FlashSwift: A Configurable and More Efficient Range Proof With Transparent Setup

Nan Wang*
nan.wang@data61.csiro.au
CSIRO's Data61, Australia

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

ABSTRACT

Bit-decomposition-based zero-knowledge range proofs in the discrete logarithm (DLOG) setting with a transparent setup, e.g., Bulletproof (IEEE S&P '18), Flashproof (ASIACRYPT '22), and SwiftRange (IEEE S&P '24), have garnered widespread popularity across various privacy-enhancing applications. These proofs aim to prove that a committed value falls within the non-negative range $[0, 2^N - 1]$ without revealing it, where N represents the bit length of the range. Despite their prevalence, the current implementations still suffer from suboptimal performance. Some exhibit reduced communication costs at the expense of increased computational costs while others experience the opposite. Presently, users are compelled to utilize these proofs in scenarios demanding stringent requirements for both communication and computation efficiency.

In this paper, we introduce, FlashSwift, a stronger DLOG-based logarithmic-sized alternative. It stands out for its greater shortness and significantly enhanced computational efficiency compared with the cutting-edge logarithmic-sized ones for the most common ranges where $N \leq 64$. It is developed by integrating the techniques from Flashproof and SwiftRange without using a trusted setup. The substantial efficiency gains stem from our dedicated efforts in overcoming the inherent incompatibility barrier between the two techniques. Specifically, when $N = 64$, our proof achieves the same size as Bulletproof and exhibits 1.1× communication efficiency of SwiftRange. More importantly, compared with the two, it achieves 2.3× and 1.65× proving efficiency, and 3.2× and 1.7× verification efficiency, respectively. At the time of writing, our proof also creates two new records of the smallest proof sizes, 289 bytes and 417 bytes, for 8-bit and 16-bit ranges among all the bit-decomposition-based ones without requiring trusted setups. Moreover, to the best of our knowledge, it is the first *configurable* range proof that is adaptable to various scenarios with different specifications, where the configurability allows to trade off communication efficiency for computational efficiency. In addition, we offer a bonus feature: FlashSwift supports the aggregation of multiple single proofs for efficiency improvement. Finally, we provide comprehensive performance benchmarks against the state-of-the-art ones to demonstrate its practicality.

KEYWORDS

Zero-knowledge range proof, bit-decomposition, logarithmic-size, configurability, discrete logarithm, transparent setup

1 INTRODUCTION

Zero-knowledge proofs have been essential foundational components in various privacy-enhancing applications since they were firstly proposed by Goldwasser, Micali and Rackoff in 1985 [18]. By utilizing a zero-knowledge proof, a prover can demonstrate the validity of a public statement without revealing any secret information except the truth of the statement.

Zero-knowledge range proofs are one of the most widely used proofs, which aim to demonstrate that a committed value lies within a specified range without revealing it. Especially, bit-decomposition-based ones in the discrete logarithm (DLOG) setting with a transparent setup, e.g., Bulletproof [7], Flashproof[1] [30] and SwiftRange [32], have gained growing popularity among various applications. These proofs utilize the folklore bit-decomposition approach to demonstrate that a committed value lies within the range $[0, 2^N - 1]$ without using a trusted setup, where N represents the bit length of the range. Transparent or non-trusted setups are becoming an increasingly desirable feature for zero-knowledge proofs, where trusted setups rely on a specific group of trusted parties to generate public parameters and destroy secret trapdoors. However, these trusted setups may pose risks of leaking trapdoor information, undermining the security of the applications that rely on these zero-knowledge proofs. On the flip side, transparent setups alleviate this concern but introduce additional challenges in the design of zero-knowledge proofs with higher efficiency.

1.1 Applications

Bit-decomposition-based range proofs are widely used in diverse applications. In blockchain-based confidential transactions (CT) [19], range proofs play a crucial role in demonstrating the possession of sufficient funds with non-negative balances while preserving privacy. Black-box accumulation (BBA) schemes [22] enable privacy-preserving point collection and redemption on cryptographic tokens. They are important pillars for diverse user-centric protocols such as payment and incentive systems. Range proofs can be used to guarantee the committed balances are non-negative after deduction when users spend points. In privacy-preserving data aggregation schemes [8, 29], data providers are required to submit range proofs to ensure the integrity of the committed data and prevent invalid data from polluting the aggregate statistics. Zero-knowledge deep learning schemes empower model owners to prove that the predictions of data samples are generated by their models without

*The corresponding author, Nan Wang's homepage: https://www.nan-wang.com

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–16

https://doi.org/XXXXXXX.XXXXXXX

[1] Flashproofs and Bulletproofs-based ones consist of multiple proofs. We use the singular names, Bulletproof(+) and Flashproof, to refer to their range instances.

disclosing any model-related information. Range proof techniques can be leveraged to compute the outputs of non-linear functions, e.g., ReLU [26]. Vehicular digital forensics systems [23] collect vehicular data to enable liability cognizance of accidents and fight against crimes in a privacy-preserving manner. The systems leverage range proofs to create digital warrants accompanied with expiration so that others can check the validity of the warrants without knowing the hidden sensitive data. Anonymous credentials [27] are popular digital signatures accompanied by a series of secret attributes, such as age, height, date of birth, etc. Users must provide range proofs to demonstrate the validity of these attributes to the issuing parties before presenting the signed credentials to verifiers to authenticate themselves without revealing their identities. In anonymous reputation systems [25], consumers can leave committed rating scores for their experiences with retailers in order to help the retailers to build reputations among industrial partners.

It is commonly accepted that the ranges for $N \in \{8, 16, 32, 64\}$ are typically adequate for the majority of privacy-enhancing applications. For instance, CT platforms typically opt for 32-bit or 64-bit ranges, while BBA schemes commonly employ 16-bit or 32-bit ranges. In the context of deep learning schemes, 8-bit ranges find utility in computing ReLU functions, as indicated in [26]. For crowdsensing applications, the requirements can be sufficiently addressed with ranges where $N \leq 32$. In the domain of anonymous credentials, 16-bit ranges are deemed suitable for parameters like age and height, whereas 32-bit ranges are sufficient for representing date-of-birth. Vehicular digital forensics systems, on the other hand, often find 16-bit ranges satisfactory for encoding warrant expiration information. Anonymous reputation systems typically make use of 3-bit range proofs.

1.2 Motivation

It is well-known that zero-knowledge proofs involve large amounts of communication and computational costs, leading to the fact that their performance tends to dominate that of the associated applications. There are three key performance measures for zero-knowledge proofs, namely proof size, proving and verification efficiency. Proof sizes influence the communication overheads of applications while proving and verification efficiency determines how quickly a proof can be generated and verified, which considerably affects the user experience and the execution of applications.

In practical scenarios, a multitude of privacy-enhancing applications impose strict demands on both communication and computational efficiency. This is primarily due to the constraints posed by limited resources such as computing power, storage capacity, bandwidth availability and battery life, etc. Moreover, the performance of the incumbent bit-decomposition-based zero-knowledge range proofs remains suboptimal, which has long been a bottleneck, constraining the broader usability and adoption of relevant applications. We elaborate on two concrete applications that are experiencing such predicament:

- One application is the blockchain-empowered verifiable crowdsensing system [8], which allows a group of users to provide crowdsensed data for privacy-preserving data aggregation. Each user must store a 16-bit Bulletproof on the Ethereum blockchain, ensuring public verifiability and substantiating the validity of their provided data. However, the storage expenses associated with these proofs on the Ethereum network present a significant challenge. Based on the current gas consumption formula (21000 + 16 * bytes), the expenses are linearly proportional to the number of bytes, where gas measures the resources used to conduct a transaction on the Ethereum blockchain. Thus, storing proofs for 1000 users alone would result in a financial loss of 8.76 million gas, equivalent to a staggering $1549 USD at the time of writing, where the prices of gas and ether were 73 gwei and $2422 USD, respectively. Additionally, the enhanced computational overheads of Bulletproof may lead to a prolonged system running time, negatively impacting user experience.
- The other application is the black-box wallets [22], a privacy-preserving payment system for resource-constrained devices, e.g., smartphones and smart cards. The system uses 16-bit Bulletproof to prove the non-negativity of users' balances after payments without compromising privacy. In order to provide a more favorable user experience with reduced interaction time, the system needs to navigate runtime-communication trade-offs. It opts for the linear-sized proof, prioritizing faster computational efficiency over the logarithmic-sized alternative. Consequently, the system must compromise on transmission time to attain a shorter computation time. Unfortunately, the system fails to benefit from Bulletproof's communication advantage.

In a nutshell, dealing with large and computationally intensive proofs is a drain on time, storage and energy. The need for shorter proofs with improved computational efficiency is always necessary and beneficial. Moreover, cutting-edge range proofs employ distinct algorithms for their constructions and are applied in diverse scenarios, contingent on their unique strengths and limitations. In case of any changes in requirements, replacing a proof with a different one for resource-constrained devices presents challenges in both communication and energy. Software upgrades can be problematic for battery-operated devices, as they tend to be energy-consuming and can significantly diminish the devices' lifespan. Configurable performance is a beneficial property for a proof that can alleviate the strain on devices during resource-intensive upgrades. This property enables a proof to adapt to various situations, eliminating the need for a replacement for another proof in the event of any changes. Therefore, we wonder curiously:

Is there a way to combine the strengths of the state-of-the-art proofs to produce a stronger and configurable alternative without resorting to a trusted setup?

1.3 Contributions

In this paper, we make a meaningful stride in crafting a stronger logarithmic-sized alternative in the DLOG setting. We propose, FlashSwift[2], through the fusion of the techniques from Flashproof and SwiftRange. FlashSwift stands out for its greater shortness and significantly enhanced computational efficiency in contrast with the state-of-the-art logarithmic-sized ones, e.g., Bulletproof and

[2]Technically, FlashSwift is an argument of knowledge, a proof whose soundness holds for computationally bounded provers. We will use the two terms interchangeably.

SwiftRange. The substantial efficiency gains stem from our dedicated efforts in overcoming the inherent incompatibility barrier between the two proof techniques. Furthermore, by combining the two proofs, FlashSwift not only gains their efficiency advantages but also a distinctive feature, *configurability*. This means that its performance can be adjusted to suit various scenarios with different specifications, providing users with more choices for their applications. To the best of our knowledge, our work is the first configurable range proof in the literature. Note that our proof does not require any trusted setup or computationally expensive pairing operations.

1.3.1 Overview. Flashproof and SwiftRange are two variants of the bit-decomposition-based range proofs, which prove that a committed value lies within the range $[0, 2^N - 1]$ if it can be represented in binary form. Flashproof is a 3-round zero-knowledge range protocol Π_{frg}, which has $O(N^{\frac{2}{3}})$ sublinear efficiency in both communication and verification. It achieves comparable verification efficiency to that of the most efficient zkSNARK [21] that only involves three pairing operations for verification benefiting from a trusted setup. SwiftRange is a logarithmic-sized range proof composed of a 5-round zero-knowledge range protocol Π_{srg} and a sequence of non-zero-knowledge compression protocols Π_{qc} in Eqn. (1):

$$\Pi_{\mathsf{SR}} = \underbrace{\Pi_{\mathsf{qc}} \diamond \cdots \diamond \Pi_{\mathsf{qc}}}_{(\log N - 3) \text{ times}} \diamond \Pi_{\mathsf{srg}} \tag{1}$$

where the final messages of Π_{srg} are replaced by the execution of a series of Π_{qc}. When $N \leq 8$, the protocol Π_{srg} alone constitutes a highly short range proof. When $N > 8$, the prover can recursively apply $(\log N - 3)$ times of the compression protocol Π_{qc} to the range protocol Π_{srg} until the witness dimension is reduced to 8 for the minimum proof size.

1.3.2 Challenges. We begin with a concise overview of the challenges to provide a brief insight to our proof with further details expounded in Section 5.1. At a high level, we combine the bit-decomposition technique of Flashproof and the compression technique of SwiftRange. Inherently, there exists an incompatibility barrier between the two techniques, which inhibits a direct fusion. After a thorough examination of both techniques' features, we discovered that Flashproof comprises three verification equations, where two of them use one same witness vector for verification. We aim to leverage SwiftRange' compression technique to logarithmically reduce the dimension of this witness vector for communication efficiency improvement. However, only one of the two vector-related equations is compressible while the other is not. Attempting to compress a single equation proves futile, as it would compromise the other, resulting in a failed verification. Consequently, the primary challenge lies in developing a comprehensive approach to enable the compression-friendliness of Flashproof's multiple verification equations without using a trusted setup.

1.3.3 Our Solution. The high-level ideas of our solution are:

(1) to integrate Flashproof's multiple verification equations into a single one.
(2) to convert the integrated equation to the compression-friendly form of SwiftRange.

We implement the ideas to devise a new 5-round compression-friendly protocol Π_{rfrg} based on the protocol Π_{frg}:

$$\Pi_{\mathsf{FS}} = \underbrace{\Pi_{\mathsf{qc}} \diamond \cdots \diamond \Pi_{\mathsf{qc}}}_{(\lceil \log \frac{N}{K} \rceil - 3) \text{ times}} \diamond \Pi_{\mathsf{rfrg}} \tag{2}$$

where $\lceil \cdot \rceil$ is a round-up-to-the-nearest-integer operator. We borrow the positive "batch-size" parameter K from Flashproof and will explain it in Section 4.1. Compared with SwiftRange, our proof involves fewer compression recursions. When $\lceil \frac{N}{K} \rceil > 8$, the prover can apply $(\lceil \log \frac{N}{K} \rceil - 3)$ times of the compression protocols to the re-devised protocol Π_{rfrg} until the witness dimension is reduced to 8 for the minimum proof size.

1.3.4 Configurability. Our proof achieves the optimal efficiency in communication and computation when $K = 2$ and $K \approx N^{\frac{1}{3}}$, respectively. We will call them *Low-Gear* (LG) proof and *High-Gear* (HG) proof in the following to distinguish them, respectively. Generally, as K incrementally ranges from 2 to $N^{\frac{1}{3}}$, our proof's computational efficiency improves, albeit at the cost of reduced communication efficiency. Thus, the parameter K can be flexibly configured in the interval $[2, N^{\frac{1}{3}}]$ so as to adapt to various scenarios with different requirements. Notably, the high-gear proof can be a stronger contender against Flashproof since it achieves greater shortness while maintaining comparable computational overheads. The substantial benefit of this configurability is that the shift between two gears requires minimal adjustments to the implementation.

Remarks: The low-gear proof is the focus of this paper since communication efficiency tends to draw more attention in many privacy-enhancing applications, e.g., confidential transactions. Note that we restrict the ranges sizes to the most common ones where $N \in \{8, 16, 32, 64\}$ and the efficiency advantages of our low-gear proof over Bulletproof and SwiftRange holds within these ranges[3].

1.3.5 Aggregation. We offer a bonus feature: FlashSwift allows for the aggregation of J single proofs, which is denoted by *J-aggregate proof*. A J-aggregate proof compacts J single proofs into one to save computational and communication overheads. In the following paper, N and J are the powers of 2 and we can pad zeros if not.

1.4 Comparisons with the State of the art

Tables 1 and 2 demonstrate the efficiency comparisons of the state-of-the-art bit-decomposition-based range proofs with ours. For our low-gear proof, it involves far fewer group exponentiations than Bulletproof and SwiftRange in proving and verification. It uses 64 fewer bytes than SwiftRange for all ranges. Despite having a quicker growth rate than Bulletproof in the number of elements, it maintains an advantage in communication costs when $N \leq 64$. Bulletproof+ [11] slightly improves Bulletproof by using three fewer elements while retaining comparable computational efficiency. Our proof still exhibits a smaller size when $N \leq 16$ while presenting a substantial advantage in computational efficiency. Additionally, our proof has lower round complexity, enabling tighter security analysis than the two logarithmic-sized proofs in the random oracle

[3]Bulletproof has higher communication efficiency when $N > 64$, but is still far less efficient in computational efficiency.

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, where the computational complexity of Bulletproof+ is comparable to that of Bulletproof based on their experimental results [11]. The same as Flashproof, we take the nearest integer $\lceil N^{\frac{1}{3}} \rfloor$ as the cubic root of N and $N^{\frac{2}{3}}$ can thus be obtained by computing $N \cdot \lceil N^{-\frac{1}{3}} \rfloor$. $F(\cdot)$ is a function, defined by Flashproof, which yields a specific value based on the input (Please see Section 4.1.2 and Appendix B for more details).

Type	Bulletproof [7]	Bulletproof+ [11]	Flashproof [30]	SwiftRange ($N > 8$) [32]	FlashSwift (LG, $N > 16$)	FlashSwift (HG, $\lceil \log N^{\frac{2}{3}} \rceil > 3$)
Prover No. of Exps ($\mathbb{G}$)	$10N + 4\log N + 7$	$\approx$ Bulletproof	$N^{\frac{2}{3}} \cdot F(N^{\frac{1}{3}} - 1) + N^{\frac{1}{3}}$	$8N - 79$	$5N - 77$	$8N^{\frac{2}{3}} + N^{\frac{2}{3}} \cdot \left(F(N^{\frac{1}{3}}) - 1\right) + N^{\frac{1}{3}} - 79$
Verifier No. of Exps ($\mathbb{G}$)	$2N + 2\log N + 7$	$\approx$ Bulletproof	$N^{\frac{2}{3}} + F(N^{\frac{1}{3}}) + N^{\frac{1}{3}} + 2$	$N + 4\log N - 7$	$\frac{N}{2} + 4\log N - 10$	$N^{\frac{2}{3}} + \lceil \log N^{\frac{2}{3}} \rceil + F(N^{\frac{1}{3}}) + N^{\frac{1}{3}} - 11$
Proof Size No. of Elements	$2\log N + 4$ ($\mathbb{G}$) 5 ($\mathbb{Z}_p$)	$2\log N + 3$ ($\mathbb{G}$) 3 ($\mathbb{Z}_p$)	$N^{\frac{2}{3}} + 2$ ($\mathbb{G}$) $N^{\frac{1}{3}} + F(N^{\frac{1}{3}}) + 1$ ($\mathbb{Z}_p$)	$4\log N - 10$ ($\mathbb{G}$) 9 ($\mathbb{Z}_p$)	$4\log N - 12$ ($\mathbb{G}$) 9 ($\mathbb{Z}_p$)	$4\lceil \log N^{\frac{2}{3}} \rceil + F(N^{\frac{1}{3}}) + N^{\frac{1}{3}} - 13$ ($\mathbb{G}$) 9 ($\mathbb{Z}_p$)
No. of Rounds	$2\log N + 5$	$2\log N + 5$	3	$2\log N - 1$	$2\log N - 3$	$2\lceil \log N^{\frac{2}{3}} \rceil - 1$

model [3] since the round complexity strongly impacts the tightness of the security loss in the random oracle model [13].

Regarding our high-gear proof, compared with Flashproof, the computational complexity is lower for 8-bit and 16-bit ranges, but slightly higher for 32-bit and 64-bit ranges[4]. The round complexity is also higher due to the use of the compression protocols. The benefit is that the involved number of elements in the proof has been greatly reduced. In a nutshell, the high-gear proof achieves higher communication efficiency at a smaller ratio of costs in computation and round in contrast to Flashproof.

What is more, to the best of our knowledge, our proofs for 8-bit and 16-bit ranges create new records of the smallest proof sizes, 289 bytes and 417 bytes, among all the bit-decomposition-based range proofs without requiring trusted setups. The records strengthen the competitiveness in more communication-critical applications.

Table 2: The proof size comparison in bytes of the state-of-the-art range proofs.

N	8	16	32	64
Bulletproof	482	546	610	674
Bulletproof+	386	450	514	578
Flashproof	385	513	738	994
SwiftRange	353	481	610	738
FlashSwift (LG)	**289**	**417**	546	674
FlashSwift (HG)	**289**	**417**	642	770

Hence, our proof can bring a significant performance enhancement for existing applications. Reconsider the black-box wallets, applying our proof eliminates the need for the runtime-communication trade-offs. In contrast to the 16-bit linear-sized Bulletproof with 1249 bytes, our logarithmic-sized proof achieves 3× transmission efficiency, 2.2× proving efficiency, and 2.9× verification efficiency, resulting in significant savings of 67% transmission time, 54% proving time, and 65% verification time. Compared with SwiftRange, using our proof leads to savings of 13% transmission time, 69% proving time, and 35% verification time. Compared to Flashproof, using ours saves 19% transmission time and 18% verification time.

[4]It is not quite straightforward to compare the computational complexity of our high-gear proof and Flashproof. Please see Section 7 for our experimental results.

Moreover, in terms of the blockchain-based crowdsensing system, the computational efficiency gains of our proof over the others are similar to those of black-box wallets. Our proof's communication advantage can help significantly save more monetary gas fees for storing proofs on the Ethereum blockchain. For instance, given 1000 users, using Bulletproof, Flashproof and SwiftRange incurs 31%, 23% and 15% more gas costs than using ours, respectively.

1.5 Outline of Our Paper

Our paper is organized as follows. First, we review the related work and introduce the cryptographic preliminaries in Section 2 and 3. We give a technical overview of Flashproof and SwiftRange in Section 4. We elaborate on the techniques of our range proof in Section 5. We present our the aggregate proof in Section 6. Finally, we give a comprehensive performance evaluation and comparison with the state-of-the-art range proofs in Section 7.

2 RELATED WORK

This section provides an overview of various existing range proofs found in the literature at the time of writing. It covers the range proofs based on three approaches, namely, bit-decomposition-based, square-decomposition-based and signature-based approaches.

Bit-Decomposition-Based Proofs. Apart from what has been discussed in Section 1.4, we describe three other well-known proofs in the DLOG setting. The range argument [5] has $O(N)$ complexity in both communication and computation. Despite achieving logarithmic shortness, the work [1] only demonstrates that a committed bit-vector consists of zeros and ones without explicitly proving a committed value falls within the range $[0, 2^N - 1]$. SymmeProof [17] is another improved Bulletproof by reducing the communication complexity through the use of special challenges. To achieve this, they require 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. A computationally expensive algorithm was suggested to convert the standard challenges to these special ones. Nevertheless, it may be challenging to achieve the conversion in practice. Furthermore, computations over non-standard groups are more computationally intensive than over standard ones.

Square-Decomposition-Based Proofs. The square decomposition involves representing a committed value as a sum of squares to prove its non-negativity. Boudot [6] proposed the first construction, which represents a value x as the sum of the greatest square less than x and another positive value. Lipmaa [24] presented the construction of using Lagrange's four squares theorem to represent a value as a sum of four squares. The construction can achieve constant efficiency in communication and verification. Groth [20] improved the construction by showing that $4x + 1$ can always be represented as a sum of three squares for a target value x. If $4x + 1$ is non-negative, then $x \geq -\frac{1}{4}$ would be a non-negative integer. Deng et al. [14] designed a constant-size range proof by adapting Bulletproof for Lagrange's four-square theorem. However, these range proofs rely on the hardness of the RSA assumption, which require a trusted setup to generate RSA modulus. Couteau et al. introduced two novel constructions in the DLOG setting: CKLR21 [13] and Sharp [12]. These constructions make use of a bounded integer commitment scheme, with the latter being an enhanced version of the former. The key feature of this scheme is its ability to convert the standard Pedersen commitment scheme from operating over $\mathbb{Z}_p$ to operating within a smaller, bounded integer range. Although these constructions offer improved efficiency in terms of computation and communication compared to bit-decomposition approaches, they do come with a drawback of relaxed soundness. This relaxation places a constraint on the provers, requiring them to work with rational witnesses within the desired integer ranges. Additionally, the proofs themselves must adapt by using a substantially reduced challenge space to accommodate standard group sizes, such as 256 bits. In order to achieve negligible soundness errors 2^{-128} and 2^{-256}, the proofs are compelled to employ larger groups or undergo multiple iterations. To address the issue of relaxed soundness, Sharp employs RSA and class groups, aiming to mitigate the limitations associated with this relaxation. Nonetheless, RSA groups require a trusted setup. While class groups offer a trustless alternative to RSA groups, their efficiency still falls short when compared to DLOG-based elliptic curve groups. For instance, as indicated by a recent study [15], 3392-bit class groups can provide a security level comparable to 256-bit elliptic curve groups.

Signature-Based Proofs. Signature-based constructions often entail the need for undesirable trusted setups. In essence, the fundamental concept revolves around the verifier pre-calculating a digital signature for each element falling within a designated range. Subsequently, the prover employs a blind signature to sign a selected element, making it computationally challenging to reveal the actual signed element. Ultimately, the verifier aims to verify whether the blind signature corresponds to the set of precomputed signatures. The most typical construction [9] was proposed by Camenisch et al. based on the Boneh-Boyen signature schemes under the q-Strong Diffie-Hellman assumption. Chaabouni et al. [10] improved the proof by doubling the communication and computation efficiency.

3 PRELIMINARIES

Let λ and $\mathrm{negl}(\lambda)$ 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. Denote the prover and the verifier by $\mathcal{P}$ and $\mathcal{V}$, respectively.

We will use vector notations in the following paper. Bold font denotes vectors or matrices. For example, $\boldsymbol{\alpha} = (\alpha_0, ..., \alpha_{n-1}) \in \mathbb{Z}_p^n$ denotes a vector of scalars. $\mathbf{g} = (g_0, ..., g_{n-1}) \in \mathbb{G}^n$, $\boldsymbol{\rho} = (\rho_0, ..., \rho_{n-1}) \in \mathbb{G}^n$ denote two generator vectors. We describe some basic vector operations below:

- $\delta = \boldsymbol{\alpha} \cdot \boldsymbol{\beta} = \sum_{i=0}^{n-1} \alpha_i \cdot \beta_i \in \mathbb{Z}_p$.
- $\boldsymbol{\delta} = \boldsymbol{\alpha} + \boldsymbol{\beta} = (\alpha_0 + \beta_0, ..., \alpha_{n-1} + \beta_{n-1}) \in \mathbb{Z}_p^n$.
- $\boldsymbol{\delta} = \boldsymbol{\alpha} \circ \boldsymbol{\beta} = (\alpha_0 \cdot \beta_0, ..., \alpha_{n-1} \cdot \beta_{n-1}) \in \mathbb{Z}_p^n$.
- $g' = \mathbf{g}^{\boldsymbol{\alpha}} = \prod_{i=0}^{n-1} g_i^{\alpha_i} \in \mathbb{G}$.
- $\mathbf{g}' = \mathbf{g}^{\boldsymbol{\alpha}} \circ \boldsymbol{\rho} = (g_0^{\alpha_0} \cdot \rho_0, ..., g_{n-1}^{\alpha_{n-1}} \cdot \rho_{n-1}) \in \mathbb{G}^n$.

where $\circ$ denotes the component-wise Hadamard product.
Moreover, we introduce an important dual-index notation shown on the left-hand side of Eqn. (3). It refers to a combination set of size $\binom{K}{2} = \frac{K^2-K}{2}$, where K is a positive integer. The two indices k and j range between 0 and $K-1$, and k is not equal to j.

$$(k, j)_{(0,0)}^{(K-1,K-1)} \wedge k \neq j \Longrightarrow (k, j)_0^{\binom{K}{2}-1} \tag{3}$$

For brevity, we will use the simplified form on the right-hand side of Eqn. (3) in the following paper.

3.1 Cryptographic Assumption

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

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

The assumption asserts that no computationally bounded adversaries can discover discrete logarithm relations that fulfill the equation $\prod_{i=0}^{n-1} g_i^{x_i} = \eta$ for any arbitrary $\eta \in \mathbb{G}$ and randomly chosen generators. In order to eliminate the need for a trusted setup, the random generators $(g_i)_{i=0}^{n-1}$ can be independently generated using a collision-resistant hash function. This hash function maps random values from $\mathbb{Z}_p^*$ to $\mathbb{G}\backslash\{1\}$, ensuring that the discrete logarithm relations between any two generators remain unknown.

3.2 Homomorphic Commitment Schemes

To formalize a homomorphic commitment scheme, we adhere to the definitions presented in [30]. The scheme comprises two probabilistic polynomial-time algorithms, denoted as $\mathcal{G}$ and Cm. The setup algorithm $\mathcal{G}(\lambda)$ generates a commitment key denoted as ck, while the commitment algorithm Cm defines a function $\mathsf{Cm}_{\mathsf{ck}} : \mathsf{M}_{\mathsf{ck}} \times \mathsf{R}_{\mathsf{ck}} \rightarrow \mathsf{C}_{\mathsf{ck}}$. In this scheme, M_{ck} represents the message space, R_{ck} represents the randomness space, and C_{ck} represents the commitment space. For any given message $m \in \mathsf{M}_{\mathsf{ck}}$, a uniformly random value $r \in \mathsf{R}_{\mathsf{ck}}$ is selected to compute a commitment $\mathsf{Cm}_{\mathsf{ck}}(m; r)$.

Definition 2 (Hiding). *A commitment scheme ($\mathcal{G}$, Cm) is hiding if a commitment does not reveal the value for all PPT adversaries $\mathcal{A}$:*

$$Pr\left[\begin{array}{l} C = \mathsf{Cm}_{\mathsf{ck}}(m_b), \\ b \in \{0,1\}, \\ b' \leftarrow \mathcal{A}(C),\ b = b' \end{array} \middle| \begin{array}{l} \mathsf{ck} \leftarrow \mathcal{G}(\lambda), \\ (m_0, m_1 \in \mathsf{M}_{\mathsf{ck}}) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \approx \frac{1}{2}$$

The scheme is perfectly hiding if the probability is $\frac{1}{2}$.

Definition 3 (Binding). *A commitment scheme ($\mathcal{G}$, Cm) is binding if a commitment can only be opened to one value for all PPT adversaries $\mathcal{A}$:*

$$Pr\left[\begin{array}{l} \mathsf{Cm}_{\mathsf{ck}}(m_0; r_0) \\ = \mathsf{Cm}_{\mathsf{ck}}(m_1; r_1), \\ m_0 \neq m_1 \end{array} \middle| \begin{array}{l} \mathsf{ck} \leftarrow \mathcal{G}(\lambda), \\ (m_0, m_1) \in \mathsf{M}_{\mathsf{ck}}, \\ (r_0, r_1) \in \mathsf{R}_{\mathsf{ck}} \end{array} \leftarrow \mathcal{A}(\mathsf{ck})\right] \leq \mathsf{negl}(\lambda)$$

The scheme is perfectly binding if the probability is 0.

Definition 4 (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 $(g_0, ..., g_{n-1}, \rho) \xleftarrow{\$} \mathbb{G}$,* we have:

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

Pedersen vector commitment is perfectly hiding and computationally binding. It satisfies the following homomorphic property:

$$\begin{aligned} &\mathsf{Cm}(x_0, ..., x_{n-1}; r_x) \cdot \mathsf{Cm}(y_0, ..., y_{n-1}; r_y) \\ &= \mathsf{Cm}(x_0 + y_0, ..., x_{n-1} + y_{n-1}; r_x + r_y) \end{aligned}$$

Pedersen commitment is a special case, where $n = 1$. We follow the notation [32] to denote commitments by capital letters in the following paper, e.g., $X = \prod_{i=0}^{n-1} g_i^{x_i} \rho^r$.

3.3 Zero-Knowledge Arguments of Knowledge

We employed the definitions from [30] to formalize zero-knowledge arguments of knowledge. A zero-knowledge argument is comprised of three interactive probabilistic polynomial-time algorithms, denoted as $\mathcal{G}$, $\mathcal{P}$, and $\mathcal{V}$. The setup algorithm $\mathcal{G}(\lambda)$ generates a common reference string denoted as σ. The prover and verifier algorithms, $\mathcal{P}$ and $\mathcal{V}$ respectively, produce a public transcript $tr \leftarrow \langle \mathcal{P}(v), \mathcal{V}(t) \rangle$ based on their respective inputs v and t. We introduce a polynomial-time decidable ternary relation, denoted as $\mathcal{R} \subset 0, 1^* \times 0, 1^* \times 0, 1^*$. A CRS-dependent language, denoted as L_σ, can be defined as follows:

$$L_\sigma = \{u \mid \exists \omega : (\sigma, u, \omega) \in \mathcal{R}\}$$

where ω represents a witness for a statement u in the relation $(\sigma, u, \omega) \in \mathcal{R}$.

Definition 5 (Argument of Knowledge). *The triple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called an argument of knowledge for the relation $\mathcal{R}$ if it satisfies the perfect completeness and computational witness-extended emulation.*

Definition 6 (Perfect Completeness). *An argument of knowledge $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ has perfect completeness if for all PPT adversaries $\mathcal{A}$:*

$$Pr\left[\begin{array}{l} (\sigma, u, \omega) \notin \mathcal{R} \text{ or} \\ \langle \mathcal{P}(\sigma, u, \omega), \mathcal{V}(\sigma, u) \rangle = 1 \end{array} \middle| \begin{array}{l} \sigma \leftarrow \mathcal{G}(\lambda), \\ (u, \omega) \leftarrow \mathcal{A}(\sigma) \end{array}\right] = 1$$

Definition 7 (Computational Witness-Extended Emulation). *An argument of knowledge $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ has witness-extended emulation if for all deterministic polynomial time $\mathcal{P}^*$, there exists an expected polynomial time emulator $\mathcal{E}$ such that for all PPT adversaries $\mathcal{A}$:*

$$Pr\left[\mathcal{A}(tr) = 1 \middle| \begin{array}{l} \sigma \leftarrow \mathcal{G}(\lambda) \\ (u, s) \leftarrow \mathcal{A}(\sigma), \\ tr \leftarrow \mathcal{O} \end{array}\right] \approx$$

$$Pr\left[\begin{array}{l} \mathcal{A}(tr) = 1 \\ \wedge\ tr \text{ is accepting} \\ \rightarrow (\sigma, u, w) \in \mathcal{R} \end{array} \middle| \begin{array}{l} \sigma \leftarrow \mathcal{G}(\lambda), \\ (u, s) \leftarrow \mathcal{A}(\sigma), \\ (tr, \omega) \leftarrow \mathcal{E}^{\mathcal{O}}(\sigma, u) \end{array}\right]$$

where the oracle is defined as $\mathcal{O} = \langle \mathcal{P}^*(\sigma, u, s), \mathcal{V}(\sigma, u) \rangle$. Whenever $\mathcal{P}^*$ presents an argument in state s, a corresponding knowledge emulator $\mathcal{E}$ comes into play, enabling the extraction of a witness for $(\sigma, u, \omega) \in \mathcal{R}$. This is achieved by rewinding the interaction to specific points and then replaying it, utilizing the original state for the prover but introducing new randomness for the verifier.

Definition 8 (Public Coin). *An argument of knowledge $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called public coin if the verifier chooses her messages uniformly at random and independently of the messages sent by the prover.*
Public-coin interactive protocols can be made non-interative through the well-known Fiat-Shamir transformation [16].

Definition 9 (Perfect Special Honest Verifier Zero-Knowledge, SHVZK). *A public coin argument of knowledge $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called perfect special honest verifier zero-knowledge argument of knowledge for $\mathcal{R}$ if there exists a PPT simulator $\mathcal{S}$ such that for all interactive PPT adversaries $\mathcal{A}$:*

$$Pr\left[\begin{array}{l} (\sigma, u, \omega) \in \mathcal{R} \\ \wedge\ \mathcal{A}(tr) = 1 \end{array} \middle| \begin{array}{l} \sigma \leftarrow \mathcal{G}(\lambda), \\ (u, \omega, e) \leftarrow \mathcal{A}(\sigma), \\ tr \leftarrow \langle \mathcal{P}(v), \mathcal{V}(t) \rangle \end{array}\right] =$$

$$Pr\left[\begin{array}{l} (\sigma, u, \omega) \in \mathcal{R} \\ \wedge\ \mathcal{A}(tr) = 1 \end{array} \middle| \begin{array}{l} \sigma \leftarrow \mathcal{G}(\lambda), \\ (u, \omega, e) \leftarrow \mathcal{A}(\sigma), \\ tr \leftarrow \mathcal{S}(u, e) \end{array}\right]$$

where e is a public coin challenge, $v = (\sigma, u, \omega)$ and $t = (\sigma, u, e)$.

4 TECHNICAL OVERVIEW

In this section, we give a brief technical overview of Flashproof and SwiftRange before elaborating on our proofs.

4.1 Flashproof's Bit-Decomposition Technique

Flashproof features its $O(N^{\frac{2}{3}})$ efficiency in both communication and verification. The high-level idea behind Flashproof is that the prover writes a 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 $w_i = 2^i b_i,\ i \in \{0, 1, ..., N-1\}$. The prover arranges all the terms $(w_i)_{i=0}^{N-1}$ in an $L \times K$ matrix in Eqn. (4), where L and K indicate the number of rows and columns, respectively, and $L \cdot K \geq N$.

$$\begin{pmatrix} v_0 \\ \vdots \\ v_{L-1} \end{pmatrix} = \begin{pmatrix} w_0 & \cdots & w_{K-1} \\ w_K & \cdots & w_{K+K-1} \\ \vdots & \ddots & \vdots \\ w_{(L-1)K} & \cdots & w_{(L-1)K+K-1} \end{pmatrix} \cdot \begin{pmatrix} e_0 \\ \vdots \\ e_{K-1} \end{pmatrix} + \begin{pmatrix} r_0 \\ \vdots \\ r_{L-1} \end{pmatrix} \quad (4)$$

Subsequently, the prover can compute a series of values $(v_l)_{l=0}^{L-1}$ after obtaining a challenge vector $(e_0, ..., e_{K-1})^\intercal$ from the verifier:

$$v_l = \sum_{k=0}^{K-1} w_{lK+k} e_k + r_l$$

where $r_l \in \mathbb{Z}_p^*$ is a random value. The argument takes advantage of a quadratic-term cancellation technique to prove each term w_{lK+k} is either the $(lK+k)$-th power of 2 or 0. If this is the case, then it can be concluded that the committed value x is within the range $[0, 2^N - 1]$.

4.1.1 3-Round Protocol Π_{frg}. Given a target commitment $X = g^x \rho^{r_x}$, the generators $g, \rho, (g_l)_{l=0}^{L-1} \xleftarrow{\$} \mathbb{G}$ and $r_x \xleftarrow{\$} \mathbb{Z}_p^*$, the range protocol Π_{frg} goes as follows:

$$\mathcal{P}: (r_l \xleftarrow{\$} \mathbb{Z}_p^*)_{l=0}^{L-1}, \quad (r_{s_k} \xleftarrow{\$} \mathbb{Z}_p^*)_{k=1}^{K} \tag{5}$$

$$(r_{q_k} \xleftarrow{\$} \mathbb{Z}_p^*)_{k=0}^{K}, \quad (r_{t_{k,j}} \xleftarrow{\$} \mathbb{Z}_p^*)_0^{\binom{K}{2}-1} \tag{6}$$

$$\mathcal{P} \Rightarrow \mathcal{V}: (T_{k,j} = \prod_{l=0}^{L-1} g_l^{t_{l,(k,j)}} \cdot \rho^{r_{t_{k,j}}})_0^{\binom{K}{2}-1} \tag{7}$$

where $t_{l,(k,j)} = w_{lK+k}(2^{lK+j} - w_{lK+j}) + w_{lK+j}(2^{lK+k} - w_{lK+k})$

$$\left(Q_k = \prod_{l=0}^{L-1} g_l^{q_{l,k}} \cdot \rho^{r_{q_k}}\right)_{k=0}^{K} \tag{8}$$

where $\left(q_{l,k} = 2r_l(2^{lK+k-1} - w_{lK+k})\right)_{k=0}^{K-1}$, $q_{l,K} = -r_l^2$

$$(S_k = g^{\sum_{l=0}^{L-1} w_{lK+k}} \rho^{r_{s_k}})_{k=0}^{K-1}, \quad S_K = g^{\sum_{l=0}^{L-1} r_l} \rho^{r_{s_K}} \tag{9}$$

$$\text{where } r_{s_0} = r_x - \sum_{k=1}^{K-1} r_{s_k} \tag{10}$$

$$\mathcal{P} \Leftarrow \mathcal{V}: (e_k \xleftarrow{\$} \mathbb{Z}_p^*)_{k=0}^{K-1}$$

$$\mathcal{P} \Rightarrow \mathcal{V}: \left(v_l = \sum_{k=0}^{K-1} w_{lK+k} e_k + r_l\right)_{l=0}^{L-1} \tag{11}$$

$$u = \sum_{0}^{\binom{K}{2}-1} r_{t_{k,j}} e_{k,j} + \sum_{k=0}^{K-1} r_{q_k} e_k + r_{q_K} \tag{12}$$

$$\epsilon = \sum_{k=0}^{K-1} r_{s_k} e_k + r_{s_K} \tag{13}$$

where $e_{k,j} = e_k \cdot e_j$ is the product of the two distinct challenges e_k and e_j and $k, j \in \{0, ..., K-1\} \wedge k \neq j$. In the first round, the prover sends a series of commitments $(T_{k,j})_0^{\binom{K}{2}-1}$, $(Q_k)_{k=0}^K$ and $(S_k)_{k=0}^K$ to the verifier. In the second round, the verifier replies with a random challenge vector $(e_k)_{k=0}^{K-1}$ consisting of K distinct challenges. Finally, in the third round, the verifier sends back a series of field elements $(v_l)_{l=0}^{L-1}$, u and ϵ. For verification, the verifier:

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

$$f_l = \sum_{k=0}^{K-1} 2^{lK+k} e_k - v_l = \sum_{k=0}^{K-1} (2^{lK+k} - w_{lK+k}) e_k - r_l \tag{14}$$

(2) computes $f_l \cdot v_l$ to generate $(\frac{1}{2}K^2 + \frac{1}{2}K + 1)$ cross-terms on the right-hand side of Eqn. (15):

$$\prod_{l=0}^{L-1} g_l^{f_l v_l} \cdot \rho^u \stackrel{?}{=} \underbrace{\prod_{k=0}^{K-1} W_{lK+k}^{e_k^2}}_{=1} \cdot \prod_{0}^{\binom{K}{2}-1} T_{k,j}^{e_{k,j}} \cdot \prod_{k=0}^{K-1} Q_k^{e_k} \cdot Q_K \tag{15}$$

where the opening of W_{lK+k} is $w_{lK+k}(2^{lK+k} - w_{lK+k})$. The verifier aims to confirm that all the quadratic terms $(W_{lK+k}^{e_k^2})_{k=0}^{K-1}$ are cancelled out. Thus, the verifier can be convinced that the $(lK+k)$-th term $w_{lK+k} \in \{0, 2^{lK+k}\}$ for $k \in \{0, ..., K-1\}$. Note that each v_l consists of K terms so that computing $f_l \cdot v_l$ can simultaneously check K terms, which plays the dominant role in yielding high verification efficiency.

(3) uses Eqn. (16) to check whether the k-th commitment S_k commits to the sum of the terms in the k-th column of the matrix for $k \in \{0, ..., K-1\}$.

$$g^{\sum_{l=0}^{L-1} v_l} \cdot \rho^{\epsilon} \stackrel{?}{=} \prod_{k=0}^{K-1} S_k^{e_k} \cdot S_K \tag{16}$$

(4) checks if the committed value x is the sum of all the terms in the matrix if Eqn. (16) passes.

$$X \stackrel{?}{=} \prod_{k=0}^{K-1} S_k \tag{17}$$

4.1.2 Optimization. Flashproof describes an optimization technique to improve the communication and verification efficiency by changing the way that the challenge vectors $(e_k)_{k=0}^{K-1}$ are generated. The verifier is allowed to randomly produce a single challenge e, such that the other challenges in the vector can be generated by taking different powers of e. In this case, some cross-terms on the right-hand side of Eqn. (15) can be combined to reduce the total number of elements in the proof and the group exponentiations for verification. Consider the case with $K = 4$, we can use the 4 challenges $(e_k)_{k=0}^3 = (e^{-1}, e, e^4, e^5)$. Originally, the computation of $f_l \cdot v_l$ will generate a polynomial with $\frac{1}{2} \cdot 4^2 + \frac{1}{2} \cdot 4 + 1 = 11$ terms. With optimization, it only involves 8 terms in Eqn. (18)[5] without any quadratic terms e^{-2}, e^2, e^8 or e^{10}:

$$w_9 e^9 + w_6 e^6 + w_5 e^5 + w_4 e^4 + w_3 e^3 + w_1 e + w_{-1} e^{-1} + w_0 \tag{18}$$

where w_* indicates the coefficients of the corresponding terms. Furthermore, for $K = 2$, we can use the challenges (e^{-1}, e) to generate 3 cross-terms. Flashproof defines an optimization function $F(K)$ that lists the number of optimized cross-terms to replace $(\frac{K^2}{2} + \frac{K}{2} + 1)$ unoptimized terms on the right-hand side of Eqn. (15). Please see Appendix B for the function $F(K)$.

4.2 SwiftRange's Compression Technique

The work [2] proposed a non-zero-knowledge compression protocol Π_{lc} for the linear setting. The protocol Π_{lc} achieves logarithmic size for proving that a committed vector $\mathbf{v}$ satisfies the linear relation $U = \mathbf{g}^{\mathbf{v}}$ for a generator vector $\mathbf{g} = (g_i)_{i=0}^{N-1} \in \mathbb{G}^N$ consisting of N distinct generators. SwiftRange presented an extended protocol

[5] This equation is borrowed from Flashproof paper [30].

Π_{qc} for the quadratic setting, which proves that a committed vector $\mathbf{v}$ satisfies the quadratic relation $U = \mathbf{h}^{\mathbf{v}} \cdot \mathbf{g}^{-\mathbf{v}^2}$, where $\mathbf{g}, \mathbf{h} \in \mathbb{G}^N$ are two generator vectors. Furthermore, SwiftRange also slightly improves the computational efficiency of the protocol Π_{lc}.

In the following, we give a brief introduction to the protocol Π_{qc}. The protocol Π_{qc} is defined with the inputs $(\mathbf{h}, \mathbf{g}, U, \mathbf{v})$. The recursive composition of the compression protocol Π_{qc} goes as below for $|\mathbf{v}| > 8$:

$$\text{If } |\mathbf{v}| \leq 8: \tag{19}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : \mathbf{v} \tag{20}$$

$$\mathcal{V} : \mathbf{h}^{\mathbf{v}} \cdot \mathbf{g}^{-\mathbf{v}^2} \stackrel{?}{=} U \tag{21}$$

$$\text{Else}: \tag{22}$$

$$\mathcal{P} \Rightarrow \mathcal{V} : A \triangleq \mathbf{g}_R^{-\mathbf{v}_L^2},\ B \triangleq \mathbf{h}_R^{\mathbf{v}_L} \cdot \mathbf{g}_R^{-2\mathbf{v}_L\mathbf{v}_R} \in \mathbb{G} \tag{23}$$

$$D \triangleq \mathbf{h}_L^{\mathbf{v}_R} \cdot \mathbf{g}_L^{-2\mathbf{v}_L\mathbf{v}_R},\ E \triangleq \mathbf{g}_L^{-\mathbf{v}_R^2} \in \mathbb{G} \tag{24}$$

$$\mathcal{P} \Leftarrow \mathcal{V} : c \xleftarrow{\$} \mathbb{Z}_p^* \tag{25}$$

$$\mathcal{P} \text{ and } \mathcal{V} : \mathbf{h}' \triangleq \mathbf{h}_L \circ \mathbf{h}_R^{c^{-1}} \in \mathbb{G}^{\frac{N}{2}} \tag{26}$$

$$\mathbf{g}' \triangleq \mathbf{g}_L \circ \mathbf{g}_R^{c^{-2}} \in \mathbb{G}^{\frac{N}{2}} \tag{27}$$

$$U' \triangleq A^{c^{-2}} \cdot B^{c^{-1}} \cdot U \cdot D^{c} \cdot E^{c^2} \tag{28}$$

$$\mathcal{P} : \mathbf{v}' = \mathbf{v}_L + c\mathbf{v}_R \in \mathbb{Z}_p^{\frac{N}{2}} \tag{29}$$

$$\text{Recursively run } \Pi_{\mathsf{qc}} \text{ on input } (\mathbf{h}', \mathbf{g}', U', \mathbf{v}') \tag{30}$$

where $\mathbf{v}_L = (v_0, ..., v_{\frac{N}{2}-1})$ and $\mathbf{v}_R = (v_{\frac{N}{2}}, ..., v_{N-1})$ are the left-half and right-half vectors of $\mathbf{v}$, respectively. $\mathbf{g}_L = (g_0, ..., g_{\frac{N}{2}-1})$ and $\mathbf{g}_R = (g_{\frac{N}{2}}, ..., g_{N-1})$ are the left-half and right-half generator vectors of $\mathbf{g}$. Similar rule applies to $\mathbf{h}_L$ and $\mathbf{h}_R$. When $|\mathbf{v}| = 8$, the recursive compression terminates for the minimum communication cost. If a new recursion is added, the vector $\mathbf{v}$ would be reduced by half to 4. However, the recursion also adds four new group elements A, B, D and E. Hence, the total proof size slightly increases since a group element is generally larger than a field element. Consequently, when $|\mathbf{v}| = 8$, the argument has the minimum proof size in bytes. Note that a prerequisite for applying the compression protocol is that the vector dimension $|\mathbf{v}|$ must be a power of 2.

5 OUR PROOF

5.1 Challenges

Expanding the brief idea in Section 1.3.2, we further expound the specific challenges we face. Recall from the technical overview in Section 4 that the two equations Eqn. (15) and Eqn. (16) use the same masking witness vector $\mathbf{v} = (v_l)_{l=0}^{L-1}$ for verification. Our goal is to apply SwiftRange's compression protocol Π_{qc} to logarithmically reduce this witness vector in order to achieve logarithmic communication efficiency while retaining comparable computational efficiency. However, only Eqn. (15) in Flashproof is compressible since the left-hand side of Eqn. (15) can be rewritten to the quadratic form of the compression protocol Π_{qc}:

$$\prod_{l=0}^{L-1} g_l^{f_l v_l} = \prod_{l=0}^{L-1} (g_l^{\sum_{k=0}^{K-1} 2^{lK+k} e_k})^{v_l} \cdot \prod_{l=0}^{L-1} g_l^{-v_l^2} = \mathbf{h}^{\mathbf{v}} \cdot \mathbf{g}^{-\mathbf{v}^2} \tag{31}$$

where $\mathbf{h} = (g_l^{\sum_{k=0}^{K-1} 2^{lK+k} e_k})_{l=0}^{L-1}$, $\mathbf{g} = (g_l)_{l=0}^{L-1}$. However, it is noticeable that Eqn. (16) does not exhibit compressibility. Specifically, Eqn. (16) satisfies a linear setting since the left-hand side of Eqn. (16) utilizes a single generator g to commit to the Hamming weight $\sum_{l=0}^{L-1} v_l$ of the vector $\mathbf{v}$. However, the linear-setting compression protocol Π_{lc} does not apply to Eqn. (16). As outlined in Section 4, the protocol Π_{lc} is only designed to operate with the linear relation $U = \prod_{i=0}^{N-1} g_i^{v_i}$ using a set of distinct generators $(g_i)_{i=0}^{N-1}$ instead of a single generator. Compressing Eqn. (15) in isolation would result in a dimension-reduced witness vector in the form $\mathbf{v}_L + c\mathbf{v}_R$, where c is an unpredictably random challenge. Consequently, it is infeasible for verifiers to use this dimension-reduced vector to reconstruct $\sum_{i=0}^{N-1} v_i$ for the validation of Eqn. (16), leading to a failed verification.

5.2 Techniques

We present a solution to enable the compression-friendliness by subtly integrating the two equations Eqn. (15) and (16) in Section 5.2.1. Furthermore, we also show a trick in Section 5.2.2 of integrating Eqn. (17) with the other two to reduce proof size.

5.2.1 Integration & Conversion. In Eqn. (32), we use a random value $y \in \mathbb{Z}_p^*$ to achieve the integration. This is based on the folklore observation [7] that checking $g^\alpha = 1 \wedge g^\beta = 1$ amounts to checking $g^{\alpha \cdot y + \beta} = 1$ for an arbitrary $y \in \mathbb{Z}_p^*$. On the left-hand side, the existence of y aims to separate $g^{\sum_{l=0}^{L-1} v_l}$ from $\prod_{l=0}^{L-1} g_l^{f_l \cdot v_l}$. On the right-hand side, the exponents of the commitments $(S_k)_{k=0}^{K-1}$ become $(e_k \cdot y)_{k=0}^{K-1}$ rather than $(e_k)_{k=0}^{K-1}$:

$$\begin{aligned} &(g^{\sum_{l=0}^{L-1} v_l})^y \cdot \prod_{l=0}^{L-1} g_l^{f_l \cdot v_l} \\ &\stackrel{?}{=} \rho^{-u} \cdot \prod_{0}^{\binom{K}{2}-1} T_{k,j}^{e_{k,j}} \cdot \prod_{k=0}^{K-1} Q_k^{e_k} \cdot \prod_{k=0}^{K-1} S_k^{e_k \cdot y} \cdot S_K^y \cdot Q_K \end{aligned} \tag{32}$$

Then we subtly convert the left-hand side of Eqn. (32) to enable the compression-friendliness:

$$\begin{aligned} (g^{\sum_{l=0}^{L-1} v_l})^y \cdot \prod_{l=0}^{L-1} g_l^{f_l \cdot v_l} &= \prod_{l=0}^{L-1} (g^y)^{v_l} \cdot \prod_{l=0}^{L-1} (g_l^{f_l})^{v_l} \\ &= \prod_{l=0}^{L-1} (g^y \cdot g_l^{\sum_{k=0}^{K-1} 2^{lK+k} e_k - v_l})^{v_l} \\ &= \prod_{l=0}^{L-1} (g^y \cdot g_l^{\sum_{k=0}^{K-1} 2^{lK+k} e_k})^{v_l} \cdot \prod_{l=0}^{L-1} g_l^{-v_l^2} \\ &= \mathbf{h}^{\mathbf{v}} \cdot \mathbf{g}^{-\mathbf{v}^2} \end{aligned} \tag{33}$$

We successfully convert the two verification equations of Flashproof to the desired compression-friendly form $\mathbf{h}^{\mathbf{v}} \cdot \mathbf{g}^{-\mathbf{v}^2}$ of the compression protocol Π_{qc}, where $\mathbf{h}$ becomes $(g^y \cdot g_l^{\sum_{k=0}^{K-1} 2^{lK+k} e_k})_{l=0}^{L-1}$ and $\mathbf{g} = (g_l)_{l=0}^{L-1}$ remains intact.

5.2.2 Use of Target Commitment. Note that Eqn. (17) has not been integrated. We present a trick to take advantage of the target commitment X to reduce the proof size by one group element. Interestingly, the verifier can obtain S_{K-1} by computing $X \cdot \prod_{k=0}^{K-2} S_k^{-1}$ based on Eqn. (17). In this case, the prover only needs to transfer $K-1$ commitments $(S_k)_{k=0}^{K-2}$ without the need for sending S_{K-1}. The part $\prod_{k=0}^{K-1} S_k^{e_k y}$ at the right-hand side of Eqn. (32) can be converted to the following by substituting $X \cdot \prod_{k=0}^{K-2} S_k^{-1}$ for S_{K-1}:

$$\begin{aligned}\prod_{k=0}^{K-1} S_k^{e_k y} &= \prod_{k=0}^{K-2} S_k^{e_k y} \cdot (X \cdot \prod_{k=0}^{K-2} S_k^{-1})^{e_{K-1} y} \\ &= \prod_{k=0}^{K-2} S_k^{(e_k - e_{K-1}) y} \cdot X^{e_{K-1} y}\end{aligned} \tag{34}$$

5.3 5-Round Protocol Π_{rfrg}

In this section, we give details on our re-devised protocol. There is another trick we want to introduce to reduce the proof size by one more element. We transform the original 3-round Flashproof protocol Π_{frg} to a new 5-round one Π_{rfrg} so that the openings of S_K are merged to those of Q_K. In the following, we incorporate all the above techniques to present a new 5-round protocol Π_{rfrg}:

$$\mathcal{P}: (r_l \xleftarrow{\$} \mathbb{Z}_p^*)_{l=0}^{L-1}, \quad (r_{s_k} \xleftarrow{\$} \mathbb{Z}_p^*)_{k=0}^{K-2} \tag{35}$$

$$(r_{q_k} \xleftarrow{\$} \mathbb{Z}_p^*)_{k=0}^{K}, \quad (r_{t_{k,j}} \xleftarrow{\$} \mathbb{Z}_p^*)_0^{K-1} \tag{36}$$

$$\mathcal{P} \Rightarrow \mathcal{V}: (T_{k,j} = \prod_{l=0}^{L-1} g_l^{t_{l,(k,j)}} \cdot \rho^{r_{t_{k,j}}})_0^{\binom{K}{2}-1} \tag{37}$$

where $t_{l,(k,j)} = w_{lK+k}(2^{lK+j} - w_{lK+j}) + w_{lK+j}(2^{lK+k} - w_{lK+k})$

$$(Q_k = \prod_{l=0}^{L-1} g_l^{q_{l,k}} \cdot \rho^{r_{q_k}})_{k=0}^{K-1} \tag{38}$$

$$\text{where } (q_{l,k} = 2r_l(2^{lK+k-1} - w_{lK+k}))_{k=0}^{K-1}$$

$$(S_k = g^{\sum_{l=0}^{L-1} w_{lK+k}} \rho^{r_{s_k}})_{k=0}^{K-2} \tag{39}$$

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

$$\mathcal{P} \Rightarrow \mathcal{V}: Q_K = g^{y \sum_{l=0}^{L-1} r_l} \prod_{l=0}^{L-1} g_l^{-r_l^2} \cdot \rho^{r_{q_K}} \tag{40}$$

$$\mathcal{P} \Leftarrow \mathcal{V}: (e_k \xleftarrow{\$} \mathbb{Z}_p^*)_{k=0}^{K-1} \tag{41}$$

$$\mathcal{P} \Rightarrow \mathcal{V}: (v_l = \sum_{k=0}^{K-1} w_{lK+k} e_k + r_l)_{l=0}^{L-1} \tag{42}$$

$$\begin{aligned}u = &\sum_{0}^{\binom{K}{2}-1} r_{t_{k,j}} e_{k,j} + \sum_{k=0}^{K-1} r_{q_k} e_k \\ &+ \sum_{k=0}^{K-2} r_{s_k}(e_k - e_{K-1}) y + r_x e_{K-1} y + r_{q_K}\end{aligned} \tag{43}$$

$$\mathcal{V}: \mathsf{LHS} \stackrel{?}{=} \mathsf{RHS} \tag{44}$$

$$\mathsf{LHS} = \mathbf{h}^{\mathbf{v}} \cdot \mathbf{g}^{-\mathbf{v}^2} \tag{45}$$

$$\begin{aligned}\mathsf{RHS} = \rho^{-u} \cdot &\prod_{0}^{\binom{K}{2}-1} T_{k,j}^{e_{k,j}} \cdot \prod_{k=0}^{K-1} Q_k^{e_k} \cdot Q_K \\ &\cdot \prod_{k=0}^{K-2} S_k^{(e_k - e_{K-1}) y} \cdot X^{e_{K-1} y}\end{aligned} \tag{46}$$

where the two composite generators $\mathbf{h}$ and $\mathbf{g}$ are defined in Eqn. (33). Note that the major difference from Π_{frg} in Section 4.1.1 is that the verifier only needs to use a single equation to check the validity of the proof rather than three. On the prover side, in the first round, the prover sends all the commitments except Q_K before receiving a random challenge y from the verifier in the second round. Then the prover uses y to compute Q_K in the third round before obtaining the challenge vector $(e_k)_{k=0}^{K-1}$ in the fourth round. The value of the field element u in the fifth round is adapted accordingly to satisfy the equality. In a nutshell, we trade off round complexity for communication complexity. Even though, the round complexity of our proof is still lower than those of Bulletproof and SwiftRange.

Corollary 1. *The range proof Π_{rfrg} is a 5-move protocol, which has perfect completeness, computational witness-extended emulation and perfect special honest-verifier zero-knowledge (SHVZK).*

The range proof Π_{rfrg} is a special case of the aggregate range proof Π_{arg}, where $J = 1$. Consequently, it is a corollary of Theorem 1.

5.4 Compressed proofs

Recall that the vector dimension $|\mathbf{v}| = \frac{N}{K}$ must be a power of 2 so that the compression protocols can be applied. Thus, we can append zeros to $\mathbf{v}$ until its dimension reaches the nearest power of 2 if $|\mathbf{v}|$ is not. Then the prover can recursively apply $(\lceil \log \frac{N}{K} \rceil - 3)$ times of the compression protocol Π_{c} to the range protocol Π_{rfrg} until the vector dimension $|\mathbf{v}|$ is reduced to 8. The prover can call the protocol Π_{qc} with the inputs $(\mathbf{h}, \mathbf{g}, U, \mathbf{v})$, where U is initialized as RHS of Eqn. (46). Finally, the verifier can use the following equation to verify the validity of the proof:

$$\hat{\mathbf{h}}^{\hat{\mathbf{v}}} \cdot \hat{\mathbf{g}}^{-\hat{\mathbf{v}}^2} \stackrel{?}{=} \mathsf{RHS} \cdot \prod_{\tau=0}^{\lceil \log \frac{N}{K} \rceil - 4} A_\tau^{c_\tau^{-2}} B_\tau^{c_\tau^{-1}} D_\tau^{c_\tau} E_\tau^{c_\tau^2}$$

where $\hat{\mathbf{v}}$, $\hat{\mathbf{h}}$ and $\hat{\mathbf{g}}$ represent the final compressed vectors of witnesses and generators of dimension 8. The verifier needs to use $(2\lceil \frac{N}{K} \rceil - 16)$ group exponentiations to compute the composite generators $\hat{\mathbf{h}}$ and $\hat{\mathbf{g}}$. Fortunately, the verification can be reduced to a single multi-exponentiation for 2× efficiency improvement by leveraging the multi-exponentiation technique [28]. The verifier can pre-compute the exponents of the base generators before performing one-off group exponentiations using Eqn. (47):

$$g^a \prod_{l=0}^{\lceil \frac{N}{K} \rceil - 1} g_l^{a_l} \stackrel{?}{=} \mathsf{RHS} \cdot \prod_{\tau=0}^{\lceil \log \frac{N}{K} \rceil - 4} A_\tau^{c_\tau^{-2}} B_\tau^{c_\tau^{-1}} D_\tau^{c_\tau} E_\tau^{c_\tau^2} \tag{47}$$

where a and $(a_l)_{l=0}^{\lceil \frac{N}{K} \rceil - 1}$ are the pre-computed exponents for the corresponding base generators. We can see that computing the left-hand side of Eqn. (47) only involves $\lceil \frac{N}{K} \rceil + 1$ group exponentiations.

Corollary 2. *The compressed range proof Π_{FS} is a $(2\lceil\log\frac{N}{K}\rceil-1)$-move protocol, which has perfect completeness, computational witness-extended emulation and perfect special honest-verifier zero-knowledge (SHVZK).*

The compressed range proof Π_{FS} is a special case of the compressed aggregate range proof Π_{carg}, where $J = 1$. Consequently, it is a corollary of Theorem 2.

5.5 Configurability

Combining the techniques from Flashproof and SwiftRange gives us two cost functions $\mathcal{F}_{|\Pi_{\mathsf{FS}}|}$ in Eqn. (48) and $\mathcal{F}_{|\mathcal{V}_{\mathsf{FS}}|}$ in Eqn. (49) to calculate the number of elements in the proof and the number of computationally-intensive group exponentiations for verification, respectively:

$$\mathcal{F}_{|\Pi_{\mathsf{FS}}|} = \frac{K^2}{2} + \frac{3K}{2} + 4\lceil\log\frac{N}{K}\rceil - 3 \tag{48}$$

$$\mathcal{F}_{|\mathcal{V}_{\mathsf{FS}}|} = \lceil\frac{N}{K}\rceil + \frac{K^2}{2} + \frac{3K}{2} + 4\lceil\log\frac{N}{K}\rceil - 10 \tag{49}$$

We can draw the following conclusions, respectively:

- Based on the derivative $\mathcal{F}'_{|\Pi_{\mathsf{FS}}|} = K - \frac{4}{K\ln 2} + \frac{3}{2}$, $\mathcal{F}_{|\Pi_{\mathsf{FS}}|}$ achieves the minimum when $K = 2$.
- Based on the derivative $\mathcal{F}'_{|\mathcal{V}_{\mathsf{FS}}|} = K - \frac{4}{K\ln 2} - \frac{N}{K^2} + \frac{3}{2}$, $\mathcal{F}_{|\mathcal{V}_{\mathsf{FS}}|}$ achieves the minimum when $K \approx N^{\frac{1}{3}}$.

The two cost functions, each attaining minimum values at distinct K values, allow for configurable performance, offering a choice between the low-gear and high-gear settings:

- Low-Gear (LG): When $K = 2$, the proof has the smallest proof size but higher computational overheads.
- High-Gear (HG): When $K \approx N^{\frac{1}{3}}$, the proof has highest computational efficiency but larger proof size.

Moreover, by utilizing Flashproof's optimization in Section 4.1.2, we redefine the cost functions $\mathcal{F}_{|\Pi_{\mathsf{FS}}|}$ and $\mathcal{F}_{|\mathcal{V}_{\mathsf{FS}}|}$ for the optimized proof:

$$\mathcal{F}_{|\Pi_{\mathsf{FS}}|} = 4\lceil\log\frac{N}{K}\rceil + F(K) + K - 4 \tag{50}$$

$$\mathcal{F}_{|\mathcal{V}_{\mathsf{FS}}|} = \lceil\frac{N}{K}\rceil + 4\lceil\log\frac{N}{K}\rceil + F(K) + K - 11 \tag{51}$$

Remarks: In unoptimized implementations, a simple adjustment involves changing the parameter K. In optimized implementations, only $(\frac{K^2}{2} + \frac{K}{2} + 1)$ cross-terms need to be replaced by $F(K)$ terms, eliminating the need for an overhaul of the entire algorithm. This adjustment is much easier compared to transitioning to an entirely different range proof.

6 AGGREGATION

6.1 Aggregate proof Π_{arg}

FlashSwift allows to simultaneously prove J committed values lie within a specific range for efficiency improvement in both communication and verification. A J-aggregate proof uses $J \cdot L$ generators $(g_{j,l})_{j=0,l=0}^{J-1,L-1}$ to achieve the construction. In this section, we essentially use the parameter L instead of $\lceil\frac{N}{K}\rceil$ for brevity. Given J target commitments $(X_j)_{j=0}^{J-1}$, we use Eqn. (52) to verify our aggregate proof based on the aggregation technique of SwiftRange [32]:

$$\mathsf{LHS} \stackrel{?}{=} \mathsf{RHS} \tag{52}$$

$$\mathsf{LHS} = \prod_{j=0}^{J-1}(g^{\sum_{l=0}^{L-1} v_{j,l}})^{y^{j+1}} \cdot \prod_{j=0}^{J-1}\prod_{l=0}^{L-1} g_{j,l}^{f_{j,l}\cdot v_{j,l}} \tag{53}$$

$$\begin{aligned}\mathsf{RHS} = \rho^{-u} \cdot \prod_{0}^{\binom{K}{2}-1} T_{k,j}^{e_{k,j}} \cdot \prod_{j=0}^{J-1}\prod_{k=0}^{K-2} S_{j,k}^{(e_k - e_{K-1})\,y^{j+1}} \\ \cdot \prod_{j=0}^{J-1} X_j^{e_{K-1}y^{j+1}} \cdot \prod_{k=0}^{K-1} Q_k^{e_k} \cdot Q_K\end{aligned} \tag{54}$$

Note that the factors $(g^{\sum_{l=0}^{L-1} v_{j,l}})_{j=0}^{J-1}$ in Eqn. (53) are separated by a series of non-linear challenges $(y^{j+1})_{j=0}^{J-1}$. Accordingly, the factors $(X_j)_{j=0}^{J-1}$ in Eqn. (54) are isolated by a series of non-linear challenges $(e_{K-1}\cdot y^{j+1})_{j=0}^{J-1}$. We can further re-write Eqn. (53) to the compression-friendly form:

$$\begin{aligned}&\prod_{j=0}^{J-1}(g^{\sum_{l=0}^{L-1} v_{j,l}})^{y^{j+1}} \cdot \prod_{j=0}^{J-1}\prod_{l=0}^{L-1} g_{j,l}^{f_{j,l}\cdot v_{j,l}} \\ =&\prod_{j=0}^{J-1}\prod_{l=0}^{L-1}(g^{y^{j+1}} \cdot g_{j,l}^{f_{j,l}})^{v_{j,l}} \\ =&\prod_{j=0}^{J-1}\prod_{l=0}^{L-1}(g^{y^{j+1}} \cdot g_{j,l}^{\sum_{k=0}^{K-1} 2^{lK+k} e_k})^{v_{j,l}} \cdot \prod_{j=0}^{J-1}\prod_{l=0}^{L-1} g_{j,l}^{-v_{j,l}^2} \\ =&\,\mathbf{h}^{\mathbf{v}} \cdot \mathbf{g}^{-\mathbf{v}^2}\end{aligned} \tag{55}$$

where $\mathbf{v} = (v_{j,l})_{j=0,l=0}^{J-1,L-1}$, $\mathbf{h} = (g^{y^{j+1}} \cdot g_{j,l}^{\sum_{k=0}^{K-1} 2^{lK+k} e_k})_{j=0,l=0}^{J-1,L-1}$ and $\mathbf{g} = (g_{j,l})_{j=0,l=0}^{J-1,L-1}$.

Theorem 1. *Our aggregate range proof Π_{arg} is 5-move protocol, which has perfect completeness, computational witness-extended emulation and perfect special honest-verifier zero-knowledge (SHVZK).*

Please see Appendix A.2 for the proof of Theorem 1.

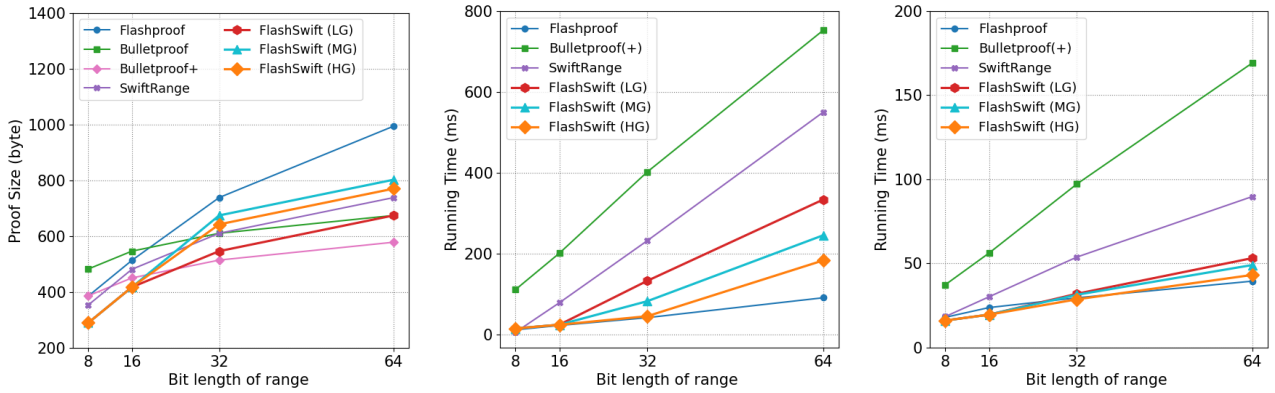
6.2 Compressed Aggregate proof Π_{carg}

By analogy, for the compressed aggregate proof, the prover needs to append zeros to $\mathbf{v}$ so as to recursively apply $(\lceil\log JL\rceil - 3)$ times of the compression protocol Π_{qc} to the aggregate range protocol Π_{arg} until the vector dimension $|\mathbf{v}|$ is reduced to 8. For an unoptimized J-aggregate proof, the number of elements would be:

$$\mathcal{F}_{|\Pi_{\mathsf{carg}}|} = 4\lceil\log JL\rceil + JK - J + \frac{K^2}{2} + \frac{K}{2} - 2 \tag{56}$$

The verification is dominated by:

$$\mathcal{F}_{|\mathcal{V}_{\mathsf{carg}}|} = JL + 4\lceil\log JL\rceil + JK + \frac{K^2}{2} + \frac{K}{2} - 10 \tag{57}$$



(a) The proof size in bytes. **(b) The proving time in milliseconds** **(c) The verification time in milliseconds**

Figure 1: The performance of the single proofs.

Compared with naively combining J single proofs, the savings are considerable:

$$\Delta\mathcal{F}_{|\Pi_{\text{carg}}|} = 4(J\lceil\log L\rceil - \lceil\log JL\rceil) + (J-1)(\frac{K^2}{2}+\frac{K}{2}+1) - 3J + 3$$

$$\Delta\mathcal{F}_{|\mathcal{V}_{\text{carg}}|} = 4(J\lceil\log L\rceil - \lceil\log JL\rceil) + (J-1)(\frac{K^2}{2}+\frac{K}{2}+1) - 11J + 11$$

Moreover, according to Eqn. (56) and (57), we can conclude that:

- Low-Gear (LG): When $K = 2$, the proof has the smallest proof size but higher computational overheads.
- High-Gear (HG): According to the derivative $\mathcal{F}'_{|\mathcal{V}_{\text{carg}}|} = K - \frac{4}{K\ln 2} - \frac{JN}{K^2} + J + \frac{3}{2}$, due to the existence of J, when K is slightly smaller than $(JN)^{\frac{1}{3}}$, the proof has highest computational efficiency. We highly recommend that readers empirically determine the optimal K through experiments.

Note that the same as the single proofs, the cross-terms $(\frac{K^2}{2}+\frac{K}{2}+1)$ can be replaced by $F(K)$ terms for optimization.

Theorem 2. *Our compressed aggregate range proof* Π_{carg} *is* $(2\lceil\log\frac{JN}{K}\rceil - 1)$*-move protocol, which has perfect completeness, computational witness-extended emulation and perfect special honest-verifier zero-knowledge (SHVZK).*

Please see Appendix A.3 for the proof of Theorem 2.

7 EXPERIMENTAL EVALUATION

7.1 Settings

We conducted comprehensive performance evaluations to compare the efficiency of FlashSwift with the other state-of-the-art range proof systems, namely Bulletproof(+), Flashproof and SwiftRange. Apart from our specially-designed low-gear (LG, K=2) and high-gear (HG, $K \approx (JN)^{\frac{1}{3}}$) settings, we also evaluated an extra medium-gear (MG, K=3) setting in our experiment to help readers understand our configurability feature. In general, our medium-gear proof demonstrates intermediate performance between the low-gear and high-gear counterparts, with a caveat. Exceptions in the proof size may occur for optimized single proofs, as Flashproof's optimization function $F(K)$ does not follow a regular pattern, as shown in Appendix B. We recommend users choose the low-gear and high-gear settings that best suit their specific applications.

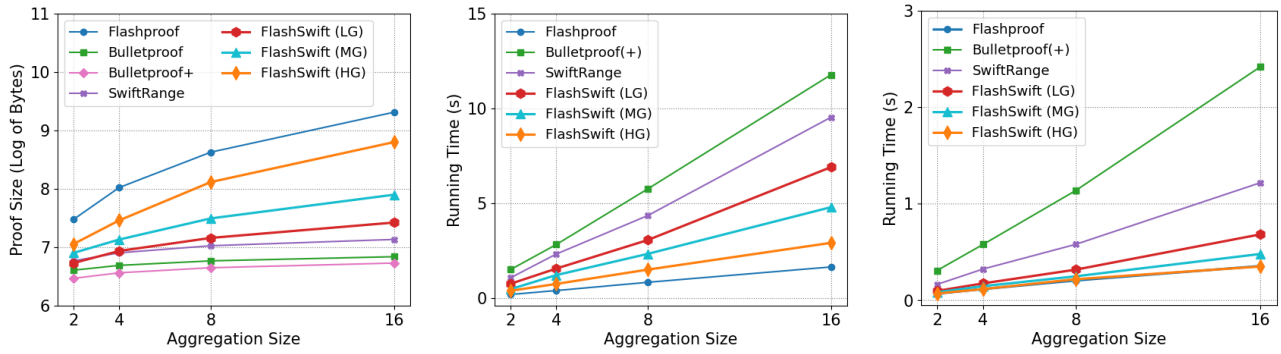
For our experiments, we utilized the same experimental settings as SwiftRange for straightforward comparisons. We employed the standard elliptic curve group of prime 254-bit order, *BN-128*[6], for Pedersen commitment schemes. Our experiments were executed in a single thread on a Java Virtual Machine 15 with an Apple M1 Pro processor, where the Java implementations were primarily aimed for performance comparison purposes and some lower-level programming languages, e.g., Rust and C, can provide higher computational efficiency in practice.

For communication overheads, we measured the proof sizes in bytes of different range proofs over a 256-bit field for the most common ranges where $N \leq 64$. We followed Bulletproof's strategy of using the compressed representation of elliptic curve points, which can be stored as a 256-bit value accompanied by an additional bit that indicates one of the two potential y coordinates. For computational overheads, we measured the running time of proving and verification of different range proofs, where we omit Bulletproof+ as it has comparable computational efficiency to Bulletproof. Moreover, we adopted the generic implementations of the unoptimized versions of FlashSwift and Flashproof for a fair comparison since the optimized versions for each combination of (J, N) require slightly specific implementations. We encourage readers to create the optimized versions that cater to their own needs. Our code is available at this github repository.

7.2 Line Charts

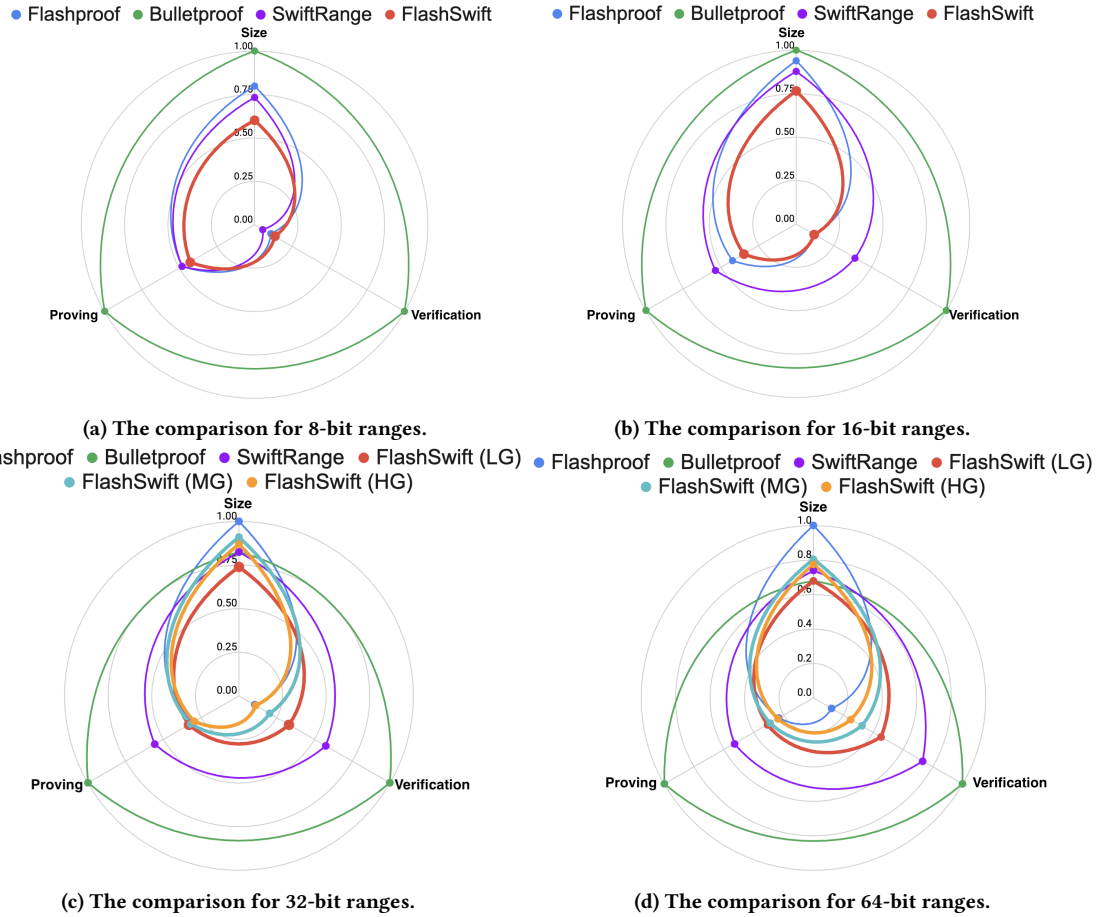
7.2.1 Communication Overhead. We generated a line chart in Figure 1a to offer a more direct proof size comparison than Table 2. The proof sizes of our low-gear proof start with the smallest sizes in bytes for 8-bit and 16-bit ranges among them all and intersect with that of Bulletproof when $N = 64$ and that of Bulletproof+ when N is slightly greater than 16. We observe a consistent 64-byte gap between SwiftRange and ours. Furthermore, our high-gear proof entails slightly greater communication costs compared to SwiftRange.

[6]Other standard groups can also be used, e.g., secp256k1.



(a) The proof size in bytes (b) The proving time in seconds (c) The verification time in seconds

Figure 2: The performance of the 64-bit aggregate proofs



(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 3: The normalized efficiency comparisons of different ranges regarding proof size, proving and verification time, where the closer the vertices to the center, the higher efficiency.

In stark contrast, a substantial reduction in communication costs becomes evident when compared with Flashproof. An exception arises wherein our medium-gear proofs for $N \in \{32, 64\}$ involve one more element than the high-gear counterparts. According to Eqn. (50), the term $4\lceil \log \frac{N}{3} \rceil$ for the medium-gear one incorporates four additional elements than the term $4\lceil \log \frac{N}{4} \rceil$ for the high-gear one, while $F(3)$+3 involves three fewer elements than $F(4)$+4. However, in most cases where unoptimized single proofs and aggregate

proofs are involved, no exceptions are observed. Turning to aggregate proofs as displayed in Figure 2a, it can be observed that the disparities between our low-gear proof and the other logarithmic-sized ones are small. The communication efficiency advantage of our high-gear proof over Flashproof is evident. Furthermore, our proof in the medium-gear configuration demonstrates a smooth efficiency improvement compared to the high-gear setting, and a reduction compared to the low-gear setting, respectively.

7.2.2 Computational Overhead. In Figure 1b and 1c, we can observe that our proofs run significantly faster than the other logarithmic-sized ones in both proving and verification. Notably, compared with Bulletproof for $N = 64$, our low-gear one achieves 2.3× and 3.2× proving and verification efficiency, respectively, which corresponds to the complexity comparison in Table 1. In comparison to SwiftRange, the efficiency improvements in proving and verification are also remarkable, which is a factor of 1.65 and 1.7, respectively. Moreover, the high-gear one retains comparable computational efficiency to Flashproof. The proving and verification efficiency is parallel to that of Flashproof when $N \leq 32$. When $N = 64$, our prover and verifier need to spend more time than those of Flashproof in executing one compression recursion for the sake of communication efficiency. In terms of verification running time, the efficiency gap is slim as the two lines almost coincide. As expected, our medium-gear proof attains an intermediate level of efficiency performance, falling between the efficiency levels of the low-gear and high-gear proofs. With respect to the aggregate proofs in Figure 2b and 2c, the performance disparities are approximately as significant as those of the single proofs.

7.3 Radar Charts

Apart from the line charts, we also provide four more straightforward radar charts in Figure 3 to present normalized efficiency comparisons across various ranges. For 8-bit and 16-bit ranges, both the charts of our three versions of proofs exactly coincide due to the use of the same $K = 2$. They demonstrate significantly superior performance than the other proofs. For 32-bit and 64-bit ranges, the advantages of the three proofs become more noticeable that the low-gear one is more communication efficient whereas the high-gear one is more computationally efficient. Our medium-gear proof demonstrates a balanced performance, positioned between the efficiency levels of the lower-gear and high-gear alternatives. Holistically, we can see that our proofs achieve the most balanced performance for all the ranges. Specifically, our low-gear proof outperforms the two logarithmic-sized ones in all three performance measures since their performance charts are fully enclosed by those of the two. Moreover, compared with Flashproof, our high-gear proof achieves a considerable advantage in communication efficiency for all four ranges, but slightly falls behind in both proving and verification efficiency.

7.4 Discussion

Based on our experimental findings, it is evident that our low-gear proof stands out with shorter size and significantly faster performance compared to the current state-of-the-art logarithmic-sized proofs. On the other hand, our high-gear proof, while slightly compromising on computational efficiency, exhibits a substantial improvement in communication efficiency, making it a preferable choice over Flashproof. Our medium-gear one exhibits a balanced performance between the former two, offering an additional choice for privacy-preserving applications. While the communication efficiency of the low-gear aggregate proof does not have logarithmic shortness, there remains considerable room for further enhancement. It currently keeps pace reasonably well with other logarithmic-sized proofs, and the minor communication gaps can be mitigated to some extent by its markedly higher computational efficiency. Furthermore, it's worth noting that single proofs suffice for meeting the requirements of many privacy-enhancing applications, such as BBA schemes, anonymous credentials, privacy-preserving crowdsensing.

8 CONCLUSION

In this paper, we presented, FlashSwift, a new logarithmic-sized zero-knowledge range proof in the DLOG setting with a transparent setup. We breach the inherent incompatibility barrier between Flashproof and SwiftRange to construct a shorter and significantly faster range proof than the state-of-the-art logarithmic-sized ones, Bulletproof and SwiftRange, for the most common ranges where $N \in \{8, 16, 32, 64\}$. To the best of our knowledge, on the one hand, our proof achieves the smallest proof sizes, 289 bytes and 417 bytes, for 8-bit and 16-bit ranges among all the bit-decomposition-based range proofs without requiring trusted setups; On the other hand, our proof is the first configurable range proof, whose communication efficiency can be traded off for computational efficiency to easily adapt to different scenarios. The high-gear proof gains a considerable communication efficiency advantage over Flashproof while incurring comparable computational efficiency. Our low-gear aggregate proof does not achieve the desirable logarithmic shortness. We anticipate overcoming this limitation in our forthcoming research endeavors.

ACKNOWLEDGMENTS

We express our gratitude to the anonymous reviewers and shepherd for their valuable and insightful comments, which contributed significantly to the improvement of the paper. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

REFERENCES

[1] Thomas Attema and Ronald Cramer. 2020. 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*. 513–543.
[2] Thomas Attema, Ronald Cramer, and Serge Fehr. 2021. Compressing Proofs of k-Out-Of-n Partial Knowledge. In *Advances in Cryptology – CRYPTO 2021*, Tal Malkin and Chris Peikert (Eds.). Springer International Publishing, Cham, 65–91.
[3] Mihir Bellare and Phillip Rogaway. 1993. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*.
[4] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. 2016. Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting. In *Advances in Cryptology – EUROCRYPT 2016*, Marc Fischlin and Jean-Sébastien Coron (Eds.). Springer Berlin Heidelberg, 327–357.
[5] Jonathan Bootle and Jens Groth. 2018. Efficient Batch Zero-Knowledge Arguments for Low Degree Polynomials. In *Public-Key Cryptography – PKC 2018*.
[6] Fabrice Boudot. [n. d.]. Efficient Proofs that a Committed Number Lies in an Interval. In *Advances in Cryptology — EUROCRYPT 2000*.

[7] Benedikt Bunz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short Proofs for Confidential Transactions and More. 315–334.

[8] Chengjun Cai, Yifeng Zheng, Yuefeng Du, Zhan Qin, and Cong Wang. 2021. Towards Private, Robust, and Verifiable Crowdsensing Systems via Public Blockchains. *IEEE Transactions on Dependable and Secure Computing* 18, 4 (2021), 1893–1907. https://doi.org/10.1109/TDSC.2019.2941481

[9] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. 2008. Efficient Protocols for Set Membership and Range Proofs. In *Advances in Cryptology - ASIACRYPT 2008*.

[10] Rafik Chaabouni, Helger Lipmaa, and Abhi Shelat. 2010. Additive Combinatorics and Discrete Logarithm Based Range Protocols. In *Information Security and Privacy*, Ron Steinfeld and Philip Hawkes (Eds.). 336–351.

[11] Heewon Chung, Kyoohyung Han, Chanyang Ju, Myungsun Kim, and Jae Hong Seo. 2022. Bulletproofs+: Shorter Proofs for a Privacy-Enhanced Distributed Ledger. *IEEE Access* 10 (2022), 42081–42096. https://doi.org/10.1109/ACCESS.2022.3167806

[12] Geoffroy Couteau, Dahmun Goudarzi, Michael Klooß, and Michael Reichle. 2022. Sharp: Short Relaxed Range Proofs. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. 609–622.

[13] Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. 2021. Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments. In *Advances in Cryptology - CRYPTO 2021*.

[14] Cong Deng, Xianghong Tang, Lin You, and Gengran Hu. 2021. Cuproof: A Novel Range Proof with Constant Size. *IACR Cryptol. ePrint Arch.* (2021).

[15] Samuel Dobson, Steven Galbraith, and Benjamin Smith. 2022. Trustless unknown-order groups. *Mathematical Cryptology* 1, 2 (Mar. 2022), 25–39. https://journals.flvc.org/mathcryptology/article/view/130579

[16] Amos Fiat and Adi Shamir. 1987. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology — CRYPTO' 86*.

[17] Shang Gao, Zhe Peng, Feng Tan, Yuanqing Zheng, and Bin Xiao. 2022. SymmeProof: Compact Zero-Knowledge Argument for Blockchain Confidential Transactions. *IEEE Transactions on Dependable and Secure Computing* (2022), 1–1.

[18] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. 1985. The knowledge complexity of interactive proof-systems. In *Symposium on the Theory of Computing*.

[19] Maxwell Gregory. 2016. Confidential Transactions. https://elementsproject.org/features/confidential-transactions/investigation.

[20] Jens Groth. 2005. Non-interactive Zero-Knowledge Arguments for Voting. In *Applied Cryptography and Network Security*.

[21] Jens Groth. 2016. On the Size of Pairing-Based Non-interactive Arguments. In *Advances in Cryptology – EUROCRYPT 2016*.

[22] Max Hoffmann, Michael Klooß, Markus Raiber, and Andy Rupp. 2020. Black-Box Wallets: Fast Anonymous Two-Way Payments for Constrained Devices. *Proceedings on Privacy Enhancing Technologies* 2020 (01 2020), 165–194. https://doi.org/10.2478/popets-2020-0010

[23] Meng Li, Yifei Chen, Chhagan Lal, Mauro Conti, Mamoun Alazab, and Donghui Hu. 2023. Eunomia: Anonymous and Secure Vehicular Digital Forensics Based on Blockchain. *IEEE Transactions on Dependable and Secure Computing* 20, 1 (2023), 225–241. https://doi.org/10.1109/TDSC.2021.3130583

[24] Helger Lipmaa. 2003. On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In *Advances in Cryptology - ASIACRYPT 2003*.

[25] Dongxiao Liu, Amal Alahmadi, Jianbing Ni, Xiaodong Lin, and Xuemin Shen. 2019. Anonymous Reputation System for IIoT-Enabled Retail Marketing Atop PoS Blockchain. *IEEE Transactions on Industrial Informatics* 15, 6 (2019), 3527–3537. https://doi.org/10.1109/TII.2019.2898900

[26] Tianyi Liu, Xiang Xie, and Yupeng Zhang. 2021. ZkCNN: Zero Knowledge Proofs for Convolutional Neural Network Predictions and Accuracy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 2968–2985. https://doi.org/10.1145/3460120.3485379

[27] Dmitry Khovratovich Michael Lodder. 2019. Anonymous Credential 2.0. https://wiki.hyperledger.org/download/attachments/6426712/Anoncreds2.1.pdf

[28] Nicholas Pippenger. 1980. On the Evaluation of Powers and Monomials. *SIAM J. Comput.* 9, 2 (may 1980), 230–250. https://doi.org/10.1137/0209022

[29] Georgia Tsaloli, Alejandro Lancho, Katerina Mitrokotsa, and Giuseppe Durisi. 2022. WiP: Verifiable, Secure and Energy-Efficient Private Data Aggregation in Wireless Sensor Networks. In *Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies* (New York, NY, USA) *(SACMAT '22)*. Association for Computing Machinery, New York, NY, USA, 61–66. https://doi.org/10.1145/3532105.3535040

[30] Nan Wang and Sid Chi-Kin Chau. 2022. Flashproofs: Efficient Zero-Knowledge Arguments of Range and Polynomial Evaluation with Transparent Setup. In *Advances in Cryptology – ASIACRYPT 2022*, Shweta Agrawal and Dongdai Lin (Eds.). 219–248.

[31] Nan Wang, Sid Chi-Kin Chau, and Dongxi Liu. 2023. SwiftRange: A Short and Efficient Zero-Knowledge Range Argument For Confidential Transactions and More. Cryptology ePrint Archive, Paper 2023/1185. https://eprint.iacr.org/2023/1185 https://eprint.iacr.org/2023/1185.

[32] Nan Wang, Sid Chi-Kin Chau, and Dongxi Liu. 2024. SwiftRange: A Short and Efficient Zero-Knowledge Range Argument For Confidential Transactions and More. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 54–54. https://doi.org/10.1109/SP54263.2024.00054

A SECURITY ANALYSIS

A.1 Useful Theorems

We provide a brief overview of the forking lemma from [4] and the security theorem of the compression protocol from SwiftRange [32], which will be essential for the security proofs of our range arguments.

A.1.1 A General Forking Lemma. Suppose we have a public-coin argument with $(2\mu-1)$ moves and μ challenges $e_1, ..., e_\mu$ in sequence. We consider $\prod_{i=1}^{\mu}\theta_i$ accepting transcripts organized in a tree format. This tree has a depth of μ and $\prod_{i=1}^{\mu}\theta_i$ leaves, with the root labeled with the statement. Each node at depth i has exactly θ_i children, each labeled with a distinct value of the i-th challenge e_i. We refer to this structure as an $(\theta_1, ..., \theta_\mu)$-tree of accepting transcripts. The forking lemma naturally generalizes the special soundness for public-coin arguments with $(2\mu-1)$ moves. Thus, by using Theorem 3, we will prove that our protocols possess witness-extended emulation.

Theorem 3 (Forking Lemma). *Let $\mathcal{G}, \mathcal{P}, \mathcal{V}$ be a $(2\mu-1)$-move, public coin interactive protocol. Let $\mathcal{E}$ be a witness extraction algorithm that succeeds with probability $1-\mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\lambda)$ in extracting a witness from an $(\theta_1, ..., \theta_\mu)$-tree of accepting transcripts in probabilistic polynomial time. Assume that $\prod_{i=1}^{\mu}\theta_i$ is bounded above by a polynomial in the security parameter λ. Then $\mathcal{G}, \mathcal{P}, \mathcal{V}$ has witness-extended emulation.*

A.1.2 Compression Protocol Theorem. The security theorem of the compression protocol Π_{qc} goes as below. Please refer to the original paper [31] for detailed security proofs.

Theorem 4. *The compression protocol Π_{qc} is a 3-move protocol, which has perfect completeness and computational 5-special soundness.*

A.2 Proof of Theorem 1

Proof. **Perfect completeness** follows by a careful inspection of the protocol. Then we depict a **perfect SHVZK simulation**. Given a challenge y, a challenge vector $(e_k)_{k=0}^{K-1}$ and a series of target commitments $(X_j)_{j=0}^{J-1}$, a simulator randomly picks up group elements $(T_{k,j})_0^{\binom{K}{2}-1}, (S_{j,k})_{j=0,k=0}^{J-1,K-2}, (Q_k)_{k=0}^{K-1} \xleftarrow{\$} \mathbb{G}$ and field elements $(v_{j,l})_{j=0,l=0}^{J-1,L-1}, u \xleftarrow{\$} \mathbb{Z}_p^*$. By the perfect hiding property, the Pedersen commitments in a real argument are uniformly random as those in the simulation. The field elements in a real argument are also uniformly random due to the random choices of $(r_{j,l})_{j=0,l=0}^{J-1,L-1}, (r_{q_k})_{k=0}^{K}$ and $(r_{s_{j,k}})_{j=0,k=0}^{J-1,K-2}$. Hence, in both real argument and simulation, the random elements uniquely determine the value Q_K by computing:

$$Q_K = \mathbf{h}^{-\mathbf{v}} \cdot \mathbf{g}^{\mathbf{v}^2} \cdot \rho^u \cdot \prod_{0}^{\binom{K}{2}-1} T_{k,j}^{-e_{k,j}} \cdot \prod_{j=0}^{J-1}\prod_{k=0}^{K-2} S_{j,k}^{(e_{K-1}-e_k)\,y^{j+1}}$$

$$\cdot \prod_{j=0}^{J-1} X_j^{-e_{K-1} y^{j+1}} \cdot \prod_{k=0}^{K-1} Q_k^{-e_k}$$

where $\mathbf{v} = (v_{j,l})_{j=0,l=0}^{J-1,L-1}$. This means we have identical distributions of real and simulated arguments with the given challenges.

Finally, we prove **witness-extended emulation**. The protocol Π_{carg} fulfils γ-special soundness. Given the verification equality as below:

$$\mathbf{h}^{\mathbf{v}} \cdot \mathbf{g}^{-\mathbf{v}^2} \stackrel{?}{=} \rho^{-u} \cdot \prod_{0}^{\binom{K}{2}-1} T_{k,j}^{e_{k,j}} \cdot \prod_{j=0}^{J-1} \prod_{k=0}^{K-2} S_{j,k}^{(e_k - e_{K-1}) y^{j+1}} \cdot \prod_{j=0}^{J-1} X_j^{e_{K-1} y^{j+1}} \cdot \prod_{k=0}^{K-1} Q_k^{e_k} \cdot Q_K$$

Let $(a_t)_{t=0}^{\gamma} \leftarrow \mathbb{Z}_p$ be a tuple of variables, where $\gamma = JK + \frac{1}{2}(K^2 + K) + 1$. We can create γ equations by multiplying these variables on both sides:

$$\begin{aligned}
&\Big((\mathbf{h}^{\mathbf{v}^{(t)}} \cdot \mathbf{g}^{-(\mathbf{v}^{(t)})^2} \cdot \rho^{u^{(t)}})^{a_t} \\
&= \Big(\prod_{0}^{\binom{K}{2}-1} T_{k,j}^{e_{k,j}^{(t)}} \cdot \prod_{j=0}^{J-1} \prod_{k=0}^{K-2} S_{j,k}^{(e_k^{(t)} - e_{K-1}^{(t)})(y^{(t)})^{j+1}} \\
&\quad \cdot \prod_{j=0}^{J-1} X_j^{e_{K-1}^{(t)} (y^{(t)})^{j+1}} \cdot \prod_{k=0}^{K-1} Q_k^{e_k^{(t)}} \cdot Q_K)^{a_t} \Big)_{t=0}^{\gamma-1}
\end{aligned}$$

Then we obtain the following by taking a linear combination of these γ equations:

$$\begin{aligned}
&\mathbf{h}^{\sum_{t=0}^{\gamma-1} \mathbf{v}^{(t)} a_t} \cdot \mathbf{g}^{-\sum_{t=0}^{\gamma-1} (\mathbf{v}^{(t)})^2 a_t} \cdot \rho^{\sum_{t=0}^{\gamma-1} u^{(t)} a_t} \\
&= \prod_{0}^{\binom{K}{2}-1} T_{k,j}^{\sum_{t=0}^{\gamma-1} e_{k,j}^{(t)} a_t} \cdot \prod_{j=0}^{J-1} \prod_{k=0}^{K-2} S_{j,k}^{\sum_{t=0}^{\gamma-1} (e_k^{(t)} - e_{K-1}^{(t)})(y^{(t)})^{j+1} a_t} \\
&\quad \cdot \prod_{j=0}^{J-1} X_j^{\sum_{t=0}^{\gamma-1} e_{K-1}^{(t)} (y^{(t)})^{j+1} a_t} \cdot \prod_{k=0}^{K-1} Q_k^{\sum_{t=0}^{\gamma-1} e_k^{(t)} a_t} \cdot Q_K^{\sum_{t=0}^{\gamma-1} a_t}
\end{aligned}$$

To ensure the left-hand side is equal to the β-th target commitment X_β, where $\beta \in \{0, ..., J-1\}$, we need the following γ equations:

$$\begin{aligned}
&\Big(\sum_{t=0}^{\gamma-1} e_{k,j}^{(t)} a_t = 0 \Big)_0^{\binom{K}{2}-1}, \\
&\Big(\sum_{t=0}^{\gamma-1} (e_k^{(t)} - e_{K-1}^{(t)})(y^{(t)})^{j+1} a_t = 0 \Big)_{k=0,j=0}^{J-1,K-2}, \\
&\Big(\sum_{t=0}^{\gamma-1} e_{K-1}^{(t)} (y^{(t)})^{\beta+1} a_t \Big) = 1, \\
&\Big(\sum_{t=0}^{\gamma-1} e_{K-1}^{(t)} (y^{(t)})^{j+1} a_t \Big)_{j \neq \beta}^{J-1} = 0, \\
&\Big(\sum_{t=0}^{\gamma-1} e_k^{(t)} a_t = 0 \Big)_{k=0}^{K-1}, \\
&\sum_{t=0}^{\gamma-1} a_t = 0
\end{aligned}$$

We can obtain a $\gamma \times \gamma$ challenge matrix:

$$\mathbf{e} = \begin{pmatrix}
e_{0,0}^{(0)} & \cdots & e_{0,0}^{(\gamma-1)} \\
\vdots & \ddots & \vdots \\
e_{k=K-1,j=K-2,k\neq j}^{(0)} & \cdots & e_{k=K-1,j=K-2,k\neq j}^{(\gamma-1)} \\
(e_0^{(0)} - e_{K-1}^{(0)}) y^{(0)} & \cdots & (e_0^{(\gamma-1)} - e_{K-1}^{(\gamma-1)}) y^{(\gamma-1)} \\
\vdots & \ddots & \vdots \\
(e_{K-2}^{(0)} - e_{K-1}^{(0)})(y^{(0)})^J & \cdots & (e_{K-2}^{(\gamma-1)} - e_{K-1}^{(\gamma-1)})(y^{(\gamma-1)})^J \\
e_{K-1}^{(0)} y^{(0)} & \cdots & e_{K-1}^{(\gamma-1)} y^{(\gamma-1)} \\
\vdots & \ddots & \vdots \\
e_{K-1}^{(0)} (y^{(0)})^{\beta+1} & \cdots & e_{K-1}^{(\gamma-1)} (y^{(\gamma-1)})^{\beta+1} \\
\vdots & \ddots & \vdots \\
e_{K-1}^{(0)} (y^{(0)})^J & \cdots & e_{K-1}^{(\gamma-1)} (y^{(\gamma-1)})^J \\
e_0^{(0)} & \cdots & e_0^{(\gamma-1)} \\
\vdots & \ddots & \vdots \\
e_{K-1}^{(0)} & \cdots & e_{K-1}^{(\gamma-1)} \\
1 & \cdots & 1
\end{pmatrix}$$

We can see that all the rows and columns of the left multiplying challenge matrix are linearly independent since these challenges are randomly generated based on the assumption of the random oracle model. The matrix is invertible for being a full-rank matrix. We can compute the openings of the β-th commitment:

$$\mathbf{e}^{-1} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{\gamma-1} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{58}$$

A solution of such variables $(a_t)_{t=0}^{\gamma-1}$ to the equations exists. We can obtain the openings of each target commitment by setting a different vector whose Hamming weight is 1 on the right-hand side of Eqn. (58).

A.3 Proof of Theorem 2

Proof. The security proof of Π_{carg} is similar to that of Π_{carg} in [32]. **Perfect completeness** follows by carefully inspecting the final verification equation:

$$\hat{\mathbf{h}}^{\hat{\mathbf{v}}} \cdot \hat{\mathbf{g}}^{-\hat{\mathbf{v}}^2} \stackrel{?}{=} \rho^{-u} \cdot \prod_{0}^{\binom{K}{2}-1} T_{k,j}^{e_{k,j}} \cdot \prod_{k=0}^{K-2} S_{j,k}^{(e_k - e_{K-1}) y^{j+1}} \cdot \prod_{j=0}^{J-1} X_j^{e_{K-1} y^{j+1}} \cdot \prod_{k=0}^{K-1} Q_k^{e_k} \cdot Q_K \cdot \prod_{\tau=0}^{\lceil \log \frac{JN}{K} \rceil - 4} A_\tau^{c_\tau^{-2}} B_\tau^{c_\tau^{-1}} D_\tau^{c_\tau} E_\tau^{c_\tau^2}$$

where $\hat{\mathbf{v}}$, $\hat{\mathbf{h}}$ and $\hat{\mathbf{g}}$ are the final compressed vectors of witnesses and generators of dimension 8.

The compressed aggregate range argument Π_{carg} exhibits a **perfect SHVZK** property, which is derived from the **perfect SHVZK**

property of the aggregate range argument Π_{arg} through the following composition structure:

$$\Pi_{\mathsf{carg}} = \underbrace{\Pi_{\mathsf{qc}} \diamond \cdots \diamond \Pi_{\mathsf{qc}}}_{(\lceil \log \frac{JN}{K} \rceil - 3) \text{ times}} \diamond \Pi_{\mathsf{arg}}$$

The simulator of Π_{carg} operates by running the simulator of Π_{arg} and then replacing the final messages of the simulated transcripts through honest executions of $\Pi_{\mathsf{qc}} \diamond \cdots \diamond \Pi_{\mathsf{qc}}$. The subsequent compression protocols Π_{qc} preserve their zero-knowledge property. The field vector $\mathbf{v}'$ is the sum of two zero-knowledge field elements $\mathbf{v_L}$ and $c\mathbf{v_R}$. Additionally, the Pedersen commitments are perfectly hiding. As a result, the zero-knowledge property of the overall protocol Π_{carg} remains unchanged.

Finally, we establish the **computational witness-extended emulation** of Π_{carg}. The protocol Π_{carg} is a combination of a γ-special sound aggregate range protocol Π_{arg}, where $\gamma = \left(JK + \frac{1}{2}(K^2 + K) + 1\right)$ and a sequence of 5-special sound compression protocols Π_{qc}. As a result, Π_{carg} can be observed as $(\gamma, 5, ..., 5)$-special sound. This implies the existence of an efficient knowledge emulator that can extract the witnesses of the commitments along the path from the leaves to the root of the $(\gamma, 5, ..., 5)$-tree of $\gamma \cdot 5^{\log(JN) - \log K - 3}$ accepting transcripts. The emulator utilizes fewer than $\gamma \cdot 5^{\log(JN) - \log K - 3} < \gamma \cdot 8^{\log(JN)} = \left(JK + \frac{1}{2}(K^2 + K) + 1\right) \cdot (JN)^3$ transcripts and thus runs in expected polynomial time in N, K and J.

B THE FUNCTION $F(K)$

Flashproof gives a list of values for the optimization function $F(K)$ of Flashproof in Table 3. Please refer to their paper [30] for more details.

Table 3: The values of $F(K)$

K	2	3	4	5	6	7	8	9	10
$F(K)$	3	6	8	11	13	20	27	32	37