

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Rarus: A Succinct and Efficient Range Proof for Polynomial-based Vector Commitment

Xinyang Yang, Wenjie Qu, Yanpei Guo, and Jiaheng Zhang, *National University of Singapore*

https://www.usenix.org/conference/usenixsecurity26/presentation/yang-xinyang

This paper is included in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

ARTIFACT EVALUATED
USENIX
AVAILABLE

Rarus: A Succinct and Efficient Range Proof for Polynomial-based Vector Commitment

Xinyang Yang[*], Wenjie Qu[*], Yanpei Guo, Jiaheng Zhang
National University of Singapore

Abstract

Range proofs enable a prover to convince a verifier that a committed value lies within a specific interval without revealing additional information. They are fundamental to privacy-preserving systems including anonymous credentials, e-voting, e-cash, and cryptocurrencies like Monero and Grin. A critical challenge is efficiently proving that multiple committed values simultaneously satisfy range constraints while minimizing communication overhead.

Vector commitment schemes provide a promising approach to this problem. Missileproof (CCS'24) recently proposed a range proof for vector commitments achieving $O(1)$ proof size and verifier time, but with prover complexity of $O(N\ell\log(N\ell))$ for proving ℓ values in $[0,2^N)$.

We present **Rarus**, an efficient range proof for polynomial-based vector commitments that achieves optimal asymptotic complexity across all metrics. Our key innovation is replacing binary decomposition with optimized b-ary decomposition, coupled with Bi-variate Zero-Test and accelerated Uni-variate Sum-Check protocols. Rarus achieves $O(1)$ proof size and verifier time, while reducing prover time to $O\left(\frac{N\ell}{\log(N\ell)}\right)\mathbb{G}+O(N\ell)\mathbb{F}$, where $\mathbb{G}$ and $\mathbb{F}$ denote group and field operations respectively. In addition, our protocol supports arbitrary ranges $[0,R)$ beyond powers of two. Experimental results demonstrate that Rarus achieves a $20\times$ speedup over both Bulletproofs and Missileproof when proving 16,384 values in $[0,2^{64})$.

1 Introduction

A zero-knowledge proof (ZKP) protocol allows a prover to prove the truth of a statement without disclosing any secret information. Zero-knowledge proofs are crucial building blocks for various security-critical applications, e.g., confidential transactions, signature schemes, and anonymous credential systems.

[1]Equal contribution.

Zero-knowledge range proofs are a family of ZKPs that prove a committed value falls within a specified range. They serve as a core building block in numerous applications, such as anonymous credentials [9], e-voting [18], e-cash [6], electronic auctions [23, 26, 31], machine learning [19, 27, 28, 30], and cryptocurrencies like Monero, Beam, and Grin. In decentralized anonymous payment systems such as Monero and ZCash [32], range proofs are used to prove that the sender's balance is greater than the transferred amount. In privacy-preserving regulatory scenarios, financial institutions may need to prove that user deposits lie within legally permitted bounds, without revealing the exact balances. In such settings, institutions submit commitments to user balances along with range proofs to demonstrate compliance. This form of privacy-preserving auditability has been explored in prior work on anonymous yet regulator-verifiable financial systems [14, 35].

1.1 The Quest for Efficiency: A Rich Landscape

Given their critical role, the construction of efficient range proofs has been a central pursuit in cryptography for decades. The landscape has evolved through several generations:

- **Bit-decomposition Schemes:** Early works [3] and the widely adopted Bulletproofs [4] rely on decomposing values into binary representations. While Bulletproofs achieved a breakthrough in proof size, their linear verification time ($O(N)\mathbb{G}$ for one element) poses a scalability barrier for resource-constrained verifiers.
- **Signature-based & Square-decomposition:** Alternatives like Camenisch et al. [5] and Sharp [11] explored different algebraic structures to optimize specific dimensions, yet often struggled to balance prover costs with verification efficiency. In addition, achieving both constant verifier time and constant proof size remains a challenge.
- **Lookup-based Arguments:** Recently, lookup argu-

Protocols	Proof size	Prover time complexity	Verifier time complexity	Setup
BulletProofs [4]	$O(\log(N\ell))\mathbb{G}+O(1)\mathbb{F}$	$O(N\ell)\mathbb{G}+O(N\ell)\mathbb{F}$	$O(N\ell)\mathbb{G}+O(N\ell)\mathbb{F}$	Transparent
Plookup [16]	$O(1)\mathbb{G}+O(1)\mathbb{F}$	$O(\ell+2^N)\mathbb{G}+O((\ell+2^N)\cdot 2^N\log(\ell+2^N))\mathbb{F}$	$O(1)\mathbb{G}+O(1)\mathbb{F}$	Trusted
Logup [20]	$O(1)\mathbb{G}+O(1)\mathbb{F}$	$O(\ell+2^N)\mathbb{G}+O((\ell+2^N)\log(\ell+2^N))\mathbb{F}$	$O(1)\mathbb{G}+O(1)\mathbb{F}$	Trusted
Lasso [34]	$O(\log\ell)\mathbb{G}$	$O(\ell+poly(2^N))\mathbb{G}+O(\ell+2^N)\mathbb{F}$	$O(\log\ell)\mathbb{F}$	Transparent
[Libert23] [22](Asiacrypt'23)	$O(1)\mathbb{G}$	$O(N\ell)\mathbb{G}$	$O(N\ell)\mathbb{G}$	Trusted
SwiftRange [36](S&P'24)	$O(\log(N\ell))\mathbb{G}+O(1)\mathbb{F}$	$O(N\ell)\mathbb{G}$	$O(N\ell)\mathbb{G}$	Trusted
Missileproof [17](CCS'24)	$O(1)\mathbb{G}+O(1)\mathbb{F}$	$O(N\ell)\mathbb{G}+O(N\ell\log(N\ell))\mathbb{F}$	$O(1)\mathbb{G}+O(1)\mathbb{F}$	Trusted
This work	$O(1)\mathbb{G}+O(1)\mathbb{F}$	$O(\frac{N\ell}{\log(N\ell)})\mathbb{G}+O(N\ell)\mathbb{F}$	$O(1)\mathbb{G}+O(1)\mathbb{F}$	Trusted

Table 1: Comparison among different applicable schemes in range proof for multiple elements.
Remark: The notation $\mathbb{G}$ is used to denote the cost of one group operation in time complexity and the size of one group element in the proof size. The notation $\mathbb{F}$ is used to denote the cost of one field operation in time complexity and the size of one field element in the proof size. Note that standard lookup arguments (e.g., Plookup, LogUp) evaluated here are not natively zero-knowledge, whereas Rarus achieves full zero-knowledge, making our performance comparison conservative.

ments (e.g., Plookup [16], LogUp [20], Lasso [34], and CQ [13]) have emerged as a powerful tool for range checks. They can easily achieve both constant verifier time and constant proof size. While highly efficient protocols like CQ significantly improve prover performance on smaller domains, lookup schemes generally require substantial preprocessing or storage overhead for large tables. This makes them resource-intensive and prone to memory exhaustion for certain large-scale or client-side applications.

Despite this extensive body of work, a "trilemma" persists: existing schemes struggle to simultaneously achieve (1) compact proofs, (2) constant-time verification, and (3) a lightweight prover.

1.2 The Bottleneck in Polynomial-based Vector Commitments

To address the verification bottleneck, some works have shifted towards *Vector Commitments* (VCs) [2,7], which allow proving properties over a vector of ℓ values with constant overhead. Specifically, *Polynomial-based Vector Commitments* (PBVC), constructed from polynomial commitment schemes like KZG [21], are particularly promising due to their algebraic properties (e.g., aggregation).

In this setting, the problem is defined as follows:

Let $\mathbb{F}$ be a finite field. Consider a uni-variate polynomial $f(X)=\sum_{i=0}^{\ell-1} a_iX^i \in \mathbb{F}_{d(X)<\ell}[X]$ that is committed using a **PBVC**. The prover needs to perform a range proof to prove that all the parameters $a_i, i=0,\ldots,\ell-1$ are in the range of $[0,R)$. So, the goal is to construct a protocol that proves the following relation $\mathcal{R}_{\text{RP}}$:

$$\{(C\in\mathbb{G};\mathbf{a}\in\mathbb{F}^\ell) : \forall i\in[0,\ell), C=\mathbf{PBVC}(\mathbf{a})\wedge a_i\in[0,R)\}$$

However, this problem has not been fully explored. The state-of-the-art in this domain, Missileproof (CCS '24) [17], successfully achieved optimal proof size and verification time of $O(1)$. However, as shown in Table 1, this efficiency comes at a steep price: a prover complexity of $O(N\ell)\mathbb{G}+O(N\ell\log(N\ell))\mathbb{F}$. For large-scale systems (e.g., stateless clients or high-frequency auditing), this prover overhead becomes the limiting factor. The gap is clear: **can we retain the $O(1)$ verification and proof size while drastically reducing the prover's computational burden?**

1.3 Our Approach: Rarus

In this work, we propose **Rarus**, a succinct and efficient range proof tailored for polynomial-based vector commitments. Rarus solves the prover bottleneck by fundamentally rethinking the decomposition strategy.

The Insight. Prior works suffer from the rigidity of binary decomposition, which leads to high-degree constraints. Rarus departs from this by adopting a *b-ary decomposition* combined with a framework called *Sumset-Representation* [8]. As illustrated in Figure 1, this allows us to decompose values into a larger base b, reducing the decomposition depth. While this usually complicates constraints, we introduce a framework of Bi-variate Zero-Test PIOP (Polynomial Interactive Oracle Proof) to handle the resulting bi-variate polynomials efficiently. Also, we optimize the existing Uni-variate SumCheck PIOP. These allow us to "fold" the verification complexity, replacing the logarithmic rounds of standard protocols with constant-round verification.

Contributions. Rarus represents a significant step forward in the efficiency of multiple elements range proofs:

- **Optimal Asymptotics & Fast Prover:** As detailed in Table 1, Rarus is the first scheme to achieve $O(1)$ verifier time and $O(1)$ proof size while reducing prover

complexity to $O(\frac{N\ell}{\log(N\ell)})\mathbb{G} + O(N\ell)\mathbb{F}$. This breaks the linear barrier for group operations.

- **Novel Technical Primitive:** We construct a Bi-variate Zero-Test PIOP framework and accelerate the standard Uni-variate Sum-Check PIOP, reducing group operations from $O(D)$ to $O(D/\log D)$, field operations from $O(D\log D)$ to $O(D\log\log D)$.
- **Concrete Performance:** We implemented Rarus and compared it against the prior works. For verifying $16,384$ ranges of $[0,2^{64})$, Rarus is **20**$\times$ **faster** than Missileproof and Bulletproofs. Compared to LogUp, it achieves a **50**$\times$ speedup, demonstrating its practicality for high-performance privacy systems.
- **Generalized Ranges:** Leveraging the method of Sumset-Representation, Rarus supports arbitrary ranges $[0,R)$, enabling precise regulatory proofs without padding.

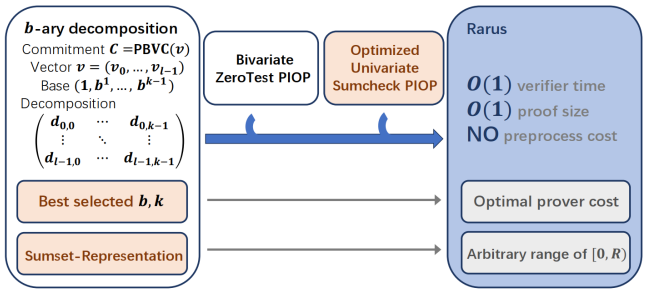


Figure 1: High-level architecture of our approach and its main technical components.

2 Preliminaries

We assume a finite field $\mathbb{F}$ is of a large prime order p and a group $\mathbb{G}$ is a multiplicative subgroup of $\mathbb{F}$. We denote a uni-variate polynomial $\mathfrak{f}$ over the field $\mathbb{F}$ by $\mathfrak{f}(X) = \sum_{i=0}^{\ell-1} a_i X^i \in \mathbb{F}_{d(X)<\ell}[X]$. Correspondingly, let the elements in $\mathbb{F}_{d(X)<\ell,d(Y)<k}[X,Y]$ denote the set of bi-variate polynomials.

As introduced in Section 1.2, the prover's goal is to construct a protocol that proves the relation $\mathcal{R}_{\text{RP}}$. In range proof problems, the range R is usually a power of 2, denoted by $R = 2^n$. However, our protocol can actually work on a general R.

2.1 Cryptographic Primitives

We rely on standard cryptographic primitives, including **Public-coin Succinct Interactive Arguments of Knowledge** [10, 33] and **Polynomial Commitment Schemes (PCS)** [21, 29]. An interactive argument system allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that a statement is true, satisfying standard properties of *completeness*, *knowledge soundness*, and *computational zero-knowledge*.

For brevity and to maintain the focus on our novel constructions, we omit the standard formal definitions and security properties (such as completeness, binding, and evaluation soundness) of these primitives, and refer interested readers to their foundational works [21, 33].

2.2 Polynomial-based Vector Commitment

Polynomial-based Vector Commitment (PBVC) is a specific instantiation of vector commitments constructed from a PCS. Let $\mathbf{a} = (a_0,\ldots,a_{\ell-1}) \in \mathbb{F}^\ell$ be the vector to be committed. We encode $\mathbf{a}$ into a uni-variate polynomial $\mathfrak{a}(X) \in \mathbb{F}[X]$ with $\deg(\mathfrak{a}) < \ell$ via Lagrange interpolation over a multiplicative subgroup $\mathbb{H}_\ell = \{1, \omega_\ell, \ldots, \omega_\ell^{\ell-1}\} \subset \mathbb{F}$, such that $\mathfrak{a}(\omega_\ell^i) = a_i$ for all $i \in [0,\ell)$.

The commitment to the vector $\mathbf{a}$ is simply the polynomial commitment $C \leftarrow \mathsf{PCS.Commit}(pp, \mathfrak{a})$. Consequently, proving that the i-th element of the vector is a_i is equivalent to running the evaluation protocol $\mathsf{PCS.Eval}(pp, C, \omega_\ell^i, a_i, \mathfrak{a})$ to show that $\mathfrak{a}(\omega_\ell^i) = a_i$.

The framework of Rarus can be applied to any generic commitment scheme. However, for our experimental evaluation and concrete complexity analysis, we instantiate it using the KZG [21] polynomial commitment scheme, and employ the PST [25] scheme (a generalized multi-variate extension of KZG) for committing to bi-variate polynomials.

2.3 Implement Sumset-Representation

The technique of bit-decomposition is frequently used in range proof protocols as Bulletproofs [4] employs a bit-decomposition on the target parameter which needs to be proved. Therefore, a basic idea of improvement is to employ a b-ary bit-decomposition instead of the binary bit-decomposition.

Concretely, we mainly focus on proving one single parameter a in a range up to $b^k - 1$. That is, $a \in [0, b^k)$, where b, k are some special positive integers regarded as the base and the bit-length of bit-decomposition, respectively. This means that Prover will decompose a into a vector $(d_0,\ldots,d_{k-1})$ such that $a = \sum_{i=0}^{k-1} b^i d_i$. Then the prover proves the range constraint of d that $\forall i = 1,\ldots,k, d_i \in [0,b)$.

However, the limitation of this method is that it cannot work for a general range up to a general R. So we need to implement some more general basis $G_i, i = 0,\ldots,k-1$, denoted by the vector $\mathbf{G}$ instead of $\mathbf{b}^k$. To achieve this, the method we use here is called Sumset-Representation, introduced by [8].

The intuition of this method can be described as follows. Let $R_0 = R - 1$. Then clearly we can decompose the range $[0, R_0] = G_0 \cdot [0, b-1] + [0, R_1]$, where we introduce $G_0 := \lfloor (R_0+1)/b \rfloor$ and $R_1 = R_0 - (b-1)G_0$. Then the formulas reduce the case $[0, R_0]$ to a smaller case $[0, R_1]$ that can

be solved similarly. Recursively, $[0,R_0] = [0,b-1]\sum_j G_j + [0,R']$, where $G_j := \lfloor \frac{R_j+1}{b} \rfloor$ and $R_{j+1} := R_j - (b-1)G_j = R_j - (b-1)\lfloor \frac{R_j+1}{b} \rfloor$.

This process stops when the interval $[0,R_{j+1}]$ is small enough so that it cannot be covered by b different non-empty intervals, that is, if $R_{j+1} \leq b-1$. Then we define $k(b,R_0) := j+1$ to be the number of steps in this recursive process. Clearly, after we are done with the recursive process, we have $R' := R_{j+1} = R_0 - \lfloor \frac{R_0}{b-1} \rfloor \cdot (b-1)$, which means in particular that if $(b-1)|R_0$ then $R' = 0$.

Finally, the sequence $(\ldots, G_j, \ldots)$ clearly decreases the slowest when for all $j, b|(R_{j+1}+1)$, since then the floor operation is not applied. But $b|(R_{j+1}+1)$ iff $b|(R_j-(b-1)G_j+1)$ iff $b|(R_j+G_j+1)$ iff (because also $b|(R_j+1)$) $b|G_j$. Thus, the sequence is the slowest to decrease if $R = b^k$ for some k. This means that the process is guaranteed to stop in $k(b,R_0) \leq \log_b(R_0+1)$ steps.

This leads us to the following theorem.

Theorem 1. *Let $b \geq 2$, $R_0 \geq b$. Let G_j, R_j and R' be defined as above. Denote $k(b,R_0) = k \leq \lceil \log_b(R_0+1) \rceil$ as above. Then $[0,R_0] = \sum_{j=0}^{k-1} G_j * [0,b-1] + [0,R']$. If $(b-1)|R_0$, then $R' = 0$.*

Proof. Given the recursive structure of the decomposition steps described above, the theorem naturally follows. □

After that, the prover can decompose a into a vector $(d_0,\ldots,d_{k-1})$ such that $a = \sum_{i=0}^{k-1} G_i d_i$. To prove a is in the range of $[0,R)$, it remains to prove that each $d_i, i = 0,\ldots,k-1$ is in the range of $[0,b)$.

2.4 The Log-derivative Method

The method we crucially use to prove the range of each $d_i, i = 0,\ldots,k-1$ is called the Log-derivative Method, introduced by [20]. The core of the method is the Derivative Verification Lemma, described as follows.

Lemma 1. *Given $b \in \mathbb{F}, d \in \mathbb{F}^k$, we have every d_i is in the range of $0 \leq d_i \leq b-1$ if and only if for some $m \in \mathbb{F}^b$, the following identity of rational functions holds*

$$\sum_{i=0}^{k-1} \frac{1}{X+d_i} = \sum_{j=0}^{b-1} \frac{m_j}{X+j}. \tag{1}$$

2.5 PIOP Gadgets

In the construction of our protocol, we need to use some existing Polynomial Interactive Oracle Proof (PIOP) gadgets. For reasons of length, we have placed the Uni-variate Zero-Test PIOP and the Uni-variate Sum-Check PIOP [1] in the Appendix A and B.

3 Technical overview

As defined in Section 2, we aim to construct a new range proof protocol that is based on polynomial-based vector commitment for proving the relation $\mathcal{R}_{RP}$. To commit the input vector $\mathbf{a}$, we require a trusted setup that provides the crs and the subgroup $\mathbb{H}_\ell$ to both the prover and the verifier. Here $\mathbb{H}_\ell$ is a multiplicative subgroup of $\mathbb{F}$ with order ℓ and generator ω_ℓ. The prover receives the commitment of $\mathbf{a}$. We assume the prover has already obtained a polynomial $\mathfrak{a}$ such that $\mathfrak{a}(Y) \in \mathbb{F}_{d(Y)<\ell}[Y]$, and $\forall \omega_\ell^i \in \mathbb{H}_\ell, \mathfrak{a}(\omega_\ell^i) = a_i$, as the interpolation can be preprocessed.

In our protocol, the prover decomposes all coefficients in $\mathbf{a}$ using the Sumset-Representation method introduced in Section 2, and commits to the results for subsequent verification. To build intuition, unlike standard b-ary decomposition which often requires dummy padding to match non-power-of-two bounds, the Sumset-Representation dynamically computes a tailored basis $\mathbf{G}$ (e.g., to bound $a_i \in [0,10)$ with base $b=3$, it derives a tight basis $\mathbf{G} = (4,2,1)$). This ensures the maximum combination perfectly bounds R without overhead. We construct two subgroups $\mathbb{H}_b$ and $\mathbb{H}_k$ in the same form. Here, b and k are the parameters in the Sumset-Representation. $\mathbb{H}_b$ is a multiplicative subgroup of $\mathbb{F}$ with order b and generator ω_b, while $\mathbb{H}_k$ is a multiplicative subgroup of $\mathbb{F}$ with order k and generator ω_k. Both parties compute the basis $\mathbf{G}$ in the Sumset-Representation, which can also be preprocessed.

The core of our protocol lies in proving four structural and boundary equations. At a high level, Equation 1 ensures the arithmetic correctness of the decomposition, while Equation 2 through Equation 4 collectively utilize the log-derivative method to assert that every decomposed digit $d_{i,j}$ strictly falls within the allowed base interval $[0,b)$. Before presenting them mathematically, we clarify the notation.

The prover generates the decomposition of a_i for $i = 0,\ldots,\ell-1$. Specifically, for each i, $\mathcal{P}$ splits a_i into a vector $\mathbf{d}_i = (d_{i,0},\ldots,d_{i,k-1})$ such that $a_i = \sum_{j=0}^{k-1} G_j d_{i,j}$. We denote the combination of all $\mathbf{d}_i$ for $i = 0,\ldots,\ell-1$ as the matrix $\mathbf{D}$. For $i = 0,\ldots,b-1$, let m_i denote the number of times i appears in the matrix $\mathbf{D}$. The verifier's first random challenge is denoted by α. For $i = 0,\ldots,\ell-1$ and $j = 0,\ldots,k-1$, let $t_{i,j} = \frac{1}{\alpha+d_{i,j}}$ and $e_i = \frac{m_i}{\alpha+i}$. The four main equations are as follows:

Equation 1: $$\forall i \in [0,\ell), a_i = \sum_{j=0}^{k-1} G_j d_{i,j} \tag{2}$$

Equation 2: $$\forall i \in [0,\ell), j \in [0,k), t_{i,j}(\alpha + d_{i,j}) - 1 = 0 \tag{3}$$

Equation 3: $$\forall i \in [0,b), e_i(\alpha+i) - m_i = 0 \tag{4}$$

Equation 4: $$\sum_{i=0}^{\ell-1}\sum_{j=0}^{k-1} t_{i,j} = \sum_{i=0}^{b-1} e_i \tag{5}$$

The first equation proves the correctness of the decom-

position, while the last three equations prove the range of $d_{i,j}$ for $i = 0,\ldots,\ell-1$ and $j = 0,\ldots,k-1$ using the log-derivative method. Our approach performs polynomial extension on all related vectors and matrices, then applies PIOP gadgets with soundness error bounded by the Schwartz-Zippel lemma. However, we observe that directly applying the bi-variate zero-test from Pianist [24] to prove Equation 2 incurs excessive overhead. Therefore, we introduce a new Bi-variate Zero-Test PIOP framework with reduced cost. To further improve efficiency, we accelerate the existing Uni-variate Sum-Check. Additionally, we apply some techniques and the Fiat-Shamir transform to make the protocol non-interactive and zero-knowledge. These techniques are detailed in Section 4.

In Section 5, we analyze the completeness and soundness of our protocol and explain how it achieves succinctness.

To optimize efficiency beyond succinctness, we model the time cost as a polynomial and analyze its monotonicity. We then derive optimal parameter settings to minimize this function, achieving not only optimal verifier time and proof size, but also a very small constant factor in the prover's group operation complexity. This optimization is presented in Section 5.

4 The Construction of Rarus

In this section, we will demonstrate the main steps of Rarus. Before that, we introduce the construction of Bi-variate Zero-Test PIOP and Optimized Uni-variate Sum-Check PIOP.

4.1 Bi-variate Zero-Test PIOP

Assume that $\mathbb{H}_{\ell_1}$ and $\mathbb{H}_{\ell_2}$ are subgroups of $\mathbb{F}$ of orders ℓ_1 and ℓ_2 and the generators ω_{ℓ_1} and ω_{ℓ_2}, respectively. Here we describe a PIOP proving that a bivariate polynomial $D(X,Y)$ evaluates to zero everywhere on $\mathbb{H}_{\ell_1},\mathbb{H}_{\ell_2}$. Formally, the PIOP proves the relation $\mathcal{R}_{\text{Bi-ZT}}$ described as follows.

$$\{(C\in\mathbb{G};\mathfrak{f}(X,Y)\in\mathbb{F}_{d(X)<D_1,d(Y)<D_2}[X,Y]) : \forall i\in[0,\ell_1), \\ j\in[0,\ell_2), C=\mathbf{PCS}(\mathfrak{f})\wedge\mathfrak{f}(\omega_{\ell_1}^i,\omega_{\ell_2}^j)=0\}$$

In Pianist [24], they introduces a protocol technique to perform bi-variate zero-test. The basic intuition of their method is that the prover computes two quotient bi-variate polynomials $\mathfrak{p}(X)\in\mathbb{F}[X,Y],\mathfrak{q}(X)\in\mathbb{F}[X,Y]$ such that

$$\mathfrak{f}(X,Y)=\mathfrak{p}(X,Y)Z_{\mathbb{H}_{\ell_1}}(X)+\mathfrak{q}(X,Y)Z_{\mathbb{H}_{\ell_2}}(Y) \tag{6}$$

where uni-variate polynomials $Z_{\mathbb{H}_{\ell_1}}(X)=\prod_{a\in\mathbb{H}_{\ell_1}}(X-a)$ and $Z_{\mathbb{H}_{\ell_2}}(Y)=\prod_{b\in\mathbb{H}_{\ell_2}}(Y-b)$ are called the vanishing polynomials. Then the verifier needs to check this equation with some random challenge. To check this equation at some random point, the verifier must query $\mathfrak{f}^O,\mathfrak{p}^O$ and $\mathfrak{q}^O$. This means that the prover must commit these three bi-variate polynomials and will have a cost of $3D_1D_2\mathbb{G}$.

Here we come up with a new construction of Bi-variate Zero-Test PIOP based on the high-level idea of Multi-variate Sum-check PIOP.

In Multi-variate Sum-check PIOP, in each step, the prover and the verifier use a random challenge to reduce the degree of the equation by 1. In the last step, the verifier only needs to verify an equation of a uni-variate polynomial. In our Bi-variate Zero-Test PIOP, we use a random challenge to transfer the equation to a uni-variate one. Then the prover sends the intermediate result. The verifier verifies it by a Uni-variate Sum-check PIOP and verifies the equation by a Uni-variate Zero-Test PIOP. Before this, we need to construct a group of polynomials.

Theorem 2. *Let $\mathfrak{u}(X,Y)=\sum_{i=0}^{\ell_2-1}X^iY^i$. Then we have that the polynomials $\{\mathfrak{u}(X,\omega_{\ell_2}^j)\}_{j=0}^{\ell_2-1}$ are ℓ_2 linearly independent uni-variate polynomials.*

Proof. The coefficients of the i^{th} uni-variate polynomial can be considered as the vector $(1,\omega^i,\ldots,(\omega^i)^{\ell_2-1})$. This means that we can combine the ℓ_2 vectors in a Vandermonde matrix. According to the property of the Vandermonde matrix, we know these n polynomials are linearly independent. □

At a high level, rather than proving the bi-variate relation directly (which requires heavy quotient polynomials), our approach logically "folds" the two-dimensional evaluation grid into a single dimension. By taking a random linear combination of the polynomial evaluated along one axis, we compress the problem into a standard uni-variate check, drastically reducing the prover's commitment overhead.

Lemma 2. *Let $r\xleftarrow{\$}\mathbb{F}$ be a random challenge. For multiplicative subgroups $\mathbb{H}_{\ell_1}$, $\mathbb{H}_{\ell_2}$ and the polynomial $\mathfrak{f}(X,Y)=\sum_{i=0}^{D_1-1}\sum_{j=0}^{D_2-1}f_{ij}X^iY^j\in\mathbb{F}_{d(X)<D_1,d(Y)<D_2}[X,Y]$: $\mathfrak{f}(X,Y)$ is identically zero on $\mathbb{H}_{\ell_1}$, $\mathbb{H}_{\ell_2}$ if the following equation holds:*

$$\forall i=0,\ldots,\ell_1-1,\ \sum_{j=0,\ldots,\ell_2-1}\mathfrak{f}(\omega_{\ell_1}^i,\omega_{\ell_2}^j)\mathfrak{u}(r,\omega_{\ell_2}^j)=0 \tag{7}$$

with a soundness error of at most $\frac{\ell_2}{|\mathbb{F}|}$.

Proof. We know that $\mathfrak{u}(X,\omega_{\ell_2}^i)$ is a set of ℓ_2 linearly independent uni-variate polynomials. So, $\mathfrak{u}'(X)=\sum_{j=0,\ldots,\ell_2-1}\mathfrak{f}(\omega_{\ell_1}^i,\omega_{\ell_2}^j)\mathfrak{u}(X,\omega_{\ell_2}^j)$ can not be a zero polynomial if not all $\mathfrak{f}(\omega_{\ell_1}^i,\omega_{\ell_2}^j)=0$. According to the Schwartz-Zippel Lemma, if $\sum_{j=0,\ldots,\ell_2-1}\mathfrak{f}(\omega_{\ell_1}^i,\omega_{\ell_2}^j)\mathfrak{u}(r,\omega_{\ell_2}^j)=0$, then $\mathfrak{f}(X,Y)$ is identically zero on $\mathbb{H}_{\ell_1}$, $\mathbb{H}_{\ell_2}$ with a soundness error of at most $\frac{\ell_2}{|\mathbb{F}|}$. □

With this lemma, the prover and the verifier only need to implement a protocol to prove this equation, which can be solved by applying a Uni-variate Sum-check PIOP and a Uni-variate Zero-Test PIOP. Now, we can formally describe the Bi-variate Zero-Test PIOP as follows:

1. **Phase 1: Construct the equation and verify the zero-test.** The verifier $\mathcal{V}$ generates a random challenge $r \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$. Then $\mathcal{P}$ computes the uni-variate polynomial $\bar{\mathfrak{f}}(X) = \sum_{j=0}^{\ell_2-1} \mathfrak{f}(X, \omega_{\ell_2}^j)\, \mathfrak{u}(r, \omega_{\ell_2}^j) \in \mathbb{F}_{d(X)<D_1}[X]$. $\mathcal{P}$ also computes a uni-variate polynomial $\mathfrak{p}(X)$ such that $\mathfrak{p}(X) \cdot Z_{\mathbb{H}_{\ell_1}} = \bar{\mathfrak{f}}(X)$. Then $\mathcal{P}$ commits $\mathfrak{p}(X)$ and sends the oracle $\mathfrak{p}^O$ to $\mathcal{V}$.

   $\mathcal{V}$ generates a random challenge $\tau_1 \xleftarrow{\$} \mathbb{F}$. Then $\mathcal{V}$ computes $Z_{\mathbb{H}_{\ell_1}}(\tau_1)$ itself and queries the oracle $\mathfrak{p}^O$ to get $\mathfrak{p}^O(\tau_1)$. $\mathcal{P}$ computes $\bar{\mathfrak{f}}(\tau_1)$ and sends the result f to $\mathcal{V}$. Then $\mathcal{V}$ checks the equation $f = \mathfrak{p}^O(\tau_1) \cdot Z_{\mathbb{H}_{\ell_1}}(\tau_1)$. If the equation holds, $\mathcal{V}$ outputs 1, otherwise outputs 0.

2. **Phase 2: Verify the sum-check of** $\mathfrak{f}(\tau_1, \omega_{\ell_2}^j)\mathfrak{u}(r, \omega_{\ell_2}^j) - \frac{f}{\ell_2}$**.** The prover $\mathcal{P}$ computes the uni-variate polynomials $\mathfrak{h}(Y) \in \mathbb{F}[Y]$ and $\mathfrak{g}(Y) \in \mathbb{F}_{d(Y)<\ell_2-1}[Y]$, such that $\mathfrak{f}(\tau_1, Y)\mathfrak{u}(r, Y) = \mathfrak{h}(Y) Z_{\mathbb{H}_{\ell_2}}(Y) + Y \cdot \mathfrak{g}(Y)$. Then $\mathcal{P}$ commits $\mathfrak{h}(Y), \mathfrak{g}(Y)$ and sends the oracle $\mathfrak{h}^O, \mathfrak{g}^O$ to $\mathcal{V}$. $\mathcal{V}$ generates a random challenge $\tau_2 \xleftarrow{\$} \mathbb{F}$. Calculating $\mathfrak{u}(r, \tau_2)$ using the formula for the sum of a geometric series only costs $O(1)\mathbb{F}$. So, $\mathcal{V}$ can compute $\mathfrak{u}(r, \tau_2)$, $Z_{\mathbb{H}_{\ell_2}}(\tau_2)$ itself and then query the oracle $\mathfrak{f}^O, \mathfrak{h}^O, \mathfrak{g}^O$ to get $\mathfrak{f}^O(\tau_1, \tau_2)$, $\mathfrak{h}^O(\tau_2)$, $\mathfrak{g}^O(\tau_2)$. $\mathcal{V}$ checks the equation $\mathfrak{f}(\tau_1, \tau_2)\mathfrak{u}(r, \tau_2) - \frac{f}{\ell_2} = \mathfrak{h}^O(\tau_2) \cdot Z_{\mathbb{H}_{\ell_2}}(\tau_2) + \tau_2 \cdot \mathfrak{g}^O(\tau_2)$. If the equation holds, $\mathcal{V}$ output 1, otherwise outputs 0.

Theorem 3. *The PIOP for $\mathcal{R}_{\text{Bi-ZT}}$ is perfectly complete and has a knowledge soundness error of at most $(D_1 + D_2 + \ell_2)/|\mathbb{F}|$.*

Proof. The completeness holds by the lemma above.

The random challenge r causes a soundness error of at most $\ell_2/|\mathbb{F}|$. Applying a Uni-variate Zero-Test PIOP causes a soundness error of at most $D_1/|\mathbb{F}|$. Applying a Uni-variate Sum-Check PIOP causes a soundness error of at most $D_2/|\mathbb{F}|$. According to Union Bound, soundness error in total is at most the sum of above, which is $(D_1 + D_2 + \ell_2)/|\mathbb{F}|$. □

$\mathcal{P}$ needs to commit $\mathfrak{f}(X,Y), \mathfrak{p}(X), \mathfrak{h}(X), \mathfrak{g}(X)$, which costs at most $D_1D_2\mathbb{G} + (D_1 - \ell_1)\mathbb{G} + (D_2 - 1)\mathbb{G} + (\ell_2 - 1)\mathbb{G}$. The dominant term here is $D_1D_2\mathbb{G}$. Therefore we have the following, the complexity of this PIOP.

- The prover time is $D_1D_2\mathbb{G} + O(D_1D_2)\mathbb{F}$.
- The verifier time is $O(1)\mathbb{G} + O(1)\mathbb{F}$.
- The proof size is $O(1)$.

Consider the commitment phase in the protocol in Pianist, in which the prover needs to commit $\mathfrak{f}$ and two quotient polynomials. This means that the cost of group operations is $3D_1D_2\mathbb{G}$. Therefore, our Bi-variate Zero-Test PIOP has an advantage in prover time with the optimal verifier time and the optimal proof size.

4.2 Optimized Uni-variate Sum-Check PIOP

Assume that $\mathbb{H}_\ell$ is a multiplicative subgroup of $\mathbb{F}$ of order ℓ with generator ω_ℓ. We aim to verify that the sum of evaluations of a polynomial $\mathfrak{f}(X)$ over $\mathbb{H}_\ell$ is zero. The PIOP proves the relation $\mathcal{R}_{\text{Uni-SC}}$ as originally defined:

$$\{(C \in \mathbb{G}; \mathfrak{f}(X) \in \mathbb{F}_{d(X)<D}[X]) : C = \mathbf{PCS}(\mathfrak{f}) \wedge \sum_{i=0}^{\ell-1} \mathfrak{f}(\omega_\ell^i) = 0\}$$

To achieve the optimized complexity $O(D/\log D)\mathbb{G}$, we utilize a domain isomorphism $\mathbb{H}_\ell \cong \mathbb{H}_{\ell_1} \times \mathbb{H}_{\ell_2}$ where $\ell = \ell_1 \cdot \ell_2$. Through this isomorphism, the univariate polynomial $\mathfrak{f}(X)$ is viewed as a bi-variate polynomial $\hat{\mathfrak{f}}(X,Y)$. Conceptually, this acts as a divide-and-conquer strategy. Instead of committing to a single massive polynomial of degree D, we logically reshape it into a grid of size $\ell_1 \times \ell_2$. By proving the sum along the rows (Phase 1) and then along a randomly selected column (Phase 2), we balance the computational load, lowering the dominant group operations from $O(D)$ to $O(D/\log D)$.

The protocol proceeds in two phases, utilizing the "Split-Sumcheck" technique:

- **Phase 1: Outer Sum-Check (Target Sum = 0).** $\mathcal{P}$ computes the partial sum polynomial $\mathfrak{p}(X) = \sum_{j=0}^{\ell_2-1} \hat{\mathfrak{f}}(X, \omega_{\ell_2}^j)$, which has degree $< \ell_1$. The condition $\sum_{i=0}^{\ell-1} \mathfrak{f}(\omega_\ell^i) = 0$ implies $\sum_{i=0}^{\ell_1-1} \mathfrak{p}(\omega_{\ell_1}^i) = 0$.

  $\mathcal{P}$ computes quotients $\mathfrak{q}_1(X), \mathfrak{h}_1(X)$ for $\mathfrak{p}(X)$ over $\mathbb{H}_{\ell_1}$. $\mathcal{P}$ commits to $\mathfrak{p}(X), \mathfrak{q}_1(X), \mathfrak{h}_1(X)$ and sends oracles $\mathfrak{p}^O, \mathfrak{q}_1^O, \mathfrak{h}_1^O$ to $\mathcal{V}$.

  $\mathcal{V}$ generates a random challenge $\tau_1 \xleftarrow{\$} \mathbb{F}$. $\mathcal{V}$ computes $Z_{\mathbb{H}_{\ell_1}}(\tau_1)$ and checks:

  $$\mathfrak{p}^O(\tau_1) = \mathfrak{q}_1^O(\tau_1) \cdot Z_{\mathbb{H}_{\ell_1}}(\tau_1) + \tau_1 \cdot \mathfrak{h}_1^O(\tau_1) \qquad (8)$$

- **Phase 2: Inner Sum-Check (Target Sum =** $\mathfrak{p}(\tau_1)$**).** Let $v = \mathfrak{p}^O(\tau_1)$. The claim reduces to proving $\sum_{j=0}^{\ell_2-1} \hat{\mathfrak{f}}(\tau_1, \omega_{\ell_2}^j) = v$. Define the univariate polynomial $\mathfrak{g}(Y) = \hat{\mathfrak{f}}(\tau_1, Y)$. $\mathcal{P}$ computes quotients $\mathfrak{q}_2(Y), \mathfrak{h}_2(Y)$ for $\mathfrak{g}(Y)$ over $\mathbb{H}_{\ell_2}$. $\mathcal{P}$ commits to $\mathfrak{q}_2(Y), \mathfrak{h}_2(Y)$ and sends oracles $\mathfrak{q}_2^O, \mathfrak{h}_2^O$ to $\mathcal{V}$.

  $\mathcal{V}$ generates a random challenge $\tau_2 \xleftarrow{\$} \mathbb{F}$. $\mathcal{V}$ computes $Z_{\mathbb{H}_{\ell_2}}(\tau_2)$ and checks:

  $$\mathfrak{g}^O(\tau_2) - \frac{v}{\ell_2} = \mathfrak{q}_2^O(\tau_2) \cdot Z_{\mathbb{H}_{\ell_2}}(\tau_2) + \tau_2 \cdot \mathfrak{h}_2^O(\tau_2) \qquad (9)$$

  Finally, $\mathcal{V}$ verifies that the value $\mathfrak{g}^O(\tau_2)$ is consistent with the original commitment C by invoking the PCS evaluation protocol for $\mathfrak{f}$ at the point corresponding to (τ_1, τ_2).

Theorem 4. *The PIOP for $\mathcal{R}_{\text{Uni-SC}}$ is perfectly complete and has a knowledge soundness error of approximately $(\ell_1+\ell_2)/|\mathbb{F}|$.*

Proof. Completeness is guaranteed by polynomial division. Soundness follows from the Schwartz-Zippel Lemma applied to random challenges τ_1, τ_2. □

We analyze the complexity by setting the parameters as follows: $\ell_1 = \log D$ and $\ell_2 = D/\log D$. In Phase 1, $\mathcal{P}$ commits to polynomials of degree ℓ_1, costing $O(\log D)\mathbb{G}$. In Phase 2, $\mathcal{P}$ commits to polynomials of degree ℓ_2, costing $O(D/\log D)\mathbb{G}$. The dominant cost is: $O(D/\log D)\,\mathbb{G}$. To compute the partial sum polynomial $\mathfrak{p}(X)$ (and quotients), $\mathcal{P}$ essentially performs ℓ_2 independent NTTs of size ℓ_1 (one for each "row" in the bivariate view). The cost is $\ell_2 \cdot O(\ell_1 \log \ell_1)$. Substituting parameters:$O(D \log\log D)\mathbb{F}$.

- The prover time is $O(D/\log D)\,\mathbb{G} + O(D\log\log D)\mathbb{F}$.
- The verifier time is $O(1)\mathbb{G} + O(1)\mathbb{F}$.
- The proof size is $O(1)$.

Type	Prover Work		Verifier/ Proof
	Group Ops	Field Ops	
Original	$O(D)$	$O(D\log D)$	$O(1)$
Optimized	$O(D/\log D)$	$O(D\log\log D)$	

Table 2: Comparison of Optimized Uni-variate Sum-check PIOP with the original one.

4.3 Protocol construction

As we imagined above, by Sumset-Representation, prover firstly needs to decompose all the coefficients in $\mathbf{a}$. We describe it concretely as follows.

To build intuition before diving into the algebraic details, unlike binary decomposition that restricts digits to $\{0,1\}$, Sumset-Representation allows a flexible base b. For instance, to prove $x \in [0,10)$, with $b=3$, we might use a basis $\mathbf{G} = (4,1)$ such that $x = 4d_0 + 1d_1$ where $d_0, d_1 \in [0,3)$. The prover first convinces $\mathcal{V}$ that this structural decomposition is correct (Equation 1). Then, instead of expensive boolean constraint checks, $\mathcal{P}$ applies the log-derivative method to assert globally that every decomposed coefficient $d_{i,j}$ falls strictly within $[0,b)$ (Equations 2-4).

$\mathcal{P}$ generates the decomposition of $a_i, i = 0,\ldots,\ell-1$. That is, for each $i = 0,\ldots,\ell-1$, $\mathcal{P}$ divides a_i into a vector $\mathbf{d}_i = (d_{i,0},\ldots,d_{i,k-1})$ such that $a_i = \sum_{j=0}^{k-1} G_j d_{i,j}$. The method here $\mathcal{P}$ used to do decomposition is repeated division which is similar to what to do in binary decomposition. Here we denote the combination of all the $\mathbf{d}_i, i = 0,\ldots,\ell-1$ by the matrix $\mathbf{D}$. Then $\mathcal{P}$ needs to convince $\mathcal{V}$ that the decomposition is correct, while the corresponding equation is denoted by Equation 1. In addition, $\mathcal{P}$ needs to prove that every $d_{i,j}$ is in the range of $[0,b)$. According to the log-derivative method, $\mathcal{P}$ needs to prove that $\sum_{i=0}^{\ell-1}\sum_{j=0}^{k-1} \frac{1}{X+d_{i,j}} = \sum_{i=0}^{b-1} \frac{m_i}{X+i}$, where m_i is the number of i appearing in the matrix $\mathbf{D}$.

We want to prove these two equality relations by Schwartz-Zippel Lemma later. So in advance, $\mathcal{P}$ converts some vectors into polynomial representations using interpolation methods. Specifically, $\mathcal{P}$ computes $\mathfrak{d}(Y,X) \in \mathbb{F}_{d(Y)<\ell, d(X)<k}[Y,X]$, s.t. $\forall \omega_\ell^i \in \mathbb{H}_\ell, \omega_k^j \in \mathbb{H}_k, \mathfrak{d}(\omega_\ell^i, \omega_k^j) = d_{i,j}$ to prove Equation 1 and Equation 2; $\mathfrak{c}(X) \in \mathbb{F}_{d(X)<k}[X]$, s.t. $\forall \omega_k^i \in \mathbb{H}_k, \mathfrak{c}(\omega_k^i) = G_i$ to prove Equation 1; $\mathfrak{I}(X) \in \mathbb{F}_{d(X)<b}[X]$, s.t. $\forall \omega_b^i \in \mathbb{H}_b, \mathfrak{I}(\omega_b^i) = i$ to prove Equation 3; $\mathfrak{m}(X) \in \mathbb{F}_{d(X)<b}[X]$, s.t. $\forall \omega_b^i \in \mathbb{H}_b, \mathfrak{m}(\omega_b^i) = m_i$ to prove Equation 3. To improve efficiency, the method employed here is NTT. $\mathcal{P}$ commits $\mathfrak{d}, \mathfrak{a}, \mathfrak{c}, \mathfrak{I}, \mathfrak{m}$ and sends their oracles to $\mathcal{V}$ for future use.

Then $\mathcal{V}$ sends a random challenge $\alpha \xleftarrow{\$} \mathbb{F} \setminus \{-b+1,\ldots,0\}$ to $\mathcal{P}$ for the range proof of $d_{i,j}$. Restricting the challenge space of α to exclude the valid ranges of $-d_{i,j}$ and $-i$ elegantly avoids the zero-denominator issue in the log-derivative method, achieving perfect completeness with negligible impact on soundness. According to the Schwartz-Zippel Lemma, the equations in the log-derivative method can be translated to $\sum_{i=0}^{\ell-1}\sum_{j=0}^{k-1} \frac{1}{\alpha+d_{i,j}} = \sum_{j=0}^{b-1} \frac{m_j}{\alpha+j}$. Here it needs to find a common denominator when we consider the degree of this polynomial equation. We can know that the degree of α is $b+\ell+k-1$, which means the soundness error here is $(b+\ell+k-1)/|\mathbb{F}|$.

To transfer the equation to polynomial version and further simplify, we introduce the matrix $\mathbf{T} = \{t_{i,j}\}_{i=0,\ldots,\ell-1, j=0,\ldots,k-1}$ and the vector $\mathbf{e} = \{e_i\}_{i=0,\ldots,b-1}$. Then $\mathcal{P}$ computes $\mathfrak{t}(Y,X) \in \mathbb{F}_{d(Y)<l, d(X)<k}[Y,X], s.t. \forall \omega_\ell^i \in \mathbb{H}_\ell, \omega_k^j \in \mathbb{H}_k, \mathfrak{t}(\omega_\ell^i, \omega_k^j) = t_{i,j} = \frac{1}{\alpha+d_{i,j}}$ to prove Equation 2 and Equation 4; $\mathfrak{e}(X) \in \mathbb{F}_{d(X)<b}[X], s.t. \forall \omega_b^i \in \mathbb{H}_b, \mathfrak{e}(\omega_b^i) = e_i = \frac{m_i}{\alpha+i}$ to prove Equation 3 and Equation 4. $\mathcal{P}$ commits $\mathfrak{t}, \mathfrak{e}$ and sends their oracles to $\mathcal{V}$ for future use. Then the equation can be translated to three equations as follows.

$$\mathfrak{t} \text{ equation:} \quad \forall i,j, \mathfrak{t}(\omega_\ell^i, \omega_k^j)(\alpha + \mathfrak{d}(\omega_\ell^i, \omega_k^j)) - 1 = 0. \tag{10}$$

$$\mathfrak{e} \text{ equation:} \quad \forall i, \mathfrak{e}(\omega_b^i)(\alpha + \mathfrak{I}(\omega_b^i)) - \mathfrak{m}(\omega_b^i) = 0 \tag{11}$$

$$\text{The original equation:} \quad \sum_{i=0}^{\ell-1}\sum_{j=0}^{k-1} \mathfrak{t}(\omega_\ell^i, \omega_k^j) = \sum_{i=0}^{b-1} \mathfrak{e}(\omega_b^i) \tag{12}$$

The equation corresponding to the correctness of the decomposition can be translated as follows.

$$\text{Decomposition:} \forall i, \mathfrak{a}(\omega_\ell^i) = \sum_{j=0}^{k-1} \mathfrak{c}(\omega_k^j)\mathfrak{d}(\omega_\ell^i, \omega_k^j). \tag{13}$$

These four equations above can be proved, respectively using the PIOP gadgets that we have introduced in Section 2, Section 4.1 and Section 4.2.

To verify equation 13, we should first transfer the equation to $\forall i = 0,\ldots,\ell-1, \sum_{j=0}^{k-1}[\mathfrak{c}(\omega_k^j)\mathfrak{d}(\omega_\ell^i,\omega_k^j) - \frac{\mathfrak{a}(\omega_\ell^i)}{k}] = 0$. Considering the polynomial $\sum_{j=0}^{k-1}[\mathfrak{c}(\omega_k^j)\mathfrak{d}(Y,\omega_k^j) - \frac{\mathfrak{a}(Y)}{k}]$, the equation tells us that all the ℓ elements in $\mathbb{H}_\ell$ are the roots of this polynomial. However, the polynomial has degree of at most $\ell-1$, which means that this polynomial must be a zero polynomial. That is $\sum_{j=0}^{k-1}[\mathfrak{c}(\omega_k^j)\mathfrak{d}(Y,\omega_k^j) - \frac{\mathfrak{a}(Y)}{k}] \equiv 0$. $\mathcal{V}$ can send a random challenge r_0 to check this with a soundness error of at most $(\ell-1)/|\mathbb{F}|$. Then proving this equation is equal to proving that

$$\sum_{j=0}^{k-1}[\mathfrak{c}(\omega_k^j)\mathfrak{d}(r_0,\omega_k^j) - \frac{\mathfrak{a}(r_0)}{k}] = 0 \tag{14}$$

Obviously, it is a uni-variate equation. $\mathcal{P}$ and $\mathcal{V}$ can execute an Optimized Uni-variate Sum-check PIOP to verify it.

$\mathcal{P}$ and $\mathcal{V}$ can execute a Bi-variate Zero-Test PIOP to verify equation 10.

$\mathcal{P}$ and $\mathcal{V}$ can execute a Uni-variate Zero-Test PIOP to verify equation 11.

For the verification of equation 12, we can first simplify the equation. According to the property of a group, we know that $\forall i = 1,\ldots,\ell-1, \sum_{a\in\mathbb{H}_\ell} a^i = 0$ (The same as b and k). While the degree of Y in $\mathfrak{t}(Y,X)$ is at most ℓ and the degree of X in $\mathfrak{t}(Y,X)$ is at most k, we can know that $\sum_{i=0}^{\ell-1}\sum_{j=0}^{k-1}\mathfrak{t}(\omega_\ell^i,\omega_k^j) = \sum_{i=0}^{\ell-1}\sum_{j=0}^{k-1}\mathfrak{t}(0,0) = \mathfrak{t}(0,0)\times\ell k$. Similarly, the degree of $\mathfrak{e}(X)$ is at most $b-1$, so we can know that $\sum_{i=0}^{b-1}\mathfrak{e}(\omega_b^i) = \sum_{i=0}^{b-1}\mathfrak{e}(0) = \mathfrak{e}(0)\times b$. Then the rest equation can be easily checked by the verifier.

These steps compose the whole protocol, which can prove $\mathcal{R}_{RP}$ successfully.

4.4 Adding Zero-knowledge

According to our construction, when the verifier queries these oracles, it can be regarded as that the prover opens the evaluation of the polynomial on a single position. This would reveal some slight information about the polynomial which can be regarded as the revealing of some information about the witness. We will add some masks on the polynomials to make the protocol zero-knowledge.

The polynomials that contain the information of the witness are $\mathfrak{a}$, $\mathfrak{d}$, $\mathfrak{m}$, $\mathfrak{t}$, and $\mathfrak{e}$. So we need to add a mask to each polynomial. The rule we need not to violate when adding the mask is that the equations in each PIOP gadget should still hold after the mask is added. To achieve this, we add the mask as follows:

$$\mathfrak{a}'(Y,X) = \mathfrak{a}(Y) + s_0 Z_{\mathbb{H}_\ell}(Y) Z_{\mathbb{H}_k}(X) \tag{15}$$
$$\mathfrak{d}'(Y,X) = \mathfrak{d}(Y,X) + s_0 Z_{\mathbb{H}_\ell}(Y) Z_{\mathbb{H}_k}(X) + s_1 Z_{\mathbb{H}_k}(X) \tag{16}$$
$$\mathfrak{m}'(X) = \mathfrak{m}(X) + s_0 Z_{\mathbb{H}_b}(X) \tag{17}$$
$$\mathfrak{t}'(Y,X) = \mathfrak{t}(Y,X) + s_0 Z_{\mathbb{H}_\ell}(Y) Z_{\mathbb{H}_k}(X) + s_1 Z_{\mathbb{H}_k}(X) \tag{18}$$
$$\mathfrak{e}'(X) = \mathfrak{e}(X) + \frac{\ell k(-s_0+s_1)}{b} Z_{\mathbb{H}_b}(X) \tag{19}$$

Here, the uni-variate polynomial $\mathfrak{a}(Y)$ is expanded to the bi-variate polynomial $\mathfrak{a}'(Y,X)$ to not violate equation 13.

And we replace $\mathfrak{a},\mathfrak{d},\mathfrak{m},\mathfrak{t},\mathfrak{e}$ by $\mathfrak{a}',\mathfrak{d}',\mathfrak{m}',\mathfrak{t}',\mathfrak{e}'$ when applying the PIOP gadgets.

Theorem 5. *After these modifications above, our protocol is zero-knowledge.*

Proof. It is demonstrated in the Appendix C. □

4.5 Putting everything together

While the protocol has already been succinct (this will be demonstrated in the next section), it can still be further optimized.

We can notice that the verifier needs to generate 10 random points for challenges, while some of them are mutually independent and without temporal correlation. Therefore, we can simplify the description of the protocol by using a shared random number approach. To be concrete, it needs to generate a random challenge in advance when constructing the decomposition equation, applying Bi-variate Zero-Test PIOP and applying Optimized Uni-variate Sum-Check PIOP, respectively. Other random challenges can be paralleled in 2. Therefore, we can reduce 10 random points to 5, which can reduce the verifier time and the proof size. Another advantage of this is that it will be easier for batch **PBVC.Open** when we compile the protocol using some PCS such as the KZG-based vector commitment scheme, which can further reduce the verifier time.

Now we try to connect everything mentioned above. In the description in Section 4.4, we make our protocol zero-knowledge by adding mask for the polynomials. Finally, we turn the protocol into a non-interactive argument of knowledge using Fiat-Shamir transform [15] and we can get our non-interactive zero-knowledge, succinct and efficient range proof protocol.

We demonstrate the complete protocol before using Fiat-Shamir transform. Shown in Figure 2.

5 Analytics of the protocol

In this section and in the following sections, we compile the protocol with KZG-based vector commitment scheme. Then we do analytics and experiments.

Protocol Π_{Rarus}: Overall Interaction between Prover and Verifier

Prover $\mathcal{P}$		**Verifier** $\mathcal{V}$
Public Parameters & Pre-computation		
Let R be the range size, $\mathbf{a}$ a vector of size ℓ. Subgroups $\mathbb{H}_\ell, \mathbb{H}_k, \mathbb{H}_b \subset \mathbb{F}$ with order ℓ, k, b. Basis vector $\mathbf{G}$ is derived from Sumset-Representation. $\mathcal{V}$ computes $\mathfrak{c}_0(X) \in \mathbb{F}_{d(X)<k}[X]$ s.t. $\forall \omega_k^i \in \mathbb{H}_k, \mathfrak{c}_0(\omega_k^i) = G_i$ and stores **PBVC.Commit**$(\mathfrak{c}_0)$.		
Step 1: Sumset Decomposition & Initial Commitments		
Decompose: For $i \in [0, \ell-1]$, $a_i = \sum_{j=0}^{k-1} G_j d_{i,j}$ Let $\mathbf{D} = \{d_{i,j}\}$, and m_i be the frequency of i in $\mathbf{D}$. **Interpolate via NTT:** • $\mathfrak{d}(\omega_\ell^i, \omega_k^j) = d_{i,j}, \quad \mathfrak{c}(\omega_k^i) = G_i$ • $\mathfrak{I}(\omega_b^i) = i, \quad \mathfrak{m}(\omega_b^i) = m_i$ **Mask (Zero-Knowledge):** Sample $s_0, s_1 \leftarrow \mathbb{F}$ • $\mathfrak{d}'(Y,X) = \mathfrak{d}(Y,X) + s_0 Z_{\mathbb{H}_\ell}(Y) Z_{\mathbb{H}_k}(X) + s_1 Z_{\mathbb{H}_k}(X)$ • $\mathfrak{a}'(Y,X) = \mathfrak{a}(Y) + s_0 Z_{\mathbb{H}_\ell}(Y) Z_{\mathbb{H}_k}(X)$ • $\mathfrak{m}'(X) = \mathfrak{m}(X) + s_0 Z_{\mathbb{H}_b}(X)$ $C_1 \leftarrow$ **PBVC.Commit**$(\mathfrak{d}', \mathfrak{a}', \mathfrak{c}, \mathfrak{I}, \mathfrak{m}')$	$\xrightarrow{C_1}$	
Step 2: Log-Derivative Extensions		
	$\xleftarrow{\alpha}$	**Challenge:** $\alpha \xleftarrow{\$} \mathbb{F} \setminus \{-b+1, \ldots, 0\}$
Interpolate via NTT: • $\mathfrak{t}(\omega_\ell^i, \omega_k^j) = \frac{1}{\alpha + d_{i,j}}, \quad \mathfrak{e}(\omega_b^i) = \frac{m_i}{\alpha + i}$ **Mask:** • $\mathfrak{t}'(Y,X) = \mathfrak{t}(Y,X) + s_0 Z_{\mathbb{H}_\ell}(Y) Z_{\mathbb{H}_k}(X) + s_1 Z_{\mathbb{H}_k}(X)$ • $\mathfrak{e}'(X) = \mathfrak{e}(X) + \frac{\ell k(-s_0+s_1)}{b} Z_{\mathbb{H}_b}(X)$ $C_2 \leftarrow$ **PBVC.Commit**$(\mathfrak{t}', \mathfrak{e}')$	$\xrightarrow{C_2}$	
Step 3: PIOP Executions & Final Verification		
	$\xleftarrow{r_0}$	**Challenge:** $r_0 \xleftarrow{\$} \mathbb{F}$
Joint Execution: $\mathcal{P}$ and $\mathcal{V}$ interactively execute the following PIOPs: 1. Optimized Uni-variate Sum-check on $\mathfrak{c}(X)\mathfrak{d}'(r_0,X) - \frac{\mathfrak{a}'(r_0,X)}{k}$ with $\mathbb{H}_k$ 2. Bi-variate Zero-Test on $\mathfrak{t}'(Y,X)(\alpha + \mathfrak{d}'(Y,X)) - 1$ with $\mathbb{H}_\ell$ and $\mathbb{H}_k$ 3. Uni-variate Zero-Test on $\mathfrak{e}'(X)(\alpha + \mathfrak{I}(X)) - \mathfrak{m}'(X)$ with $\mathbb{H}_b$		**Open Oracles & Check Validity:** ${\mathfrak{a}'}^O(r_0) \leftarrow$ **PBVC.Open**$(\mathfrak{a}', r_0)$ ${\mathfrak{t}'}^O(0,0) \leftarrow$ **PBVC.Open**$(\mathfrak{t}', (0,0))$ ${\mathfrak{e}'}^O(0) \leftarrow$ **PBVC.Open**$(\mathfrak{e}', 0)$ Verify ${\mathfrak{t}'}^O(0,0) \times \ell k \stackrel{?}{=} {\mathfrak{e}'}^O(0) \times b$

Figure 2: Overall protocol of Rarus structured as an interactive argument of knowledge. Local computations are strictly separated into prover/verifier columns, with explicit message flows denoted by arrows.

5.1 Security Analytics

Theorem 6. *The PIOP for $\mathcal{R}_{RP}$ achieves perfect completeness (an error of 0) and a knowledge soundness error of at most $\frac{3b+6k+4\ell}{|\mathbb{F}|}$.*

The formal proof of completeness and soundness is deferred to Appendix D.

5.2 Complexity Analytics

According to the description in Section 4, the verifier cost during the whole protocol is $O(1)$.

In the protocol, the verifier needs to query the oracles 15 times. Each query represents a constant proof size. So, the total proof size is $O(1)$. However, when we implement the KZG-based vector commitment scheme, we can notice that some queries are for different oracles but at a common position. Therefore, we can apply the batch opening method when the prover opens the evaluations.

The concrete proof size when implementing the KZG-based vector commitment scheme is exactly $20\mathbb{G}_1 + 29\mathbb{F}$. On a standard 254-bit curve where elements in $\mathbb{G}_1$ and $\mathbb{F}$ are 32 bytes each, this equates to precisely 1568 bytes (≈ 1.53 KB). This correctly accounts for all polynomial commitments, evaluations, and the 4 evaluation points (each a 32-byte scalar in $\mathbb{F}$) required for the 4 batch openings.

The main cost of the prover is the construction time of the four equations and the cost of committing phase for polynomials. For constructing the four equations, the bottleneck cost is on polynomial multiplication in each equation. For efficiency, the method used here is based on NTT. A single polynomial multiplication requires one NTT and one INTT. In the construction of Equation 1, the cost of polynomial multiplication is $O(k\log k)$. In the construction of Equation 2, when the prover constructs f_1 and f_2, the cost of polynomial multiplication is $O(\ell k\log\ell) + O(k\log k)\mathbb{F}$. In the construction of Equation 3, the cost of polynomial multiplication is $O(b\log b)\mathbb{F}$.

In the description of the completed protocol, the prover must commit $\mathfrak{d}', \mathfrak{a}', \mathfrak{c}, \mathfrak{I}, \mathfrak{m}', \mathfrak{t}', \mathfrak{e}'$ and all the auxiliary polynomials in each PIOP gadget. The total cost here for commitment is $(3\ell k + 3b)\mathbb{G}$. Note that the reported group operations in our implementation are optimized using Multi-Scalar Multiplication (MSM) algorithms provided by the `mcl` library, which significantly amortizes the cost compared to naive individual scalar multiplications.

A trivial idea to obtain a succinct prover time is to set $b = \ell k$. If we do this, the total prover time can be $O(b\log b)\mathbb{F} + O(b)\mathbb{G}$. According to the definition in Sumset-Representation, there is $b^k \geq R$. If here $R = 2^N$, we have $b^b = b^{k\ell} \geq R^\ell = 2^{N\ell}$ from which we can calculate that $b = O(\frac{N\ell}{\log(N\ell)})$ as follows:

We have $b\log b = N\ell$. Now let $b = \frac{N\ell}{\log(N\ell)} \cdot c(N\ell)$, where c is some uni-variate function. Then we have

$$\frac{N\ell}{\log(N\ell)} \cdot c(N\ell) \cdot \log\left(\frac{N\ell}{\log(N\ell)} \cdot c(N\ell)\right) = N\ell$$

Simplify this, we have

$$c(N\ell) \cdot \left(1 + \frac{\log(c(N\ell))}{\log(N\ell)} - \frac{\log\log(N\ell)}{\log(N\ell)}\right) = 1$$

When $N\ell$ tends to infinity, $\frac{\log(c(N\ell))}{\log(N\ell)}$ and $\frac{\log\log(N\ell)}{\log(N\ell)}$ should both tend to 0. That means $c(N\ell)$ should tend to some constant. Therefore, we get $b = O(\frac{N\ell}{\log(N\ell)})$.

Then the total prover time will be $O(\frac{N\ell}{\log(N\ell)})\mathbb{G} + O(N\ell)\mathbb{F}$.

5.3 Parameter Optimization

We must let b^k be at least 2^N. So, we find that $3\ell k + 3b$ is at least $\frac{3N\ell}{\log b} + 3b$, which we can denote by a polynomial $u(b)$ in b. Then we have

$$u'(b) = 3 - \frac{3N\ell}{b(\log b)^2}$$

We know that b is an integer no less than 2. In addition, we can calculate $u'(2) < 0$ and $\lim_{b\to\infty} u'(b) > 0$. So, $u(b)$ has a unique minimum within the domain.

Suppose $u(b)$ reaches its minimum at $b_0 = \frac{c(N\ell)N\ell}{(\log N\ell)^2}$, where c is some uni-variate function. Then we have

$$u'(b_0) = 3 - \frac{3}{c(N\ell)} \frac{(\log(N\ell))^2}{(\log(N\ell) + \log c(N\ell) - 2\log\log(N\ell))^2}$$

We know that $u'(b_0) = 0$. This means that when $N\ell$ tends to infinity, $\frac{3}{c(N\ell)} \frac{(\log N\ell)^2}{(\log(N\ell)+\log c(N\ell)-2\log\log(N\ell))^2}$ tends to be a constant. So, we get that $c(N\ell)$ should tend to some constant. Therefore, if we want to obtain the optimal prover time, we should let $b = O(\frac{N\ell}{(\log N\ell)^2})$. Then we have that

$$\ell k = \frac{N\ell}{\log b} = \frac{N\ell}{\log(N\ell) + \log c - 2\log\log(N\ell)} \approx \frac{N\ell}{\log(N\ell)}$$

After this optimization, the total prover time will be $\frac{3N\ell}{\log(N\ell)}\mathbb{G} + O\left(\frac{N\ell\log\ell + N\log N}{\log(N\ell)}\right)\mathbb{F}$.

On Parameter Selection and Subgroup Existence. To efficiently implement the NTT operations in our protocol, the sizes of the multiplicative subgroups (ℓ, k, and b) must divide $p-1$, where p is the field order. In practice, our implementation handles this via a structured adjustment: we snap the theoretical optimal base b to the nearest power of 2. Subsequently, we deterministically derive ℓ and k based on the relations $b = k\ell$ and $b^k \geq R$, while ensuring that both ℓ and k are strictly powers of 2. In this way, our choice of b, ℓ, k guarantees protocol correctness and soundness. For our experimental evaluations, we utilize the BN254 curve, which

features a 2-adicity of 2^{28}. Since our maximum experimental parameters only require subgroup sizes up to $b \leq 2^{16}$, the multiplicative subgroups provided by the curve are abundantly large enough to seamlessly support our protocol.

6 Experiments

In this section, we implemented our protocol in C++ and conduct a comprehensive evaluation of its performance.

Experimental Setup. All experiments were executed on a high-performance server equipped with an Intel(R) Xeon(R) Platinum 8460Y+ CPU @ 2.00GHz and 192GB of RAM, running the Ubuntu 22.04 LTS operating system. Our protocol was implemented using the `mcl` library over the BN254 curve for all cryptographic operations, with all KZG commitments strictly instantiated in $\mathbb{G}_1$. All benchmarks were executed in a single-threaded environment.

We first empirically confirm the theoretical complexity analysis of our verifier. As demonstrated in our results, both the verifier time and proof size of Rarus exhibit $O(1)$ complexity, remaining constant regardless of the problem scale (N or ℓ). Specifically, across all test cases, the proof size stabilizes at exactly 1.53 KB (1568 bytes), and the verification time remains consistently below 0.03 s. This low and constant overhead confirms that our protocol is highly suitable for resource-constrained verifiers.

In the following subsections, we focus on the prover performance, highlighting the substantial efficiency gains of our protocol compared to state-of-the-art range proofs and lookup arguments, particularly in large-scale scenarios.

6.1 Performance Comparison with Range Proof Protocols

To benchmark the efficiency of Rarus, we compare it against two established range proof protocols: Bulletproofs and Missileproof. The comparisons are conducted with a fixed range $R = 2^{64}$, varying the vector size ℓ from 64 to 16,384. Parameter settings for Rarus are detailed in Table 3.

The performance results are illustrated in Figure 3. We observe a decisive advantage in prover efficiency:

- **Scalability:** As the vector size ℓ increases, the prover time for both Bulletproofs and Missileproofs grows rapidly. For instance, at $\ell = 16,384$, Bulletproofs requires over 463 s. In stark contrast, Rarus completes the proof in just 22.76 s—achieving a speedup of approximately **20**×.
- **Efficiency at Scale:** Even at a moderate scale of $\ell = 4,096$, Rarus (4.39 s) is roughly **25**× faster than Bulletproofs (111.2 s). This demonstrates that our protocol significantly reduces the computational bottleneck for the prover.

Vector Size (ℓ)	**64**	**256**	**1024**	**4096**	**16384**
Parameter b	256	256	256	256	65536
Parameter k	8	8	8	8	4

Table 3: Parameter setting in Rarus for testing different ℓ

- **Trade-offs and On-chain Verification:** While both protocols achieve asymptotic $O(1)$ verification, Missileproof achieves a lower concrete verification time (≈ 0.0025 s) compared to Rarus (≈ 0.02 s). This discrepancy stems from the underlying constant factors: verifying Rarus requires 35 pairing operations, whereas Missileproof requires only 12. We acknowledge that for strictly gas-constrained on-chain environments, Missileproof's verification overhead is superior. However, Rarus maintains a highly competitive constant proof size (1.53 KB) while providing a 20× faster prover. This tradeoff is highly optimized for off-chain auditing, client-side proving, and high-frequency decentralized applications where prover scalability is the primary bottleneck.

6.2 Performance Comparison with Lookup Protocols

Since range proofs can also be constructed using lookup arguments, we compare Rarus with two prominent lookup protocols: Plookup and Logup. Later, we will discuss the comparison between Rarus and CQ.

A critical limitation of traditional lookup protocols is their dependency on the table size (i.e., the range size R). To evaluate this, we fix the batch size ℓ and vary the range bit-length N from 16 to 20 ($R = 2^{16}$ to 2^{20}). The parameter configurations for Rarus are listed in Table 4.

The results, shown in Figure 4, highlight the superior robustness of our protocol:

- **Sensitivity to Range Size:** The prover time for Plookup and Logup degrades significantly as the range expands. Specifically, as R grows from 2^{16} to 2^{20}, Plookup's proving time explodes from 9.7 s to 152.3 s. Logup also sees a sharp increase to 28.8 s.
- **Robustness of Rarus:** In contrast, Rarus demonstrates exceptional stability. Its prover time grows logarithmically with the range size. Our protocol's prover cost is dominated by the batch size ℓ rather than the range R.
- **Speedup:** At $R = 2^{20}$, Rarus is approximately **63**× faster than Logup and over **330**× faster than Plookup. This result confirms that Rarus is uniquely positioned to handle proofs over large ranges where traditional lookup arguments become computationally infeasible.

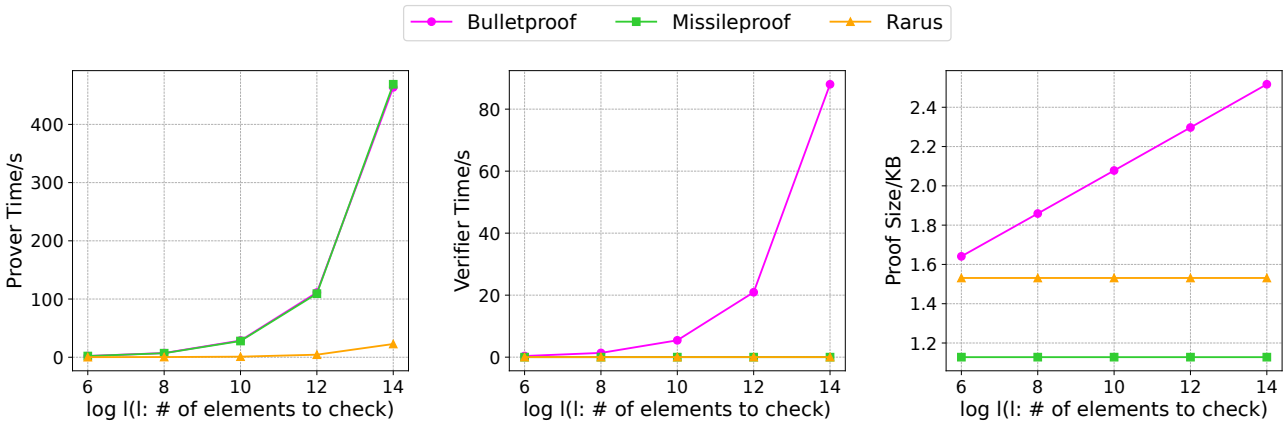


Figure 3: Performance comparison with range proof protocols

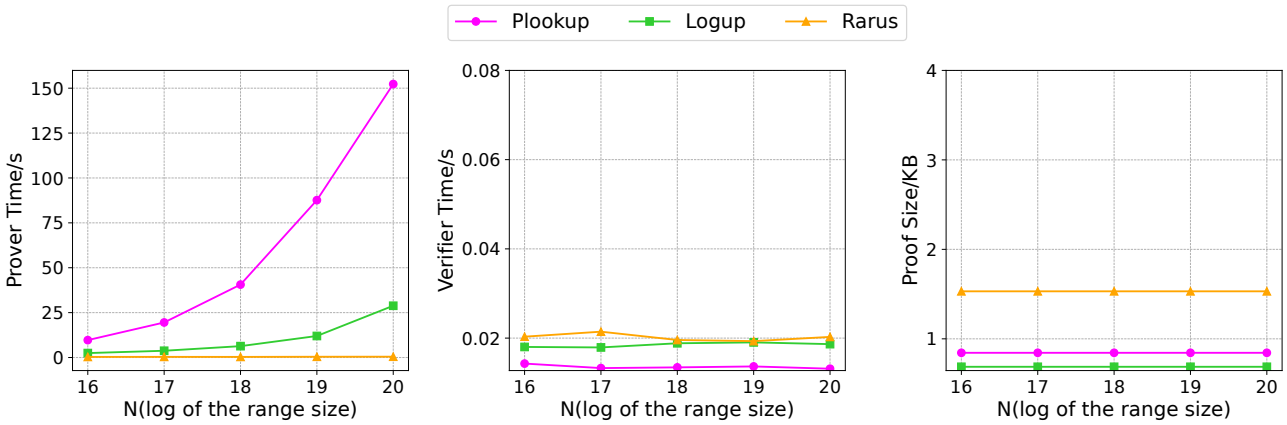


Figure 4: Performance comparison with lookup protocols

Range Bits (N)	**16**	**17**	**18**	**19**	**20**
Parameter b	256	512	512	1024	1024
Parameter k	2	2	2	2	2

Table 4: Parameter setting in Rarus for testing different N

6.3 Comparison with CQ [13]

CQ is a highly efficient lookup protocol that has been successfully applied to range proofs. However, a naïve application of CQ to large range problems (e.g., $R = 2^{64}$) introduces prohibitive preprocessing costs. Specifically, the preprocessing phase incurs a computational complexity of $O(N \log N)$ and a space complexity of $O(N)$, where N is the table size.

To handle a range of $R = 2^{64}$, practical implementations of CQ typically require splitting the range into smaller subtables, commonly of size 2^{32}. Even with this decomposition, the preprocessing step demands operations on the order of $2^{32}\mathbb{G} + 2^{32}\mathbb{F}$ and requires storing a lookup table of size at least $2^{32}\mathbb{F}$. As shown in Table 5, this imposes a massive memory footprint and significant setup time, often triggering Out-of-Memory (OOM) errors on standard hardware. In contrast, our protocol eliminates these substantial preprocessing overheads in both time and space, enabling deployment on consumer-grade devices.

Furthermore, even at smaller, manageable scales where CQ does not face memory exhaustion (e.g., $R = 2^{16}$ to 2^{20}), Rarus maintains a fundamental performance advantage. Empirical tests reveal that even if we completely ignore CQ's preprocessing overhead, its actual prover time remains between 0.7 s and 0.8 s for these ranges. Under the exact same settings, Rarus consistently outperforms CQ, achieving actual prover times of 0.33 s to 0.45 s (as shown in Figure 4 data). This demonstrates that Rarus is fundamentally faster for the prover even before memory bottlenecks occur.

Beyond preprocessing and smaller domains, our protocol also demonstrates superior efficiency in prover time at scale. Consider the case where $R = 2^{64}$ and the batch size $\ell = 2^{14}$.

Setting	Protocol	Preprocessing		Prover
		Time	Memory	Dom. Cost
$R = 2^{64}$ $\ell = 2^{14}$	CQ	> 4 hrs†	**OOM**	$16\ell\mathbb{G}$
	Rarus	**0 s**	**Negl.**	$\mathbf{13.5}\ell\mathbb{G}$
$R = 2^{64}$ $\ell = 2^{20}$	CQ	> 4 hrs†	**OOM**	$16\ell\mathbb{G}$
	Rarus	**0 s**	**Negl.**	$\mathbf{10}\ell\mathbb{G}$
$R = 2^{128}$ $\ell = 2^{14}$	CQ	> 8 hrs†	**OOM**	$32\ell\mathbb{G}$
	Rarus	**0 s**	**Negl.**	$\mathbf{23.5}\ell\mathbb{G}$

† Estimated based on complexity; OOM: Out-of-Memory.

Table 5: Comparison of Preprocessing and Proving costs. Due to excessive memory requirements of CQ for $N = 2^{32}$ tables, execution resulted in Out-of-Memory (OOM) errors. Values marked with † are analytically estimated.

Since CQ must decompose the range into two 2^{32} sub-tables, it requires the protocol to be invoked twice. The dominant cost in CQ is approximately $8\ell\mathbb{G} \times 2 = 16\ell\mathbb{G}$. In comparison, our protocol's prover cost (without zero-knowledge) is characterized by $(2\ell k + 2\ell + 3b)\mathbb{G}$. By selecting parameters $b = 2^{13}$ and $k = 5$, our prover cost is approximately $13.5\ell\mathbb{G}$, which is notably lower than that of CQ ($16\ell\mathbb{G}$).

This performance gap widens as the batch size ℓ increases. For instance, with $R = 2^{64}$ and $\ell = 2^{20}$, we can adjust our parameters to $b = 2^{16}$ and $k = 4$. The resulting prover cost is approximately $10\ell\mathbb{G}$, significantly outperforming CQ's $16\ell\mathbb{G}$. Furthermore, our protocol adapts more efficiently to larger ranges. For $R = 2^{128}$ and $\ell = 2^{14}$, CQ requires four protocol invocations, increasing the dominant term to $32\ell\mathbb{G}$. By setting $b = 2^{13}$ and $k = 10$, our protocol incurs only $23.5\ell\mathbb{G}$, maintaining a clear efficiency advantage.

6.4 Versatility for Large-Scale Applications

A significant limitation of existing schemes is the dichotomy between domain size and batch size. Conventional range proofs incur prohibitive prover costs at large batch sizes (ℓ), whereas lookup protocols suffer from memory exhaustion when the domain (N) exceeds typical bounds (e.g., 2^{28}). In contrast, Rarus demonstrates unique versatility, effectively handling *both* massive domains and high-throughput batches simultaneously.

To evaluate this, we configured a stress test with a domain size of $R = 2^{128}$ and varied the batch size ℓ from 2^{14} to 2^{18}. Parameter settings are provided in Table 6.

Significance of Large Parameters. Handling such extreme parameters is crucial for emerging privacy-preserving applications.

- **Large Domains ($N = 128$):** Applications involving 128-bit UUIDs, IPv6 address filtering, or overflow checks for cryptographic primitives (e.g., preventing wrap-around in 128-bit counters) require range proofs far beyond the standard 64-bit integer limit.
- **Large Batches (ℓ):** In large-scale privacy-preserving financial audits, such as Proof of Solvency for centralized exchanges [12], auditors must cryptographically verify that hundreds of thousands of individual user balances are non-negative to prevent the malicious injection of fake liabilities. Aggregating these massive scale range checks into a single vector commitment drastically reduces the verification footprint while maintaining strict user confidentiality.

Vector Size (ℓ)	**16,384** (2^{14})	**65,536** (2^{16})	**262,144** (2^{18})
Parameter b	256	65,536	65,536
Parameter k	16	8	8

Table 6: Parameter setting in Rarus for large-scale experiments ($R = 2^{128}$)

The performance results are summarized in Table 7. We observe that even under the extreme condition of $R = 2^{128}$ and a batch size of $\ell = 262,144$, the prover time remains within a practical range (≈ 15 minutes). Notably, the verification cost remains constant (< 0.022 s) and the proof size is invariant (1.53 KB). This confirms that Rarus is capable of supporting next-generation cryptographic applications that demand both wide value ranges and high-throughput processing, a combination that renders prior works impractical.

Metric	$\ell = 16,384$	$\ell = 65,536$	$\ell = 262,144$
Prover Time (s)	37.55	241.2	935.5
Verifier Time (s)	0.0206	0.0204	0.0214
Proof Size (KB)	1.53	1.53	1.53

Table 7: Performance of Rarus with large domain ($R = 2^{128}$) and varying batch sizes

7 Conclusion

We presented Rarus, a succinct range proof for polynomial-based vector commitments. By integrating Sumset-Representation with a novel Bi-variate Zero-Test and an optimized Uni-variate Sum-Check, Rarus achieves asymptotic optimality. It maintains $O(1)$ proof size and verifier time while delivering a $20\times$ prover speedup over state-of-the-art protocols. Supporting arbitrary ranges scalably without padding, Rarus provides a highly practical, foundational primitive for next-generation privacy-preserving audits and compliant financial systems.

Ethical Considerations

The development of Rarus is motivated by the critical need to balance individual privacy with regulatory compliance in digital financial systems. While zero-knowledge proofs are powerful tools for confidentiality, we recognize the ethical implications of their deployment.

A primary ethical contribution of Rarus is enabling "proof of solvency" and regulatory compliance without compromising user data. Unlike fully anonymous systems that may inadvertently facilitate illicit activities, Rarus is designed to support scenarios where ranges (e.g., transaction limits or asset reserves) must be cryptographically verified by auditors, fostering trust in decentralized infrastructure.

By reducing the prover's computational cost by orders of magnitude, Rarus significantly lowers the energy footprint of generating proofs. This efficiency also lowers the hardware barrier for participation, allowing users with standard consumer-grade devices to generate proofs for complex vectors, thereby democratizing access to privacy-enhancing technologies.

We acknowledge that cryptographic primitives are dual-use technologies. While Rarus is optimized for compliance-friendly applications (e.g., proving values lie within legal bounds), it could theoretically be used to conceal transaction details in non-compliant settings. We advocate for responsible deployment where range proofs serve as a mechanism for transparency rather than obfuscation.

Open Science

To support the reproducibility of our results and facilitate further research, we have made the source code of Rarus publicly available.

The complete C++ implementation of our protocol, including the KZG-based vector commitment integration and the Bivariate Zero-Test PIOP gadgets, is permanently archived on Zenodo at https://doi.org/10.5281/zenodo.20322978. The repository includes build instructions and a full list of dependencies.

Regarding the experimental evaluation, we benchmarked our system against standard open-source implementations of the baseline protocols (e.g., Bulletproofs). As these benchmarking frameworks are maintained by third parties, we do not include their source code in our repository. Instead, we have provided detailed documentation of the parameter configurations (b, k, ℓ) and hardware settings in Section 6, enabling researchers to reproduce the performance comparisons using these standard public libraries.

References

[1] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P Ward. Aurora: Transparent succinct arguments for r1cs. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*, pages 103–128. Springer, 2019.

[2] Dan Boneh, Benedikt Bünz, Ben Fisch, and Aleksey Szepieniec. Edrax: A cryptocurrency with stateless transaction validation. In *Advances in Cryptology–ASIACRYPT 2020*, pages 447–475. Springer, 2020.

[3] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 431–444, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[4] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.

[5] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, volume 5350 of *Lecture Notes in Computer Science*, pages 234–252. Springer, 2008.

[6] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. Cryptology ePrint Archive, Paper 2005/060, 2005.

[7] Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography – PKC 2013*, pages 55–72, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[8] Rafik Chaabouni, Helger Lipmaa, and Abhi Shelat. Additive combinatorics and discrete logarithm based range protocols. In Ron Steinfeld and Philip Hawkes, editors, *Information Security and Privacy*, pages 336–351, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[9] David Chaum. Showing credentials without identification. In Franz Pichler, editor, *Advances in Cryptology — EUROCRYPT' 85*, pages 241–244, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[10] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*, pages 738–768. Springer, 2020.

[11] Geoffroy Couteau, Dahmun Goudarzi, Michael Klooß, and Michael Reichle. Sharp: Short relaxed range proofs. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 609–622, 2022.

[12] Gaby G. Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 720–731, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2810103.2813674.

[13] Liam Eagen, Dario Fiore, and Ariel Gabizon. cq: Cached quotients for fast lookups. *Cryptology ePrint Archive*, 2022.

[14] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In *Advances in Cryptology – ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I*, page 649–678, Berlin, Heidelberg, 2019. Springer-Verlag. doi:10.1007/978-3-030-34578-5_23.

[15] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[16] Ariel Gabizon and Zachary J Williamson. plookup: A simplified polynomial protocol for lookup tables. *Cryptology ePrint Archive*, 2020.

[17] Rui Gao, Zhiguo Wan, Yuncong Hu, and Huaqun Wang. A succinct range proof for polynomial-based vector commitment. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3152–3166, 2024.

[18] Jens Groth. Non-interactive zero-knowledge arguments for voting. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 467–482, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[19] Yanpei Guo, Wenjie Qu, Linyu Wu, Shengfang Zhai, Lionel Z Wang, Ming Xu, Yue Liu, Binhang Yuan, Dawn Song, and Jiaheng Zhang. Immaculate: A practical llm auditing framework via verifiable computation. *arXiv preprint arXiv:2602.22700*, 2026.

[20] Ulrich Haböck. Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Paper 2022/1530, 2022. URL: https://eprint.iacr.org/2022/1530.

[21] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 177–194. Springer, 2010.

[22] Benoit Libert. Vector commitments with proofs of smallness: Short range proofs and more. Cryptology ePrint Archive, Paper 2023/800, 2023. URL: https://eprint.iacr.org/2023/800.

[23] Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure vickrey auctions without threshold trust. Cryptology ePrint Archive, Paper 2001/095, 2001.

[24] T. Liu, T. Xie, J. Zhang, D. Song, and Y. Zhang. Pianist: Scalable zkrollups via fully distributed zero-knowledge proofs. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 39–39, Los Alamitos, CA, USA, may 2024. IEEE Computer Society.

[25] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In *Theory of Cryptography Conference*, pages 222–242. Springer, 2013.

[26] David C. Parkes, Michael O. Rabin, Stuart M. Shieber, and Christopher Thorpe. Practical secrecy-preserving, verifiably correct and trustworthy auctions. *Electronic Commerce Research and Applications*, 7(3):294–312, 2008. Special Section: New Research from the 2006 International Conference on Electronic Commerce. URL: https://www.sciencedirect.com/science/article/pii/S1567422308000197, doi:10.1016/j.elerap.2008.04.006.

[27] Zipeng Qiu, Wenjie Qu, Jiaheng Zhang, and Binhang Yuan. V3db: Audit-on-demand zero-knowledge proofs for verifiable vector search over committed snapshots. *arXiv preprint arXiv:2603.03065*, 2026.

[28] Wenjie Qu, Yanpei Guo, Yue Ying, and Jiaheng Zhang. Verfcnn, optimal complexity zksnark for convolutional neural networks. *Cryptology ePrint Archive*, 2025.

[29] Wenjie Qu, Yanpei Guo, and Jiaheng Zhang. Lightning, field-agnostic super-efficient polynomial commitment scheme. *Cryptology ePrint Archive*, 2026.

[30] Wenjie Qu, Yijun Sun, Xuanming Liu, Tao Lu, Yanpei Guo, Kai Chen, and Jiaheng Zhang. {zkGPT}: An efficient non-interactive zero-knowledge proof framework for {LLM} inference. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2045–2063, 2025.

[31] Michael O. Rabin, Yishay Mansour, S. Muthukrishnan, and Moti Yung. Strictly-black-box zero-knowledge and efficient validation of financial transactions. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, pages 738–749, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[32] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

[33] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In *Annual International Cryptology Conference*, pages 704–737. Springer, 2020.

[34] Srinath Setty, Justin Thaler, and Riad Wahby. Unlocking the lookup singularity with lasso. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 180–209. Springer, 2024.

[35] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *NDSS*, 2018.

[36] N. Wang, S. Chau, and D. Liu. Swiftrange: A short and efficient zero-knowledge range argument for confidential transactions and more. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 57–57, 2024.

A Uni-variate Zero-Test PIOP

Here we describe a PIOP proving that a uni-variate polynomial evaluates to zero everywhere on a subgroup $\mathbb{H}_\ell \subset \mathbb{F}$ of order ℓ with generator ω_ℓ. Formally, the PIOP proves the relation:

$$\mathcal{R}_{\text{Uni-ZT}} = \{(C \in \mathbb{G}; \mathfrak{f}(X) \in \mathbb{F}_{d(X)<D}[X]) :$$
$$C = \mathbf{PBVC}(\mathfrak{f}) \wedge \forall i \in [0,\ell), \mathfrak{f}(\omega_\ell^i) = 0\}$$

Lemma 3. *Let $Z_{\mathbb{H}_\ell}(X) = \prod_{a\in\mathbb{H}_\ell}(X-a)$ be the vanishing polynomial. $\mathfrak{f}(X)$ is identically zero on $\mathbb{H}_\ell$ if and only if there exists $\mathfrak{h}(X)$ such that $\mathfrak{f}(X) = \mathfrak{h}(X)Z_{\mathbb{H}_\ell}(X)$.*

Proof. This is a standard algebraic property. $\forall a \in \mathbb{H}_\ell, \mathfrak{f}(a) = 0$ requires all elements in $\mathbb{H}_\ell$ to be roots of $\mathfrak{f}(X)$, which holds if and only if $\mathfrak{f}(X)$ is perfectly divisible by $Z_{\mathbb{H}_\ell}(X)$. □

Protocol & Complexity. The protocol proceeds as follows: $\mathcal{P}$ computes the quotient $\mathfrak{h}(X) = \mathfrak{f}(X)/Z_{\mathbb{H}_\ell}(X)$, commits to it, and sends the oracle $\mathfrak{h}^O$ to $\mathcal{V}$. Next, $\mathcal{V}$ samples a random challenge $\tau \xleftarrow{\$} \mathbb{F}$, computes $Z_{\mathbb{H}_\ell}(\tau)$, and queries the oracles to obtain $\mathfrak{f}^O(\tau)$ and $\mathfrak{h}^O(\tau)$. Finally, $\mathcal{V}$ accepts if $\mathfrak{f}^O(\tau) = \mathfrak{h}^O(\tau) \cdot Z_{\mathbb{H}_\ell}(\tau)$.

Theorem 7. *The PIOP for $\mathcal{R}_{Uni\text{-}ZT}$ is perfectly complete and has a knowledge soundness error of $D/|\mathbb{F}|$.*

The completeness holds by Lemma 3. By the Schwartz-Zippel Lemma, the soundness error is bounded by the maximum degree D over $|\mathbb{F}|$. This PIOP requires $O(D)\mathbb{G} + O(D)\mathbb{F}$ prover time, $O(1)\mathbb{G} + O(1)\mathbb{F}$ verifier time, and $O(1)$ proof size.

B Uni-variate Sum-Check PIOP

Here we describe a PIOP proving that the sum of all evaluations on $\mathbb{H}_\ell$ of a uni-variate polynomial equals zero. Formally, it proves the relation:

$$\mathcal{R}_{\text{Uni-SC}} = \{(C \in \mathbb{G}; \mathfrak{f}(X) \in \mathbb{F}_{d(X)<D}[X]) :$$
$$C = \mathbf{PBVC}(\mathfrak{f}) \wedge \sum_{i=0}^{\ell-1} \mathfrak{f}(\omega_\ell^i) = 0\}$$

Lemma 4. *For a polynomial $\mathfrak{f}(X) \in \mathbb{F}_{d(X)<D}[X]$, we have $\sum_{a\in\mathbb{H}_\ell}\mathfrak{f}(a) = 0$ if and only if there exist $\mathfrak{h}(X) \in \mathbb{F}[X]$ and $\mathfrak{g}(X) \in \mathbb{F}_{d(X)<\ell-1}[X]$ such that $\mathfrak{f}(X) = \mathfrak{h}(X)Z_{\mathbb{H}_\ell}(X) + X \cdot \mathfrak{g}(X)$.*

Proof. Dividing $\mathfrak{f}(X)$ by $Z_{\mathbb{H}_\ell}(X)$ yields $\mathfrak{f}(X) = \mathfrak{h}(X)Z_{\mathbb{H}_\ell}(X) + \mathfrak{u}(X)$, where $\deg(\mathfrak{u}) < \ell$. Let $\mathfrak{u}(X) = \sum_{i=0}^{\ell-1} u_i X^i$. By the group property $\sum_{a\in\mathbb{H}_\ell} a^i = 0$ for $0 < i < \ell$, the sum of evaluations simplifies to $\sum_{a\in\mathbb{H}_\ell}\mathfrak{f}(a) = \sum_{a\in\mathbb{H}_\ell}\mathfrak{u}(a) = \ell \cdot u_0$. Thus, the total sum is zero if and only if $u_0 = 0$. This implies $\mathfrak{u}(X)$ has no constant term, allowing it to be factored exactly as $X \cdot \mathfrak{g}(X)$ for some $\mathfrak{g}(X)$ of degree $< \ell - 1$. □

Protocol & Complexity. The protocol proceeds as follows: $\mathcal{P}$ computes $\mathfrak{h}(X)$ and $\mathfrak{g}(X)$ satisfying Lemma 4, commits to them, and sends the oracles $\mathfrak{h}^O, \mathfrak{g}^O$ to $\mathcal{V}$. Then, $\mathcal{V}$ samples a challenge $\tau \xleftarrow{\$} \mathbb{F}$, computes $Z_{\mathbb{H}_\ell}(\tau)$, and queries $\mathfrak{f}^O(\tau), \mathfrak{h}^O(\tau)$, and $\mathfrak{g}^O(\tau)$. $\mathcal{V}$ accepts if $\mathfrak{f}^O(\tau) = \mathfrak{h}^O(\tau) \cdot Z_{\mathbb{H}_\ell}(\tau) + \tau \cdot \mathfrak{g}^O(\tau)$.

Theorem 8. *The PIOP for $\mathcal{R}_{Uni\text{-}SC}$ is perfectly complete and has a knowledge soundness error of $D/|\mathbb{F}|$.*

Similarly, completeness holds via Lemma 4, and the Schwartz-Zippel Lemma bounds the soundness error by $D/|\mathbb{F}|$. The complexity matches the Zero-Test PIOP: $O(D)\mathbb{G} + O(D)\mathbb{F}$ prover time, $O(1)\mathbb{G} + O(1)\mathbb{F}$ verifier time, and $O(1)$ proof size.

C Proof of Theorem 5

The above protocol is zero-knowledge assuming that the underlying polynomial commitment scheme is perfectly hiding. That is, for any probabilistic polynomial-time verifier $\mathcal{V}^*$, we want to prove that there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that the view of $\mathcal{V}^*$ in a real execution is computationally indistinguishable from its view in a simulated execution.

Let $\mathcal{V}^*$ be an arbitrary PPT verifier. We construct a simulator $\mathcal{S}$ as follows:

- It is clear that,

$$\begin{aligned}
&\mathfrak{c}(X)\mathfrak{d}'(r_0,X) - \frac{\mathfrak{a}'(r_0,X)}{k} \\
&= \mathfrak{c}(X)\mathfrak{d}(r_0,X) + s_0\mathfrak{c}(X)Z_{\mathbb{H}_\ell}(r_0)Z_{\mathbb{H}_k}(X) \\
&+ s_1\mathfrak{c}(X)Z_{\mathbb{H}_k}(X) - \frac{\mathfrak{a}(r_0) + s_0 Z_{\mathbb{H}_\ell}(r_0)Z_{\mathbb{H}_k}(X)}{k} \\
&= \mathfrak{h}_1(X)Z_{\mathbb{H}_k}(X) + X\mathfrak{g}_1(X) + s_0\mathfrak{c}(X)Z_{\mathbb{H}_\ell}(r_0)Z_{\mathbb{H}_k}(X) \\
&+ s_1\mathfrak{c}(X)Z_{\mathbb{H}_k}(X) - \frac{s_0}{k}Z_{\mathbb{H}_\ell}(r_0)Z_{\mathbb{H}_k}(X)\mathfrak{f}_1'(Y) \\
&= \sum_{j=0}^{k-1}((\mathfrak{t}(Y,\omega_k^j)+0)(\alpha+\mathfrak{d}(Y,\omega_k^j)+0)-1)\mathfrak{u}(\beta,\omega_k^j) \\
&= \mathfrak{f}_1(Y) \\
&= \mathfrak{p}(Y)Z_{\mathbb{H}_l}(Y) \\
&\quad \mathfrak{f}_2'(X) - \frac{f'}{k} \\
&= [(\mathfrak{t}(r_0,X) + s_0 Z_{\mathbb{H}_\ell}(r_0)Z_{\mathbb{H}_k}(X) + s_1 Z_{\mathbb{H}_k}(X)) \\
&\times (\alpha + \mathfrak{d}(r_0,X) + s_0 Z_{\mathbb{H}_\ell}(r_0)Z_{\mathbb{H}_k}(X) + s_1 Z_{\mathbb{H}_k}(X)) - 1] \\
&\times \mathfrak{u}(\beta,X) - \frac{f'}{k} \\
&= (\mathfrak{t}(r_0,X)(\alpha+\mathfrak{d}(r_0,X)))\mathfrak{u}(\beta,X) - \frac{f'}{k} \\
&+ (\mathfrak{t}(r_0,X) + \alpha + \mathfrak{d}(r_0,X)) \\
&\quad \times (s_0 Z_{\mathbb{H}_\ell}(r_0)Z_{\mathbb{H}_k}(X) + s_1 Z_{\mathbb{H}_k}(X)) \\
&+ (s_0 Z_{\mathbb{H}_\ell}(r_0)Z_{\mathbb{H}_k}(X) + s_1 Z_{\mathbb{H}_k}(X))^2
\end{aligned}$$

$$
\begin{aligned}
&= (\mathfrak{t}(r_0,X)(\alpha+\mathfrak{d}(r_0,X)))\mathfrak{u}(\beta,X) - \frac{f'}{k} \\
&+ (\mathfrak{t}(r_0,X)+\alpha+\mathfrak{d}(r_0,X)+s_0 Z_{\mathbb{H}_\ell}(r_0)Z_{\mathbb{H}_k}(X)) \\
&\times (s_0 Z_{\mathbb{H}_\ell}(r_0)+s_1)Z_{\mathbb{H}_k}(X) \\
&\quad \mathfrak{e}'(X)(\alpha+\mathfrak{I}(X)) - \mathfrak{m}'(X) \\
&= \mathfrak{h}_3(X)Z_{\mathbb{H}_b}(X) \\
&= (\mathfrak{e}(X) + \frac{\ell k(-s_0+s_1)}{b} Z_{\mathbb{H}_b}(X))(\alpha+\mathfrak{I}(X)) \\
&- \mathfrak{m}(X) - s_0 Z_{\mathbb{H}_b}(X) \\
&= \mathfrak{e}(X)(\alpha+\mathfrak{I}(X)) - \mathfrak{m}(X) \\
&+ (\frac{\ell k(-s_0+s_1)}{b} - s_0) Z_{\mathbb{H}_b}(X)
\end{aligned}
$$

from which we have that

$$
\begin{aligned}
&\mathfrak{h}_1'(X) = \mathfrak{h}_1(X) + s_0\mathfrak{c}(X)Z_{\mathbb{H}_\ell}(r_0) + s_1\mathfrak{c}(X) - \frac{s_0}{k}Z_{\mathbb{H}_\ell}(r_0) \\
&\mathfrak{g}_1'(X) = \mathfrak{g}_1(X) \\
&\mathfrak{p}'(Y) = \mathfrak{p}(Y) \\
&\mathfrak{h}_2'(X) = \mathfrak{h}_2(X) + (\mathfrak{t}(r_0,X)+\alpha+\mathfrak{d}(r_0,X) \\
&\qquad + s_0 Z_{\mathbb{H}_\ell}(r_0)Z_{\mathbb{H}_k}(X))(s_0 Z_{\mathbb{H}_\ell}(r_0)+s_1) \\
&\mathfrak{g}_2'(X) = \mathfrak{g}_2(X) \\
&\mathfrak{h}_3'(X) = \mathfrak{h}_3(X) + (\frac{\ell k(-s_0+s_1)}{b} - s_0)
\end{aligned}
$$

$\mathcal{S}$ samples random polynomials $\mathfrak{h}_1^*(X)$, $\mathfrak{g}_1^*(X)$, $\mathfrak{p}^*(Y)$, $\mathfrak{h}_2^*(X)$, $\mathfrak{g}_2^*(X)$, $\mathfrak{h}_3^*(X)$ of appropriate degrees which are the same as $\mathfrak{h}_1(X) + s_0\mathfrak{c}(X)Z_{\mathbb{H}_\ell}(r_0) + s_1\mathfrak{c}(X) - \frac{s_0}{k}Z_{\mathbb{H}_\ell}(r_0)$, $\mathfrak{g}_1(X)$, $\mathfrak{p}(Y)$, $\mathfrak{h}_2(X) + (\mathfrak{t}(r_0,X) + \alpha + \mathfrak{d}(r_0,X) + s_0 Z_{\mathbb{H}_\ell}(r_0)Z_{\mathbb{H}_k}(X))(s_0 Z_{\mathbb{H}_\ell}(r_0) + s_1)$, $\mathfrak{g}_2(X)$, $\mathfrak{h}_3(X) + (\frac{\ell k(-s_0+s_1)}{b} - s_0)$, respectively.

- $\mathcal{S}$ also samples random challenge points $\alpha', \beta', r_0', r_1' \in \mathbb{F}$.
- $\mathcal{S}$ computes the results of:
  - $h_1^*(r_1')Z_{\mathbb{H}_k}(r_1') + r_1'\mathfrak{g}_1^*(r_1')$
  - $\mathfrak{p}^*(r_0')Z_{\mathbb{H}_\ell}(r_0')$
  - $h_2^*(r_1')Z_{\mathbb{H}_k}(r_1') + r_1'\mathfrak{g}_2^*(r_1')$
  - $h_1^*(r_0')Z_{\mathbb{H}_b}(r_0')$

  These values correspond to the results of:
  - $\mathfrak{c}^*(r_1')\mathfrak{d}^*(r_0',r_1') - \frac{\mathfrak{a}^*(r_0',r_1')}{k}$
  - f^*
  - $(\mathfrak{t}^*(r_0',r_1')(\alpha'+\mathfrak{d}^*(r_0,r_1))-1)\mathfrak{u}(\beta',r_1') - \frac{f^*}{k}$
  - $\mathfrak{e}^*(r_0)(\alpha'+\mathfrak{I}(r_0)) - \mathfrak{m}^*(r_0)$

  Therefore, $\mathcal{S}$ can calculate the evaluation of f^*, $\mathfrak{d}^*(r_0',r_1')$, $\mathfrak{a}^*(r_0',r_1')$, $\mathfrak{m}^*(r_0')$, $\mathfrak{t}^*(r_0',r_1')$, $\mathfrak{e}^*(r_0')$. Note that $\mathcal{S}$ only calculates the evaluation at the challenge points, not the full polynomials.
- $\mathcal{S}$ generates commitments:

$$
\begin{aligned}
&\mathsf{Com}_{\mathfrak{d}^*} = \mathsf{Commit}(\mathfrak{d}^*(Y,X)), \\
&\mathsf{Com}_{\mathfrak{a}^*} = \mathsf{Commit}(\mathfrak{a}^*(Y,X)), \\
&\mathsf{Com}_{\mathfrak{m}^*} = \mathsf{Commit}(\mathfrak{m}^*(X)), \\
&\mathsf{Com}_{\mathfrak{t}^*} = \mathsf{Commit}(\mathfrak{t}^*(Y,X)), \\
&\mathsf{Com}_{\mathfrak{e}^*} = \mathsf{Commit}(\mathfrak{e}^*(X)), \\
&\mathsf{Com}_{\mathfrak{h}_1^*} = \mathsf{Commit}(\mathfrak{h}_1^*(X)), \\
&\mathsf{Com}_{\mathfrak{g}_1^*} = \mathsf{Commit}(\mathfrak{g}_1^*(X)), \\
&\mathsf{Com}_{\mathfrak{p}^*} = \mathsf{Commit}(\mathfrak{p}^*(Y)), \\
&\mathsf{Com}_{\mathfrak{h}_2^*} = \mathsf{Commit}(\mathfrak{h}_2^*(X)), \\
&\mathsf{Com}_{\mathfrak{g}_2^*} = \mathsf{Commit}(\mathfrak{g}_2^*(X)), \\
&\mathsf{Com}_{\mathfrak{h}_3^*} = \mathsf{Commit}(\mathfrak{h}_3^*(X))
\end{aligned}
$$

  using the hiding property of the commitment scheme.
- $\mathcal{S}$ simulates valid proofs of opening:

$$
\begin{aligned}
&\pi_{\mathfrak{d}^*} = \mathsf{Open}(\mathfrak{d}^*, r_0', r_1'), \\
&\pi_{\mathfrak{a}^*} = \mathsf{Open}(\mathfrak{a}^*, r_0', r_1'), \\
&\pi_{\mathfrak{m}^*} = \mathsf{Open}(\mathfrak{m}^*, r_0'), \\
&\pi_{\mathfrak{t}^*} = \mathsf{Open}(\mathfrak{t}^*, r_0', r_1'), \\
&\pi_{\mathfrak{e}^*} = \mathsf{Open}(\mathfrak{e}^*, r_0'), \\
&\pi_{\mathfrak{h}_1^*} = \mathsf{Open}(\mathfrak{h}_1^*, r_1'), \\
&\pi_{\mathfrak{g}_1^*} = \mathsf{Open}(\mathfrak{g}_1^*, r_1'), \\
&\pi_{\mathfrak{p}^*} = \mathsf{Open}(\mathfrak{p}^*, r_0'), \\
&\pi_{\mathfrak{h}_2^*} = \mathsf{Open}(\mathfrak{h}_2^*, r_1'), \\
&\pi_{\mathfrak{g}_2^*} = \mathsf{Open}(\mathfrak{g}_2^*, r_1'), \\
&\pi_{\mathfrak{h}_3^*} = \mathsf{Open}(\mathfrak{h}_3^*, r_0')
\end{aligned}
$$

  using the zero-knowledge property (or extractability) of the commitment scheme.
- Finally, $\mathcal{S}$ sends the following to $\mathcal{V}^*$:
  - $(\alpha', \beta', r_0', r_1')$
  - $(\mathsf{Com}_{\mathfrak{d}^*}, \mathsf{Com}_{\mathfrak{a}^*}, \mathsf{Com}_{\mathfrak{m}^*}, \mathsf{Com}_{\mathfrak{t}^*}, \mathsf{Com}_{\mathfrak{e}^*})$
  - $(\mathsf{Com}_{\mathfrak{h}_1^*}, \mathsf{Com}_{\mathfrak{g}_1^*}, \mathsf{Com}_{\mathfrak{p}^*}, \mathsf{Com}_{\mathfrak{h}_2^*}, \mathsf{Com}_{\mathfrak{g}_2^*}, \mathsf{Com}_{\mathfrak{h}_3^*})$
  - $(\mathfrak{d}^*(r_0',r_1'), \mathfrak{a}^*(r_0',r_1'), \mathfrak{m}^*(r_0'), \mathfrak{t}^*(r_0',r_1'), \mathfrak{e}^*(r_0'))$
  - $(\mathfrak{h}_1^*(r_1'), \mathfrak{g}_1^*(r_1'), \mathfrak{p}^*(r_0'), \mathfrak{h}_2^*(r_1'), \mathfrak{g}_2^*(r_1'), \mathfrak{h}_3^*(r_1'))$
  - $(\pi_{\mathfrak{d}^*}, \pi_{\mathfrak{a}^*}, \pi_{\mathfrak{m}^*}, \pi_{\mathfrak{t}^*}, \pi_{\mathfrak{e}^*})$
  - $(\pi_{\mathfrak{h}_1^*}, \pi_{\mathfrak{g}_1^*}, \pi_{\mathfrak{p}^*}, \pi_{\mathfrak{h}_2^*}, \pi_{\mathfrak{g}_2^*}, \pi_{\mathfrak{h}_2^*})$

Since the polynomial commitment scheme is perfectly (or computationally) hiding, the simulated commitments are indistinguishable from real commitments. Furthermore, since the equalities

$$\mathfrak{c}(r_1')\mathfrak{d}^*(r_0',r_1') - \frac{\mathfrak{a}^*(r_0',r_1')}{k} = \mathfrak{h}_1^*(r_1')Z_{\mathbb{H}_k}(r_1') + r_1'\mathfrak{g}_1^*(r_1')$$
$$f^* = \mathfrak{p}^*(r_0')Z_{\mathbb{H}_\ell}(r_0')(\mathfrak{t}^*(r_0',r_1')(\alpha' + \mathfrak{d}^*(r_0',r_1')) - 1)$$
$$\times \mathfrak{u}(\beta', r_1') - \frac{f^*}{k} = \mathfrak{h}_2^*(r_1')Z_{\mathbb{H}_k}(r_1') + r_1'\mathfrak{g}_2^*(r_1')$$
$$\mathfrak{e}^*(r_0')(\alpha' + \mathfrak{I}(r_0')) - \mathfrak{m}^*(X) = \mathfrak{h}_3^*(r_0')Z_{\mathbb{H}_b}(r_0')$$

holds by construction, and the opening proofs are simulated correctly, the verifier's view is identically (or computationally indistinguishably) distributed compared to a real interaction.

Therefore, the simulator $\mathcal{S}$ successfully simulates the view of any verifier $\mathcal{V}^*$ without knowledge of $\mathfrak{a}(X)$, proving that the protocol is zero-knowledge.

D Proof of Theorem 6 (Completeness and Soundness)

Proof. **Completeness.** Assume that the prover $\mathcal{P}$ is honest and holds a valid witness for the relation $\mathcal{R}_{\mathrm{RP}}$. The verification protocol involves checking these identities at a random challenge point α. The Log-derivative identity involves terms of the form $\frac{1}{\alpha+d_{i,j}}$ and $\frac{m_i}{\alpha+i}$. The evaluation of these rational functions is well-defined if and only if the denominator is non-zero. By restricting the verifier's sampling space for α to $\mathbb{F}\setminus\{-b+1,\ldots,0\}$, we explicitly exclude the valid ranges of $-d_{i,j}$ and $-i$. This elegantly avoids the zero-denominator issue, ensuring the rational functions are strictly well-defined and thus achieving perfect completeness. Note that removing a small constant number of elements from the overwhelmingly large field $\mathbb{F}$ has a negligible impact on the soundness probability.

Soundness. As we mentioned above, the random challenge α causes a soundness error of at most $\frac{b+k+\ell+2}{|\mathbb{F}|}$ and the random challenge r_0 in Equation 1 causes a soundness error of at most $\frac{\ell}{|\mathbb{F}|}$. Applying an Optimized Uni-variate Sum-Check PIOP in Equation 1 causes a soundness of at most $\frac{2k-1}{|\mathbb{F}|}$. Applying a Bi-variate Zero-Test PIOP in Equation 2 causes a soundness error of at most $\frac{2\ell+3k}{|\mathbb{F}|}$. Applying a Uni-variate Zero-Test PIOP in Equation 3 causes a soundness of at most $\frac{2b-1}{|\mathbb{F}|}$.

According to Union Bound, the soundness error of the whole protocol is at most $\frac{3b+6k+4\ell}{|\mathbb{F}|}$. □

E Supplementary Experimental Data

	Scheme	Polynomial Size (e.g., ℓ)				
		64	256	1024	4096	16384
Prover Time (s)	BulletProof	2.133	7.215	28.75	111.2	463.6
	Missile Proof	2.070	6.949	27.87	109.2	468.9
	This work	0.1733	0.3523	0.9877	4.387	22.76
Verifier Time (s)	BulletProof	0.374	1.368	5.432	20.93	88.02
	Missile Proof	0.002542	0.002473	0.002502	0.002612	0.002523
	This work	0.01998	0.01986	0.02221	0.02039	0.02120
Proof Size (KB)	BulletProof	1.641	1.859	2.078	2.297	2.517
	Missile Proof	1.128	1.128	1.128	1.128	1.128
	This work	1.531	1.531	1.531	1.531	1.531

Table 8: Performance comparison with range proof protocols

	Scheme	Range (e.g., R)				
		2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
Prover Time (s)	Plookup	9.692	19.52	40.62	87.64	152.3
	Logup	2.471	3.763	6.334	11.99	28.83
	This work	0.3325	0.3620	0.3431	0.4242	0.4583
Verifier Time (s)	Plookup	0.01429	0.01328	0.01344	0.01365	0.01314
	Logup	0.01802	0.01791	0.01884	0.01904	0.01865
	This work	0.02032	0.02144	0.01958	0.01932	0.02027
Proof Size (KB)	Plookup	0.8438	0.8438	0.8438	0.8438	0.8438
	Logup	0.6875	0.6875	0.6875	0.6875	0.6875
	This work	1.531	1.531	1.531	1.531	1.531

Table 9: Performance comparison with lookup protocols