomial SumCheck PIOP that $(\mathfrak{m}^O, \mathfrak{f}^O; \mathfrak{m}(X, Y), \mathfrak{f}(X)) \in \mathcal{R}_{BUSum}$. The soundness error is $O(\frac{l+n}{|\mathbb{F}|})$.

Phase 3: $\mathcal{P}$ proves that for all $a \in \mathbb{H}_l$ and all $b \in \mathbb{H}_n$, $\mathfrak{m}(a, b)$ $(\mathfrak{m}(a, b) - \mathfrak{p}(b)) = 0$.

- Denote $\mathfrak{M}(X, Y) = \mathfrak{m}(X, Y)(\mathfrak{m}(X, Y) - \mathfrak{p}(Y))$. In order to convince $\mathcal{V}$ of the second condition (bivariate polynomial ZeroTest), $\mathcal{P}$ and $\mathcal{V}$ run a bivariate polynomial ZeroTest PIOP for $(\mathfrak{M}^O; \mathfrak{M}(X, Y)) \in \mathcal{R}_{BiZT}$. (Actually, $\mathcal{V}$ does not own an oracle $\mathfrak{M}^O$ directly. When he wants to query $\mathfrak{M}^O$ to get $\mu_{\mathfrak{M}} = \mathfrak{M}(\tau_x, \tau_y)$, he can query $\mathfrak{m}^O$ and $\mathfrak{p}^O$ to get $\mu_{\mathfrak{m}} = \mathfrak{m}(\tau_x, \tau_y)$ and $\mu_{\mathfrak{p}} = \mathfrak{p}(\tau_y)$. Then he can get $\mu_{\mathfrak{M}} = \mu_{\mathfrak{m}} \cdot (\mu_{\mathfrak{m}} - \mu_{\mathfrak{p}})$). The soundness error is $O(\frac{l+n}{|\mathbb{F}|})$.

Remark. In summary, the whole PIOP is shown in Fig 6 in the appendix. In practice, the PIOP for the ZeroTest and SumCheck can run in parallel. So $\mathcal{V}$ can choose $\xi_x = \tau_x$ and $\xi_y = \tau_y$.

- **Completeness and knowledge soundness**. Follows PIOP toolbox in section 4, The PIOP for $\mathcal{R}_{\text{VCRP}_{\text{PIOP}}}$ is perfectly complete and the soundness error is the total degree over the field size, $\delta_{\text{VCRP}_{\text{PIOP}}} = \delta_{BUSum} + \delta_{BiZT} = O(\frac{l+n}{|\mathbb{F}|})$.
- **Complexities**.
  - round complexity: 6-round (the two parts of the PIOP can be run in parallel).
  - prover complexity: $O(l \log l \cdot n \log n)$ field operations (FFT for the matrix M).

Table 2: Comparison between different polynomial commitment schemes for a bivariate polynomial of degree (d, d) (For simplity, we assume that $d(X) = d(Y) = d$). Transparent means no trusted setup. e is the extension factor in the FRI scheme.

Protocols	Transparent	Group	|pp|	Proof size	Prover complexity	Verifier complexity
KZG-based [18]	no	bilinear $\mathbb{G}_B$	$O(d^2)$	$2\mathbb{G}_B$	$O(d^2)$MUL	2 Pairing
DARK [17]	yes	Unknown order groups $\mathbb{G}_U$	$O(1)$	$4\log d\mathbb{G}_U$	$O(d^2)$MUL	$O(\sqrt{d^2})$MUL
FRI-based [29]	yes	Hash output field $\mathbb{F}_H$	$O(1)$	$O(e\log^2 d)\mathbb{F}_H$	$O(ed^2)$Hash	$O(e\log^2 d)$Hash

Notes: some notations used here are the same as that in Table. 1.

– proof size: 2 oracles to bivariate polynomials and 3 oracles to univariate polynomials and 1 field element.
– verifier complexity: query oracles to bivariate polynomials 2 times and oracles to univariate polynomials 5 times.

5.2 Non-interactive argument for vector commitment range proof

Subsection 5.1 presents a 6-round PIOP for $\mathcal{R}_{\text{VCRP}}$ ($\mathcal{R}_{\text{VCRP}_{\text{PIOP}}}$), which has perfect completeness and negligible soundness error. As shown in Theorem 3.1, given a polynomial commitment scheme PC that is hiding and Eval is honest-verifier zero-knowledge and has witness-extended emulation, then the PIOP compilation can output Π, a secure zero knowledge argument of knowledge for $\mathcal{R}_{\text{VCRP}}$.

Here we gave an instance of a zero-knowledge argument protocol for $\mathcal{R}_{\text{VCRP}}$ by compiling the PIOP with a KZG-based commitment scheme [18][26]. The efficiency and complexities of Π is as follows:

- soundness error: $O(\frac{l+n}{|\mathbb{F}|})$;
- round complexity: 6;
- prover complexity: $O(l\log l \cdot n\log n)$ field operations and $O(ln)$ group exponentiations in $\mathbb{G}$;
- proof size: $O(1)$;
- verifier complexity: $O(1)$;
- size of the updatable strctured reference string (SRS): $O(ln)$;

This completes the proof of Theorem 5.1. □

Then using Fiat-Shamir transformation, we can turn this interactive argument into a non-interactive argument of knowledge by replacing the interaction random point with the output of hash functions.

6 FURTHER DISCUSSION

6.1 Transparent setup version of MissileProof

Choice of the polynomial commitment scheme. The compiler can compile a PIOP with different polynomial commitment schemes to get different security properties.

In Table 2, we list commonly used polynomial commitment schemes and compared their properties.

KZG-based schemes [18] [26]. The non-interactive argument we presented in section 7 relies on the KZG-based commitment scheme which works on a bilinear pairing group and stand out for having the optimal proof size and the widest range of application scenarios. However, its drawback lies in the requirement for a trusted setup to generate a set of updatable structured reference string.

Other schemes. If a system values the transparent setup, one can compile the PIOP with DARK [17] or FRI-based polynomial commitment scheme [19][29][30][31] to get a non-interactive argument that do not need the trusted setup.

6.2 Batch-VCRP

If $\mathcal{P}$ needs to run the range proof for multiple vector commitments at the same time, compared to directly running the protocol multiple times, the prover time, proof size and the verifier complexity can be significantly reduced via the batch openings of the polynomial commitments. Here we give the definition for the batch vector range proof relation $\mathcal{R}_{\text{Bat-VCRP}}$.

Relation $\mathcal{R}_{\text{Bat-VCRP}}$. The relation $\mathcal{R}_{\text{VCRP}}$ is the set of all pairs: $(\mathbb{x}, \mathbb{w}) = (\mathbf{C} \in \mathbb{G}^t; [\mathbf{v}_k]_{k\in[0,t)} \in (\mathbb{F}^l)^t)$, where

$$\forall k \in [0,t), \forall i \in [0,l), C_k = \mathsf{VC.Commit}(\mathbf{v}_k) \wedge v_{k,i} \in [0, 2^n)$$

The batch opening protocol of polynomial commitment can greatly reduce the proof size and verification time of the proof of relationship $\mathcal{R}_{\text{Bat-VCRP}}$.

Let $\mathfrak{p}(Y) \in \mathbb{F}_{d(Y)<n}[Y]$ be a univariate polynomial satisfying that for all $b_j \in \mathbb{H}_n, \mathfrak{p}(b_j) = 2^j$. For t vectors to be proved and $k \in [0,t)$, let $\mathfrak{f}_k(X) \in \mathbb{F}_{d(X)<l}[X]$ be the univariate polynomial that extends the vector $\mathbf{v}_k$. Same as mentioned before, $\mathcal{P}$ needs to prove that there exists t bivariate polynomial $\mathfrak{m}_k(X,Y) \in \mathbb{F}_{d(X)<l,d(Y)<n}[X,Y]$, s.t.

$$1. \forall k \in [0,t), \sum_{b\in\mathbb{H}_n} \mathfrak{m}_k(X,b) = \mathfrak{f}_k(X)$$

$$2. \forall k \in [0,t), \forall a \in \mathbb{H}_l, b \in \mathbb{H}_n, \mathfrak{m}_k(a,b)(\mathfrak{m}_k(a,b) - \mathfrak{p}(b)) = 0$$

Based on the idea of batch processing, we can turn the conditions to:

$$1. \sum_{k=0}^{t-1} \sum_{b\in\mathbb{H}_n} (\mathfrak{m}_k(X,b) - \mathfrak{f}_k(X)) * Z^k = 0$$

$$2. \forall a \in \mathbb{H}_l, b \in \mathbb{H}_n, \sum_{k=0}^{t-1} \mathfrak{m}_k(a,b)(\mathfrak{m}_k(a,b) - \mathfrak{p}(b)) * Z^k = 0$$

$\mathcal{V}$ can randomly choose $\tau_z \xleftarrow{\$} \mathbb{F}$ and send it to $\mathcal{P}$. $\mathcal{P}$ is left to prove that

$$1. \sum_{k=0}^{t-1} \sum_{b\in\mathbb{H}_n} (\mathfrak{m}_k(X,b) - \mathfrak{f}_k(X)) * \tau_z^k = 0$$

$$2. \sum_{k=0}^{t-1} \mathfrak{m}_k(a,b)(\mathfrak{m}_k(a,b) - \mathfrak{p}(b)) * \tau_z^k = 0$$

which are essentially a bi-to-univariate SumCheck relation and a bivariate ZeroTest relation.

Using this batch processing method, in addition to sending t bivariate polynomials $\mathfrak{m}_k(X, Y)$, the prover only needs to send 6 oracles to witness polynomials to complete one SumCheck and one ZeroTest argument, which is reduced from $O(n)$ to $O(1)$. Moreover, we list the complexities of the batched range proof.

- **Completeness and knowledge soundness**. Following the PIOP toolbox introduced in section 4, the PIOP for $\mathcal{R}_{\text{Bat-VCRP}}$ is perfectly complete and the soundness error is the maximum degree over the field size, $\delta_{\text{Bat-VCRP}} = O(\frac{t+l+n}{|\mathbb{F}|})$. □
- **Complexities**.
  - round complexity: 6-round.
  - prover complexity: $O(t \cdot l \log l \cdot n \log n)$ field operations and $O(tln)$ group exponentiations in the bilinear pairing group $\mathbb{G}_1$.
  - proof size: $t + 2$ oracles to bivariate polynomials and 4 oracles to univariate polynomials and 1 field element.
  - verifier complexity: query oracles to bivariate polynomials $t + 2$ times and oracles to univariate polynomials 6 times.

6.3 Arbitrary range

In the previous discussion, we explored how to prove that all values in a vector are within the range $[0, 2^n)$. However, in practical scenarios, it is often necessary to prove that each element in a vector belongs to arbitrary ranges. Here we introduce a way to prove that all values in a vector are within arbitrary ranges, namely, $\forall v_i \in \mathbf{v}, v_i \in [min_i, max_i]$.

To prove a secret value v lies in range $[a, b]$, it is sufficient to prove both $v - a$ and $b - v$ are non-negative. When n is large (e.g., $n = 64$), proving $v \in [0, 2^n)$ is essentially equivalent to proving $v \geq 0$. So for two bound vectors $\mathbf{min} = (min_0, ..., min_{l-1})$ and $\mathbf{max} = (max_0, ..., max_{l-1})$, which can be extended to two polynomials $\mathfrak{min}(X)$ and $\mathfrak{max}(X) \in \mathbb{F}_{d(X)<l}[X]$, $\forall v_i \in \mathbf{v}, v_i \in [min_i, max_i] \Leftrightarrow \forall a \in \mathbb{H}_l, \mathfrak{f}(a) - \mathfrak{min}(a) \in [0, 2^n) \wedge \mathfrak{max}(a) - \mathfrak{f}(a) \in [0, 2^n)$, where n is an enough large integer. Therefore, the vector range proof for arbitrary ranges can be obtained by running the single MissileProof protocol twice.

6.4 Range proof for subvector

In reality, we often only need to prove that all elements of a subset $\mathbb{S}$ of a vector are within an range. We can achieve this proof by fine-tuning the relation $\mathcal{R}_{\text{VCRP}_{\text{PIOP}}}$ to $\mathcal{R}_{\text{sub-VCRP}_{\text{PIOP}}}$, which is the set of all pairs: $(\mathbb{x}, \mathbb{w}) = (\mathfrak{f}^O, \mathfrak{p}^O; \mathfrak{f} \in \mathbb{F}_{d(X)<l}[X])$, where exists a bivariate polynomial $\mathfrak{m}(X, Y) \in \mathbb{F}_{d(X)<l, d(Y)<n}[X, Y]$, s.t.

$$\sum_{b \in \mathbb{H}_n} \mathfrak{m}(X, b) = \mathfrak{f}(X)$$
$$\wedge \ \forall a \in \mathbb{S}, b \in \mathbb{H}_n, \mathfrak{m}(a, b)(\mathfrak{m}(a, b) - \mathfrak{p}(b)) = 0$$

The PIOP for $\mathcal{R}_{\text{sub-VCRP}_{\text{PIOP}}}$ can be easily get by replacing the vanishing polynomial $z_{\mathbb{H}_l}$ with a new vanishing polynomial $z_{\mathbb{S}} = \prod_{a \in \mathbb{S}} (X - a)$.

7 COMPLEXITIES ANALYSIS AND EXPERIMENT DEMONSTRATION

This section introduces the complexities comparison among our work and other works at first. Then we list the concrete performance comparison. At last we implement an anti-money washing stateless blockchain system equipped with a smart contract based on the MissileProof.

7.1 Benchmark

We evaluated the performance of the MissileProof protocol. We implemented the MissileProof scheme using go (go version go1.20.2 linux/amd64) on a virtual machine with the machine image of ubuntu-20.04.2.0-desktop-amd64, 3.20 GHz processer and 8 GB memory. For the standard group based protocol, we use the elliptic curve secp256k1, on which a point is stored as 64 bytes. For the bilinear pairing based group, we use the curve bn256. A point of $\mathbb{G}_1$ is stored as 64 bytes. Every field element is stored as 32 bytes. In Table 3, we give the concrete complexity analysis of our work and compare them with other related works. In Table 4, we tested various overheads of our work and compare it with other schemes.

Commitment length. Since the MissileProof, Sharp and TurboPlonk are range proofs for vector commitments, the prover only needs to send one group element in $\mathbb{G}_1$. However, the IPA-based scheme needs to send pedersen commitments for each element, resulting in huge overhead. This is the natural advantage of our work.

Proof size. Our scheme has the optimal proof size, 0.81 Kb. This is also one of the biggest advantages of our scheme compared with other schemes. The proof size of TurboPlonk is very large, 6.09 Kb, 7.52 times that of MissileProof. The proof size of Sharp is linear to the vector length, which is unacceptable in real-life scenarios.

Proving cost. Proving cost is the only shortcoming of the MissileProof. It takes 421 s to generate a range proof for a vector of length 16384. The prover cost is between TurboPlonk and IPA-based schemes. It is worth noting that this weakness is not fatal: 1. Prover overhead can be further reduced by introducing a distributed generation system. 2. In a blockchain system, proof is generated off-chain, and verified on-chain. The price of computing power off-chain is much lower than that on-chain.

Verification cost. The verification cost of this scheme is also optimal, 0.0087 s. The verification efficiency is 1.6 times that of TurboPlonk. This also facilitates the verification being deployed on the blockchain.

Conclusion. Experimental data shows that our work has the best performance in terms of the commitment length, proof size and verification time.

The only scheme that could pose a threat to our solution is TurboPlonk. In comparison, the main advantage of MissileProof lies in its smaller proof size and verification cost. Someone might ask: Can we use recursive SNARKs (like Halo) to prove the proof verification process of TurboPlonk, further reducing its proof size and verification time? The answer is yes, but there are two issues to consider: 1. Recursive SNARKs introduce additional prover overhead. 2. Recursive SNARKs impose additional constraints on the choice of elliptic curves, limiting their applicability in a more widespread manner.

Table 3: Comparison of concrete complexities among different range proof schemes for multiple elements.

Protocols	Commitment length	Proof size	Prover complexity	Verifier complexity
BulletProof [10]	$l\mathbb{G}$	$(2\log n+2\log l+4)\mathbb{G}+$ $5\mathbb{F}$	$l(13n+2\log n-1)E+$ $l(14n-2)M$	$l(7n+2\log n+9)E+$ $l(n+3)M$
Daza et al. [9]	$l\mathbb{G}$	$(7\log l+7\log n+12)\mathbb{G}_1+$ $(2\log l+2\log n+5)\mathbb{F}$	$l(14n+11)E_1+$ $l(35n+15)M$	$(2l+9\log l+9\log n+24)E_1+(\log l+\log n)E_2+$ $(2\log l+2\log n+1)M+(6\log l+6\log n)P$
TurboPlonk [15]	$\mathbb{G}_1$	$(n+1)\mathbb{G}_1+$ $(n+1)\mathbb{F}$	$(4ln+2n+2)E_1+$ $(l\log l\cdot n+4ln)M$	$(4n+4)E_1+1E_2+$ $(2n)M+2P$
This work	$\mathbb{G}_1$	$10\mathbb{G}_1+$ $8\mathbb{F}$	$(8ln+4n+4l)E_1+$ $(l\log l\cdot(n\log n+1)+6ln+5l+7n)M$	$18E_1+6E_2+$ $22M+5P$

Notes: the notations used here are the same as that in Table. 1. The term "Commitment length" refers to the length of the commitments to the all values in the vector that the prover needs to provide to the verifier. Green represents the best, while red represents the worst.

Table 4: Experiment results of MissileProof and comparison with other works. The bit length n is fixed to 64 and l denotes the length of the committed vector.

	Schemes	$l=64$	$l=256$	$l=1024$	$l=4096$	$l=16384$
Commitment length (Kb)	BulletProof	4	16	64	256	1024
	Daza	4	16	64	256	1024
	Sharp	0.03	0.03	0.03	0.03	0.03
	TurboPlonk	0.03	0.03	0.03	0.03	0.03
	This work	**0.03**	**0.03**	**0.03**	**0.03**	**0.03**
Proof Size (Kb)	BulletProof	1.90	2.16	2.41	2.66	2.91
	Daza	6.75	7.75	8.75	9.75	10.75
	Sharp	6.56	25.50	101.25	404.25	1616.25
	TurboPlonk	6.09	6.09	6.09	6.09	6.09
	This work	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**
Proving cost (s)	BulletProof	2.37	9.49	37.96	151.85	607.42
	Daza	2.60	10.41	41.66	166.67	666.70
	TurboPlonk	0.73	2.92	11.77	47.38	190.82
	Sharp	0.11	0.42	1.65	6.32	25.52
	This work	**1.53**	**6.22**	**25.35**	**103.34**	**421.15**
Verification cost (s)	BulletProof	1.30	5.21	20.84	83.39	333.56
	Daza	0.11	0.14	0.22	0.51	1.59
	Sharp	0.01	0.05	0.20	0.83	3.34
	TurboPlonk	0.014	0.014	0.014	0.014	0.014
	This work	**0.0087**	**0.0087**	**0.0087**	**0.0087**	**0.0087**

Notes: Green represents the best, while red represents the worst.

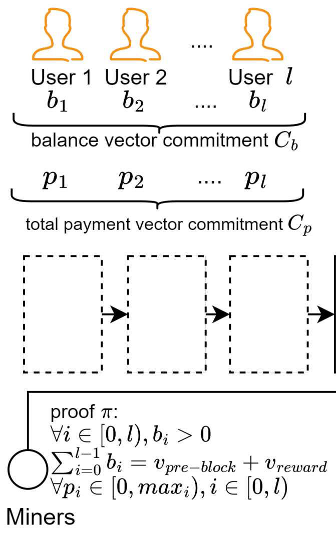

User 1
User 2
....
User l
b_1
b_2
....
b_l
balance vector commitment C_b
p_1
p_2
....
p_l
total payment vector commitment C_p
C_b
C_p
smart contract
proof π:
$\forall i \in [0,l), b_i > 0$
$\sum_{i=0}^{l-1} b_i = v_{pre-block} + v_{reward}$
$\forall p_i \in [0, max_i), i \in [0,l)$
Miners

Figure 4: Stateless blockchain and the regulation smart contract

7.2 Application: an anti-money-laundering regulatable stateless blockchain

As an application demonstration, we implement an anti-money-laundering regulatable stateless blockchain [8] on a private Ethereum (a modified Ethereum v1.9.20 client equipped with a set of precompiled contracts to support bn256 curves [4]). The smart contract is written in solidty 10 and contains about 300 lines of code. We use Truffle 11 to deploy smart contracts. As shown in Fig 4, the anti-money-laundering regulatable stateless blockchain consistently maintains and updates two vector commitments:

- C_b, a balance vector commitment to all users' balances. Miners do not need to store all blocks of the entire chain; they can quickly verify the transactions solely relying on the balance vector commitment (same as the original paper [8]).
- C_p, a total payment vector commitment to all users' total payment. It commits to the vector of total amounts spent by every user.

Both vector commitments will be updated simultaneously by miners. Regulators have three constraints on blockchain:

[4]https://eips.ethereum.org/EIPS/eip-1108

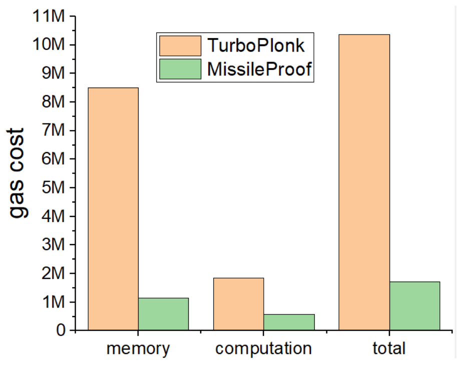


Figure 5: Gas cost of the storage and verification of the proofs

- 1. Blockchain rules require that (regulators don't care about the details of specific transactions):
  - 1.1 All user balances are non-negative. $\forall i \in [0,l), b_i > 0$.

  - 1.2 The total balance of all users is the sum of total balance of the previous block and the block reward. $\sum_{i=0}^{l-1} b_i = v_{pre-block} + v_{reward}$.
- 2. Anti-money-laundering laws that any user $\mathcal{U}_i$ cannot transfer money p_i larger than his payment limit max_i. $\forall i \in [0, l), p_i < max_i$.

Based on MissileProof, we designed a smart contract for constraints verification. Note that both constraint 1.1 and 2 are vector range proofs and constraint 1.2 can be proved using a univariate SumCheck protocol. Miners will generate two MissileProofs and one SumCheck proof to prove that all constraints follow.

After uploading the proofs, regulators call the smart contracts for verification. Compared to regulators directly obtaining all user balances and verifying them one by one, this way supports quick regulation without revealing any specific data. It not only protects user privacy but also achieves rapid regulation. As shown in Fig 5, MissileProof is much better than TurboPlonk in both the gas cost of calculation and the gas cost of storage proof. The total gas cost is reduced from 10.4 M to 1.8 M, reducing consumption by 82%. This reflects MissileProof's huge potential in the blockchain field.

8 CONCLUSION

This paper introduces MissileProof, a zero-knowledge succinct non-interactive argument of knowledge for vector range proof. We reduce this argument to a bi-to-univariate SumCheck problem and the bivariate polynomial ZeroTest problem, and design two PIOPs for them. Then we construct a PIOP for the vector range proof and compile it with a KZG-based extractable polynomial commitment. Via the Fiat-Shamir transformation, we obtain a zero-knowledge succinct non-interactive argument of knowledge for range proof. Our scheme has the optimal proof size ($O(1)$), the optimal commitment length ($O(1)$), and the optimal verification time ($O(1)$), at the expense of slightly sacrificing proof time ($O(l \log l \cdot n \log n)$ operations on the prime field for FFT and $O(ln)$ group exponentiations in $\mathbb{G}$). Experimental data shows that our work has the best performance in terms of the commitment length, proof size and verification time. Additionally, we implement a regulatable anti-money-laundering blockchain system. The gas consumption of the smart contract is reduced by 85%.

REFERENCES

[1] D. Chaum, "Showing credentials without identification transferring signatures between unconditionally unlinkable pseudonyms," in *Advances in Cryptology, AUSCRYPT 1990, Sydney, Australia.* Springer, 1990, pp. 245–264.

[2] J. Groth, "Non-interactive zero-knowledge arguments for voting," in *Applied Cryptography and Network Security, ACNS 2005, New York, NY, USA.* Springer, 2005, pp. 467–482.

[3] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact e-cash," in *Advances in Cryptology - EUROCRYPT 2005, Aarhus, Denmark.* Springer, 2005, pp. 302–321.

[4] H. Lipmaa, N. Asokan, and V. Niemi, "Secure vickrey auctions without threshold trust," in *Financial Cryptography, FC 2002, Southampton, Bermuda.* Springer, 2003, pp. 87–101.

[5] D. C. Parkes, M. O. Rabin, S. M. Shieber, and C. A. Thorpe, "Practical secrecy-preserving, verifiably correct and trustworthy auctions," in *Proceedings of the 8th international conference on Electronic commerce, ICEC 2006, Fredericton, New Brunswick, Canada*, 2006, pp. 70–81.

[6] M. O. Rabin, Y. Mansour, S. Muthukrishnan, and M. Yung, "Strictly-black-box zero-knowledge and efficient validation of financial transactions," in *International Colloquium on Automata, Languages, and Programming, ICALP 2012, Warwick, UK.* Springer, 2012, pp. 738–749.

[7] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *IEEE symposium on security and privacy, S&P 2014.* IEEE, 2014, pp. 459–474.

[8] A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich, "Aggregatable subvector commitments for stateless cryptocurrencies," in *International Conference on Security and Cryptography for Networks, SCN 2020, Amalfi, Italy.* Springer, 2020, pp. 45–64.

[9] V. Daza, C. Ràfols, and A. Zacharakis, "Updateable inner product argument with logarithmic verifier and applications," in *Public-Key Cryptography, PKC 2020, Edinburgh, UK.* Springer, 2020, pp. 527–557.

[10] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *IEEE symposium on security and privacy, S&P 2018, San Francisco, CA, USA.* IEEE, 2018, pp. 315–334.

[11] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Advances in Cryptology, EUROCRYPT 2000, Bruges, Belgium.* Springer, 2000, pp. 431–444.

[12] G. Couteau, M. Klooß, H. Lin, and M. Reichle, "Efficient range proofs with transparent setup from bounded integer commitments," in *Advances in Cryptology, EUROCRYPT 2021, Zagreb, Croatia.* Springer, 2021, pp. 247–277.

[13] G. Couteau, D. Goudarzi, M. Klooß, and M. Reichle, "Sharp: Short relaxed range proofs," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA*, 2022, pp. 609–622.

[14] T. Attema and R. Cramer, "Compressed-protocol theory and practical application to plug & play secure algorithmics," in *Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA.* Springer, 2020, pp. 513–543.

[15] A. Gabizon and Z. J. Williamson, "Proposal: The turbo-plonk program syntax for specifying snark programs," 2020. [Online]. Available: https://docs.zkproof.org/pages/standards/accepted-workshop3/proposal-turbo_plonk.pdf

[16] L. Pearson, J. Fitzgerald, H. Masip, M. Bellés-Muñoz, and J. L. Muñoz-Tapia, "Plonkup: Reconciling plonk with plookup," *Cryptology ePrint Archive*, vol. 2022, p. 86, 2022.

[17] B. Bünz, B. Fisch, and A. Szepieniec, "Transparent snarks from dark compilers," in *Advances in Cryptology, EUROCRYPT 2020, Zagreb, Croatia.* Springer, 2020, pp. 677–706.

[18] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Advances in Cryptology, ASIACRYPT 2010, Singapore.* Springer, 2010, pp. 177–194.

[19] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Fast reed-solomon interactive oracle proofs of proximity," in *international colloquium on automata, languages, and programming, ICALP 2018, Prague, Czech Republic*, vol. 14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018, pp. 1–17.

[20] E. Ben-Sasson, A. Chiesa, and N. Spooner, "Interactive oracle proofs," in *Theory of Cryptography, TCC 2016, Beijing, China.* Springer, 2016, pp. 31–60.

[21] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology, CRYPTO 1986, Santa Barbara, California, USA.* Springer, 1986, pp. 186–194.

[22] D. Catalano and D. Fiore, "Vector commitments and their applications," in *Public-Key Cryptography, PKC 2013, Nara, Japan.* Springer, 2013, pp. 55–72.

[23] S. Gorbunov, L. Reyzin, H. Wee, and Z. Zhang, "Pointproofs: Aggregating proofs for multiple vector commitments," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS 2020, Virtual Event, USA*, 2020, pp. 2007–2023.

[24] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward, "Marlin: Preprocessing zksnarks with universal and updatable srs," in *Advances in Cryptology, EUROCRYPT 2020, Zagreb, Croatia.* Springer, 2020, pp. 738–768.

[25] S. Setty, "Spartan: Efficient and general-purpose zksnarks without trusted setup," in *Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA.* Springer, 2020, pp. 704–737.

[26] C. Papamanthou, E. Shi, and R. Tamassia, "Signatures of correct computation," in *Theory of Cryptography Conference, TCC 2013, Tokyo, Japan.* Springer, 2013, pp. 222–242.

[27] B. Chen, B. Bünz, D. Boneh, and Z. Zhang, "Hyperplonk: Plonk with linear-time prover and high-degree custom gates," in *Advances in Cryptology, EUROCRYPT 2023, Lyon, France.* Springer, 2023, pp. 499–530.

[28] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward, "Aurora: Transparent succinct arguments for r1cs," in *Advances in Cryptology, EUROCRYPT 2019, Darmstadt, Germany.* Springer, 2019, pp. 103–128.

[29] A. Kattis, K. Panarin, and A. Vlasov, "Redshift: Transparent snarks from list polynomial commitment iops," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1400, 2019.

[30] J. Zhang, T. Xie, Y. Zhang, and D. Song, "Transparent polynomial delegation and its applications to zero knowledge proof," in *IEEE Symposium on Security and Privacy, S&P 2020, San Francisco, CA, USA.* IEEE, 2020, pp. 859–876.

[31] Z. Zhang, W. Li, Y. Guo, K. Shi, S. S. Chow, X. Liu, and J. Dong, "Fast rs-iop multivariate polynomial commitments and verifiable secret sharing," in *USENIX Security 2024, Philadelphia, PA, USA*, 2024.

A PROOFS OF LEMMAS

A.1 Proof of Lemma 3.2

Proof: $\Rightarrow$: Dividing $\mathfrak{f}(X)$ by $z_{\mathbb{H}_l}(X)$ allows us to write $f(X) = \mathfrak{q}(X) \cdot z_{\mathbb{H}_l}(X) + \mathfrak{d}(X)$, where $\mathfrak{q}(X)$ is the quotient polynomial and $\mathfrak{d}(X)$ is the remainder polynomial of degree less than l. Denote that $\mathfrak{d}(X) = \sum_{i=0}^{l-1} d_i X^i$. By the Fact 3.1, we have:

$$\begin{aligned}\sum_{a\in\mathbb{H}_l} \mathfrak{d}(a) &= \sum_{a\in\mathbb{H}_l} \sum_{i=0}^{l-1} d_i a^i \\ &= \sum_{i=1}^{l-1} \sum_{a\in\mathbb{H}_l} d_i a^i + \sum_{a\in\mathbb{H}_l} d_0 a^0 \\ &= l \cdot d_0\end{aligned}$$

So $\sum_{a\in\mathbb{H}_l} \mathfrak{f}(a) = \sum_{a\in\mathbb{H}_l} \mathfrak{d}(a) + \sum_{a\in\mathbb{H}_l} \mathfrak{q}(a) z_{\mathbb{H}_l}(a) = l \cdot d_0 + 0$. Actually, $v = \sum_{a\in\mathbb{H}_l} \mathfrak{f}(a) = l \cdot d_0$.

So we can set $\mathfrak{u}(X) = \frac{\mathfrak{d}(X) - d_0}{X} \in \mathbb{F}_{d(X)<l-2}[X]$, such that $\mathfrak{f}(X) = \frac{v}{l} + X \cdot \mathfrak{u}(X) + \mathfrak{q}(X) z_{\mathbb{H}_l}(X)$.

$\Leftarrow$: Denote the polynomial $\mathfrak{u}(X) \in \mathbb{F}_{d(X)<l-1}[X] = \sum_{i=0}^{l-2} u_i X^i$, then:

$$\begin{aligned}\sum_{a\in\mathbb{H}_l} \mathfrak{f}(a) &= \sum_{a\in\mathbb{H}_l} (\frac{v}{l} + a \cdot \mathfrak{u}(a) + \mathfrak{q}(a) z_{\mathbb{H}_l}(a)) \\ &= v + \sum_{i=0}^{l-2} u_i \sum_{a\in\mathbb{H}_l} a^{i+1} + 0 \\ &= v\end{aligned}$$

□

B WHOLE PROTOCOL OF MISSILEPROOF

C PROPERTIES OF PCS SCHEMES

- **Hiding**. For all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:

$$\left| \Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathbf{D}); \\ (\mathfrak{m}_0, \mathfrak{m}_1, st) \leftarrow \mathcal{A}_0(\mathsf{pp}); \\ b \overset{\$}{\leftarrow} \{0,1\}; \\ (C, S) \leftarrow \mathsf{Commit}(\mathsf{pp}; \mathfrak{m}_b); b' \leftarrow \mathcal{A}_1(\mathsf{pp}, C, \mathfrak{m}_0, \mathfrak{m}_1, st) : \\ b' = b \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

- **Binding**. For any PPT adversary $\mathcal{A}$, and any bivariate polynomial $\mathfrak{m} \in \mathbb{F}_{d(x)<D_x, d(y)<D_y}[X, Y]$,

$$\Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathbf{D}); (C, \mathfrak{m}_0, \mathfrak{m}_1, S_0, S_1) \leftarrow \mathcal{A}(\mathsf{pp}); \\ 1 \leftarrow \mathsf{Open}(\mathsf{pp}, C, \mathfrak{m}_0, S_0) \wedge 1 \leftarrow \mathsf{Open}(\mathsf{pp}, C, \mathfrak{m}_1, S_1) \\ \wedge \mathfrak{m}_0 \neq \mathfrak{m}_1 \end{array} \right] \leq \mathsf{negl}(\lambda).$$

- **Knowledge soundness**. Eval is a public-coin succinct interactive argument of knowledge which has knowledge soundness for the following relation given $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathbf{D})$:

$$\begin{aligned}\mathcal{R}_{\mathsf{Eval}}(\mathsf{pp}) = \{\langle (C, (\tau_x, \tau_y), v), (\mathfrak{m}, S) \rangle : \mathfrak{m} \in \mathbb{F}[X, Y] \\ \wedge\ \mathfrak{m}(\tau_x, \tau_y) = v \wedge \mathsf{Open}(\mathsf{pp}, C, \mathfrak{m}, S) = 1\}\end{aligned}$$

- **Zero knowledge**. Eval is a public-coin succinct interactive argument of knowledge with zero-knowledge for the relation $\mathcal{R}_{\mathsf{Eval}}$ given $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathbf{D})$.

Remark. The definition provided above is for bivariate polynomials. It can be simply converted to the definition for univariate polynomial commitment schemes. We won't elaborate on this here.

The `MulRangeProof` PIOP

input of $\mathcal{P}$: $(\mathsf{pp}, \mathbb{F}, \mathbb{H}_l, \mathbb{H}_n, \mathfrak{f}(X), \mathfrak{p}(Y), \mathfrak{r}(R, Y), \mathbf{v})$
input of $\mathcal{V}$: $(\mathsf{pp}, \mathbb{F}, \mathbb{H}_l, \mathbb{H}_n, \mathfrak{f}^{\mathcal{O}}, \mathfrak{p}^{\mathcal{O}}, \mathfrak{r}(R, Y))$

Phase 1: generate a decomposition bivariate polynomial

$\mathcal{P}$: for $\forall i \in [0, l)$, split v_i into a binary vector $(v_{i,0}, ..., v_{i,n-1})$.
  generate a matrix $M \in \mathbb{F}^{l \times n}, M_{i,j} = v_{i,j} \cdot 2^j$.
  compute $\mathfrak{m}(X, Y) \in \mathbb{F}_{d(X)<l, d(Y)<n}[X, Y]$, s.t. $\forall a_i \in \mathbb{H}_l, b_j \in \mathbb{H}_n, \mathfrak{m}(a_i, b_j) = M_{i,j}$.
$\mathcal{P} \rightarrow \mathcal{V}$: $\{\mathfrak{m}^{\mathcal{O}}\}$.

Phase 2: prove that $\sum_{b \in \mathbb{H}_n} \mathfrak{m}(X, b) = \mathfrak{f}(X)$

$\mathcal{P}$: compute $\mathfrak{u}(X, Y) = \frac{\mathfrak{m}(X,Y) - \frac{1}{n}\mathfrak{f}(X)}{Y}$.
$\mathcal{P} \rightarrow \mathcal{V}$: $\{\mathfrak{u}^{\mathcal{O}}\}$.
$\mathcal{V}$: check if $\mathfrak{u}(X, Y) \in \mathbb{F}_{d(X)<l, d(Y)<n-1}[X, Y]$.
  $\xi_x, \xi_y \xleftarrow{\$} \mathbb{F}$.
  query oracles $\mathfrak{m}^{\mathcal{O}}, \mathfrak{u}^{\mathcal{O}}$ and $\mathfrak{f}^{\mathcal{O}}$ to get $\mu_{\mathfrak{m}} = \mathfrak{m}(\xi_x, \xi_y), \mu_{\mathfrak{u}} = \mathfrak{u}(\xi_x, \xi_y), \mu_{\mathfrak{f}} = \mathfrak{f}(\xi_x)$.
  check if $\mu_{\mathfrak{m}} \overset{?}{=} \xi_y \cdot \mu_{\mathfrak{u}} + \frac{1}{n}\mu_{\mathfrak{f}}$.

Phase 3: prove that $\forall a \in \mathbb{H}_l, b \in \mathbb{H}_n, \mathfrak{m}(a, b)(\mathfrak{m}(a, b) - \mathfrak{p}(b)) = 0$

$\mathcal{P}$: compute $\mathfrak{M}(X, b) = \mathfrak{m}(X, b)(\mathfrak{m}(X, b) - \mathfrak{p}(b))$
  compute $\mathfrak{M}^*(R, X) = \sum_{b \in \mathbb{H}_n} \mathfrak{M}(X, b)\mathfrak{r}(R, b)$
    // then $\mathcal{P}$ wants to prove that $\forall a \in \mathbb{H}_l, \mathfrak{M}^*(R, a) = 0$.
$\mathcal{V}$: $\tau_r \xleftarrow{\$} \mathbb{F}$.
$\mathcal{V} \rightarrow \mathcal{P}$: $\{\tau_r\}$.
$\mathcal{P}$: compute $\mathfrak{e}(X) = \frac{\mathfrak{M}^*(\tau_r, X)}{z_{\mathbb{H}_l}(X)}$.
$\mathcal{P} \rightarrow \mathcal{V}$: $\{\mathfrak{e}^{\mathcal{O}}\}$.
$\mathcal{V}$: $\tau_x \xleftarrow{\$} \mathbb{F}$.
$\mathcal{V} \rightarrow \mathcal{P}$: $\{\tau_x\}$.
$\mathcal{P}$: compute $\mu_{\mathfrak{M}^*} = \mathfrak{M}^*(\tau_r, \tau_x)$.
$\mathcal{P} \rightarrow \mathcal{V}$: $\{\mu_{\mathfrak{M}^*}\}$.
$\mathcal{V}$: query oracle $\mathfrak{e}^{\mathcal{O}}$ to get $\mu_{\mathfrak{e}} = \mathfrak{e}(\tau_x)$.
  check if $\mu_{\mathfrak{e}} \cdot z_{\mathbb{H}_l}(\tau_x) \overset{?}{=} \mu_{\mathfrak{M}^*}$.
    // then $\mathcal{P}$ wants to prove that $\mu_{\mathfrak{M}^*} = \sum_{b \in \mathbb{H}_n} \mathfrak{M}(\tau_x, b)\mathfrak{r}(\tau_r, b)$.

SumCheck protocol: prove that $\mu_{\mathfrak{M}^*} = \sum_{b \in \mathbb{H}_n} \mathfrak{M}(\tau_x, b)\mathfrak{r}(\tau_r, b)$

$\mathcal{P}$: compute $\mathfrak{q}(Y), \mathfrak{g}(Y) \in \mathbb{F}_{d(Y) \le n-2}[Y]$, s.t. $\mathfrak{M}(\tau_x, Y)\mathfrak{r}(\tau_r, Y) = \mathfrak{q}(Y)z_{\mathbb{H}_n}(Y) + Y\mathfrak{g}(Y) + \frac{\mu_{\mathfrak{M}^*}}{n}$.
$\mathcal{P} \rightarrow \mathcal{V}$: $\{\mathfrak{q}^{\mathcal{O}}, \mathfrak{g}^{\mathcal{O}}\}$.
$\mathcal{V}$: check if $\mathfrak{g}(Y) \in \mathbb{F}_{d(Y)<n-1}[Y]$
  $\tau_y \xleftarrow{\$} \mathbb{F}$.
  query oracles $\mathfrak{m}^{\mathcal{O}}, \mathfrak{q}^{\mathcal{O}}, \mathfrak{p}^{\mathcal{O}}, \mathfrak{g}^{\mathcal{O}}$ to get $\mu_{\mathfrak{m}} = \mathfrak{m}(\tau_x, \tau_y), \mu_{\mathfrak{q}} = \mathfrak{q}(\tau_y), \mu_{\mathfrak{p}} = \mathfrak{p}(\tau_y)$ and $\mu_{\mathfrak{g}} = \mathfrak{g}(\tau_y)$.
  check if $\mu_{\mathfrak{m}}(\mu_{\mathfrak{m}} - \mu_{\mathfrak{p}})\mathfrak{r}(\tau_r, \tau_y) \overset{?}{=} \mu_{\mathfrak{q}} z_{\mathbb{H}_n}(\tau_y) + \tau_y \mu_{\mathfrak{g}} + \frac{\mu_{\mathfrak{M}^*}}{n}$.

Figure 6: MissileProof PIOP for $\mathcal{R}_{\mathrm{VCRP}_{\mathrm{PIOP}}}$