SwiftRange: A Short and Efficient Zero-Knowledge Range Argument For Confidential Transactions and More

Nan Wang
Australian National University and CSIRO's Data61, Australia
nan.wang@data61.csiro.au

Sid Chi-Kin Chau
Australian National University and CSIRO's Data61, Australia
sid.chau@acm.org

Dongxi Liu
CSIRO's Data61, Australia
dongxi.liu@data61.csiro.au

Abstract—**Zero-knowledge range proofs play a critical role in confidential transactions (CT) on blockchain systems. They are used to prove the non-negativity of committed transaction payments without disclosing the exact values. Logarithmic-sized range proofs with transparent setups, e.g., Bulletproofs, which aim to prove a committed value lies in the range $[0, 2^N - 1]$ where N is the bit length of the range, have gained growing popularity for communication-critical blockchain systems as they increase scalability by allowing a block to accommodate more transactions. In this paper, we propose SwiftRange, a new type of logarithmic-sized zero-knowledge range argument with a transparent setup in the discrete logarithm setting. Our argument can be a drop-in replacement for range proofs in blockchain-based confidential transactions. Compared with Bulletproofs, our argument has higher computational efficiency and lower round complexity while incurring comparable communication overheads for CT-friendly ranges, where $N \in \{32, 64\}$. Specifically, a SwiftRange achieves 1.61× and 1.32× proving efficiency with no more than 1.1× communication costs for both ranges, respectively. More importantly, our argument offers a 2.3× increase in verification efficiency. Furthermore, our argument has a smaller size when $N \leq 16$, making it competitive for many other communication-critical applications. Our argument supports the aggregation of multiple single arguments for greater efficiency in communication and verification. Finally, we benchmarked our argument against the state-of-the-art range proofs to demonstrate its practicality.**

Index Terms—**Zero-knowledge range argument, discrete logarithm, logarithmic-size, transparent setup, confidential transactions, blockchain**

1. Introduction

Zero-knowledge proofs are crucial building blocks for a variety of secure applications, e.g., confidential transactions, signature schemes and anonymous credential systems, since the first proposal in 1985 [1]. A zero-knowledge proof allows a prover to prove the truth of a statement without disclosing any secret information. More formally, given an NP-language $\mathcal{L}$, a prover can convince a verifier of knowing a witness ω for a statement $u \in \mathcal{L}$. A zero-knowledge proof should satisfy three key properties:

- *Completeness.* A prover can convince a verifier of $u \in \mathcal{L}$, if $u \in \mathcal{L}$.
- *Soundness.* A prover cannot convince a verifier of $u \in \mathcal{L}$ if $u \notin \mathcal{L}$.
- *Zero-knowledge.* The proof does not reveal anything except $u \in \mathcal{L}$.

Having been developed for two decades, zero-knowledge range proofs aim to prove that a committed value falls within a specified range. They have been extensively used in various applications, e.g., federated learning [2], e-cash [3], multi-coupon systems [4] and electronic voting systems [5], [6], where the privacy of the precise values needs to be preserved. Recently, they have become more popular because of the rise of confidential transactions (CT) [7] on blockchain systems. Blockchain is a distributed ledger technology that allows multiple parties to have a shared and synchronized view of a ledger. It employs cryptographic techniques to store data in linked blocks, making the ledger immutable and tamper-proof. Blockchain is expected to shape the future of our world by enabling secure, transparent, decentralized peer-to-peer transactions without the need for intermediaries. Nevertheless, privacy concerns have been a hindrance to the broader adoption of blockchain, as it does not inherently ensure privacy. Confidential transactions protect privacy by using commitment schemes to hide transaction values, which is particularly crucial for businesses and individuals with sensitive information. Thus, zero-knowledge range proofs are used to demonstrate the possession of sufficient funds with non-negative balances in a privacy-preserving manner.

Logarithmic-sized range proofs with transparent setups have gained increasing popularity, which often leverage the bit-decomposition approach to prove a committed value lies in the range $[0, 2^N - 1]$, where N is the bit length of the range. On the one hand, shorter range proofs help increase the number of transactions in a block, enhancing scalability by a higher throughput of transactions, since scalability is a major bottleneck for blockchain systems. On the other hand, transparent or non-trusted setups are becoming a desirable feature for zero-knowledge proofs. Trusted setups require a specific group of trusted parties to generate public parameters and destroy secret trapdoors,

TABLE 1: The efficiency comparison of the state-of-the-art bit-decomposition-based range arguments, where N is assumed to be the power of 2. $\mathbb{G}$ indicates a cyclic group of prime order p and $\mathbb{Z}_p$ is the ring of integers modulo p. We compare the group exponentiations as they dominate the computational overheads. The complexity of proving and verification of Bulletproofs and our work has been optimized for higher efficiency. The computational overheads of Bulletproofs+ are comparable to those of Bulletproofs based on their experimental results.

Type	Flashproofs [8]	Bulletproofs [9]	Bulletproofs+ [10]	SwiftRange ($N \leq 8$)	SwiftRange ($N > 8$)
Prover No. of Exps ($\mathbb{G}$)	$\frac{1}{2}N^{\frac{4}{3}} + N^{\frac{2}{3}} + N^{\frac{1}{3}} - \frac{1}{2}N$	$10N + 4\log N + 7$	$\approx$ Bulletproofs	1	$8N - 79$
Verifier No. of Exps ($\mathbb{G}$)	$\frac{3}{2}(N^{\frac{2}{3}} + N^{\frac{1}{3}} + 2)$	$2N + 2\log N + 7$	$\approx$ Bulletproofs	$N + 5$	$N + 4\log N - 7$
Proof Size No. of Elements	$N^{\frac{2}{3}} + 2$ ($\mathbb{G}$) $\frac{1}{2}(N^{\frac{2}{3}} + 3N^{\frac{1}{3}} + 4)$ ($\mathbb{Z}_p$)	$2\log N + 4$ ($\mathbb{G}$) 5 ($\mathbb{Z}_p$)	$2\log N + 3$ ($\mathbb{G}$) 3 ($\mathbb{Z}_p$)	2 ($\mathbb{G}$) $N + 1$ ($\mathbb{Z}_p$)	$4\log N - 10$ ($\mathbb{G}$) 9 ($\mathbb{Z}_p$)
No. of Rounds	3	$2\log N + 5$	$2\log N + 5$	5	$2\log N - 1$

which not only undermine blockchain decentralization nature but also expose the risk of leaking trapdoor information that may compromise the underlying security. Benefiting from the logarithmic shortness and transparent setup, the state-of-the-art Bulletproofs [9] are widely used in various blockchain systems, e.g., Monero, Grin, Beam. However, the proving and verification overheads of Bulletproofs are dominated by $10N$ and $2N$ expensive group exponentiations, respectively. The range proofs with higher computational efficiency and comparable proof sizes can be more beneficial to blockchain-based confidential transactions. Firstly, higher proving efficiency helps generate a proof faster to improve user experience. More importantly, higher verification efficiency leads to higher transaction validation efficiency, which can also greatly enhance the scalability.

1.1. Contributions

In this paper, our major contribution is the proposal for SwiftRange, a new type of logarithmic-sized zero-knowledge range argument in the discrete logarithm (DLOG) setting with a transparent setup. We aim to provide a robust alternative to the range proofs for blockchain and many other privacy-preserving systems.

At a high level, SwiftRange is an interactive protocol Π_{crg} between a prover and a verifier. It is a composition[1] of a 5-round zero-knowledge range protocol Π_{rg} and a sequence of non-zero-knowledge compression protocols Π_{ac}. Our compression protocol is an adaptation of the protocol Π_{c} in [11] from linear relations to quadratic ones. Our adaptation addresses the coupling issue of the protocol Π_{c} to improve the verification efficiency. The range protocol Π_{rg} alone can provide highly short range arguments when $N \leq 8$.

$$\Pi_{\mathsf{crg}} = \underbrace{\Pi_{\mathsf{ac}} \diamond \cdots \diamond \Pi_{\mathsf{ac}}}_{\log N - 3 \text{ times}} \diamond \Pi_{\mathsf{rg}} \tag{1}$$

When $N > 8$, the prover can recursively apply $\log N - 3$ times of the compression protocol Π_{ac} to the range protocol Π_{rg} until the witness dimension is reduced to 8 for the minimum proof size as shown in Eqn. (1).

Our argument is short and computationally efficient without the need for computationally expensive pairing operations. For general ranges where $N > 8$, it involves $4\log N - 1$ elements, $8N - 79$ and $N + 4\log N - 7$ group exponentiations for proving and verification, and $2\log N - 1$ rounds, where the negative constant values result from the early termination of the protocols. Moreover, we present an optimization technique to use a single group exponentiation in the proof generation when $N \leq 8^2$. Our argument also supports aggregation for greater efficiency in communication and verification, where a prover can prove J committed values in batches by using additional $4\log J$ group elements over the size of a single argument. Finally, we benchmarked our argument against the state-of-the-art range arguments to demonstrate its practicality. N and J are the powers of 2 throughout the paper, and we can pad with zeros if not.

1.2. Comparisons with the State-of-the-art

Flashproofs [8] and Bulletproofs[3] [9] are two particular bit-decomposition-based range arguments in the DLOG setting with a transparent setup. Our secondary contribution is to provide a comprehensive performance comparison of the three arguments, allowing users to decide the appropriate one based on their strengths and weaknesses.

Flashproofs are 3-round zero-knowledge range arguments that achieve $O(N^{\frac{2}{3}})$ sub-linear efficiency in communication and verification. They have a highly efficient verifier, which incurs comparable gas costs on smart contract platforms to those of the most efficient zkSNARK [12] that rely on a trusted setup. Smart contracts are publicly verifiable computer programs running on blockchain systems, which serve as a "decentralized executor". Note that the executions on smart contract systems require gas fees proportional to the amount of computational operations. Bulletproofs have

[1] We follow the notation of [11] and write $\Pi_{\mathsf{b}} \diamond \Pi_{\mathsf{a}}$ for the composition of two interactive protocols Π_{a} and Π_{b}, where the composition indicates that the final message of Π_{a} is replaced by the execution of Π_{b}.

[2] The proving complexity of our argument would be dominated by $O(N)$ group multiplications when $N \leq 8$.

[3] Flashproofs and Bulletproofs consist of multiple zero-knowledge proofs. We only refer to their range proofs in our paper.

gained wide popularity among CT applications due to their logarithmic succinctness. The basic idea consists in using an inner product argument to recursively compress a 5-round zero-knowledge range argument until the witness dimension is reduced to 1 for the minimum communication complexity. They involve $O(N)$ number of group exponentiations in proving and verification. Bulletproofs+ [10] are an improved version, which require 3 fewer elements and achieve comparable computational efficiency to Bulletproofs. According to their experimental results, for 64-bit ranges, Bulletproofs+ run 6.2% faster in proving but 5.8% slower in verification. For 32-bit ranges, the efficiency discrepancy is smaller.

TABLE 2: The comparison of the proof sizes in bytes, where 32-bit and 64-bit ranges are commonly used for confidential transactions on blockchain systems.

N	8	16	32	64	128
Flashproofs	385	513	738	994	1444
Bulletproofs	482	546	610	674	739
Bulletproofs+	386	450	514	578	643
SwiftRange	**353**	481	610	738	867

Tables 1 and 2 show two efficiency comparisons of the state-of-the-art range arguments with ours. Flashproofs have a sub-linearly efficient verifier and constant round complexity but produce larger proof sizes for $N > 16$, making it better suited for confidential transactions on smart contract systems, where the verification efficiency matters the most. In this paper, we essentially focus on the comparisons between Bulletproofs-based arguments and ours as all of them feature logarithmically short proof sizes. These short arguments are more suitable candidates for confidential transactions on blockchain systems where block sizes are critical. Regarding the proof size, despite having a two-fold growth rate in the communication complexity, our argument involves comparable communication costs to the Bulletproofs-based ones for CT-friendly ranges where $N \in \{32, 64\}$, since the latter ones have large constants in their complexity. Specifically, for 32-bit ranges, our argument is of the same size as Bulletproofs and 18.7% larger than Bulletproofs+. For 64-bit ranges, our argument demands 9.5% and 27.7% more communication costs than the two, respectively. For 16-bit ranges, our argument is only 6.9% larger than Bulletproofs+ but 11.9% smaller than Bulletproofs. Furthermore, our argument for 8-bit ranges achieves the smallest 353 bytes proof size among these four, making it more competitive for communication-critical applications. For the computational overhead, our argument requires fewer computationally expensive group exponentiations in proving and verification than Bulletproofs-based ones. Specifically, SwiftRange achieves 1.61× and 1.32× proving efficiency for 32-bit and 64-bit ranges, respectively. More importantly, our argument improves the verification efficiency by a factor of 2.3. Another advantage lies in the security aspect. Our protocol enables tighter security than Bulletproofs-based ones in the random oracle model due to the lower round complexity as the round complexity has a strong impact on the tightness of the security loss in the random oracle model [13].

The performance of zero-knowledge proofs, namely proof size and verification efficiency, significantly impacts CT platforms' scalability, measured by transactions per second (TPS). The higher the TPS, the more transactions the platforms can handle in a given time frame. Our argument can serve as a robust alternative to Bulletproofs for confidential transactions on blockchain systems. A typical application is Monero, a leading CT platform that has dynamic demand-driven block size. Compared to Bulletproofs, our argument achieves 2.3× verification efficiency at the expense of no more than 1.1× communication cost for CT-friendly ranges. For a block consisting of hundreds of transactions, our argument has great potential to not only offset the decrease in TPS caused by the proof size gap but also to further enhance the overall TPS.

1.3. Outline of Our Paper

Our paper is organized as follows. First, we review the related work in Section 2, and introduce the cryptographic preliminaries in Section 3. We give an overview of the bit-decomposition approach in Section 4, and elaborate the core techniques of our range argument in Section 5. We describe the adapted compression protocol in Section 6. We present our full protocol and the aggregation of multiple arguments in Section 7. We provide a comprehensive performance evaluation and comparison with the state-of-the-art range arguments in Section 8. Finally, we discuss some typical applications in Section 9.

2. Related Work

This section gives a survey of most of the other existing range proofs in the literature at the time of writing, which incorporates three mainstream constructions of range proofs: bit-decomposition-based, square-decomposition-based and signature-based ones.

Bit-Decomposition Constructions. There are three other range arguments in the DLOG setting. The argument [14] has $O(N)$ complexity in communication and computation. Another argument [15] achieves logarithmic shortness, involving $2\lceil\log(2N+5)\rceil + 8$ elements. However, it merely proves that a committed bit-vector comprises zeros and ones to implicitly demonstrate that the constituted value lies in the range $[0, 2^N - 1]$, which has limited applications. The most recent one, SymmeProof [16], is another improved version of Bulletproofs, which benefits from special challenges to halve the communication complexity. Their approach requires each challenge c to satisfy a quadratic residue $c^2 \equiv 1 \pmod{p}$ over a non-standard elliptic curve group of composite order p, which may be hard to achieve in practice. On the one hand, the authors described a computationally expensive method to obtain these special challenges and suggested converting the standard challenges to these special ones when producing the arguments. However, they did not provide an efficient algorithm to achieve the conversion.

On the other hand, the computations over the non-standard groups are far less efficient than over the standard ones.

Square-Decomposition Constructions. The square decomposition consists in representing a committed value as a sum of squares to prove its non-negativity. The exploration of square-decomposition-based range proofs has witnessed over two decades of history. The first construction was proposed by Boudot [17], which represents a target value x into the sum of the greatest square less than x and a positive value. Lipmaa [18] used Lagrange's four squares theorem [19] to represent a value by the sum of four squares. Groth [6] improved the construction based on his observation that 4x+1 can always be represented into the sum of three squares for a value x. If $4x+1$ is non-negative, then $x \geq -\frac{1}{4}$ would be a non-negative integer. Deng et al. [20] designed a constant-size range proof based on the RSA assumption by adapting Bulletproofs for Lagrange's four-square theorem. However, these range proofs rely on RSA-based integer-commitment schemes, which require a trusted setup to generate RSA modulus.

Recently, two new range proofs, CKLR21 [13] and Sharp [21], revived the square-decomposition approach and presented the constructions in the DLOG setting by leveraging a bounded integer commitment scheme, where the latter is an improved version of the former. They proposed an encoding scheme to transform the standard Pedersen commitment scheme from over $\mathbb{Z}_p$ to over a small bounded integer range. Their techniques enable higher efficiency than bit-decomposition constructions in computation and communication with relaxed soundness. The proofs used a considerably smaller challenge space to accommodate the standard group sizes, e.g., 256-bit, which leads to increased soundness errors. Otherwise, the proofs must use dramatically large group sizes or multiple iterations to maintain negligible soundness errors. Moreover, their approaches restrict the provers to rational witnesses rather than integer ones in the target ranges, which becomes a hindrance to the applications of blockchain-based confidential transactions that have stringent security requirements. Sharp also presented two constructions in RSA and class groups to mitigate the relaxed soundness. Nevertheless, on the one hand, RSA groups require trusted setups. On the other hand, class groups are often too large to be used in practice. A recent study [22] suggests that 3392-bit class groups can barely achieve 128-bit security as 256-bit DLOG-based elliptic curve groups.

Signature-Based Constructions. Signature-based constructions require the verifier to pre-compute a digital signature for each element in the range. The prover then signs a selected element with a blind signature, making it computationally hard to know the signed element. Finally, the verifier checks if the blinded signature belongs to the range of the precomputed signatures. Camenisch et al. [23] proposed a range proof based on the Boneh-Boyen signature schemes under the q-Strong Diffie-Hellman assumption. An improved version was proposed [24] to reduce the communication and computational complexity by a factor of 2. However, both range proofs require a trusted setup.

3. Preliminaries

Let λ and $\text{negl}(\lambda)$ be the security parameter and a negligible function. Denote a cyclic group of prime order p by $\mathbb{G}$, and the ring of integers modulo p by $\mathbb{Z}_p$. Let $\mathbb{Z}_p^*$ be $\mathbb{Z}_p \backslash \{0\}$. Let $g, \rho, (g_i)_{i=0}^{n-1} \xleftarrow{\$} \mathbb{G}$ be uniformly random generators from $\mathbb{G}$. Let $x \xleftarrow{\$} \mathbb{Z}_p^*$ be uniformly random element from $\mathbb{Z}_p^*$. Denote the vector spaces of dimension n over $\mathbb{G}$ and $\mathbb{Z}_p$ by $\mathbb{G}^n$ and $\mathbb{Z}_p^n$, respectively. PPT stands for probabilistic polynomial time.

We will use the vector notations in our protocol. Bold font denotes vectors or matrices. For example, $\mathbf{a} = (a_0, ..., a_{n-1}) \in \mathbb{Z}_p^n$ denotes a vector of scalars. $\mathbf{g} = (g_0, ..., g_{n-1}) \in \mathbb{G}^n$ and $\boldsymbol{\rho} = (\rho_0, ..., \rho_{n-1}) \in \mathbb{G}^n$ denote two generator vectors. $|\mathbf{a}|$ denotes the dimension of the vector $\mathbf{a}$. We define some basic vector operations below:

- $c = \langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a} \cdot \mathbf{b} = \sum_{i=0}^{n-1} a_i \cdot b_i \in \mathbb{Z}_p$.
- $\mathbf{c} = \mathbf{a} + \mathbf{b} = (a_0 + b_0, ..., a_{n-1} + b_{n-1}) \in \mathbb{Z}_p^n$.
- $\mathbf{c} = \mathbf{a} \circ \mathbf{b} = (a_0 \cdot b_0, ..., a_{n-1} \cdot b_{n-1}) \in \mathbb{Z}_p^n$.
- $g' = \mathbf{g}^{\mathbf{a}} = \prod_{i=0}^{n-1} g_i^{a_i} \in \mathbb{G}$.
- $\mathbf{g}' = \mathbf{g}^{\mathbf{a}} \circ \boldsymbol{\rho} = (g_0^{a_0} \cdot \rho_0, ..., g_{n-1}^{a_{n-1}} \cdot \rho_{n-1}) \in \mathbb{G}^n$.

where $\langle \cdot, \cdot \rangle$ and $\circ$ denote the inner product and the component-wise Hadamard product, respectively.

3.1. Cryptographic Assumption

Definition 1 (**Discrete Logarithm (DLOG)**)**.** *The discrete logarithm assumption holds for all PPT adversaries $\mathcal{A}$:*

$$Pr\left[\begin{array}{l|l} (x_i)_{i=0}^{n-1} \leftarrow \mathcal{A}\big((g_i)_{i=0}^{n-1}\big), & \mathbb{G} \leftarrow \mathcal{G}(\lambda), \\ \prod_{i=0}^{n-1} g_i^{x_i} = \eta & (g_i)_{i=0}^{n-1} \xleftarrow{\$} \mathbb{G} \end{array}\right] \leq \text{negl}(\lambda)$$

where $\mathcal{G}(\lambda)$ is the setup algorithm.

The assumption states that no computationally bounded adversaries can find such non-trivial discrete logarithm relations that satisfy $\prod_{i=0}^{n-1} g_i^{x_i} = \eta$ for an arbitrary $\eta \in \mathbb{Z}_p^*$ and randomly chosen generators. To avoid a trusted setup, the random generators $(g_i)_{i=0}^{n-1}$ can be independently generated by using a collision-resistant hash function to map from random values in $\mathbb{Z}_p^*$ to $\mathbb{G}\backslash\{1\}$. Thus, the non-trivial discrete logarithm relations among the generators are unknown.

3.2. Pedersen Commitment Schemes

In this work, we consider the Pedersen commitment scheme and the Pedersen vector commitment scheme under the DLOG assumption. Both commitment schemes are:

- *Perfectly Hiding:* Computationally unbounded adversaries cannot infer any information about the committed values.
- *Computationally Binding:* Computationally bounded adversaries have negligible probability of opening a commitment to two distinct values.

We define the Pedersen vector commitment scheme and the Pedersen commitment scheme is a special case where $n = 1$.

Definition 2 (Pedersen Vector Commitment). *Given the message space $\mathcal{M} = \mathbb{Z}_p^n$, the randomness space $\mathcal{R} = \mathbb{Z}_p^*$, the commitment space $C = \mathbb{G}$ of prime order p and $(g_0, ..., g_{n-1}, \rho) \xleftarrow{\$} \mathbb{G}$:*

$$\mathrm{Com}(x_0, ..., x_{n-1}; r) \triangleq \prod_{i=0}^{n-1} g_i^{x_i} \rho^r$$

The Pedersen vector commitment satisfies the following homomorphic properties:

- $\mathrm{Com}(x_0, ..., x_{n-1}; r_x) \cdot \mathrm{Com}(y_0, ..., y_{n-1}; r_y)$ $= \mathrm{Com}(x_0 + y_0, ..., x_{n-1} + y_{n-1}; r_x + r_y)$
- $\mathrm{Com}(x_0, ..., x_{n-1}; r_x)^y = \mathrm{Com}(x_0 \cdot y, ..., x_{n-1} \cdot y; r_x \cdot y)$

We will use capital letters to denote commitments in the following paper, e.g., $X = \prod_{i=0}^{n-1} g_i^{x_i} \rho^r$.

3.3. Zero-Knowledge Argument of Knowledge

A zero-knowledge argument of knowledge is an interactive protocol Π for a relation $\mathcal{R}$ between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. It is a zero-knowledge proof with computational soundness in the sense that no probabilistic polynomial-time provers can deceive a verifier into accepting it. It takes an NP public statement u, and the prover's *witness* ω (private input), then outputs the verifier's decision on whether to accept or reject the prover's claim of knowing the witness. The messages communicated in the protocol are called a *transcript*. An interactive protocol is called *public-coin* if the verifier' messages are all randomly generated and independent of the prover' messages. The verifier' messages are also called *challenges*. The public-coin protocols can be transformed into non-interactive ones using Fiat-Shamir heuristics [25], where the prover can use the hash values of previously revealed messages as the public-coin challenges.

An interactive protocol is (perfecly) *complete* if the predicate $(u, \omega) \in \mathcal{R}$ is always true on any input. We consider $(2\mu + 1)$-move public-coin protocols, which have $(\theta_1, ..., \theta_\mu)$*-special soundness* as demonstrated in [11]. An interactive protocol is $(\theta_1, ..., \theta_\mu)$*-special sound* if there exists an efficient algorithm that, given a statement u and a $(\theta_1, ..., \theta_\mu)$-tree accepting transcripts, outputs a witness ω for u. A $(\theta_1, ..., \theta_\mu)$-tree accepting transcripts is a set of $\prod_{i=1}^{\mu} \theta_i$ accepting transcripts arranged in a tree structure, where the nodes are the prover' messages, and the edges are the verifier' challenges. Every transcript includes the messages along the path from the root to a leaf node. θ*-special soundness* is a special case of $(\theta_1, ..., \theta_\mu)$*-special soundness*, where $\mu = 1$. An interactive protocol is (perfect) *special honest-verifier zero-knowledge (SHVZK)* if given the challenges, there exists an efficient simulator that can always simulate an indistinguishable transcript of the argument without the knowledge of the witness.

4. Overview of Bit-Decomposition

Bit-decomposition is a popular approach for constructing range proofs. The goal is to find an efficient method to prove the following relation $\mathcal{R}$:

$$\left\{g, \rho, X \in \mathbb{G}, N, x, r_x \in \mathbb{Z}_p : X = g^x \rho^{r_x} \ \wedge \ x \in [0, 2^N - 1]\right\}$$

We begin with a brief overview of the techniques of Flashproofs and Bulletproofs before presenting ours.

4.1. Flashproofs

Flashproofs [8] leverage a quadratic-term cancellation technique to efficiently prove that a committed value can be represented in binary form. The intuition behind Flashproofs is that a prover expresses the committed value $x = \sum_{i=0}^{N-1} 2^i b_i$ as a sequence of terms $(w_0, w_1, ..., w_{N-1})$ for the range $[0, 2^N - 1]$, where $b_i \in \{0, 1\}$ and $w_i = 2^i b_i,\ i \in \{0, 1, ..., N - 1\}$. Then the prover folds the sequence and arranges all the terms $(w_i)_{i=0}^{N-1}$ in an $L \times K$ matrix, where L and K indicate the number of rows and columns, respectively. The prover computes a series of values $(v_l = \sum_{k=0}^{K-1} w_{lK+k} e_k + r_l)_{l=0}^{L-1}$ after acquiring a challenge vector $(e_0, ..., e_{K-1})^\intercal$ from the verifier, where r_l is a random value.

$$\begin{pmatrix} v_0 \\ \vdots \\ v_{L-1} \end{pmatrix} = \begin{pmatrix} w_0 & \cdots & w_{K-1} \\ w_K & \cdots & w_{K+K-1} \\ \vdots & \ddots & \vdots \\ w_{(L-1)K} & \cdots & w_{(L-1)K+K-1} \end{pmatrix} \cdot \begin{pmatrix} e_0 \\ \vdots \\ e_{K-1} \end{pmatrix} \tag{2}$$

The verifier computes a value f_l by subtracting v_l from $\sum_{k=0}^{K-1} 2^{lK+k} e_k$ for each $l \in \{0, ..., L - 1\}$:

$$f_l = \sum_{k=0}^{K-1} 2^{lK+k} e_k - v_l = \sum_{k=0}^{K-1} (2^{lK+k} - w_{lK+k}) e_k - r_l \tag{3}$$

Next, the verifier computes $f_l \cdot v_l$ to generate a series of cross-terms in Eqn. (5). The objective is to confirm that the quadratic terms $(e_k^2)_{k=0}^{K-1}$ in the challenges are all cancelled out. In this case, the verifier can conclude that $w_{lK+k} \in \{0, 2^{lK+k}\}$ for each $lK + k$ where $k \in \{0, ..., K - 1\}$.

$$f_l \cdot v_l \overset{?}{=} \underbrace{\sum_{k=0}^{K-1} w_{lK+k} (2^{lK+k} - w_{lK+k}) e_k^2}_{=\ 0,\ \text{if}\ w_{lK+k} \in \{0, 2^{lK+k}\}} \tag{4}$$

$$+ \sum_{k=0, j=1}^{k=K-2, j=K-1} t_{l,k,j} e_{k,j} + \sum_{k=0}^{K-1} q_{l,k} e_k + q_{l,K} \tag{5}$$

where $t_{l,k,j}$ and $q_{l,k}$ are the coefficients that must be committed by the prover before seeing the challenges.

Finally, the prover flattens the two-dimension matrix to a one-dimension vector and proves that x is the sum of K values, such that $x = \sum_{k=0}^{K-1} s_k$, where $s_k = \sum_{l=0}^{L-1} w_{lK+k}$ is the sum of L coefficients $(w_{lK+k})_{l=0}^{L-1}$ in the k-th column.

4.2. Bulletproofs

Bulletproofs [9] take advantage of an improved inner product argument [26], which allows a prover to convince a verifier of knowing two committed vectors **a** and **b** to satisfy the following relation:

$$\left\{\begin{matrix}\mathbf{g}, \boldsymbol{\rho} \in \mathbb{G}^N, P \in \mathbb{G}, \\ c \in \mathbb{Z}_p, \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^N\end{matrix} \ : \ P = \mathbf{g}^{\mathbf{a}} \boldsymbol{\rho}^{\mathbf{b}}, c = \langle a, b \rangle \right\} \tag{6}$$

The intuition behind Bulletproofs is that a prover prepares a vector commitment to the vector **b** and the vector $\mathbf{a} = \mathbf{b} - \mathbf{1^N}$, where **b** is the bit vector of x. The prover constructs Eqn. (7) to prove the three constraints: (1) $\langle \mathbf{b}, \mathbf{2^N} \rangle = x$, (2) $\langle \mathbf{b} - \mathbf{1^N} - \mathbf{a}, \mathbf{y^N} \rangle = 0$ and (3) $\langle \mathbf{b}, \mathbf{a} \circ \mathbf{y^N} \rangle = 0$.

$$z^2 \cdot \langle \mathbf{b}, \mathbf{2^N} \rangle + z \cdot \langle \mathbf{b} - \mathbf{1^N} - \mathbf{a}, \mathbf{y^N} \rangle + \langle \mathbf{b}, \mathbf{a} \circ \mathbf{y^N} \rangle = z^2 \cdot x \tag{7}$$

where $y, z \in \mathbb{Z}_p^*$ are two random challenges provided by the verifier. $\mathbf{1^N} = (1, 1, ..., 1)$ is a vector of 1. $\mathbf{2^N} = (2^0, 2^1, ..., 2^{N-1})$ and $\mathbf{y^N} = (y^0, y^1, ..., y^{N-1})$ are the vectors of powers of 2 and y. Eqn. (7) can be rewritten as follows:

$$\langle \mathbf{b} - z \cdot \mathbf{1^N}, \mathbf{y^N} \circ (\mathbf{a} + z \cdot \mathbf{1^N}) + z \cdot \mathbf{2^N} \rangle = z^2 \cdot x + \delta(y, z) \tag{8}$$

where $\delta(y, z) = (z - z^2) \cdot \langle \mathbf{1^N}, \mathbf{y^N} \rangle - z^3 \cdot \langle \mathbf{1^N}, \mathbf{2^N} \rangle \in \mathbb{Z}_p$.

The prover recursively compresses the inner product of the two vectors $\mathbf{b} - z \cdot \mathbf{1^N}$ and $\mathbf{y^N} \circ (\mathbf{a} + z \cdot \mathbf{1^N}) + z \cdot \mathbf{2^N}$ in $\log N$ rounds until the vector dimension is reduced to 1. The technique helps achieve $O(\log N)$ communication efficiency but requires at least $10N$ and $2N$ group exponentiations for proving and verification, respectively. Notably, the $2N$ group exponentiations in verification are attributed to using $2N$ generators $\mathbf{g}, \boldsymbol{\rho} \in \mathbb{G}^N$ to commit to the two vectors **b** and **a**.

In addition, Bulletproofs+ [10] follow the same idea but replace the inner product argument with a weighted inner product argument for slightly higher communication efficiency.

5. Techniques of Our Argument

5.1. Intuition

Different from Bulletproofs, we design a new variant of the bit-decomposition approach to construct zero-knowledge range arguments, which needs only one committed vector (rather than two in Bulletproofs). Given a non-negative value x, we can write it as a bit vector $\mathbf{b} = (b_0, b_1, ..., b_{N-1})$, where $b_i \in \{0, 1\}$ is the i-th bit. Then x can be represented as $x = \sum_{i=0}^{N-1} 2^i b_i$ and committed as $X = g^{\sum_{i=0}^{N-1} 2^i b_i} \rho^{r_x} \in \mathbb{G}$, where $g, \rho \in \mathbb{G}, r_x \overset{\$}{\leftarrow} \mathbb{Z}_p^*$. Notably, we apply a trick by rewriting X as $\prod_{i=0}^{N-1} (g^{2^i})^{b_i} \rho^{r_x}$, which can also be regarded as a commitment to the bit vector **b**, where the i-th generator is g^{2^i}. We define a polynomial function $f(\mathbf{b})$ for the bit vector **b** over group elements:

$$f(\mathbf{b}) \triangleq (\underbrace{\prod_{i=0}^{N-1} (g^{2^i})^{b_i}}_{=X \cdot \rho^{-r_x}})^e \cdot \underbrace{\prod_{i=0}^{N-1} g_i^{b_i - b_i^2}}_{=1} = (X \cdot \rho^{-r_x})^e \tag{9}$$

where $g, (g_i)_{i=0}^{N-1} \overset{\$}{\leftarrow} \mathbb{G}$ are $N + 1$ distinct base generators and $e \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ is an arbitrary value.

The function $f(\mathbf{b})$ consists of two multiplicative factors and should be evaluated to $(X \cdot \rho^{-r_x})^e$ if **b** is a valid bit vector. The high-level ideas are explained as below:

- The first factor $\prod_{i=0}^{N-1} (g^{2^i})^{b_i}$ aims to represent the witness x hidden in X.
- The second factor $\prod_{i=0}^{N-1} g_i^{b_i - b_i^2}$ should be equal to 1 for a valid bit vector. The hardness of the discrete logarithm relations between any two generators implies for each $i \in \{0, ..., N - 1\}$ that:

$$b_i - b_i^2 = b_i \cdot (1 - b_i) = 0 \Longleftrightarrow b_i \in \{0, 1\}$$

- The value e is used to compartmentalize the two factors. The equality must hold for an arbitrary e to ensure that they do not affect each other.

Next, we re-write Eqn. (9) as follows:

$$f(\mathbf{b}) = \prod_{i=0}^{N-1} (g^{2^i e} g_i)^{b_i} \cdot \prod_{i=0}^{N-1} g_i^{-b_i^2} = (X \cdot \rho^{-r_x})^e \tag{10}$$

$$\Longrightarrow f(\mathbf{b}) = \mathbf{h^b} \cdot \mathbf{g^{-b^2}} = (X \cdot \rho^{-r_x})^e \tag{11}$$

where we define two generator vectors by $\mathbf{g} \triangleq (g_0, ..., g_{N-1}) \in \mathbb{G}^N$ and $\mathbf{h} \triangleq (g^e g_0, ..., g^{2^{N-1} e} g_{N-1}) \in \mathbb{G}^N$. Note that **h** is a composite generator vector built on the base generators g and $(g_i)_{i=0}^{N-1}$. Next, we will construct a 5-round zero-knowledge range protocol Π_{rg} to prove the equality in Eqn. (11).

5.2. Zero-knowledge Range Protocol Π_{rg}

Let $\mathcal{P}$ and $\mathcal{V}$ be the prover and verifier, respectively. Given a commitment $X = g^x \rho^{r_x}$, $\mathcal{P}$ aims to convince $\mathcal{V}$ of knowing the witness $x \in [0, 2^N - 1]$ hidden in X. At a high level, $\mathcal{P}$ generates a linearly masking vector $\mathbf{z} = \mathbf{b} + \mathbf{m} \cdot y$ and substitutes it for the bit-vector **b** in proving $\mathbf{h^b} \cdot \mathbf{g^{-b^2}}$. $\mathcal{P}$ must ensure that the new function $\mathbf{h^z} \cdot \mathbf{g^{-z^2}}$ evaluates to the product of a series of pre-defined commitments, including X. The protocol Π_{rg} to prove Eqn. (11) is described as follows:

$$\mathcal{P} : (m_i)_{i=0}^{N-1}, r_q, r_t \overset{\$}{\leftarrow} \mathbb{Z}_p^* \tag{12}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : Q \triangleq \rho^{r_q} \prod_{i=0}^{N-1} g_i^{-m_i^2} \in \mathbb{G} \tag{13}$$

$$\mathcal{P} \Leftarrow \mathcal{V} : e \overset{\$}{\leftarrow} \mathbb{Z}_p^* \tag{14}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : T \triangleq \rho^{r_t} (g^{\sum_{i=0}^{N-1} 2^i m_i})^e \prod_{i=0}^{N-1} g_i^{(1-2b_i) m_i} \in \mathbb{G} \tag{15}$$

$$\mathcal{P} \Leftarrow \mathcal{V} : y \overset{\$}{\leftarrow} \mathbb{Z}_p^* \tag{16}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : \mathbf{z} \triangleq (z_i = b_i + m_i \cdot y)_{i=0}^{N-1} \in \mathbb{Z}_p^N \tag{17}$$

$$s \triangleq r_x \cdot e + r_t \cdot y + r_q \cdot y^2 \in \mathbb{Z}_p \tag{18}$$

$$\mathcal{V} : \mathbf{h^z} \cdot \mathbf{g^{-z^2}} \overset{?}{=} \rho^{-s} \cdot X^e \cdot T^y \cdot Q^{y^2} \tag{19}$$

where $-\mathbf{z}^2 \triangleq -\mathbf{z} \circ \mathbf{z}$.
We elaborate on Eqn. (19) with a more detailed explanation:

$$\begin{aligned}
\mathbf{h}^{\mathbf{z}} \cdot \mathbf{g}^{-\mathbf{z}^2} &= \prod_{i=0}^{N-1} (g^{2^i e} g_i)^{z_i} \cdot \prod_{i=0}^{N-1} g_i^{-z_i^2} && (20)\\
&= \prod_{i=0}^{N-1} g^{2^i e b_i + 2^i e m_i y} \cdot g_i^{b_i + m_i y - b_i^2 - 2 b_i m_i y - m_i^2 y^2} && (21)\\
&= \rho^{-s} \cdot (\underbrace{\rho^{r_x} g^{\sum_{i=0}^{N-1} 2^i b_i}}_{=X})^e \cdot \underbrace{\prod_{i=0}^{N-1} g_i^{b_i - b_i^2}}_{=1} && (22)\\
&\quad \cdot \underbrace{\left(\rho^{r_t} (g^{\sum_{i=0}^{N-1} 2^i m_i})^e \prod_{i=0}^{N-1} g_i^{(1-2b_i)m_i} \right)}_{=T}{}^{y} && (23)\\
&\quad \cdot (\underbrace{\rho^{r_q} \prod_{i=0}^{N-1} g_i^{-m_i^2}}_{=Q})^{y^2} && (24)
\end{aligned}$$

Computing $\mathbf{h}^{\mathbf{z}} \cdot \mathbf{g}^{-\mathbf{z}^2}$ will generate a tuple of multiplicative factors. The verifier needs to ensure that all the four bases ρ, X, T and Q on the right-hand side of Eqn. (19) have the proper exponents, $-s$, e, y and y^2, respectively. In the following, we provide some intuition of the protocol.

- The computation of $\mathbf{z}$ and $-\mathbf{z}^2$ results in the existence of the two exponents y and y^2 on the right-hand side, which indicates that each element of the vector $\mathbf{z}$ and y satisfies a linear relation.
- The linear form $\mathbf{z} = \mathbf{b} + \mathbf{m} \cdot y$ rather than $\mathbf{z} = \mathbf{b} \cdot y + \mathbf{m}$ should be satisfied. Otherwise, a non-one factor $\prod_{i=0}^{N-1} g_i^{m_i - m_i^2} \neq 1$ would appear on the right-hand side instead of $\prod_{i=0}^{N-1} g_i^{b_i - b_i^2} = 1$. We apply a trick here to remove the dependencies between the protocols Π_{rg} and Π_{ac} by using the former linear form to construct $\mathbf{b} - \mathbf{b}^2$ in Eqn. (22), where the witness vector $\mathbf{b}$ is put as the $\mathbf{z}$-intercepts instead of the slopes. Using the latter linear form requires to compute $e\mathbf{z} - \mathbf{z}^2$ to construct $e^2 \cdot (\mathbf{b} - \mathbf{b}^2)$. This would introduce the challenge e in the subsequent compression protocols, which increases the computational overhead.
- $\mathcal{V}$ computes the generators $(g^{2^i e})_{i=0}^{N-1}$ to ensure X is composed of the generator g by checking whether the exponent of X is the random challenge e.
- Based on the above analysis, computing $\mathbf{h}^{\mathbf{z}} \cdot \mathbf{g}^{-\mathbf{z}^2}$ gives rise to the appearance of the constant factor $\prod_{i=0}^{N-1} g_i^{b_i - b_i^2}$ on the right-hand side. $\mathcal{P}$ is required to provide X before seeing e and (T, Q) before seeing y. Without knowing e and y beforehand, it is computationally infeasible for a probabilistic polynomial-time $\mathcal{P}$ to pre-define X, T and Q such that these three factors X^e, T^y and Q^{y^2} can cancel out the factor $\prod_{i=0}^{N-1} g_i^{b_i - b_i^2}$ unless $b_i \in \{0, 1\}$ $\forall$ $i \in \{0, ..., N-1\}$. Thus, $\mathcal{V}$ would be convinced with an overwhelming probability that:

$$\big(b_i \in \{0,1\}\big)_{i=0}^{N-1} \Longleftrightarrow \prod_{i=0}^{N-1} g_i^{b_i - b_i^2} = 1$$

Remarks: The range protocol Π_{rg} achieves $O(N)$ complexity in communication and computation. Compared with Bulletproofs, our approach only uses $N+1$ distinct generators, $(g_i)_{i=0}^{N-1}$ and g. Thus, using the multi-exponentiation technique [27], the verification of our argument would be dominated by N group exponentiations instead of $2N$ required by Bulletproofs-based arguments. We will elaborate on the technique in Section 7.1.2. We stress that the protocol Π_{rg} can directly instantiate a highly short range argument for $N \leq 8$. In the next section, we describe an adapted compression protocol Π_{ac} to achieve $O(\log N)$ communication complexity for $N > 8$.

Corollary 1. *The range argument* Π_{rg} *is a 5-move protocol, which has perfect completeness, computational 3-special soundness and perfect special honest-verifier zero-knowledge (SHVZK).*
The range argument is a special case of the aggregate range argument Π_{arg} in Section 7.2 where $J = 1$. Thus, it is a corollary of Theorem 2.

6. Compression for Logarithmic Size

To achieve $O(\log N)$ communication complexity for $N > 8$, we employ a different compression protocol Π_{ac} from the one in Bulletproofs. We adapt the compressed Σ-protocol in [11] that was originally designed to reduce the communication complexity of proving a committed vector $\mathbf{z}$ satisfies a public linear relation $f(\mathbf{z}) = \mathbf{g}^{\mathbf{z}}$ from linear to logarithmic. Our protocol Π_{ac} is seamlessly compatible with our range protocol Π_{rg}. We first introduce the compressed Σ-protocol Π_{c} in [11], before describing our adapted protocol Π_{ac}.

6.1. Compression Protocol Π_{c}

Given a witness vector $\mathbf{z} \in \mathbb{Z}_p^N$, a generator vector $\mathbf{g} \in \mathbb{G}^N$ and an input $f(\mathbf{z}) \in \mathbb{G}$, the protocol Π_{c} to prove $f(\mathbf{z}) = \mathbf{g}^{\mathbf{z}}$ is described as follows:

$$\mathcal{P} \Rightarrow \mathcal{V} : A \triangleq \mathbf{g}_R^{\mathbf{z}_L},\ B \triangleq \mathbf{g}_L^{\mathbf{z}_R} \in \mathbb{G} \tag{25}$$

$$\mathcal{P} \Leftarrow \mathcal{V} : c \xleftarrow{\$} \mathbb{Z}_p^* \tag{26}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : \mathbf{z}' \triangleq \mathbf{z}_L + c\mathbf{z}_R \in \mathbb{Z}_p^{\frac{N}{2}} \tag{27}$$

$$\mathcal{V} : \mathbf{g}' \triangleq \mathbf{g}_L^c \circ \mathbf{g}_R \in \mathbb{G}^{\frac{N}{2}} \tag{28}$$

$$f(\mathbf{z}') = (\mathbf{g}')^{\mathbf{z}'} \overset{?}{=} A \cdot f(\mathbf{z})^c \cdot B^{c^2} \tag{29}$$

where $\mathbf{z}_L = (z_0, ..., z_{\frac{N}{2}-1})$ and $\mathbf{z}_R = (z_{\frac{N}{2}}, ..., z_{N-1})$ are the left-half and right-half vectors of $\mathbf{z}$, respectively. Similarly, $\mathbf{g}_L = (g_0, ..., g_{\frac{N}{2}-1})$ and $\mathbf{g}_R = (g_{\frac{N}{2}}, ..., g_{N-1})$. One can apply Π_{c} recursively, until $|\mathbf{z}'| = 2$, to achieve the minimum proof size.

We elaborate on Eqn. (29) with a more detailed explanation:

$$(\mathbf{g}')^{\mathbf{z}'} = (\mathbf{g}_L^c \circ \mathbf{g}_R)^{\mathbf{z}_L + c\mathbf{z}_R} = \underbrace{\mathbf{g}_R^{\mathbf{z}_L}}_{=A} \cdot (\underbrace{\mathbf{g}_L^{\mathbf{z}_L}\mathbf{g}_R^{\mathbf{z}_R}}_{=f(\mathbf{z})})^c \cdot (\underbrace{\mathbf{g}_L^{\mathbf{z}_R}}_{=B})^{c^2} \tag{30}$$

The compression technique hinges on Eqn. (27), which halves the dimension of the vector $\mathbf{z}$. $\mathcal{V}$ can construct new generators $\mathbf{g}'$ in Eqn. (28) by raising the left-half generators $\mathbf{g_L}$ to the power of the challenge c and compute $(\mathbf{g}')^{\mathbf{z}'}$ to check whether the input $f(\mathbf{z})$ is the base of the challenge c on the right-hand side of Eqn. (29). Then $\mathcal{P}$ can continue to run the protocol Π_{c} with the input $f(\mathbf{z}')$ to further compress the vector $\mathbf{z}'$. In this case, $\mathcal{P}$ can recursively run the protocols to reduce the dimension of the witness vector. However, this technique has a coupling issue among the recursions that the output $f(\mathbf{z})$ in Eqn. (29) must be raised to the power of the challenge c. Inductively, the exponent of $f(\mathbf{z})$ will be magnified by a factor of the challenge of each subsequent recursion. Thus, $\mathcal{V}$ must consider the challenges of all the subsequent recursions when computing the final exponent of $f(\mathbf{z})$. We will optimize the protocol to mitigate this issue and reduce the computational overheads. Note that the compression protocol Π_{c} has perfect completeness and 3-special soundness. It does not have zero-knowledge as the elements of the vector $\mathbf{z}'$ are not masked by random values.

6.2. Adapted Compression Protocol Π_{ac}

Since the function in Eqn. (9) satisfies a quadratic group relation rather than a linear one, we adapt the compression protocol Π_{c} to the quadratic setting. Given a witness vector $\mathbf{z} \in \mathbb{Z}_p^N$, two generator vectors $\mathbf{g}, \mathbf{h} \in \mathbb{G}^N$ and an input $f(\mathbf{z}) \in \mathbb{G}$, the adapted compression protocol Π_{ac} to prove $f(\mathbf{z}) = \mathbf{h}^{\mathbf{z}}\mathbf{g}^{-\mathbf{z}^2}$ is described as follows:

$$\mathcal{P} \Rightarrow \mathcal{V} : A \triangleq \mathbf{g}_R^{-\mathbf{z}_L^2},\ B \triangleq \mathbf{h}_R^{\mathbf{z}_L}\mathbf{g}_R^{-2\mathbf{z}_L\mathbf{z}_R} \in \mathbb{G} \tag{31}$$

$$D \triangleq \mathbf{h}_L^{\mathbf{z}_R}\mathbf{g}_L^{-2\mathbf{z}_L\mathbf{z}_R},\ E \triangleq \mathbf{g}_L^{-\mathbf{z}_R^2} \in \mathbb{G} \tag{32}$$

$$\mathcal{P} \Leftarrow \mathcal{V} : c \xleftarrow{\$} \mathbb{Z}_p^* \tag{33}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : \mathbf{z}' \triangleq \mathbf{z}_L + c\mathbf{z}_R \in \mathbb{Z}_p^{\frac{N}{2}} \tag{34}$$

$$\mathcal{V} : \mathbf{h}' \triangleq \mathbf{h}_L \circ \mathbf{h}_R^{c^{-1}} \in \mathbb{G}^{\frac{N}{2}} \tag{35}$$

$$\mathbf{g}' \triangleq \mathbf{g}_L \circ \mathbf{g}_R^{c^{-2}} \in \mathbb{G}^{\frac{N}{2}} \tag{36}$$

$$(\mathbf{h}')^{\mathbf{z}'}(\mathbf{g}')^{-\mathbf{z}'^2} \overset{?}{=} A^{c^{-2}}B^{c^{-1}}f(\mathbf{z})\ D^cE^{c^2} \tag{37}$$

We elaborate on Eqn. (37) with a more detailed explanation:

$$\begin{aligned}(\mathbf{h}')^{\mathbf{z}'}(\mathbf{g}')^{-\mathbf{z}'^2} &= (\mathbf{h}_L\mathbf{h}_R^{c^{-1}})^{\mathbf{z}_L+c\mathbf{z}_R}(\mathbf{g}_L\mathbf{g}_R^{c^{-2}})^{-\mathbf{z}_L^2-2\mathbf{z}_L\mathbf{z}_Rc-\mathbf{z}_R^2c^2} && (38)\\ &= (\underbrace{\mathbf{g}_R^{-\mathbf{z}_L^2}}_{=A})^{c^{-2}} \cdot (\underbrace{\mathbf{h}_R^{\mathbf{z}_L}\mathbf{g}_R^{-2\mathbf{z}_L\mathbf{z}_R}}_{=B})^{c^{-1}} && (39)\\ &\quad \cdot \underbrace{\mathbf{h}_L^{\mathbf{z}_L}\mathbf{h}_R^{\mathbf{z}_R}\mathbf{g}_L^{-\mathbf{z}_L^2}\mathbf{g}_R^{-\mathbf{z}_R^2}}_{=f(\mathbf{z})} && (40)\\ &\quad \cdot (\underbrace{\mathbf{h}_L^{\mathbf{z}_R}\mathbf{g}_L^{-2\mathbf{z}_L\mathbf{z}_R}}_{=D})^c \cdot (\underbrace{\mathbf{g}_L^{-\mathbf{z}_R^2}}_{=E})^{c^2} && (41)\end{aligned}$$

where $\mathbf{z}_L\mathbf{z}_R$, $\mathbf{z}_L^2$ and $\mathbf{z}_R^2$ are all Hadamard products, which are equivalent to $\mathbf{z}_L \circ \mathbf{z}_R$, $\mathbf{z}_L \circ \mathbf{z}_L$ and $\mathbf{z}_R \circ \mathbf{z}_R$, respectively.

We allow $\mathcal{V}$ to construct new generators $\mathbf{h}'$ and $\mathbf{g}'$ before substituting the vector $\mathbf{z}$ with the compressed vector $\mathbf{z}'$ in Eqn. (37). The quadratic relations force each recursion to use 4 additional group elements A, B, D and E on the right-hand side, which are raised to the power of c^{-2}, c^{-1}, c and c^2, respectively. The major difference from the protocol Π_{c} is that we construct new generators $\mathbf{h}'$ and $\mathbf{g}'$ by raising the two right-half generators, $\mathbf{h}_R$ and $\mathbf{g}_R$, to the power of c^{-1} and c^{-2} in Eqn. (35) and (36) such that $f(\mathbf{z})$ would be a constant factor without any exponent on the right-hand side of Eqn. (37). This optimization effectively decouples the recursions to save computational overheads such that the group elements of previous recursions no longer need to consider the challenges of subsequent recursions for verification. Likewise, for the protocol Π_{c}, $\mathcal{V}$ can construct a new generator $\mathbf{g}'$ by raising the right-half generator $\mathbf{g}_R$ to the power of c^{-1} for the sake of decoupling. Then $\mathcal{V}$ needs to compute $A^{-c} \cdot f(\mathbf{z}) \cdot B^c$ on the right-hand side of Eqn. (29). Note that the protocol Π_{ac} only contributes to reducing the communication complexity to logarithmic with little help in improving the computational efficiency.

Theorem 1. *The adapted compression protocol* Π_{ac} *is a 3-move protocol, which has perfect completeness and computational 5-special soundness.*

Please see Appendix A for the proof of Theorem 1.

7. Compressed Range Protocol Π_{crg}

We combine the range protocol Π_{rg} with a sequence of the adapted compression protocol Π_{ac} to obtain the final compressed range protocol Π_{crg} when $N > 8$. To minimize the communication overheads, the prover can recursively apply $\log N - 3$ times the compression protocol Π_{ac} to the range protocol Π_{rg} until the witness dimension is reduced to 8. We present the recursive composition of our compression protocol Π_{ac}, which is defined with the input $(\mathbf{h}, \mathbf{g}, W, \mathbf{z})$, where $\mathbf{z}$ is the witness vector of dimension $|\mathbf{z}| > 8$:

$$\text{If } |\mathbf{z}| \leq 8 : \tag{42}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : \mathbf{z} \tag{43}$$

$$\mathcal{V} : \mathbf{h}^{\mathbf{z}}\mathbf{g}^{-\mathbf{z}^2} \overset{?}{=} W \tag{44}$$

$$\text{Else :} \tag{45}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : A \triangleq \mathbf{g}_R^{-\mathbf{z}_L^2},\ B \triangleq \mathbf{h}_R^{\mathbf{z}_L}\mathbf{g}_R^{-2\mathbf{z}_L\mathbf{z}_R} \in \mathbb{G} \tag{46}$$

$$D \triangleq \mathbf{h}_L^{\mathbf{z}_R}\mathbf{g}_L^{-2\mathbf{z}_L\mathbf{z}_R},\ E \triangleq \mathbf{g}_L^{-\mathbf{z}_R^2} \in \mathbb{G} \tag{47}$$

$$\mathcal{P} \Leftarrow \mathcal{V} : c \xleftarrow{\$} \mathbb{Z}_p^* \tag{48}$$

$$\mathcal{P} \text{ and } \mathcal{V} : \mathbf{h}' \triangleq \mathbf{h}_L \circ \mathbf{h}_R^{c^{-1}} \in \mathbb{G}^{\frac{N}{2}} \tag{49}$$

$$\mathbf{g}' \triangleq \mathbf{g}_L \circ \mathbf{g}_R^{c^{-2}} \in \mathbb{G}^{\frac{N}{2}} \tag{50}$$

$$W' \triangleq A^{c^{-2}} \cdot B^{c^{-1}} \cdot W \cdot D^c \cdot E^{c^2} \tag{51}$$

$$\mathcal{P} : \mathbf{z}' = \mathbf{z}_L + c\mathbf{z}_R \in \mathbb{Z}_p^{\frac{N}{2}} \tag{52}$$

$$\text{Recursively run } \Pi_{\mathsf{ac}} \text{ on input } (\mathbf{h}', \mathbf{g}', W', \mathbf{z}') \tag{53}$$

To construct the argument Π_{crg}, $\mathcal{P}$ generates the range argument Π_{rg} in Section 5.2 before calling Π_{ac} with $W = \rho^{-s}X^eT^yQ^{y^2}$. Therefore, the final verification equation would be:

$$\hat{\mathbf{h}}^{\hat{\mathbf{z}}} \cdot \hat{\mathbf{g}}^{-\hat{\mathbf{z}}^2} \overset{?}{=} \rho^{-s}X^eT^yQ^{y^2} \prod_{l=1}^{\log N-3} A_l^{c_l^{-2}} B_l^{c_l^{-1}} D_l^{c_l} E_l^{c_l^2} \tag{54}$$

where $\hat{\mathbf{z}}$, $\hat{\mathbf{h}}$ and $\hat{\mathbf{g}}$ are the final compressed vectors of witnesses and generators.

We generalize a formula to calculate the number of elements $|\Pi| = \gamma + 4(\log N - \log\gamma) + 3$, where $\gamma = |\hat{\mathbf{z}}|$ represents the terminating dimension. Consequently, when $\gamma \in \{4, 8\}$, $|\Pi|$ reaches its minimum based on the derivative of the formula $1 - \frac{4}{\gamma \cdot \ln 2}$. The argument with $\gamma = 4$ requires four more group elements than that with $\gamma = 8$ due to an extra recursion. Hence, the argument with $\gamma = 8$ has the minimum proof size in bytes since a group element is generally larger than a field element.

Corollary 2. *The compressed range protocol* Π_{crg} *is a* $(2\log N - 1)$*-move protocol, which has perfect completeness, computational (3, 5, ..., 5)-special soundness and perfect special honest-verifier zero-knowledge (SHVZK).*

The compressed range argument is a special case of the compressed aggregate range argument Π_{carg} in Section 7.2, where $J = 1$. Therefore, it is a corollary of Theorem 3.

7.1. Optimizations

We introduce two optimization techniques to improve the efficiency of proving and verification, respectively.

7.1.1. Proving. We present an idea to improve proving efficiency, which applies to all the other range proof systems in Table 1. The prover can pre-compute a portion of group exponentiations and store the pairs of group elements and their exponents, e.g., $(m_i, g^{m_i})_{i=0}^{\infty}$, in a lookup table. When producing proofs, the prover can retrieve them directly from the look-up table to save computational overheads without the need for re-computation. The pre-computed group exponentiations include those where the bases and their exponents are already known by the prover beforehand. We summarize two cases where the group exponentiations can be pre-computed:

1) When the exponents are public constants or bits, e.g., $g^{2^i b_i}$, $g_i^{(1-2b_i)}$.
2) When the exponents are random values that are only known to the prover, e.g., $g_i^{m_i}$, $g_i^{-m_i^2}$, ρ^{r_t} and ρ^{r_q}.

With the help of the optimization technique, the prover only needs to perform one single group exponentiation, raising the value $g^{\sum_{i=0}^{N-1} 2^i m_i}$ to the power of e in Eqn. (23) when generating the commitment T in the range protocol Π_{rg}. This optimization saves $2N + 3$ group exponentiations in generating T and Q.

7.1.2. Verification. We employ the multi-exponentiation technique [27] used by Bulletproofs to improve the verification efficiency. In our compressed range protocol, the verifier must compute two generator vectors $\mathbf{h}'$ and $\mathbf{g}'$ in each round, which requires a total of $2N - 16$ group exponentiations within $\log N - 3$ rounds:

$$\underbrace{2 \cdot \frac{N}{2} + 2 \cdot \frac{N}{4} + 2 \cdot \frac{N}{8} + \ldots + 2 \cdot 8}_{\log N-3 \text{ terms}} = 2N - 16$$

However, the generators $\mathbf{h}'$ and $\mathbf{g}'$ are all computed based on the initial base generators $(g_i)_{i=0}^{N-1}$ and g. Thus, the verification can be reduced to a single multi-exponentiation of these generators by delaying all the exponentiations to the last round. The verifier can aggregate the exponents of these generators before performing one-off exponentiations for verification using Eqn. (55):

$$g^a \prod_{i=0}^{N-1} g_i^{a_i} \overset{?}{=} \rho^{-s}X^eT^yQ^{y^2} \prod_{l=1}^{\log N-3} A_l^{c_l^{-2}} B_l^{c_l^{-1}} D_l^{c_l} E_l^{c_l^2} \tag{55}$$

where a and $(a_i)_{i=0}^{N-1}$ indicate the aggregated exponents for the corresponding generators. Finally, the multi-exponentiation optimization helps reduce $2N - 16$ group exponentiations to only $N + 1$.

7.2. Aggregate Range Argument

SwiftRange also supports the aggregation of multiple single arguments for further efficiency improvements in communication and verification. The aggregation allows a prover to prove J committed values lie in a specific range at the same time. We call it a J-aggregate argument. Given J witnesses $(x_j)_{j=0}^{J-1}$ and their commitments $(X_j = g^{x_j}\rho^{r_{x_j}})_{j=0}^{J-1}$, each witness can be written in its binary form $x_j = \sum_{i=0}^{N-1} 2^i b_{j,i}$, where $b_{j,i} \in \{0, 1\}$. We use $J \cdot N$ generators to combine J range arguments based on the following equality:

$$f(\mathbf{b}) = \prod_{j=0}^{J-1} \Big(\prod_{i=0}^{N-1} g^{2^i b_{j,i}} \Big)^{e^{j+1}} \cdot \underbrace{\prod_{j=0}^{J-1} \prod_{i=0}^{N-1} g_{j,i}^{b_{j,i} - b_{j,i}^2}}_{=1} = \prod_{j=0}^{J-1} X_j^{e^{j+1}} \cdot \rho^{-s}$$

where $e \in \mathbb{Z}_p^*$ is an arbitrary value and $s = \sum_{j=0}^{J-1} r_{x_j} \cdot e^{j+1}$ We can re-write the above equation:

$$f(\mathbf{b}) = \prod_{j=0}^{J-1} \prod_{i=0}^{N-1} (g^{2^i e^{j+1}} g_{j,i})^{b_{j,i}} \cdot \prod_{j=0}^{J-1} \prod_{i=0}^{N-1} g_{j,i}^{-b_{j,i}^2} \tag{56}$$

$$= \prod_{j=0}^{J-1} X_j^{e^{j+1}} \cdot \rho^{-s} \tag{57}$$

$$\implies f(\mathbf{b}) = \mathbf{h}^{\mathbf{b}} \cdot \mathbf{g}^{-\mathbf{b}^2} = \mathbf{X}^{\mathbf{e}} \cdot \rho^{-s} \tag{58}$$

where $\mathbf{h} = \mathbf{h}_0 \ || \ \mathbf{h}_1 \ || \ \ldots \ || \ \mathbf{h}_{J-1}$ and $||$ is a concatenation operator. $\mathbf{h}_j = (g^{e^j} g_{j,0}, g^{2e^j} g_{j,1}, \ldots, g^{2^{N-1}e^j} g_{j,N-1})$, $\mathbf{g} =$

$(g_{0,0}, ..., g_{0,N-1}, g_{1,0}, ..., g_{1,N-1}, ..., g_{J-1,0}, ..., g_{J-1,N-1})$, $\mathbf{b} = (b_{0,0}, ..., b_{0,N-1}, b_{1,0}, ..., b_{1,N-1}, ..., b_{J-1,0}, ..., b_{J-1,N-1})$. $\mathbf{X} = (X_0, ..., X_{J-1})$ and $\mathbf{e} = (e, ..., e^J)$.

Next, we describe a generalized zero-knowledge aggregate range protocol Π_{arg}. Given J commitments $(X_j = g^{x_j} \rho^{r_{x_j}})_{j=0}^{J-1}$, $\mathcal{P}$ can simultaneously convince $\mathcal{V}$ of knowing the witnesses $(x_j \in [0, 2^N - 1])_{j=0}^{J-1}$:

$$\mathcal{P} : (m_{j,i})_{j=0,i=0}^{J-1,N-1}, r_q, r_t \xleftarrow{\$} \mathbb{Z}_p^* \tag{59}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : Q \triangleq \rho^{r_q} \prod_{j=0}^{J-1} \prod_{i=0}^{N-1} g_{j,i}^{-m_{j,i}^2} \tag{60}$$

$$\mathcal{P} \Leftarrow \mathcal{V} : e \xleftarrow{\$} \mathbb{Z}_p^* \tag{61}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : T \triangleq \rho^{r_t} g^{\sum_{j=0}^{J-1} (\sum_{i=0}^{N-1} 2^i m_{j,i}) e^{j+1}} \prod_{j=0}^{J-1} \prod_{i=0}^{N-1} g_{j,i}^{(1-2b_{j,i}) m_{j,i}} \tag{62}$$

$$\mathcal{P} \Leftarrow \mathcal{V} : y \xleftarrow{\$} \mathbb{Z}_p^* \tag{63}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : \mathbf{z} \triangleq (z_{j,i} = b_{j,i} + m_{j,i} \cdot y)_{j=0,i=0}^{J-1,N-1} \in \mathbb{Z}_p^{JN} \tag{64}$$

$$s \triangleq \sum_{j=0}^{J-1} r_{x_j} \cdot e^{j+1} + r_t \cdot y + r_q \cdot y^2 \tag{65}$$

$$\mathcal{V} : \mathbf{h}^{\mathbf{z}} \cdot \mathbf{g}^{-\mathbf{z}^2} \overset{?}{=} \rho^{-s} \cdot \prod_{j=0}^{J-1} X_j^{e^{j+1}} \cdot T^y \cdot Q^{y^2} \tag{66}$$

The dimension of the witness vector is $J \times N$. We can recursively apply the protocol Π_{ac} to Π_{arg} to obtain the compressed aggregate range protocol Π_{carg} until the dimension of the vector $\mathbf{z}$ is reduced to 8. The number of elements for a J-aggregate argument would be $4 \log(JN) - 1$. The single argument is a special case of the aggregate argument, where $J = 1$. We can observe that doubling the aggregation size would involve 4 additional group elements. The verification of a J-aggregate argument can be reduced to a multi-exponentiation of $JN + 4\log(JN) + J - 8$, which is more cost-effective than naively performing $JN + 4J \log N - 7J$ group exponentiations to verify J single arguments.

Theorem 2. *The aggregate range argument* Π_{arg} *is a 5-move protocol, which has perfect completeness, computational (J+2)-special soundness and perfect special honest-verifier zero-knowledge (SHVZK).*

Please see Appendix B for the proof of Theorem 2.

Theorem 3. *The compressed aggregate range argument* Π_{carg} *is a* $\big(2\log(JN) - 1\big)$*-move protocol, which has perfect completeness, computational (J+2, 5, ..., 5)-special soundness and perfect special honest-verifier zero-knowledge (SHVZK).*

Please see Appendix C for the proof of Theorem 3.

8. Experimental Evaluation

We conducted comprehensive performance evaluations to benchmark SwiftRange against the state-of-the-art Flashproofs and Bulletproofs-based arguments. In our experiments, we employed the standard elliptic curve group on Ethereum, *BN-128*[4], for the Pedersen commitment schemes. This elliptic curve has 254-bit keys and guarantees the same 127-bit security as RSA groups of 3072-bit order based on the NIST recommendations[5]. Furthermore, we used the well-known Bouncy Castle Crypto APIs [28] to implement the BN-128 elliptic curve for a fair comparison since they were initially used in the Java implementations of Bulletproofs[6] and Flashproofs[7]. All the experiments were executed on the Java Virtual Machine 15 in a single thread with an Apple M1 Pro processor. Note that the Java implementations were only intended for performance evaluation, which may not yield the optimal efficiency in practice. Rust and C programming languages are more suitable candidates for practical efficiency.

8.1. Line Charts

8.1.1. Computational Overhead. We measured the running time of proving and verification of Flashproofs, Bulletproofs and SwiftRange, where the pre-computation optimization was applied to the three arguments and the multi-exponentiation optimization was applied to the two compression-friendly ones for fair efficiency comparisons.

For single arguments, Figure 1a and 1b graphically show the running time of proving and verification in milliseconds. Our experimental results show that Flashproofs run the fastest, followed by SwiftRange. Bulletproofs compare unfavorably with SwiftRange in terms of computational efficiency. Specifically, SwiftRange achieves 1.61× and 1.32× proving efficiency for 32-bit and 64-bit ranges, respectively. What is more, SwiftRange runs 2.3× as fast as Bulletproofs in verification for both ranges. We also compared the proving running time of our unoptimized and optimized single arguments in Table 3. It is noteworthy that our pre-computation optimization remarkably improves the proving efficiency. Regarding the aggregate arguments, Figure 2a and 2b present the running time comparisons in seconds for 64-bit ranges. The running time of both compression-friendly arguments grows linearly with the increased aggregation size. Compared with Bulletproofs, SwiftRange still maintains a slight and a significant advantages in the efficiency of proving and verification, respectively. However, neither of them can outmatch Flashproofs. Additionally, Table 4 shows a decent efficiency improvement of SwiftRange in verification that benefits from our aggregation technique in Section 7.2.

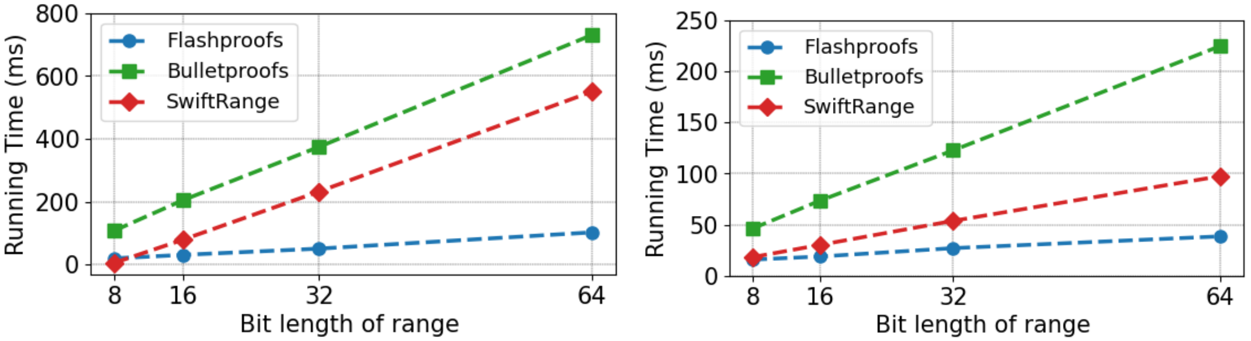
8.1.2. Communication Overhead. We measured the proof sizes in bytes over a 256-bit field. To save space, we used the compressed form of elliptic curve points, which can be stored as a 256-bit value plus an extra bit indicating one of the two possible y coordinates.

[4] Other standard elliptic curves can also be used, e.g., secp256k1.

[5] https://www.keylength.com/en/4/

[6] https://github.com/bbuenz/BulletProofLib

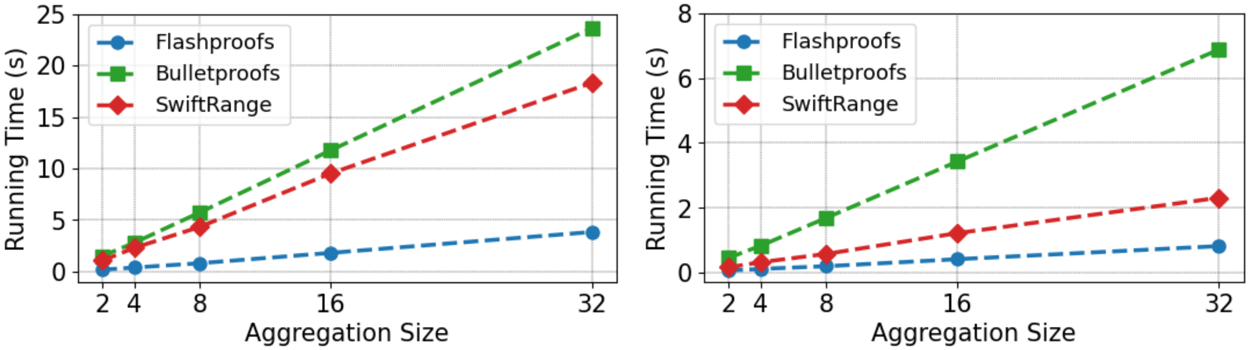
[7] https://github.com/wangnan-vincent/Flashproofs



(a) The proving running time in milliseconds.

(b) The verification running time in milliseconds.

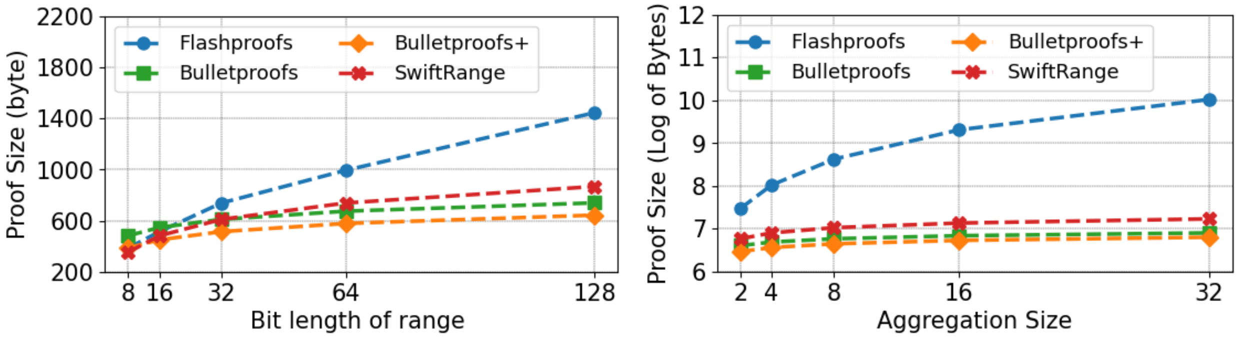
Figure 1: The computational overheads of single arguments.



(a) The proving running time in seconds.

(b) The verification running time in seconds.

Figure 2: The computational overheads of aggregate range arguments.



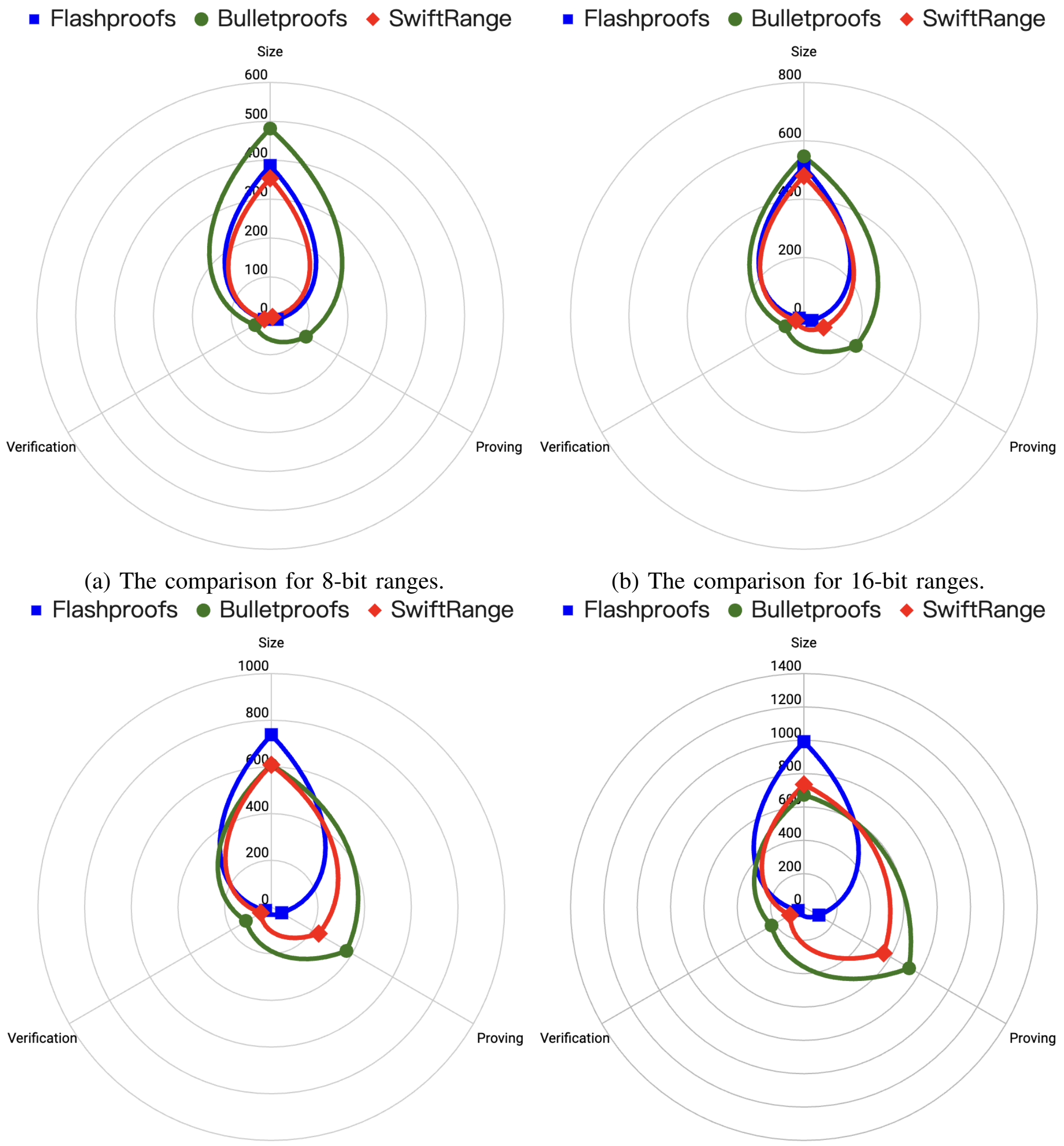
(a) The proof sizes of single arguments.

(b) The proof sizes of 64-bit aggregate arguments.

Figure 3: The communication overheads of range arguments.

We plotted a line graph in Figure 3a to provide a more straightforward comparison of single arguments than Table 2. The proof sizes of the compression-friendly arguments grow logarithmically as N increases, whereas that of Flashproofs grows far quicker than the other three. SwiftRange exhibits a slightly sharper growth in the proof size from the smallest 353 bytes as N grows. Figure 3b illustrates a comparison of 64-bit aggregate range arguments. We can see that the compression-friendly arguments have a significant advantage over Flashproofs. Our aggregate argument presents a logarithmically quicker growth with the increased aggregate size than Bulletproofs-based ones. Specifically, 22 million[8] 64-bit 4-aggregate Bulletproofs and Bulletproofs+ take up around 16.4 GB and 14.4 GB, which are 19.2% and 29.1% smaller than SwiftRange, respectively. For 32-

[8] In Bulletproofs [9], the authors estimated the proof sizes of the aggregate range arguments required for 22 million Bitcoin transactions. So we apply the same to Bulletproofs+ and SwiftRange for comparison.



(a) The comparison for 8-bit ranges.

(b) The comparison for 16-bit ranges.

(c) The comparison for 32-bit ranges.

(d) The comparison for 64-bit ranges.

Figure 4: The efficiency comparisons of different ranges in radar charts concerning proof size, proving and verification running time, where the closer the vertices to the center, the higher efficiency.

TABLE 3: The comparison of proving running time in milliseconds of different ranges between our optimized and unoptimized single arguments.

N	8	16	32	64
Unoptimized (ms)	30.6	117.5	309.2	702.5
Optimized (ms)	5.6	78.7	232	550
Saving (%)	82%	33%	25%	22%

bit arguments, the figures are reduced to 14.7% and 26%, respectively. However, we can argue that both arguments demand more than twice the verification time as SwiftRange.

TABLE 4: The comparison of verification running time in milliseconds between a J-aggregate argument and J single arguments for 64-bit ranges.

J	2	4	8	16	32
J-Aggregate (ms)	164	324	579	1218	2310
J Singles (ms)	195	390	780	1560	3120

8.2. Radar Charts

We provide four radar charts in Figure 4 to give more straightforward efficiency comparisons of the three bit-decomposition-based arguments for different ranges, which

TABLE 5: Suggested applications of different range sizes.

Range	8-bit	16-bit	32-bit	64-bit
Applications	age, humidity, temperature, exam grades, etc.	utilities bills, vehicle speed, healthcare data (e.g. blood pressure), height, weight, postcode, etc.	transaction amounts, timestamps, votes, etc.	transaction amounts, timestamps, votes, etc.

can help users select appropriate range arguments for their specific scenarios.

For 8-bit ranges, SwiftRange outperforms Flashproofs in both proving and communication efficiency. For 16-bit ranges, SwiftRange loses its proving advantage but still incurs less communication overheads than Flashproofs. Bulletproofs are the least efficient arguments for both ranges since the radar charts of Bulletproofs fully enclose those of Flashproofs and SwiftRange. The features of the three arguments start to emerge when range sizes become larger. The communication advantage of Bulletproofs and SwiftRange over Flashproofs stands out prominently when $N \geq 32$. In comparison with Bulletproofs alone, SwiftRange starts to incur more communication overheads when $N \geq 64$. We can also see that the efficiency gap of proving between Bulletproofs and SwiftRange gradually diminishes whereas that of verification becomes larger as N increases.

In summary, Flashproofs are more appropriate for computation-critical applications, whereas Bulletproofs and SwiftRange are better candidates for communication-critical applications. For 32-bit and smaller ranges, SwiftRange entirely outperforms Bulletproofs. For 64-bit ranges, SwiftRange maintains the significant advantage over Bulletproofs in verification efficiency but slightly falls behind in communication efficiency. For ranges larger than 64-bit, SwiftRange is more suitable for the scenarios where computational efficiency matters far more than communication efficiency.

9. Typical Applications

Our range argument can be applied to various practical applications for preserving privacy, not limited to confidential transactions on blockchain systems. We exemplify some applications of different range sizes in Table 5. We believe that our range argument can be applied to even more scenarios beyond the examples as below:

Black-Box Accumulation (BBA). Black-box accumulation enables privacy-preserving point collection and redemption on cryptographic tokens, which is an important building-block for various user-centric protocols such as loyalty, payment and incentive systems. The studies [29], [30] proposed privacy-preserving BBA schemes for resource-constrained wearable devices and smart phones. When a user spends points, she needs to use range proofs to ensure the committed balances are non-negative after deduction. Thus, the scheme [30] used 16-bit and 32-bit Bulletproofs to guarantee the validity of the updated balances at the time of their writing. Based on the Figure 4b, it can easily be observed that our argument is a stronger alternative for these systems as it has higher computational and communication efficiency for 16-bit and 32-bit ranges than Bulletproofs.

Crowdsensing. Privacy-preserving crowdsensing [31] relies on users voluntarily contributing their data to gain global knowledge that benefits all participants. One typical application is privacy-preserving transportation planning, where users share their speed data to enable others to predict traffic conditions and travel times. Each user is required to submit speed commitments for a large number of roads, while keeping their own location hidden. Rather than generating separate range proofs for each commitment to demonstrate the non-negativity of the committed speed, each user can provide a short aggregate argument to save communication costs. Subsequently, all users verify each other's arguments before jointly performing multi-party computation protocols to compute the average speed values of all roads in a privacy-preserving manner. Benefiting from the logarithmic shortness and high verification efficiency, our range argument serves as a more competitive candidate than Bulletproofs-based arguments to achieve real-time feedback for timely traffic predictions.

Collaborative Consumption. Privacy-preserving collaborative consumption is becoming one of the most typical practices in real-world scenarios. It enables fair cost-sharing for using resources or services among a group of users. For example, [32] proposed an energy sharing application, where a group of users can share the costs of an energy storage service at peak electricity usage periods to avoid purchasing costly power from the grid. Each user is required to submit demand commitments for a series of time slots along with range arguments to demonstrate the non-negativity of the committed demand. All the users can then jointly compute the fairly-split payments in a secure manner. Our argument is a suitable solution that provides high communication and computational efficiency.

Smart IoT. An increasing number of battery-powered IoT (Internet of Things) devices equipped with sophisticated sensors have been deployed ubiquitously thanks to ease of use and flexible installation. Untrusted IoT devices are encouraged to report their collected data via wireless communication modules to form global knowledge collectively. For the sake of privacy concerns, the data must be committed or encrypted before submission. Range proofs can effectively prevent malicious devices from polluting global knowledge by restricting the submitted data to allowed ranges. Besides, the lifespan of IoT devices essentially hinges on energy consumption. The more transferred data and the more intensive computations, the more energy consumption. Our range argument, especially the 8-bit one, can be a robust candidate for these scenarios.

10. Conclusion

In this paper, we presented SwiftRange, a new type of logarithmic-sized zero-knowledge range argument with a transparent setup in the discrete logarithm setting. Our range argument can be a drop-in substitute for Bulletproofs-based arguments in blockchain-based confidential transactions and many other privacy-preserving applications as it has higher computational efficiency and lower round complexity while incurring comparable communication overheads for CT-friendly ranges. Moreover, our argument has a smaller proof size when the range size is smaller than 32-bit, making it suitable for more communication-critical applications. In the future, we will improve its efficiency by further reducing the proof size or verification running time. We also hope that our work can be applied to a wider range of privacy-preserving applications [31]–[33].

Acknowledgments

We would like to express our sincere gratitude to the anonymous reviewers and shepherd for their valuable comments and suggestions, which helped significantly improve this paper. This work was supported by ARC Discovery Project No: GA69027/DP200101985.

References

[1] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Symposium on the Theory of Computing*, 1985.

[2] H. Lycklama, L. Burkhalter, A. Viand, N. Küchler, and A. Hithnawi, "Rofl: Robustness of secure federated learning," 2021. [Online]. Available: https://arxiv.org/abs/2107.03311

[3] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact e-cash," in *Advances in Cryptology – EUROCRYPT 2005*, R. Cramer, Ed., 2005.

[4] S. Canard, A. Gouget, and E. Hufschmitt, "A handy multi-coupon system," in *Applied Cryptography and Network Security*, J. Zhou, M. Yung, and F. Bao, Eds., 2006.

[5] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard, "Practical multi-candidate election system," ser. PODC '01, 2001.

[6] J. Groth, "Non-interactive zero-knowledge arguments for voting," in *Applied Cryptography and Network Security*, 2005.

[7] M. Gregory, "Confidential transactions," https://elementsproject.org/features/confidential-transactions/investigation, 2016.

[8] N. Wang and S. C.-K. Chau, "Flashproofs: Efficient zero-knowledge arguments of range and polynomial evaluation with transparent setup," in *Advances in Cryptology – ASIACRYPT 2022*, S. Agrawal and D. Lin, Eds., 2022, pp. 219–248.

[9] B. Bunz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," 05 2018, pp. 315–334.

[10] H. Chung, K. Han, C. Ju, M. Kim, and J. H. Seo, "Bulletproofs+: Shorter proofs for a privacy-enhanced distributed ledger," *IEEE Access*, vol. 10, pp. 42 081–42 096, 2022.

[11] T. Attema, R. Cramer, and S. Fehr, "Compressing proofs of k-out-of-n partial knowledge," in *Advances in Cryptology – CRYPTO 2021*, T. Malkin and C. Peikert, Eds. Cham: Springer International Publishing, 2021, pp. 65–91.

[12] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology – EUROCRYPT 2016*, 2016.

[13] G. Couteau, M. Klooß, H. Lin, and M. Reichle, "Efficient range proofs with transparent setup from bounded integer commitments," in *Advances in Cryptology - CRYPTO 2021*, 2021.

[14] J. Bootle and J. Groth, "Efficient batch zero-knowledge arguments for low degree polynomials," in *Public-Key Cryptography – PKC 2018*, 2018.

[15] T. Attema and R. Cramer, "Compressed-protocol theory and practical application to plug & play secure algorithmics," in *Advances in Cryptology – CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part III*, 2020, p. 513–543.

[16] S. Gao, Z. Peng, F. Tan, Y. Zheng, and B. Xiao, "Symmeproof: Compact zero-knowledge argument for blockchain confidential transactions," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2022.

[17] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Advances in Cryptology — EUROCRYPT 2000*.

[18] H. Lipmaa, "On diophantine complexity and statistical zero-knowledge arguments," in *Advances in Cryptology - ASIACRYPT 2003*, 2003.

[19] E. W. Weisstein, "Lagrange's four-square theorem," https://mathworld.wolfram.com/LagrangesFour-SquareTheorem.html, 2021.

[20] C. Deng, X. Tang, L. You, and G. Hu, "Cuproof: A novel range proof with constant size," *IACR Cryptol. ePrint Arch.*, 2021.

[21] G. Couteau, D. Goudarzi, M. Klooß, and M. Reichle, "Sharp: Short relaxed range proofs," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22, 2022, p. 609–622.

[22] S. Dobson, S. Galbraith, and B. Smith, "Trustless unknown-order groups," *Mathematical Cryptology*, vol. 1, no. 2, p. 25–39, Mar. 2022. [Online]. Available: https://journals.flvc.org/mathcryptology/article/view/130579

[23] J. Camenisch, R. Chaabouni, and A. Shelat, "Efficient protocols for set membership and range proofs," in *Advances in Cryptology - ASIACRYPT 2008*, 2008.

[24] R. Chaabouni, H. Lipmaa, and A. Shelat, "Additive combinatorics and discrete logarithm based range protocols," in *Information Security and Privacy*, R. Steinfeld and P. Hawkes, Eds., 2010, pp. 336–351.

[25] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology — CRYPTO' 86*, 1987.

[26] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *Advances in Cryptology – EUROCRYPT 2016*, 2016.

[27] N. Pippenger, "On the evaluation of powers and monomials," *SIAM J. Comput.*, vol. 9, no. 2, p. 230–250, may 1980. [Online]. Available: https://doi.org/10.1137/0209022

[28] BouncyCastle, "Bouncycastle," https://www.bouncycastle.org/.

[29] G. Hartung, M. Hoffmann, M. Nagel, and A. Rupp, "Bba+: Improving the security and applicability of privacy-preserving point collection," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, 2017, p. 1925–1942. [Online]. Available: https://doi.org/10.1145/3133956.3134071

[30] M. Hoffmann, M. Klooß, M. Raiber, and A. Rupp, "Black-box wallets: Fast anonymous two-way payments for constrained devices," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, pp. 165–194, 01 2020.

[31] H. Zhu, N. Wang, S. C.-K. Chau, and M. Khonji, "Blockchain-enabled decentralized anonymous crowdsourcing based on anonymous payments," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, ser. ICBC, 2023.

[32] N. Wang, S. C.-K. Chau, and Y. Zhou, "Privacy-preserving energy storage sharing with blockchain," in *Proceedings of the ACM International Conference on Future Energy Systems*, ser. e-Energy, 2021, p. 185–198. [Online]. Available: https://doi.org/10.1145/3447555.3464869

[33] S. C.-K. Chau and Y. Zhou, "Blockchain-enabled decentralized privacy-preserving group purchasing for retail energy plans," in *Proceedings of the ACM International Conference on Future Energy Systems*, ser. e-Energy, 2022, pp. 172–187. [Online]. Available: https://doi.org/10.1145/3538637.3538848

Appendix A. Proof of Theorem 1

Proof. **Perfect completeness** follows by carefully inspecting the verification equation in Π_{ac}:

$$(\mathbf{h}')^{\mathbf{z}'}(\mathbf{g}')^{-\mathbf{z}'^2} = \underbrace{(\mathbf{g}_R^{-\mathbf{z}_L^2})^{c^{-2}}}_{=A} \cdot \underbrace{(\mathbf{h}_R^{\mathbf{z}_L}\mathbf{g}_R^{-2\mathbf{z}_L\mathbf{z}_R})^{c^{-1}}}_{=B} \cdot \underbrace{\mathbf{h}_L^{\mathbf{z}_L}\mathbf{h}_R^{\mathbf{z}_R}\mathbf{g}_L^{-\mathbf{z}_L^2}\mathbf{g}_R^{-\mathbf{z}_R^2}}_{=f(\mathbf{z})} \cdot \underbrace{(\mathbf{h}_L^{\mathbf{z}_R}\mathbf{g}_L^{-2\mathbf{z}_L\mathbf{z}_R})^{c}}_{=D} \cdot \underbrace{(\mathbf{g}_L^{-\mathbf{z}_R^2})^{c^2}}_{=E}$$

where $\mathbf{z}_L\mathbf{z}_R$, $\mathbf{z}_L^2$ and $\mathbf{z}_R^2$ are all Hadamard products, which are equivalent to $\mathbf{z}_L \circ \mathbf{z}_R$, $\mathbf{z}_L \circ \mathbf{z}_L$ and $\mathbf{z}_R \circ \mathbf{z}_R$, respectively.

Then, we prove **computational 5-special soundness**. Note that the soundness of our zero-knowledge argument is based on the hardness of the discrete logarithm assumption that the non-trivial discrete logarithm relations among the public generators are unknown to all computationally bounded adversaries. If the soundness holds, we can easily extract the witnesses in expected polynomial-time. Otherwise, if the soundness fails to hold, we can instead extract the discrete logarithm relations, which breaks the discrete logarithm assumption. A knowledge emulator runs the argument in expected polynomial time and rewinds the prover until it acquires five accepting transcripts. Then we compute the challenge matrix $\mathbf{c}$ in Eqn. (67), which is invertible for being a special Vandermonde matrix:

$$\mathbf{c} = \begin{pmatrix} c_0^{-2} & c_0^{-1} & 1 & c_0 & c_0^2 \\ c_1^{-2} & c_1^{-1} & 1 & c_1 & c_1^2 \\ c_2^{-2} & c_2^{-1} & 1 & c_2 & c_2^2 \\ c_3^{-2} & c_3^{-1} & 1 & c_3 & c_3^2 \\ c_4^{-2} & c_4^{-1} & 1 & c_4 & c_4^2 \end{pmatrix} \tag{67}$$

Finally, we can acquire the openings of the commitments A, B, $f(\mathbf{z})$ and D, E at each generator on the right-hand side of Eqn. (37). For the openings at the $\frac{N}{2}$ generators of the left-hand vector $\mathbf{h_L}$ and the right-hand vector $\mathbf{h_R}$, we can compute the openings at these generators using Eqn. (68) and (69), respectively:

$$\mathbf{c}^{-1} \cdot \begin{pmatrix} {z'_0}^{(0)} & \cdots & {z'_{\frac{N}{2}-1}}^{(0)} \\ {z'_0}^{(1)} & \cdots & {z'_{\frac{N}{2}-1}}^{(1)} \\ {z'_0}^{(2)} & \cdots & {z'_{\frac{N}{2}-1}}^{(2)} \\ {z'_0}^{(3)} & \cdots & {z'_{\frac{N}{2}-1}}^{(3)} \\ {z'_0}^{(4)} & \cdots & {z'_{\frac{N}{2}-1}}^{(4)} \end{pmatrix} \tag{68}$$

$$\mathbf{c}^{-1} \cdot \begin{pmatrix} c_0^{-1}{z'_0}^{(0)} & \cdots & c_0^{-1}{z'_{\frac{N}{2}-1}}^{(0)} \\ c_1^{-1}{z'_0}^{(1)} & \cdots & c_1^{-1}{z'_{\frac{N}{2}-1}}^{(1)} \\ c_2^{-1}{z'_0}^{(2)} & \cdots & c_2^{-1}{z'_{\frac{N}{2}-1}}^{(2)} \\ c_3^{-1}{z'_0}^{(3)} & \cdots & c_3^{-1}{z'_{\frac{N}{2}-1}}^{(3)} \\ c_4^{-1}{z'_0}^{(4)} & \cdots & c_4^{-1}{z'_{\frac{N}{2}-1}}^{(4)} \end{pmatrix} \tag{69}$$

Similarly, we can obtain the openings at the $\frac{N}{2}$ generators of the left-hand vector $\mathbf{g_L}$ and the right-hand vector $\mathbf{g_R}$ based on Eqn. (70) and (71), respectively:

$$\mathbf{c}^{-1} \cdot \begin{pmatrix} -({z'_0}^{(0)})^2 & \cdots & -({z'_{\frac{N}{2}-1}}^{(0)})^2 \\ -({z'_0}^{(1)})^2 & \cdots & -({z'_{\frac{N}{2}-1}}^{(1)})^2 \\ -({z'_0}^{(2)})^2 & \cdots & -({z'_{\frac{N}{2}-1}}^{(2)})^2 \\ -({z'_0}^{(3)})^2 & \cdots & -({z'_{\frac{N}{2}-1}}^{(3)})^2 \\ -({z'_0}^{(4)})^2 & \cdots & -({z'_{\frac{N}{2}-1}}^{(4)})^2 \end{pmatrix} \tag{70}$$

$$\mathbf{c}^{-1} \cdot \begin{pmatrix} -c_0^{-2}({z'_0}^{(0)})^2 & \cdots & -c_0^{-2}({z'_{\frac{N}{2}-1}}^{(0)})^2 \\ -c_1^{-2}({z'_0}^{(1)})^2 & \cdots & -c_1^{-2}({z'_{\frac{N}{2}-1}}^{(1)})^2 \\ -c_2^{-2}({z'_0}^{(2)})^2 & \cdots & -c_2^{-2}({z'_{\frac{N}{2}-1}}^{(2)})^2 \\ -c_3^{-2}({z'_0}^{(3)})^2 & \cdots & -c_3^{-2}({z'_{\frac{N}{2}-1}}^{(3)})^2 \\ -c_4^{-2}({z'_0}^{(4)})^2 & \cdots & -c_4^{-2}({z'_{\frac{N}{2}-1}}^{(4)})^2 \end{pmatrix} \tag{71}$$

Once we have these openings, the following relations hold for each challenge c:

$$\begin{aligned} A^{c^{-2}}B^{c^{-1}}f(\mathbf{z})\ D^{c}E^{c^2} &= \mathbf{h}_L^{\mathbf{z}_L+c\mathbf{z}_R} \cdot \mathbf{h}_R^{c^{-1}(\mathbf{z}_L+c\mathbf{z}_R)} \\ &\quad \cdot \mathbf{g}_L^{-(\mathbf{z}_L^2+2\mathbf{z}_L\mathbf{z}_R c+\mathbf{z}_R^2 c^2)} \\ &\quad \cdot \mathbf{g}_R^{-c^{-2}(\mathbf{z}_L^2+2\mathbf{z}_L\mathbf{z}_R c+\mathbf{z}_R^2 c^2)} \\ &= (\mathbf{h_L} \cdot \mathbf{h_R}^{c^{-1}})^{\mathbf{z}_L+c\mathbf{z}_R} \\ &\quad \cdot (\mathbf{g_L} \cdot \mathbf{g_R}^{c^{-2}})^{-(\mathbf{z}_L+c\mathbf{z}_R)^2} \end{aligned}$$

Note that $h_i = g^{2^i e}g_i$, and $\mathbf{h_L}, \mathbf{h_R}$ can be expressed by $\mathbf{g_L}, \mathbf{g_R}, g$. If the above relations do not hold, then we can extract the non-trivial discrete logarithm relations among the independently generated g and $(g_i)_{i=0}^{N-1}$, which breaks the DLOG assumption.

Appendix B. Proof of Theorem 2

Proof. **Perfect completeness** follows by carefully inspecting the verification equation in Π_{arg}:

$$\begin{aligned}
&\mathbf{h}^{\mathbf{z}} \cdot \mathbf{g}^{-\mathbf{z}^2} \\
&= \rho^{-s} \cdot \prod_{j=0}^{J-1} (\underbrace{\rho^{r_{x_j}} g^{\sum_{i=0}^{N-1} 2^i b_{j,i}}}_{=X_j})^{e^{j+1}} \cdot \underbrace{\prod_{j=0}^{J-1} \prod_{i=0}^{N-1} g_{j,i}^{b_{j,i}-b_{j,i}^2}}_{=1} \\
&\quad \cdot \underbrace{\big(\rho^{r_t} g^{\sum_{j=0}^{J-1} (\sum_{i=0}^{N-1} 2^i m_{j,i}) e^{j+1}} \prod_{j=0}^{J-1} \prod_{i=0}^{N-1} g_{j,i}^{(1-2b_{j,i}) m_{j,i}} \big)^{y}}_{=T} \\
&\quad \underbrace{\cdot (\rho^{r_q} \prod_{j=0}^{J-1} \prod_{i=0}^{N-1} g_{j,i}^{-m_{j,i}^2})^{y^2}}_{=Q}
\end{aligned}$$

Then we describe a **perfect SHVZK** simulation. A zero-knowledge argument ensures that no information, other than the witness, can be deduced. To establish that an argument possesses SHVZK, a general approach involves creating a simulator that is aware of the challenge and can produce the entire interaction of the argument without having knowledge of the witness. Given two random challenges (e, y) and a series of target commitments $(X_j)_{j=0}^{J-1}$, a simulator can simulate the transcript $(T, Q, \mathbf{z}, s)$: it randomly chooses one group element Q, a vector of field elements $\mathbf{z}$ and a field element s so that T can be uniquely determined according to the equation:

$$T = \big(\mathbf{h}^{\mathbf{z}} \cdot \mathbf{g}^{-\mathbf{z}^2} \cdot \rho^{s} \cdot \prod_{j=0}^{J-1} X_j^{-e^{j+1}} \cdot Q^{-y^2}\big)^{y^{-1}}$$

By the perfectly hiding property, the Pedersen commitments in a real argument are uniformly random, as in the simulation. The field elements in a real argument are also uniformly random due to the random choices of $(m_{j,i})_{j=0,i=0}^{J-1,N-1}$, $(r_{x_j})_{j=0}^{J-1}$, r_t and r_q. Therefore, we have identical distributions of real and simulated arguments for the given challenges.

Finally, we prove **computational** $(J+2)$**-special soundness**. An emulator runs the argument with random challenges and rewinds the prover until it obtains $J+2$ accepting transcripts. Then we compute the challenge matrix $\mathbf{y}$ in Eqn. (72), which is invertible since all the rows and columns are linearly independent.

$$\mathbf{y} = \begin{pmatrix} e_0 & \dots & e_0^J & y_0 & y_0^2 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ e_{J+1} & \dots & e_{J+1}^J & y_{J+1} & y_{J+1}^2 \end{pmatrix} \tag{72}$$

Firstly, we can acquire the openings at the generator g and ρ of the commitments $(X_j)_{j=0}^{J-1}$, T and Q on the right-hand side of Eqn. (66) by computing:

$$\mathbf{y}^{-1} \cdot \begin{pmatrix} \sum_{j=0}^{J-1} \sum_{i=0}^{N-1} 2^i e^{j+1} z_{j,i}^{(0)} & s_0 \\ \vdots & \vdots \\ \sum_{j=0}^{J-1} \sum_{i=0}^{N-1} 2^i e^{j+1} z_{j,i}^{(J+1)} & s_{J+1} \end{pmatrix}$$

Then we can obtain the openings at the JN generators $(g_{j,i})_{j=0,i=0}^{J-1,N-1}$ of the commitments by computing:

$$\mathbf{y}^{-1} \cdot \begin{pmatrix} z_{0,0}^{(0)} - (z_{0,0}^{(0)})^2 & \dots & z_{J-1,N-1}^{(0)} - (z_{J-1,N-1}^{(0)})^2 \\ \vdots & \ddots & \vdots \\ z_{0,0}^{(J+1)} - (z_{0,0}^{(J+1)})^2 & \dots & z_{J-1,N-1}^{(J+1)} - (z_{J-1,N-1}^{(J+1)})^2 \end{pmatrix}$$

Once we have the openings, the following relations hold for the challenges e and y:

$$\begin{aligned}
&\prod_{j=0}^{J-1} X_j^{e^{j+1}}\ T^{y}\ Q^{y^2} \\
&= g^{\sum_{j=0}^{J-1} (\sum_{i=0}^{N-1} 2^i b_{j,i}) e^{j+1}} \\
&\quad \cdot (\rho^{r_t} g^{\sum_{j=0}^{J-1} (\sum_{i=0}^{N-1} 2^i m_{j,i}) e^{j+1}} \prod_{j=0}^{J-1} \prod_{i=0}^{N-1} g_{j,i}^{(1-2b_{j,i}) m_{j,i}})^{y} \\
&\quad \cdot (\rho^{r_q} \prod_{j=0}^{J-1} \prod_{i=0}^{N-1} g_{j,i}^{-m_{j,i}^2})^{y^2} \\
&= \prod_{j=0}^{J-1} \prod_{i=0}^{N-1} g^{e^{j+1} 2^i (b_{j,i} + m_{j,i} y)} \\
&\quad \cdot \prod_{j=0}^{J-1} \prod_{i=0}^{N-1} g_{j,i}^{-(b_{j,i} + m_{j,i} y)^2} \cdot \rho^{\sum_{j=0}^{J-1} r_{x_j} \cdot e^{j+1} + r_t \cdot y + r_q \cdot y^2}
\end{aligned}$$

Otherwise, if the above relations do not hold, then we can extract the non-trivial discrete logarithm relations among the independently generated g, ρ and $(g_{j,i})_{j=0,i=0}^{J-1,N-1}$. which breaks the DLOG assumption.

Appendix C. Proof of Theorem 3

Proof. **Perfect completeness** follows by carefully inspecting the generalized final verification equation in Eqn. (73) derived from Eqn. (54):

$$\hat{\mathbf{h}}^{\hat{\mathbf{z}}} \cdot \hat{\mathbf{g}}^{-\hat{\mathbf{z}}^2} \overset{?}{=} \rho^{-s} \prod_{j=0}^{J-1} X_j^{e^{j+1}} T^{y} Q^{y^2} \prod_{l=1}^{\log N - 3} A_l^{c_l^{-2}} B_l^{c_l^{-1}} D_l^{c_l} E_l^{c_l^2} \tag{73}$$

where $\hat{\mathbf{z}}$, $\hat{\mathbf{h}}$ and $\hat{\mathbf{g}}$ are the final compressed vectors of witnesses and generators.

The **perfect SHVZK** of the compressed aggregate range argument Π_{carg} follows from the **perfect SHVZK** property of the aggregate range argument Π_{arg} based on the following composition structure:

$$\Pi_{\mathsf{carg}} = \underbrace{\Pi_{\mathsf{ac}} \diamond \cdots \diamond \Pi_{\mathsf{ac}}}_{\log N - 3 \text{ times}} \diamond \Pi_{\mathsf{arg}}$$

A simulator of Π_{carg} runs the simulator of Π_{arg} and substitutes the final messages of the simulated transcripts by honestly executing $\Pi_{\mathsf{ac}} \diamond \cdots \diamond \Pi_{\mathsf{ac}}$. The subsequent compressed protocol in each recursion Π_{carg} remains zero knowledge. The field vector $\mathbf{z}'$ is the sum of two zero-knowledge field elements $\mathbf{z_L}$ and $c\mathbf{z_R}$. Furthermore, the Pedersen commitments are perfectly hiding. Thus, the zero-knowledge property of the overall protocol Π_{carg} remains unchanged.

Finally, we follow the soundness proof in [11] to argue the **computational special soundness** of Π_{carg}. The protocol Π_{carg} is the composition of $(J + 2)$-special sound aggregate range protocol Π_{arg} and a sequence of 5-special sound compression protocols Π_{ac}. Thus, it can be observed that Π_{carg} is $(J + 2, 5, ..., 5)$-special sound such that there exists an efficient knowledge emulator that can extract the witnesses of the commitments along the path from the leaves to the root of the $(J + 2, 5, ..., 5)$-tree of $(J + 2) \cdot 5^{\log N - 3}$ accepting transcripts. We can see that the emulator uses $(J + 2) \cdot 5^{\log N - 3} < (J + 2) \cdot 8^{\log N} = (J + 2) \cdot N^3$ transcripts and thus runs in expected polynomial-time in N and J.