SymmeProof: Compact Zero-Knowledge Argument for Blockchain Confidential Transactions

Shang Gao, Zhe Peng, Feng Tan, Yuanqing Zheng *Member, IEEE*, and Bin Xiao *Senior Member, IEEE*

***Abstract*—To reduce the transmission cost of blockchain confidential transactions, we propose SymmeProof, a novel communication efficient non-interactive zero-knowledge range proof protocol without a trusted setup. We design and integrate two new techniques in SymmeProof, namely vector compression and inner-product range proof. The proposed vector compression is able to reduce the communication cost to $\log(n)$ for n-size vectors. The proposed inner-product range proof converts a range proof relation into an inner-product form, which can further reduce the range proof size with the vector compression technique. Based on these two techniques, SymmeProof can eventually achieve a $\log(n)$-size range proof. The proposed SymmeProof can be used in many important applications such as blockchain confidential transactions as well as arguments for arithmetic circuit satisfiability. We evaluate the performance of SymmeProof. The results show that SymmeProof substantially outperforms representative methods such as Bulletproofs in the proof size without a trusted setup.**

***Index Terms*—Blockchain, privacy preservation, confidential transactions, zero-knowledge argument, range proofs, Bulletproofs.**

I. Introduction

Blockchain-based cryptocurrencies can avoid tampering attempts from minority attackers by maintaining a copy of all transactions at distributed participants. Among all the implementations, Bitcoin [1] is the first fully decentralized cryptocurrency, which requires all details of transactions for validation. Although Bitcoin can provide some weak anonymity by using many identities (*pseudonyms*), the amount of money transferred in transactions (i.e., *confidentiality*) is public to everyone. This serious limitation makes Bitcoin unsuitable for confidential scenarios, such as a second-price auction which requires confidentiality to incentivize truthful bidding.

To this end, confidential transactions (CT) [2], [3], [4], [5] use homomorphic commitments to hide amounts. In order to support public verifications on the transactions, zero-knowledge proof or zero-knowledge argument techniques[1] can be applied by conducting verifications on the confidential transactions (commitments). Some current CT zero-knowledge argument (e.g. zero-knowledge succinct non-interactive argument of knowledge, zk-SNARK) requires a trusted setup [6], [7], [8]. The security of these protocols is based on the assumption that the setup is performed properly. Zk-SNARK can be further improved by replacing the costly setup with an updatable and universal setup [9], such as Sonic [10], PLONK [11], and Lunar [12]. However, the universal setup is still not practical without a trusted third party, which breaks the decentralized property of the blockchain systems. The zero-knowledge scalable and transparent argument of knowledge (zk-STARK) [13] can reduce the trusted setup procedure and has many promising applications in cryptocurrencies. Unfortunately, the proof size of zk-STARK is much larger than existing approaches. As the proofs need to be transmitted over the whole network and stored for a long time, zk-STARK incurs high communication and storage overhead.

To improve the efficiency of zk-STARKs while reducing the trusted setup (transparent zk-STARKs), Supersonic [14] is proposed. With a new μ-multivariate polynomial commitment scheme, the proof size of Supersonic is reduced to $O(\mu \log(n))$. Dory [15] reduces the prover's time complexity in Supersonic but incurs a larger proof size. Fractal [16] further enables the post-quantum property in transparent proofs. Bootle et al. [17] present another "two-set splitting" technique for vector compressing which can reduce the size of an inner-product argument to $6\log(n)$ for arithmetic circuit satisfiability. Bulletproofs protocol [18] further leverages Bootle's vector compressing idea and applies to range proofs. By simultaneously dealing with three elements, a more compressed inner-product argument with $2\log(n)$ size is proposed. Besides, since Bulletproofs protocol optimizes the underlying range proof technique of blockchain, it is compatible with current application-level compression approaches such as Mimblewimble [19], [20]. Though the verification time of Bulletproofs is polynomial which is less efficient than some zk-SNARK approaches, Bulletproofs protocol is built on the *falsifiable*[2] discrete logarithm assumption. Based on the result from Gentry and Wichs, "*there is no black-box reduction security proof for any (zk-)SNARK under falsifiable assumptions*" [21], which implies poly-logarithm verification is impossible for Bulletproofs.

Our goal is to design more efficient protocols for range proofs without a trusted setup, which significantly reduces the cost of transmission and storage. Specifically, we aim to

Shang Gao, Yuanqing Zheng, and Bin Xiao are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: shanggao@comp.polyu.edu.hk, csyqzheng@comp.polyu.edu.hk, csbxiao@comp.polyu.edu.hk).

Zhe Peng is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong (e-mail: pengzhe@comp.hkbu.edu.hk).

Feng Tan is with the Shanghai Artificial Intelligence Institute, China (e-mail: Tf.uestc@gmail.com).

[1]Precisely, zero-knowledge proofs (with statistical soundness) are different from zero-knowledge arguments (with computational soundness). In this paper, we only discuss "arguments" and use "proofs" and "arguments" interchangeably since "proofs" cannot have communication less than the witness size, which makes "proofs" impractical in most real-world applications.

[2]A cryptographic assumption is falsifiable if it can be modeled as a game between an adversary and an efficient challenger which the challenger can finally determine whether the adversary won the game [21].

reduce the size of range proof, which could eventually make CT techniques practical in the real-world applications such as cryptocurrencies.

In this paper, we propose SymmeProof, a logarithmic-size honest verifier zero-knowledge range proof argument. Comparing with the previous logarithmic-size Bulletproofs, the size of SymmeProof is only *half* of the Bulletproofs under the same setting. SymmeProof has perfect completeness, perfect honest verifier zero-knowledge, and computational soundness (based on challenge space size and discrete logarithm assumption). Meanwhile, it preserves most of advantages of Bulletproofs, such as reducing setup procedures, aggregating multiple proofs, and is compatible with other compression techniques [22], [20]. Furthermore, SymmeProof can also be generalized to other applications such as arithmetic circuit satisfiability. We summarize our contributions as follows:

- *Vector compression.* We analyze the techniques in Bulletproofs and propose a more compact vector compression technique based on randomization and quadratic residue, which can reduce the size of existing proofs by half. Based on this technique, we construct inner-product arguments with only $\log(n)$ communication cost.
- *Fewer challenges.* We propose a new technique that can convert range proofs to an inner-product form with fewer challenges. Comparing with the existing range proof protocol (e.g., Bulletproofs), our technique can reduce the communication size in interactive scenarios and the computational cost (hash function) in non-interactive scenarios.
- *Combination of techniques.* We combine our proposed techniques to build a range proof protocol named SymmeProof, which can significantly reduce the communication cost ($\log(n)$ proof size) without a trusted setup. We also compare SymmeProof with the state-of-the-art methods and evaluate its performance.
- *More applications.* We discuss describe the generalization of SymmeProof to many important applications such as multi-party computation protocol and arithmetic circuit satisfiability. As a matter of fact, since SymmeProof optimizes the underlying vector compression techniques, it can be applied to most of today's vector argument, inner-product argument, and range proof applications to improve their performances.

The rest of this paper is organized as follows. Section II present some related work and background knowledge of range proofs and zero-knowledge arguments. In Section III, we analyze Bulletproofs and present our new techniques. Based our new ideas, we construct inner-product arguments and range proofs in Section IV and Section V respectively, and further build SymmeProof, a logarithm-size range proof protocol in VI. Evaluations are conducted in Section VII. We discuss SymmeProof in general settings and further applications in Section VIII. Finally, we conclude this paper in Section IX.

II. Preliminaries

A. Related Work

Range proof. To keep the amount of transactions as a secret, confidential transactions [2] hide the input and output amounts in Pedersen commitments and encapsulate zero-knowledge proofs to ensure (1) all the inputs and outputs are positive, and (2) the sum of inputs equals outputs. The requirements can be converted into range proofs to show the secret (amount or sum) lies in a range [18]. Today's implementations (e.g. Provisions protocols [23]) heavily rely on range proofs to avoid malicious transactions, which becomes a bottleneck of the protocols since they have a $O(n)$ proof size. For instance, records in Provisions protocol (about 2 million accounts) contain proofs of 18GB. More than 70% space (about 13GB) is used for range proofs.

Mimblewimble [19] explores the fact that the difference between outputs, inputs, and transaction fees should be 0 for a valid transaction. Therefore, by regarding an ECDSA public key as a commitment to 0, the verifier can use the public key as the signature of the difference. This idea reduces the transmission of scriptSig (the signatures of unspent transaction outputs) as well as the proof size. A further improvement [20] presents a blockchain compression technique which only contains some block headers and unspent addresses (outputs). Meanwhile, verifiers can also verify the entire blockchain without having spent addresses. Though Mimblewimble can compress the blockchain to reduce the size, the proof size is still linear.

Bootle et al. propose a vector compression technique for arithmetic circuit satisfiability [17], which can reduce the size of a vector argument to $2\log(n)$. By adopting this technique in the inner-product argument, the proof size is reduced to $6\log(n)$ (individually dealing with the two vectors and their product, which is three times of a single vector argument). Motivated by Bootle's idea, Bulletproofs protocol [18] simultaneously deals with vectors and the product in an inner-product argument and reduces the range proof size to $2\log(n)$. Since these approaches optimize the range proof technique, they can work with Mimblewimble to build a more practical blockchain which significantly reduces the size of the Bitcoin network.

Zero-knowledge argument. To enable the programmability of Bitcoin, Ethereum [24] adopts the idea of smart contract to support complex transactions for different applications. For some privacy-related scenarios, a non-interactive zero-knowledge (NIZK) technique can be used to protect the users' inputs [25], [26], [27], [28], [29], [30]. However, the communication and computation cost makes NIZK not suitable for smart contracts since the communication over the blockchain network is expensive and the computational power is limited. A further improvement, zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK) [7], [8], can reduce the communication and computation cost. Unfortunately, some zk-SNARK techniques require a trusted setup. The cost of a trusted setup is also significant. For instance, HAWK [31] with zk-SNARK technique needs a trusted third party, or a costly multi-party computation to generate long common reference strings for different smart

contracts.

The trusted setup can be replaced with a universal setup: a one-time setup for any computations [9], which is used in Sonic [10], PLONK [11], and Lunar [12]. These systems combine a reduction of circuit satisfiability (for the probabilistic tests of polynomials) with polynomial commitment schemes to build SNARKs. The polynomial commitment scheme's setup also needs to be trusted, but can be updated and reused for any computations, which only needs to be performed once. However, these systems still need to involve a trusted party to perform the setup (at least once).

One can fully reduce the setup by adopting scalable computational integrity (SCI) [32] or zero-knowledge scalable and transparent argument of knowledge (zk-STARK) [13] with succinct proofs and efficient verifiers. Zk-STARK uses public and random parameters to generate proofs, which completely removes the need for a trusted third party or a costly multi-party computation. Furthermore, the proofs with zk-STARK can be extended to quantum-secure range proofs (using ElGamal commitments instead of Pedersen commitments), which have a longer life cycle than those under discrete logarithm settings (can resist potential attacks from quantum computers). Though zk-STARK is promising, its limitation is also obvious: the proof size is too large which makes it impractical to be applied to most blockchain applications. Comparing with a 288B zk-SNARK proof, zk-STARK requires 45KB to 200KB storage, which is more than 160 times of zk-SNARK.

Supersonic uses a new polynomial commitment scheme for μ-multivariate polynomials [14] based on the Diophantine Arguments of Knowledge (DARK) proof system [33]. The commitment scheme can fully remove the trusted setup with class groups of an imaginary quadratic order. Supersonic can build $O(\mu\log(n))$-size proofs which are verifiable in $O(\mu\log(n))$ time. The size can be less than 10% of the STARK's proof size. However, Block et al. identify a gap in the soundness proof of DARK and modify DARK to overcome the gap [34]. Recently, Bünz and Fisch derive a tight upper bound on the Schwartz-Zippel lemma to fill the gap [35].

Compressing the secret can also reduce the proof size of zk-SNARK [18]. Revealing the witness of a compressed secret is sufficient to convince the verifier about the validity of the original secret [36]. Bootle et al. leverage this idea to compress an n dimensional vector to $n/2$ dimensions and recursively reduce to 1 size with $2\log(n)$ additional transmissions. Furthermore, they adopt the compression technique to build inner-product arguments for arithmetic circuits satisfiability with $6\log(n)$ size [17]. Motivated by Bootle's compression idea, Bulletproofs protocol optimizes Bootle's inner-product arguments to build range proofs with $2\log(n)$ size [18]. Though the verification time of Bulletproofs ($O(n)$) is longer than Supersonic, its proof size is only 10% of Supersonic's proof size.

The compression technique can efficiently generate small-size proofs without the trusted setup. In this paper, we also discuss new compression techniques to generate proofs which has much smaller size than both Bootle's proofs and Bulletproofs.

B. Pedersen Commitment

Definition 1. ***Commitment [18].*** *A non-interactive commitment scheme is a pair of probabilistic polynomial time algorithms,* $(\mathrm{CGen}, \mathrm{Com})$. CGen *is a setup algorithm which generates a commitment key* $\mathrm{ck} \leftarrow \mathrm{CGen}(1^\lambda)$ *with a secure parameter* λ. $\mathrm{Com}_{\mathrm{ck}}$ *is a commitment algorithm which maps from the message space* M_{ck} *and randomness space* R_{ck} *to the commitment space* C_{ck}, $\mathsf{M}_{\mathrm{ck}} \times \mathsf{R}_{\mathrm{ck}} \rightarrow \mathsf{C}_{\mathrm{ck}}$ *(*M_{ck}, R_{ck}, *and* C_{ck} *are defined by* ck*). For a message* $m \in \mathsf{M}_{\mathrm{ck}}$, *we can uniformly pick a random* $r \in \mathsf{R}_{\mathrm{ck}}$ *and compute the commitment* $c = \mathrm{Com}_{\mathrm{ck}}(m, r) \in \mathsf{C}_{\mathrm{ck}}$.

For simplicity, we use Com *to represent* $\mathrm{Com}_{\mathrm{ck}}$.

Definition 2. ***Hiding Commitment [18].*** *A non-interactive commitment scheme* $(\mathrm{CGen}, \mathrm{Com})$ *is a hiding commitment if it reveals no information about the committed message. For all non-uniform polynomial time stateful interactive adversaries* $\mathcal{A}$, *there exists a negligible function* $\mu(\lambda)$ *such that*

$$\left| P\left[\mathcal{A}(c) = b \middle| \begin{array}{l} \mathrm{ck} \leftarrow \mathrm{CGen}(1^\lambda); r \leftarrow \mathsf{R}_{\mathrm{ck}}; \\ (m_0, m_1) \in \mathsf{M}_{\mathrm{ck}}^2 \leftarrow \mathcal{A}(\mathrm{ck}); \\ b \leftarrow \{0,1\}; c \leftarrow \mathrm{Com}(m_b, r) \end{array} \right] - \frac{1}{2} \right| \leqslant \mu(\lambda).$$

The scheme is ***perfectly hiding*** *when* $\mu(\lambda) = 0$.

Definition 3. ***Binding Commitment [18].*** *A non-interactive commitment scheme* $(\mathrm{CGen}, \mathrm{Com})$ *is a binding commitment if a commitment can only be opened to one message. For all non-uniform polynomial time stateful interactive adversaries* $\mathcal{A}$, *there exists a negligible function* $\mu(\lambda)$ *such that*

$$P\left[\begin{array}{l} \mathrm{Com}(m_0, r_0) = \mathrm{Com}(m_1, r_1) \\ \quad \wedge \quad m_0 \neq m_1 \end{array} \middle| \begin{array}{l} \mathrm{ck} \leftarrow \mathrm{CGen}(1^\lambda); \\ (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(\mathrm{ck}) \end{array} \right] \leqslant \mu(\lambda).$$

The scheme is ***perfectly binding*** *when* $\mu(\lambda) = 0$.

Definition 4. ***Pedersen Commitment [18].*** *A Pedersen commitment ensures the security based on discrete logarithm assumptions.* $\mathsf{M}_{\mathrm{ck}}, \mathsf{R}_{\mathrm{ck}} = \mathbb{Z}_p, \mathsf{C}_{\mathrm{ck}} = \mathbb{G}$ *of order* p.

$$\mathrm{CGen} : g, h \leftarrow \mathbb{G}$$
$$\mathrm{Com}(m, r) = g^m h^r.$$

Definition 5. ***Pedersen Vector Commitment [18].*** *A Pedersen vector commitment also ensures the security based on discrete logarithm assumptions.* $\mathsf{M}_{\mathrm{ck}} = \mathbb{Z}_p^n, \mathsf{R}_{\mathrm{ck}} = \mathbb{Z}_p, \mathsf{C}_{\mathrm{ck}} = \mathbb{G}$ *of order* p.

$$\mathrm{CGen} : \mathbf{g} = [g_1, \cdots, g_n] \leftarrow \mathbb{G}^n, h \leftarrow \mathbb{G}$$
$$\mathrm{Com}(\mathbf{m}, r) = h^r \prod_{i=1}^{n} g_i^{m_i} = \mathbf{g}^{\mathbf{m}} h^r.$$

The Pedersen vector commitment is perfectly hiding and computational binding under the discrete logarithm assumption. Specifically, the commitment scheme is binding under the assumption that a prover cannot find a non-zero vector $(r, m_1, \cdots, m_n)$ such that $h^r \prod_{i=1}^{n} g_i^{m_i} = 1$. The $(r, m_1, \cdots, m_n)$ is also known as a non-trivial discrete logarithm relation for $(h, g_1, \cdots, g_n)$. In some cases that hiding is not required, we can ensure binding by setting $r = 0$.

C. Zero-Knowledge Argument of Knowledge

Let R be a relationship that defines a language in NP. w is called a witness for a statement u if $(u, w) \in R$.

We consider a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, both of which are probabilistic polynomial time interactive algorithms. When $\mathcal{P}$ and $\mathcal{V}$ are interacting on inputs s and t, the transcript produced by them is denoted by $tr \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$. We write $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b$ depending on whether the verifier rejects ($b = 0$), or accepts ($b = 1$).

Definition 6. ***Argument of Knowledge [18].*** *$(\mathcal{P}, \mathcal{V})$ is an argument of knowledge for the relationship R if perfect completeness and computational witness-extended emulation (defined as follows) hold.*

Definition 7. ***Perfect Completeness [18].*** *$(\mathcal{P}, \mathcal{V})$ has perfect completeness if for all non-uniform polynomial time stateful interactive adversaries $\mathcal{A}$:*

$$P\left[\begin{array}{c|c} \langle \mathcal{P}(u,w), \mathcal{V}(u) \rangle = 1 \\ \vee \quad (u,w) \neq R \end{array} \middle| \; (u,w) \leftarrow \mathcal{A}(1^\lambda) \right] = 1.$$

Definition 8. ***Computational Witness-Extended Emulation [18].*** *$(\mathcal{P}, \mathcal{V})$ is computational witness-extended emulation if for all deterministic polynomial time $\mathcal{P}^*$ there exists an expected polynomial time emulator $\mathcal{E}$ such that for non-uniform polynomial time stateful interactive adversaries $\mathcal{A}$ there exists a negligible function $\mu(\lambda)$ such that:*

$$\left| \begin{array}{l} P\left[\mathcal{A}(tr) = 1 \middle| (u,s) \leftarrow \mathcal{A}(1^\lambda); tr \leftarrow \langle \mathcal{P}^*(u,s), \mathcal{V} \rangle\right] - \\ P\left[\begin{array}{c|c} \mathcal{A}(tr) = 1 \quad \wedge & (u,s) \leftarrow \mathcal{A}(1^\lambda) \\ (tr \text{ is accepting} \Rightarrow (u,w) \in R) & (tr,w) \leftarrow \mathcal{E}^{\mathcal{O}}(u) \end{array}\right] \end{array} \right| \leqslant \mu(\lambda),$$

where $\mathcal{O} = \langle \mathcal{P}^*(u,s), \mathcal{V}(u) \rangle$. The oracle called by $\mathcal{E}^{\mathcal{O}}$ can rewind to a specific point and resume with fresh randomness for the verifier from this point onwards.

Definition 9. ***Public Coin [18].*** *$(\mathcal{P}, \mathcal{V})$ is called public coin if all messages sent from the verifier to the prover are directly and uniformly chosen at random, and are independently of the prover's message, i.e., the challenges correspond to the verifier's randomness ρ.*

An argument is zero-knowledge if it does not leak information about w except what can be inferred from the truth of the statement. We will present arguments that have special honest-verifier zero-knowledge, which means that given the verifier's challenge values in advance, it is possible to efficiently simulate the entire argument without knowing the witness.

Definition 10. ***Perfect Special Honest-Verifier Zero-Knowledge [18].*** *A public coin argument $(\mathcal{P}, \mathcal{V})$ is perfect special honest-verifier zero-knowledge (SHVZK) for R if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all interactive non-uniform polynomial time adversaries $\mathcal{A}$*

$$\begin{aligned} & P\left[(u,w) \in R \wedge \mathcal{A}(tr) = 1 \middle| \begin{array}{l} (u,w,\rho) \leftarrow \mathcal{A}(1^\lambda); \\ tr \leftarrow \langle \mathcal{P}(u,w), \mathcal{V}(u;\rho) \rangle \end{array}\right] \\ = & P\left[(u,w) \in R \wedge \mathcal{A}(tr) = 1 \middle| \begin{array}{l} (u,w,\rho) \leftarrow \mathcal{A}(1^\lambda); \\ tr \leftarrow \mathcal{S}(u,\rho) \end{array}\right], \end{aligned}$$

where ρ is the public coin randomness used by the verifier.

Lemma 1. ***Forking Lemma [18].*** *Let $(\mathcal{P}, \mathcal{V})$ be a public coin interactive protocol with $(2k+1)$ moves. Let $(n_1, \cdots, n_k)$-tree be an extraction tree of accepting transcripts that can be efficiently built by distinct challenges and $\mathcal{E}$ be a witness extraction algorithm that always succeeds in extracting a witness from an $(n_1, \cdots, n_k)$-tree in probability polynomial time. Suppose $\prod_{i=1}^{k} n_i$ is bounded by a polynomial in the security parameter λ. Then $(\mathcal{P}, \mathcal{V})$ has witness-extended emulation.*

The proof of Lemma 1 can be referred to [17], [18].

D. Notations

We use $\mathbb{G}$ to denote a cyclic group with p order[3] (for an elliptic curve $\mathscr{E}$, $p = |\mathscr{E}|$), and $\mathbb{Z}_p$ to the ring of integers modulo p. We use $\mathbb{C}$ to represent a challenge space. Accordingly, the size of the challenge space is $|\mathbb{C}|$. $\mathbb{G}^n$ be the n-dimension vector space over $\mathbb{G}$ (similar for $\mathbb{Z}_p^n$). Let $g, h, u \in \mathbb{G}$ denotes generators of $\mathbb{G}$. Commitments (which are group elements) are capitalized and blinding factors are denoted by Greek letters, i.e., $C = g^a h^\alpha$ is a commitment to a. We use $x \leftarrow \mathbb{Z}_p$ to represent the uniform sampling of an element from $\mathbb{Z}_p$. For $m, n \in \mathbb{Z}$, $gcd(m, n)$ denotes the greatest common divisor of m and n.

Vector notations are defined as follows. We use bold font to represent vectors. For instance, $\mathbf{g} \in \mathbb{G}^n$ denotes a vector with group elements $g_0, g_1, \cdots, g_{n-1}$, where $g_i \in \mathbb{G}$. Suppose $k \in \mathbb{Z}_p$, we denote $\mathbf{k}^n$ as $\mathbf{k}^n = [1, k, k^2, \cdots, k^{n-1}]$. Furthermore, for $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$ and $\mathbf{g} \in \mathbb{G}^n$, we denote $\prod_{i=0}^{n-1} g_i^{a_i}$ as $\mathbf{g}^{\mathbf{a}}$ and the Hadamard product of $\mathbf{a}$ and $\mathbf{b}$ as $\mathbf{a} \circ \mathbf{b} = [a_0 \cdot b_0, a_1 \cdot b_1, \cdots, a_{n-1} \cdot b_{n-1}]$.

We write a vector polynomial $p(X) \in \mathbb{Z}_p^n[X]$ as $p(X) = \sum_{i=0}^{d} \mathbf{p}_i \cdot X^i$, where each $\mathbf{p}_i$ is a vector in $\mathbb{Z}_p^n$ as a coefficient. The inner product of two vector polynomials $l(X)$ and $r(X)$ is defined as follows:

$$\langle l(X), r(X) \rangle = \sum_{i=0}^{d} \sum_{j=0}^{i} \langle \mathbf{l}_i, \mathbf{r}_j \rangle \cdot X^{i+j} \in \mathbb{Z}_p[X]. \quad (1)$$

Clearly, based on this definition, we can prove that evaluating the polynomials at x and then taking the inner product is same as evaluating a new inner-product polynomial at x. Therefore, there exists an inner-product polynomial such that $t(X) = \langle l(X), r(X) \rangle$.

Finally, we use "$\{$(Public Input; Witness) : Relation$\}$" format to denote the "Relation" of the "Public Input" and "Witness".

III. Motivation and Key Ideas

A. Bulletproofs Analysis

Bünz et al. introduce Bulletproos [18] with a logarithmic-size range proof to show a secret v is in $[0, 2^n - 1]$. Specifically, it presents a new inner-product argument and writes the range proof relation in an inner-product form for the inner-product argument. We first describe how the new inner-product argument works.

[3] p is not necessarily be a large prime to ensure the hardness of discrete logarithm problems. Based on Pohlig-Hellman algorithm, we need at least one factor of p (e.g. q_0) is a large prime, which ensures a same security with curves of q_0 order.

The inner-product argument literately compresses two vectors in an inner-product relation into two scalars. Consider two n-size vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$. For $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$ and $u \in \mathbb{G}$, the commitment of $\mathbf{a}$ and $\mathbf{b}$ and their inner-product is $A = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{\langle \mathbf{a},\mathbf{b}\rangle}$. The prover will first make a "two-set splitting" to split an n-size vector into two $(n/2)$-size vectors, i.e., $\mathbf{a} = [\mathbf{a}_1, \mathbf{a}_2]$, $\mathbf{b} = [\mathbf{b}_1, \mathbf{b}_2]$, $\mathbf{g} = [\mathbf{g}_1, \mathbf{g}_2]$, and $\mathbf{h} = [\mathbf{h}_1, \mathbf{h}_2]$, where $\mathbf{a}_1 = [a_0, \cdots, a_{n/2-1}]$ and $\mathbf{a}_2 = [a_{n/2}, \cdots, a_{n-1}]$, etc. Then, based on a compressing scalar (i.e., a random challenge) $x \in \mathbb{Z}_p$ provided by the verifier, the prover compresses $\mathbf{a}$, $\mathbf{b}$, $\mathbf{g}$, and $\mathbf{h}$ to $\widehat{\mathbf{a}}$, $\widehat{\mathbf{b}}$, $\widehat{\mathbf{g}}$, and $\widehat{\mathbf{h}}$ as follows:

$$\begin{aligned} \widehat{\mathbf{a}} &= x\mathbf{a}_1 + x^{-1}\mathbf{a}_2, \qquad & \widehat{\mathbf{g}} &= \mathbf{g}_1^{x^{-1}} \circ \mathbf{g}_2^{x}, \\ \widehat{\mathbf{b}} &= x^{-1}\mathbf{b}_1 + x\mathbf{b}_2, \qquad & \widehat{\mathbf{h}} &= \mathbf{h}_1^{x} \circ \mathbf{h}_2^{x^{-1}}. \end{aligned} \tag{2}$$

Since the new commitment $\widehat{A}$ becomes

$$\begin{aligned} \widehat{A} &= \widehat{\mathbf{g}}^{\widehat{\mathbf{a}}} \cdot \widehat{\mathbf{h}}^{\widehat{\mathbf{b}}} \cdot u^{\langle \widehat{\mathbf{a}}, \widehat{\mathbf{b}}\rangle} \\ &= \mathbf{g}_1^{\mathbf{a}_1}\mathbf{g}_2^{\mathbf{a}_2} \cdot \left(\mathbf{g}_2^{\mathbf{a}_1}\right)^{x^2} \cdot \left(\mathbf{g}_1^{\mathbf{a}_2}\right)^{x^{-2}} \\ &\quad \cdot \mathbf{h}_1^{\mathbf{b}_1}\mathbf{h}_2^{\mathbf{b}_2} \cdot \left(\mathbf{h}_1^{\mathbf{b}_2}\right)^{x^2} \cdot \left(\mathbf{h}_2^{\mathbf{b}_1}\right)^{x^{-2}} \\ &\quad \cdot u^{\langle \mathbf{a}_1,\mathbf{b}_1\rangle + \langle \mathbf{a}_2,\mathbf{b}_2\rangle} \cdot \left(u^{\langle \mathbf{a}_1,\mathbf{b}_2\rangle}\right)^{x^2} \cdot \left(u^{\langle \mathbf{a}_2,\mathbf{b}_1\rangle}\right)^{x^{-2}} \\ &= A \cdot L^{x^2} \cdot R^{x^{-2}}, \end{aligned}$$

where $L = \mathbf{g}_2^{\mathbf{a}_1}\mathbf{h}_1^{\mathbf{b}_2}u^{\langle \mathbf{a}_1,\mathbf{b}_2\rangle}$ and $R = \mathbf{g}_1^{\mathbf{a}_2}\mathbf{h}_2^{\mathbf{b}_1}u^{\langle \mathbf{a}_2,\mathbf{b}_1\rangle}$, the prover must also publish L and R before receiving x. By iterating the compression process from "$\mathbf{a}, \mathbf{b}$ to $\widehat{\mathbf{a}}, \widehat{\mathbf{b}}$", the prover can reduce the n-size vectors $\mathbf{a}$ and $\mathbf{b}$ to two scalars a and b. Therefore, instead of sending two n-size vectors, the prover only needs to send $2\log(n)$ group elements (L and R in each round of iteration), which can significantly reduce the proof size.

Note that the inner-product argument dose not ensure hiding. In the range proof of Bulletproofs, it needs to mask the secret and convert into an inner-product form before applying the inner-product argument. Specifically, for a range proof of $v \in [0, 2^n - 1]$, the prover can express the proof as having a secret vector $\mathbf{a}_L = [a_0, \cdots, a_{n-1}]$ such that: (1) $\mathbf{a}_L$ is the binary encoding of v, i.e., $v = \sum_{i=0}^{n-1} 2^i a_i$; and (2) each element of $\mathbf{a}_L$ (a_i) is either 0 or 1. Setting $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n$, we have the following relations:

$$\begin{aligned} \mathbf{a}_L \circ \mathbf{a}_R &= \mathbf{0}^n, \\ \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R &= \mathbf{0}^n, \\ \langle \mathbf{a}_L, \mathbf{2}^n\rangle &= v. \end{aligned} \tag{3}$$

Taking two challenges $y, z \in \mathbb{Z}_p$, the prover can prove that Equation (3) hold by proving that

$$\begin{aligned} &z^2 \cdot \langle \mathbf{a}_L, \mathbf{2}^n\rangle + z \cdot \langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R, \mathbf{y}^n\rangle + \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n\rangle \\ =&z^2 \cdot v, \end{aligned} \tag{4}$$

which can be further converted to an inner-product form of $\mathbf{a}_L$ and $\mathbf{a}_R$ (actually $(\mathbf{a}_L - z \cdot \mathbf{1}^n + \mathbf{s}_L \cdot x)$ and $(\mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot x) + z^2 \cdot \mathbf{2}^n)$ with another challenge x and some masking vectors $\mathbf{s}_L$ and $\mathbf{s}_R$) [18]. Therefore, the prover needs to send two n-size vectors $\mathbf{l} = \mathbf{a}_L - z \cdot \mathbf{1}^n + \mathbf{s}_L \cdot x$ and $\mathbf{r} = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot x) + z^2 \cdot \mathbf{2}^n$ that blindly represent $\mathbf{a}_L$ and $\mathbf{a}_R$.

Finally, the prover can adopt the inner-product argument to compress $\mathbf{l}$ and $\mathbf{r}$ to two scalars l and r. The proof includes $(2\lceil \log_2(n)\rceil + 4)$ group elements, 5 $\mathbb{Z}_p$ elements, and $(4 + \lceil \log_2(n)\rceil)$ challenges.

We have three observations in Bulletproofs. First, in the inner-product argument, x and x^{-1} in Equation (2) reduce the challenge space. Writing $\widehat{\mathbf{a}} = x^{-1} \cdot (x^2\mathbf{a}_1 + \mathbf{a}_2)$, the value of $x^2 \pmod p$ maps to a much smaller space than the space of x. Specifically, suppose $\mathbb{C} = \{x\}$ and $\mathbb{C}' = \{x^2 | x \in \mathbb{C}\}$, we have $|\mathbb{C}'| \leqslant 0.5 \times |\mathbb{C}|$. Thus, the prover will have a higher chance to correctly guess x^2 and pass the verification without knowing the secret. However, this is not a serious security problem as the $|\mathbb{C}'|$ is still super-poly. Even the attacker's winning advantage is increased, it is still negligible.

The second observation is the proof size of Bulletproofs is mainly contributed by the L's and R's in the inner-product argument. If we can send one group element X instead of L and R in each iteration, the communication complexity is reduced by half of the Bulletproofs. However, this is a challenging problem as the exponents of L and R are different. It is important to find a practical approach while maintaining the security of the protocol.

Finally, Bulletproofs protocol uses *two* challenges, y and z, to mask the equations in Equation (3). If we regard them as a $(2n+1)$-size vector,

$$[\mathbf{a}_L \circ \mathbf{a}_R, \quad \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R, \quad \langle \mathbf{a}_L, \mathbf{2}^n\rangle] = [\mathbf{0}^{2n}, v],$$

the prover only needs *one* challenge y from the verifier to build the equation:

$$\begin{aligned} &y^{2n} \cdot \langle \mathbf{a}_L, \mathbf{2}^n\rangle + y^n \cdot \langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R, \mathbf{y}^n\rangle + \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n\rangle \\ =&y^{2n} \cdot v. \end{aligned} \tag{5}$$

This is based on the fact that in a Σ-protocol for an n-size vector $(b_0, \cdots, b_{n-1})$, using $n-1$ challenges, $(x_1, \cdots, x_{n-1})$, to build the response as $f = b_0 + \sum_{i=1}^{n-1} x_i b_i$ is equivalent to build $f = \sum_{i=0}^{n-1} x^i b_i$ with only one challenge.

B. New Techniques

Reducing additional commitments. Recall the improved inner-product argument of Bulletproofs, the prover must separately publish L and R parts in each iterated step as they share different exponents. As L and R only contribute to deriving the new commitment for the verifier, we may reduce the communication cost if we find new approaches to derive the new commitment with fewer parameters. For instance, if we can find an approach to batch L and R parts with $X = L \cdot R$, the prover only needs to send one group element, X, in each step. Therefore, the communication complexity is reduced by half of Bulletproofs.

Consider the common input of an argument of knowledge of one vectors, $(\mathbb{G}, p, \mathbf{g}, A)$, where $\mathbf{g}$ can be split into two $(n/2)$-size vectors $\mathbf{g}_1$ and $\mathbf{g}_2$. We try to prove knowledge of a vector $\mathbf{a} = [\mathbf{a}_1, \mathbf{a}_2]$ such that

$$A = \mathbf{g}^{\mathbf{a}} = \mathbf{g}_1^{\mathbf{a}_1} \cdot \mathbf{g_2}^{\mathbf{a}_2}.$$

We compress $\mathbf{a}$ and $\mathbf{g}$ with the challenge x provided by the verifier: $\widehat{\mathbf{a}} = \mathbf{a}_1 + x\mathbf{a}_2$ and $\widehat{\mathbf{g}} = \mathbf{g}_1 \circ \mathbf{g}_2^{x^{-1}}$. Therefore, the new

commitment $\widehat{A}$ becomes

$$\widehat{A} = \widehat{\mathbf{g}}^{\widehat{\mathbf{a}}} = \mathbf{g}_1^{\mathbf{a}_1} \cdot \mathbf{g}_2^{\mathbf{a}_2} \cdot (\mathbf{g}_1^{\mathbf{a}_2})^x \cdot (\mathbf{g}_2^{\mathbf{a}_1})^{x^{-1}}.$$

If we want to send one group element which contains both $\mathbf{g}_1^{\mathbf{a}_2}$ and $\mathbf{g}_2^{\mathbf{a}_1}$, the exponents of the two parts must be equal. Note that these calculations are conducted on a cyclic group $\mathbb{G}$ with order of p. Therefore, the challenge x should satisfy a quadratic residue

$$x^2 = 1 \mod p. \tag{6}$$

Hence, when the verifier challenges with an x that satisfies the quadratic residue in Equation (6)[4], the prover only needs to send one group element $X = \mathbf{g}_1^{\mathbf{a}_2}\mathbf{g}_2^{\mathbf{a}_1}$ before the challenge. Both the prover and verifier can compute the new commitment $\widehat{A}$ to $\widehat{\mathbf{a}}$:

$$\widehat{A} = \mathbf{g}_1^{\mathbf{a}_1} \cdot \mathbf{g}_2^{\mathbf{a}_2} \cdot (\mathbf{g}_1^{\mathbf{a}_2})^x \cdot (\mathbf{g}_2^{\mathbf{a}_1})^{x^{-1}} = \mathbf{g}_1^{\mathbf{a}_1} \cdot \mathbf{g}_2^{\mathbf{a}_2} \cdot (\mathbf{g}_1^{\mathbf{a}_2} \cdot \mathbf{g}_2^{\mathbf{a}_1})^x = A \cdot X^x.$$

Since x must satisfy Equation (6), the size of challenge space can be reduced accordingly (smaller than p)[5]. We discuss how to derive the challenge space and its size, and build a super-poly size space in Appendix A. We also provide approaches to generate appropriate curves and give a specific curve in Section VII-B. When the challenge space size is *not* be super-poly, the protocol is not secure. An attacking strategy against a small challenge space size is presented in Appendix A. Nevertheless, we still consider this approach can be practical with new secure elliptic curves. We also hope our solution could provide some insights into other settings such as lattice, where q does not need to be prohibitively large.

Fewer challenges. Besides vector compression, we also propose new approaches to reduce the challenges in a range proof protocol. Based on our previous observation, the prover only needs *one* challenge y from the verifier to build the Equation (5), which can be further converted to an inner-product form of $\mathbf{l} = \mathbf{a}_L - y^n \cdot \mathbf{1}^n + \mathbf{s}_L \cdot x$ and $\mathbf{r} = \mathbf{y}^n \circ (\mathbf{a}_R + y^n \cdot \mathbf{1}^n + \mathbf{s}_R \cdot x) + y^{2n} \cdot \mathbf{2}^n$ with another challenge x and some masking vectors $\mathbf{s}_L$ and $\mathbf{s}_R$. The prover can further run the inner-product argument to reduce the proof size to logarithmic.

For interactive proof protocols, fewer challenges can reduce the computational cost of the verifier (generating fewer secure random numbers) and the communication cost. In non-interactive scenarios, the prover and verifier can invoke fewer hash functions under Fiat-Shamir heuristic [37], which will save computational power.

IV. New Vector Compression Protocols

Both [17] and [18] adopt compression techniques to reduce the communication cost when dealing with vectors. These techniques can be used in inner-product arguments to build communication efficient zero-knowledge proofs for range proofs or arithmetic circuit satisfiability. When dealing with n-size vectors, the communication complexity of an inner-product argument in [17] and [18] is $6\log(n)$ and $2\log(n)$ respectively. Based on the idea of reducing additional commitments in Section III-B, we present a new compression technique to reduce the communication complexity to $\log(n)$, which is sound but *not* zero-knowledge. Furthermore, for inner-product argument, the communication complexity of our approach is $\log(n)$, which is only half of the Bulletproofs size [18] and $1/6$ of Bootle's proof size [17].

A. Single Vector Argument

We formally describe the argument of knowledge of an n-size vector $\mathbf{a}$ between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$.

Common input: $(\mathbb{G}, \mathbb{C}, p, \mathbf{g}, A)$ such that $\mathbb{C} \subset \mathbb{Z}_p^*$, $\mathbf{g} \in \mathbb{G}^n$, and $A \in \mathbb{G}$.

Prover's witness: $\mathbf{a}$ that satisfies $\mathbf{g}^{\mathbf{a}} = A$.

Argument if $n = 1$:

$\mathcal{P} \to \mathcal{V} : a$.

$\mathcal{V} \to \mathcal{P}$: ACCEPT if $A = g^a$, otherwise REJECT.

Reduction if $n \neq 1$:

$\mathcal{P}$ computes:

$$\mathbf{a}_1 = [a_0, a_1, \cdots, a_{n/2-1}], \mathbf{a}_2 = [a_{n/2}, a_{n/2+1}, \cdots, a_{n-1}],$$
$$\mathbf{g}_1 = [g_0, g_1, \cdots, g_{n/2-1}], \mathbf{g}_2 = [g_{n/2}, g_{n/2+1}, \cdots, g_{n-1}],$$
$$X = \mathbf{g}_1^{\mathbf{a}_2}\mathbf{g}_2^{\mathbf{a}_1}$$

$\mathcal{P} \to \mathcal{V} : X$.

$\mathcal{V} : x \leftarrow \mathbb{C}$.

$\mathcal{V} \to \mathcal{P} : x$.

$\mathcal{P}$ and $\mathcal{V}$ compute:

$$\widehat{\mathbf{g}} = \mathbf{g}_1 \circ \mathbf{g}_2^{x^{-1}} \in \mathbb{G}^{n/2}, \quad \widehat{A} = A \cdot X^x \in \mathbb{G}.$$

$\mathcal{P}$ computes:

$$\widehat{\mathbf{a}} = \mathbf{a}_1 + x\mathbf{a}_2 \in \mathbb{Z}_p^{n/2}.$$

$\mathcal{P}$ and $\mathcal{V}$ recursively compute a reduced statement from $(\mathbb{G}, \mathbb{C}, p, \widehat{\mathbf{g}}, \widehat{A})$, where $\mathcal{P}$'s witness is $\widehat{\mathbf{a}}$.

Since the prover only transmits one X in each iteration, the communication complex is only $\log(n)$.

Theorem 2. ***Single Vector Argument.*** *When $|\mathbb{C}|$ is super-poly, the above protocol of argument of knowledge of one vector has perfect completeness and computational witness-extended emulation for either extracting a non-trivial discrete logarithm relation in $\mathbf{g}$ or extracting a valid witness $\mathbf{a}$.*

The proof of Theorem 2 are presented in Appendix B.

B. Inner-Product Argument

We extend the compression technique to the inner-product argument of knowledge of two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$. The inner-product relation can be expressed as follows:

$$\{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, P \in \mathbb{G}; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{\langle \mathbf{a}, \mathbf{b} \rangle}\}. \tag{7}$$

Common input: $(\mathbb{G}, \mathbb{C}, p, \mathbf{g}, \mathbf{h}, u, P)$ such that $\mathbb{C} \subset \mathbb{Z}_p^*$, $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$, $u, P \in \mathbb{G}$.

[4]In fact, the requirement can be $x^2 = k \mod p$, where $k \in \mathbb{Z}_p^*$. X will become $(\mathbf{g}_1^{\mathbf{a}_2})^k \mathbf{g}_2^{\mathbf{a}_1}$.

[5]To ensure a large challenge space size, we build $p = q_0 q_1^{e_1} \cdots q_k^{e_k}$, which q_0 is a large prime and others are small primes. Details are presented in Appendix A.

Prover's witness: $\mathbf{a}$ and $\mathbf{b}$ that satisfy:

$$\mathbf{g}^{\mathbf{a}} \cdot \mathbf{h}^{\mathbf{b}} \cdot u^{\langle \mathbf{a}, \mathbf{b} \rangle} = P.$$

Argument if $n = 1$**:**

$\mathcal{P} \to \mathcal{V} : a, b.$

$\mathcal{V} \to \mathcal{P}$: ACCEPT if $P = g^a h^b u^{ab}$, otherwise REJECT.

Reduction if $n \neq 1$**:**

$\mathcal{P}$ computes:

$$\mathbf{a}_1 = [a_0, a_1, \cdots, a_{n/2-1}], \mathbf{a}_2 = [a_{n/2}, a_{n/2+1}, \cdots, a_{n-1}],$$
$$\mathbf{b}_1 = [b_0, b_1, \cdots, b_{n/2-1}], \mathbf{b}_2 = [b_{n/2}, b_{n/2+1}, \cdots, b_{n-1}],$$
$$\mathbf{g}_1 = [g_0, g_1, \cdots, g_{n/2-1}], \mathbf{g}_2 = [g_{n/2}, g_{n/2+1}, \cdots, g_{n-1}],$$
$$\mathbf{h}_1 = [h_0, h_1, \cdots, h_{n/2-1}], \mathbf{h}_2 = [h_{n/2}, h_{n/2+1}, \cdots, h_{n-1}],$$
$$z_X = \langle \mathbf{a}_1, \mathbf{b}_2 \rangle + \langle \mathbf{a}_2, \mathbf{b}_1 \rangle, \quad X = \mathbf{g}_1^{\mathbf{a}_2} \mathbf{g}_2^{\mathbf{a}_1} \mathbf{h}_1^{\mathbf{b}_2} \mathbf{h}_2^{\mathbf{b}_1} u^{z_X}.$$

$\mathcal{P} \to \mathcal{V} : X.$

$\mathcal{V} : x \leftarrow \mathbb{C}.$

$\mathcal{V} \to \mathcal{P} : x.$

$\mathcal{P}$ and $\mathcal{V}$ compute:

$$\widehat{\mathbf{g}} = \mathbf{g}_1 \circ \mathbf{g}_2^{x^{-1}} \in \mathbb{G}^{n/2}, \quad \widehat{\mathbf{h}} = \mathbf{h}_1 \circ \mathbf{h}_2^{x} \in \mathbb{G}^{n/2}$$
$$\widehat{P} = P \cdot X^x \in \mathbb{G}.$$

$\mathcal{P}$ computes:

$$\widehat{\mathbf{a}} = \mathbf{a}_1 + x\mathbf{a}_2 \in \mathbb{Z}_p^{n/2}, \quad \widehat{\mathbf{b}} = \mathbf{b}_1 + x^{-1}\mathbf{b}_2 \in \mathbb{Z}_p^{n/2}. \tag{8}$$

$\mathcal{P}$ and $\mathcal{V}$ recursively compute a reduced statement from $(\mathbb{G}, \mathbb{C}, p, \widehat{\mathbf{g}}, \widehat{\mathbf{h}}, u, \widehat{P})$, where $\mathcal{P}$'s witness is $\widehat{\mathbf{a}}$ and $\widehat{\mathbf{b}}$.

Similar to the single vector argument scenario, the computation complexity is $\log(n)$, which is much less than the size of Bulletproofs [18] and Bootle's proof [17].

Theorem 3. ***Inner-Product Argument.*** *When the challenge space size is super-poly, the above protocol of inner-product argument of knowledge of two vector has perfect completeness and computational witness-extended emulation for either extracting a non-trivial discrete logarithm relation between* $\mathbf{g}, \mathbf{h}, u$ *or extracting a valid witness* $\mathbf{a}, \mathbf{b}$*.*

The proof of Theorem 3 is given in Appendix C.

V. New Range Proof Protocol

We propose a zero-knowledge protocol to use the inner-product argument for range proofs, which reduces the number of challenges *without* sacrificing challenge space size (i.e., independent from the vector compression technique).

The range proof can be constructed with a Pedersen commitment $V \in \mathbb{G}$ that is used for the inner-product argument. Specifically, let $V \in \mathbb{G}$ be the Pedersen commitment on $v \in \mathbb{Z}_p$ with randomness γ. Formally speaking, the range proof relation can be expressed as:

$$\{(V, g, u \in \mathbb{G}; v, \gamma \in \mathbb{Z}_p) : V = g^v u^\gamma \wedge v \in [0, 2^n - 1]\}. \tag{9}$$

A. Inner-Product Range Proof

Recall the "fewer challenges" idea in Section III-B. We use an n-size vector $\mathbf{a}_L$ to encode the secret v, i.e., $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$, and another vector $\mathbf{a}_R$ which $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n$. Based on a challenge y from the verifier, the prover can use Equation (5) to prove Equation (9). We further re-write Equation (5) into an inner-product form:

$$\begin{aligned} & \left\langle \mathbf{a}_L - y^n \cdot \mathbf{1}^n, \quad \mathbf{y}^n \circ (\mathbf{a}_R + y^n \cdot \mathbf{1}^n) + y^{2n} \cdot \mathbf{2}^n \right\rangle \\ = \quad & y^{2n} \cdot v + \delta(y), \end{aligned} \tag{10}$$

where $\delta(y) = (y^n - y^{2n}) \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle - y^{3n} \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle \in \mathbb{Z}_p$. Thus, we can leverage the idea of inner-product argument to compress each vector.

Notice that we also need two random vectors, $\mathbf{s}_L, \mathbf{s}_R \in \mathbb{Z}_p^n$, to hide $\mathbf{a}_L$ and $\mathbf{a}_R$. The zero knowledge protocol is:

$\mathcal{P}$ computes:

$\mathbf{a}_L \in \{0, 1\}^n$ s.t. $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$

$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n$

$\alpha, \rho \leftarrow \mathbb{Z}_p$

$A = u^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} \in \mathbb{G}$ *// commitment to* $\mathbf{a}_L, \mathbf{a}_R$

$\mathbf{s}_L, \mathbf{s}_R \leftarrow \mathbb{Z}_p^n$ *// blinding vector for* $\mathbf{a}_L, \mathbf{a}_R$

$S = u^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G}$ *// commitment to* $\mathbf{s}_L, \mathbf{s}_R$

$\mathcal{P} \to \mathcal{V} : A, S$

$\mathcal{V} : y \leftarrow \mathbb{Z}_p^*$ *// challenge space is* $\mathbb{Z}_p^*$ *instead of* $\mathbb{C}$

$\mathcal{V} \to \mathcal{P} : y$

To build the two vectors in an inner-product argument, the prover defines two linear vector polynomials $r(X), l(X) \in \mathbb{Z}_p^n[X]$ and one polynomial $t(X) \in \mathbb{Z}_p[X]$:

$$l(X) = (\mathbf{a}_L - y^n \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X,$$
$$r(X) = \mathbf{y}^n \circ (\mathbf{a}_R + y^n \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + y^{2n} \cdot \mathbf{2}^n,$$
$$t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2.$$

Since the constant terms of $t(X)$, t_0, is the result of Equation (10), the prover needs to prove t_0 satisfies:

$$t_0 = y^{2n} \cdot v + \delta(y).$$

Therefore, the prover needs to commit all other coefficients of $t(X)$, $t_1, t_2 \in \mathbb{Z}_p$, and engage in a polynomial identity test with the verifier to show $t(X) = \langle l(X), r(X) \rangle$ as follows:

$\mathcal{P}$ computes:

$\tau_1, \tau_2 \leftarrow \mathbb{Z}_p$ *// blinding* t_1 *and* t_2

$T_i = u^{\tau_i} g^{t_i} \in \mathbb{G}, i \in \{1, 2\}$ *// commitment to* t_1 *and* t_2

$\mathcal{P} \to \mathcal{V} : T_1, T_2$

$\mathcal{V} : x \leftarrow \mathbb{Z}_p^*$ *// challenge space is* $\mathbb{Z}_p^*$ *instead of* $\mathbb{C}$

$\mathcal{V} \to \mathcal{P} : x$

$\mathcal{P}$ computes:

$$\begin{aligned} & \mathbf{l} = l(x) = (\mathbf{a}_L - y^n \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot x \in \mathbb{Z}_p^n \\ & \mathbf{r} = r(x) = \mathbf{y}^n \circ (\mathbf{a}_R + y^n \cdot \mathbf{1}^n + \mathbf{s}_R \cdot x) + y^{2n} \cdot \mathbf{2}^n \in \mathbb{Z}_p^n \\ & \tilde{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p \\ & \tau_x = y^{2n} \cdot \gamma + \tau_1 \cdot x + \tau_2 \cdot x^2 \in \mathbb{Z}_p \\ & \mu = \alpha + \rho \cdot x \in \mathbb{Z}_p \end{aligned} \tag{11}$$

$\mathcal{P} \to \mathcal{V} : \mathbf{l}, \mathbf{r}, \tilde{t}, \tau_x, \mu$

Notice that we cannot directly apply the compression strategy of inner-product argument to $\mathbf{l}$ and $\mathbf{r}$. It is because $\mathbf{a}_R$ part in $\mathbf{r}$ contains the coefficient $\mathbf{y}^n$. When adopting

the inner-product argument directly, $\mathbf{y}^n$ will be "mixed into" $\widehat{\mathbf{r}} = \mathbf{r}_1 + z\mathbf{r}_2$ (z is a challenge in the inner-product argument). Since the commitment $A = u^{\alpha}\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R}$ does not contain $\mathbf{y}^n$, it is important to cancel out $\mathbf{y}^n$ in $\mathbf{r}$ to make the coefficient of $\mathbf{a}_R$ being 1 (or a constant). Therefore, we change the group elements $\mathbf{h}$ with the following transfer:

$$h'_i = h_i^{y^{-i}} \in \mathbb{Z}, i \in [0, n-1]. \quad // \quad \mathbf{h}' = [h_0, h_1^{y^{-1}}, h_{n-1}^{y^{1-n}}]$$

Then we have $\mathbf{h}^{\mathbf{y}^n \circ \mathbf{a}_R} = (\mathbf{h}')^{\mathbf{a}_R}$, which can be used to construct $\mathbf{g}^{\mathbf{l}}(\mathbf{h}')^{\mathbf{r}}u^{\mu}$ as $A \cdot S^x \cdot \mathbf{g}^{-\mathbf{y}^n} \cdot (\mathbf{h}')^{y^n \cdot \mathbf{y}^n + y^{2n} \cdot \mathbf{2}^n}$. Therefore, we build a commitment of $\mathbf{l}$ and $\mathbf{r}$, $P = \mathbf{g}^{\mathbf{l}}(\mathbf{h}')^{\mathbf{r}}u^{\mu}$, on group elements $\mathbf{g}$, $\mathbf{h}'$, and u.

The verifier further checks: (1) $\tilde{t} \stackrel{?}{=} t_0 + t_1 \cdot x + t_2 \cdot x^2$; (2) $\mathbf{l} \stackrel{?}{=} (\mathbf{a}_L - y^n \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot x$; (3) $\mathbf{r} \stackrel{?}{=} \mathbf{y}^n \circ (\mathbf{a}_R + y^n \cdot \mathbf{1}^n + \mathbf{s}_R \cdot x) + y^{2n} \cdot \mathbf{2}^n$; and (4) $\tilde{t} \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle$. Thus, the final steps of the range proof protocol are:

$\mathcal{V} \to \mathcal{P}$: ACCEPT if

$$g^{\tilde{t}}u^{\tau_x} = V^{y^{2n}} \cdot g^{\delta(y)} \cdot T_1^x \cdot T_2^{x^2}, \quad // \textit{ checking (1)}$$

and $$\mathbf{g}^{\mathbf{l}}(\mathbf{h}')^{\mathbf{r}}u^{\mu} = A \cdot S^x \cdot \mathbf{g}^{-\mathbf{y}^n} \cdot (\mathbf{h}')^{y^n \cdot \mathbf{y}^n + y^{2n} \cdot \mathbf{2}^n},$$
$$// \textit{ checking (2) and (3)}$$

and $$\tilde{t} = \langle \mathbf{l}, \mathbf{r} \rangle; \quad // \textit{ checking (4)}$$

otherwise REJECT.

In the above range proof protocol, the prover needs to transmit two n-size vectors $\mathbf{l}$ and $\mathbf{r}$, four group elements A, S, T_1 and T_2, and three $\mathbb{Z}_p$ elements $(\tilde{t}, \tau_x, \mu)$. The verifier needs to send two random challenges x and y. Please note that since the challenges are generated from $\mathbb{Z}_p^*$, this approach will *not* sacrifice challenge space size and is compatible with existing prime order elliptic curves.

Corollary 4. ***Range Proof.*** *The above protocol of inner-product range proof has perfect completeness, perfect honest verifier zero-knowledge and computational special soundness.*

Proof: The range proof is a special case of the aggregated range proof when $m = 1$ without compression. Therefore, it is a direct corollary of Theorem 6. ∎

VI. SymmeProof

A. Logarithmic Range Proof

We leverage the inner-product argument (Section IV-B) to improve the efficiency of range proof (Section V-A) by literately reducing $\mathbf{l}$ and $\mathbf{r}$ to a single scalar.

We briefly describe the compression protocol as follows:

Common input: $(\mathbb{G}, \mathbb{C}, \mathbf{g}, \mathbf{h}, g, u, V, A, S$ where $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$ and $g, u, V, A, S \in \mathbb{G}$.

Prover's witness: $\mathbf{s}_L$, $\mathbf{s}_R$, $\mathbf{a}_L$, $\mathbf{a}_R$ and v that satisfy:

$$\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v, \quad \mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n, \quad \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^n,$$
$$V = g^v u^{\gamma}, \quad A = u^{\alpha}\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R}, \quad S = u^{\rho}\mathbf{g}^{\mathbf{s}_L}\mathbf{h}^{\mathbf{s}_R}.$$

Inner-product range proof: (Section V-A)

$\mathcal{V}$ generates $y \in \mathbb{Z}_p^*$ and sends to $\mathcal{P}$.

$\mathcal{P}$ generates $\tau_1, \tau_2 \in \mathbb{Z}_p$, computes and sends $T_1 = u^{\tau_1}g^{t_1}$ and $T_2 = u^{\tau_2}g^{t_2}$ to $\mathcal{V}$.

$\mathcal{V}$ generates $x \in \mathbb{Z}_p^*$ and sends to $\mathcal{P}$.

$\mathcal{P}$ computes $\tau_x, \mu, \tilde{t}, \mathbf{l}, \mathbf{r}$ based on Equation (11) and sends $\tau_x, \mu, \tilde{t}$ to $\mathcal{V}$.

$\mathcal{P}$ and $\mathcal{V}$ compute:.

$$\mathbf{h}', \quad \mathbf{g}' = \mathbf{g}, \quad n = size(\mathbf{l}),$$
$$P = A \cdot S^x \cdot \mathbf{g}^{-\mathbf{y}^n} \cdot (\mathbf{h}')^{y^n \cdot \mathbf{y}^n + y^{2n} \cdot \mathbf{2}^n}.$$

$\mathcal{V}$ generates $z \in \mathbb{Z}_p^*$ and sends to $\mathcal{P}$.

$\mathcal{P}$ and $\mathcal{V}$ compute:.

$$P' = P \cdot u^{z \cdot \tilde{t}}.$$

Argument if $n = 1$**:**

$\mathcal{P} \to \mathcal{V} : l, r.$

$\mathcal{V} \to \mathcal{P}$: ACCEPT if:

$$g^{\tilde{t}}u^{\tau_x} = V^{y^2} \cdot g^{y^2 - 3y^3} \cdot T_1^x \cdot T_2^{x^2},$$
and $$(g')^l(h')^r u^{\mu} = A \cdot S^x \cdot (g')^{-y} \cdot (h')^{3y^2},$$
and $$\tilde{t} = l \cdot r;$$

otherwise REJECT.

Reduction if $n \neq 1$**, on** $(\mathbb{G}, \mathbb{C}, \mathbf{g}', \mathbf{h}', u, P'; \mathbf{l}, \mathbf{r})$**:** (Section IV-B)

$\mathcal{P}$ computes X and sends it to $\mathcal{V}$.

$\mathcal{V}$ generates $x' \in \mathbb{C}$ and sends to $\mathcal{P}$.

$\mathcal{P}$ and $\mathcal{V}$ compute $\widehat{\mathbf{g}'}, \widehat{\mathbf{h}'}, \widehat{u}, \widehat{P'}$ based on Equation (2).

$\mathcal{P}$ computes $\widehat{\mathbf{l}}, \widehat{\mathbf{r}}$ based on Equation (2).

$\mathcal{P}$ and $\mathcal{V}$ recursively compute a reduced statement from $(\mathbb{G}, \mathbb{C}, \widehat{\mathbf{g}'}, \widehat{\mathbf{h}'}, \widehat{u}, \widehat{P'}; \widehat{\mathbf{l}}, \widehat{\mathbf{r}})$, where $\mathcal{P}$'s witness is $\widehat{\mathbf{l}}$ and $\widehat{\mathbf{r}}$.

Corollary 5. ***Logarithmic Range Proof.*** *The above protocol of logarithmic range proof has perfect completeness, perfect honest verifier zero-knowledge and computational special soundness.*

Proof: The range proof is a special case of the aggregated range proof when $m = 1$. Therefore, it is a direct corollary of Theorem 6. ∎

B. Aggregating Range Proofs

Many scenarios require the prover to perform multiple range proofs at the same time. One application is one confidential transaction always involves multiple outputs (i.e., unspent transaction outputs) to allow the sender to collect unspent funds, which requires one range proof for each account. We show how to aggregate m range proofs into one to reduce the communication cost. The multiple range proofs relation is:

$$\{(g, u \in \mathbb{G}, \mathbf{V} \in \mathbb{G}^m; \mathbf{v}, \boldsymbol{\gamma} \in \mathbb{Z}_p^m) : \\ V_j = g^{v_j}u^{\gamma_i} \wedge v_j \in [0, 2^n - 1] \quad \forall j \in [1, m]\}. \tag{12}$$

The main idea of this approach is to use one nm-size vector to represent all v_j. We describe how to aggregate range proofs with our inner-product argument. Specifically, we generate one vector $\mathbf{a}_{L,j}$ for each v_j that satisfies Equation (9), and combine all $\mathbf{a}_{L,j}$ into one $(2nm + m)$-size vector, $[\mathbf{a}_L \circ \mathbf{a}_R, \mathbf{a}_R - \mathbf{1}^{nm} - \mathbf{a}_L, \langle \mathbf{a}_{L,2}, \mathbf{2}^n \rangle, \langle \mathbf{a}_{L,2}, \mathbf{2}^n \rangle, \cdots, \langle \mathbf{a}_{L,m}, \mathbf{2}^n \rangle] = [\mathbf{0}^{2nm}, v_i, v_2, \cdots, v_m]$. The aggregated protocol is similar to the single range proof protocol described in Section V-A, with the following adjustments (more specifically, we only modify the inner-product range proof part).

Firstly, let's define an nm-size vector $\mathbf{2}_j^{nm} = [\mathbf{0}^{(j-1)n}, \mathbf{2}^n, \mathbf{0}^{(m-j)n}]$, and an n-size vector $\mathbf{h}_j = [h_{(j-1)n}, h_{(j-1)n+1}, \cdots, h_{jn-1}]$ (i.e., $(j-1)n$ to $jn-1$ elements of the nm-size vector $\mathbf{h}$). After the verifier challenges with $y \in \mathbb{Z}_p^*$, Equation (5) will become:

$$\begin{aligned} & y^{2nm} \cdot \sum_{j=1}^{m} y^{j-1} \cdot \langle \mathbf{a}_{L,j} \circ \mathbf{2}^n, \mathbf{y}^n \rangle + \\ & y^{nm} \cdot \langle \mathbf{a}_L - \mathbf{1}^{nm} - \mathbf{a}_R, \mathbf{y}^{nm} \rangle + \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^{nm} \rangle \\ = & \sum_{j=1}^{m} y^{2nm+j-1} \cdot v_j. \end{aligned} \tag{13}$$

Accordingly, Equation (10) will become:

$$\begin{aligned} & \left\langle \mathbf{a}_L - y^{nm} \cdot \mathbf{1}^{nm}, \mathbf{y}^{nm} \circ (\mathbf{a}_R + y^{nm} \cdot \mathbf{1}^{nm}) + \sum_{j=1}^{m} y^{2nm+j-1} \cdot \mathbf{2}_j^{nm} \right\rangle \\ = & \sum_{j=1}^{m} y^{2nm+j-1} v_j + \delta(y), \end{aligned}$$

where $\delta(y) = (y^{nm} - y^{2nm}) \cdot \langle \mathbf{1}^{nm}, \mathbf{y}^{nm} \rangle - \sum_{j=1}^{m} y^{2nm+j} \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle$.

Secondly, since $\mathbf{s}_L$ and $\mathbf{s}_R$ become nm-size vectors, we should also adjust $r(X)$, $l(X)$, $t(X)$, and τ_x accordingly:

$$\begin{aligned} l(X) = & \mathbf{a}_L - y^{nm} \cdot \mathbf{1}^{nm} + \mathbf{s}_L \cdot X \in \mathbb{Z}_p^{nm}[X] \\ r(X) = & \mathbf{y}^{nm} \circ (\mathbf{a}_R + y^{nm} \cdot \mathbf{1}^{nm} + \mathbf{s}_R \cdot X) \\ & + \sum_{j=1}^{m} y^{2nm+j-1} \cdot \mathbf{2}_i^{nm} \in \mathbb{Z}_p^{nm}[X] \\ \tau_x = & \tau_1 \cdot x + \tau_2 \cdot x^2 + \sum_{j=1}^{m} y^{2nm+j-1} \gamma_j \in \mathbb{Z}_p. \end{aligned}$$

Finally, the verifier will check

$$g^{\tilde{t}} u^{\tau_x} \stackrel{?}{=} \mathbf{V}^{y^{2nm} \cdot \mathbf{y}^m} \cdot g^{\delta(y)} \cdot T_1^x \cdot T_2^{x^2}, \quad \text{and}$$

$$\mathbf{g}^{\mathbf{l}} (\mathbf{h}')^{\mathbf{r}} u^{\mu} \stackrel{?}{=} A \cdot S^x \cdot \mathbf{g}^{-\mathbf{y}^{nm}} \cdot (\mathbf{h}')^{y^{nm} \cdot \mathbf{y}^{nm}} \prod_{j=1}^{m} (\mathbf{h}'_j)^{y^{2nm+j-1} \cdot \mathbf{2}^n},$$

and $\tilde{t} \stackrel{?}{=} l \cdot r$.

The aggregated range proof requires a prover to send $\lceil \log_2(n \cdot m) \rceil + 4$ group elements and 5 elements in $\mathbb{Z}_p$. Therefore, the proof size only grows by $\lceil \log_2(m) \rceil$, which is much less than treating them individually (multiplied by m).

Theorem 6. ***Aggregated Range Proof.*** *The above protocol of aggregated range proof has perfect completeness, perfect honest verifier zero-knowledge and computational special soundness.*

The details of the proof of Theorem 6 are given in Appendix D.

C. Non-Interactive Range Proof

Though we only discuss interactive range proof protocols so far, we can convert our logarithmic range proof protocols into non-interactive ones with Fiat-Shamir heuristic [37]. All challenges are generated by hashes of the transcript of the interaction up to that point. For instances, $y = H(\mathbf{g}, \mathbf{h}, u, V, A, S)$ and $x = H(\mathbf{g}, \mathbf{h}, u, V, A, S, y, T_1, T_2)$ in the logarithmic range proof. Please note that when the challenge space is $\mathbb{C}$, the hash function should map to $\mathbb{C}$ instead of $\mathbb{Z}_p^*$. We suggest to use a hash function that maps to $\mathbb{Z}_{|\mathbb{C}|}$, and then maps $\mathbb{Z}_{|\mathbb{C}|}$ to $\mathbb{C}$ based on the approach discussed in Appendix A.

To avoid a trusted setup in each protocol, approaches such as adopting a common random string, or using a small public seed to generate public parameters with hash functions can be used.

TABLE I: Performance of Multiple Range Proof Arguments under m range proofs.

	Setup	$\#\mathbb{G}$ elements	$\#\mathbb{Z}_p$ elements
Groth'16 [8]	yes	3	0
Sonic [10]	universal	20	16
PLONK [11]	universal	7	7
Lunar [12]	universal	10	2
Σ-Protocol	no	mn	$3 \cdot mn + 1$
Mimblewimble [19]	no	$0.63 \cdot mn$	$1.26 \cdot nm + 1$
Supersonic [14]	no	$\lceil 2\log_2(mn) \rceil$	$\lceil (\mu+1)\log_2(mn) \rceil$
Dory [15]	no	$\lceil 6\log_2(mn) \rceil + 13$	8
Bulletproofs [18]	no	$\lceil 2\log_2(mn) \rceil + 4$	5
SymmeProof	no	$\lceil \log_2(mn) \rceil + 4$	5

VII. Evaluation

A. Theoretical Analysis

Considering m range proofs, we compare the range proof performance of SymmeProof other approaches, including trusted setup zk-SNARK systems [8], universal setup systems [10], [11], [12], and transparent systems (no setup is required) [20], [14], [15], [18]. The theoretical communication cost of each protocol is depicted in Table I. The μ in Supersonic indicates evaluating μ points of the polynomial in quadratic arithmetic programs. For simplicity, $\mathbb{G}$ elements include all group points even for different curves.

Please note we do not include the challenge size when calculating proof size, since challenges can be replaced by the results of hash functions in non-interactive protocols based on the Fiat-Shamir heuristic (Section VI-C). Nevertheless, reducing the challenges can improve the computational efficiency of non-interactive protocols (fewer hash functions). The communication cost the SymmeProof is significantly lower than other transparent systems, which is only half of the Bulletproofs size. Besides, SymmeProof preserves the nice features of Bulletproofs: the proof size only grows by $\lceil \log_2(m) \rceil$ for m range proofs, while the proof size of Σ-protocol and Mimblewimble grows by m times.

B. Performance Evaluation

We evaluate the performance of SymmeProofs. Since Symmeproofs require $|\mathbb{C}|$ must be super-poly to ensure security, it may not be adopted on today's prime order elliptic curves. We have shown the strategy to find a secure $|\mathbb{C}|$ in Appendix A. Methods of finding elliptic curves with a specific composite order can be referred to [38], [39]. We generate a simple composite order elliptic curve based on [38], the parameters of the curve $\mathscr{E}$ is as follows. This curve $\mathscr{E}$ is secure based

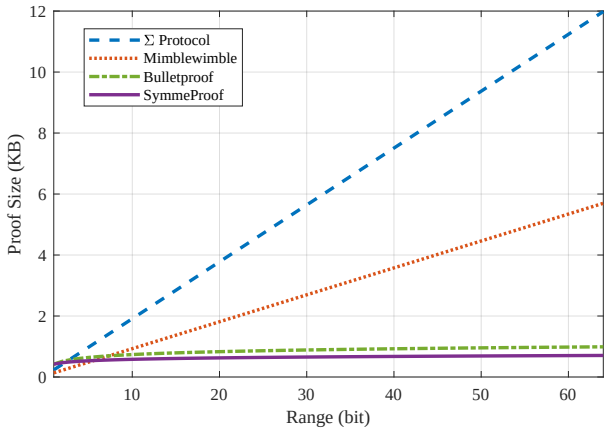


Fig. 1: Range proof size of different proofs.

on the Pohlig-Hellman algorithm. More secured curves can be generated via Cocks-Pinch method [39].

$$\mathscr{E} : y^2 = x^3 + 1 \mod q, \tag{14}$$

where $q = p \times 8 \times \prod_{i=1}^{24} p_i - 1$. p is the order of NIST P-256 curve (a big secure prime), and $\prod_{i=1}^{24} p_i$ is the product of 24-smallest odd primes (in $[3, 97]$)[6]. The order of the curve is $|\mathscr{E}| = p+1$ (as with the approach in Appendix A). Specifically, we give the values of q, $|\mathscr{E}|$, and the base point (G_x, G_y) as follows:

$$\begin{aligned} q =& 0x6f0251b8ffc37d37974f63154276f8240f3661e026c615 \\ & ff7008db6bf1d078a73930ae91e68da9e4739879c5c9e6817. \\ |\mathscr{E}| =& 0x6f0251b8ffc37d37974f63154276f8240f3661e026c615 \\ & ff7008db6bf1d078a73930ae91e68da9e4739879c5c9e6818. \\ G_x =& 0x7c9402ba2a66450571c1bcdb1e74c4f3259d71331f428ec \\ & b1c849a9dae9cf39c132e1089c77efedc5f6ee7796a2945. \\ G_y =& 0x81e2c493c34bbca6ca6ec554ac4daf9df84a2ed38386954 \\ & 23c38afe7e660bceb91aa5888d39fec9e4db535eb20d342. \end{aligned}$$

For $\mathbb{G}$ elements, we use the compressed format which stores each element in 48 bytes. SymmeProof is implemented with Golang based on the self-implemented elliptic package (the curve $\mathscr{E}$ showed above). A reference implementation is shown in [40]. All tests are run on a computer equipped with an i7-8750H CPU and 8GM memory.

The range proof size of Σ-protocol [41], Mimblewimble [20], Bulletproofs [18], and SymmeProof is depicted in Figure 1. When the range is small (less than 3 bit), SymmeProof is not efficient due to the constant elements and additional bits of the curve points in transmissions. When the range is large, SymmeProof is the best. It seems that the SymmeProof's size is more than half of the Bulletproofs. It is because we only evaluate the range up to 64 bits. Considering $\log_2(64) = 6$, the $\log(n)$ part is not significant comparing with the constant part. When n is large, we can regard SymmeProof is only half of the Bulletproofs size.

We show the size of multiple proofs in Figure 2. We zoom in proof size in range $[0, 8]$ to more clearly compare SymmeProof and Bulletproofs. As expected, Σ-protocol and Mimblewimble

[6]In the lattice settings, q does not need to include a large prime, which indicates q does not need to be prohibitively large.

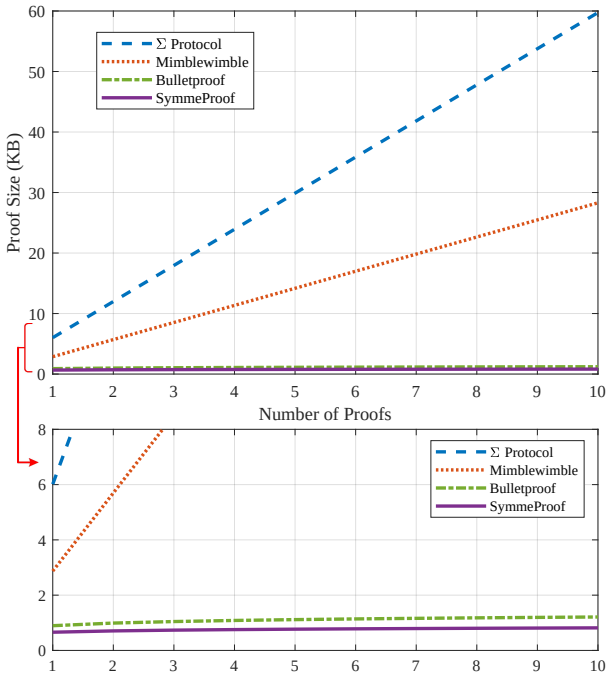


Fig. 2: Multiple range proofs size of different proofs under 32 bits of range.

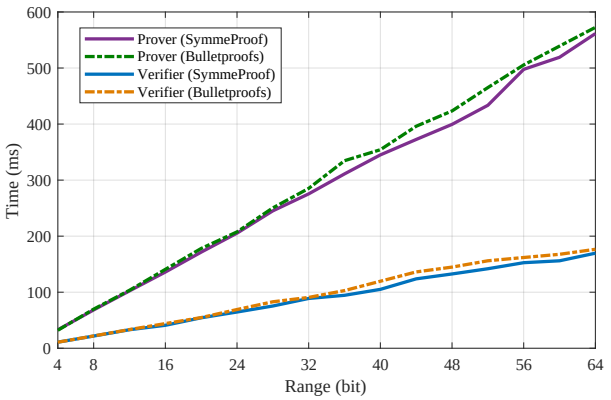


Fig. 3: Time cost of the prover and verifier under SymmeProofs.

grow linearity with the number of proofs. While both Bulletproofs and SymmeProof only grow an additional $\log_2(m)$ size, which is much smaller than other proofs. Meanwhile, SymmeProof is smaller than Bulletproofs regardless of m. As we discussed earlier, SymmeProof's size is more than half of the Bulletproofs since we fix n at 32.

Finally, we compare the time cost of the prover and verifier in our approach with Bulletproofs. All of them grow linearity with the range size. As SymmeProof is built on the "two-set splitting" technique, its performance is similar to Bulletproofs. Our performance is a little bit better since 1) we compute $\widehat{\mathbf{a}} = \mathbf{a}_1 + x\mathbf{a}_2$ rather than $\widehat{\mathbf{a}} = x\mathbf{a}_1 + x^{-1}\mathbf{a}_2$ to avoid computing $x\mathbf{a}_1$ in each iteration, 2) we do not need to compute the inverse values since $x^{-1} = x \mod p$, and 3) we use fewer challenges (fewer hash function calls). The prover's cost is higher than the verifier's cost since it needs to conduct more group operations

such as generating commitments to $\mathbf{s}_L$ and $\mathbf{s}_R$. As all tests are run on our curve $\mathscr{E}$ defined in Equation (14), which does not have assembly codes to speed up group field operations, our result (time) is much higher than Bulletproofs in [18]. The time to generate a proof for a range with 64 bits is 561ms, and the verification time is 169ms. Nevertheless, our performance can be significantly improved by assembling implementations.

VIII. Discussion

A. SymmeProof in Power-of-k Settings

In our design of SymmeProof, we consider the challenge x satisfies a quadratic equation in Equation (6). Here we discuss a more general case that x satisfies

$$x^k = 1 \mod p. \tag{15}$$

Equation (6) is a special case when $k = 2$.

Similar to our analysis in Appendix A, we can compute challenges and derive the size of the challenge space by solving the module functions

$$x^k = 1 \mod q_i^{e_i}, \qquad i \in [0, n], \tag{16}$$

where q_i's and e_i's are defined in Appendix A (by replacing the power-of-2 as k in Equation (18)). Based on Lagrange's theorem, Equation (16) has at most k solutions. Thus, the challenge space size may increase with k. For instance, when k and q are distinct odd primes, there are $gcd(k, q^e - q^{e-1})$ solutions to $x^k = 1 \mod q^e$.

Equation (15) indicates "k-set splitting". We further describe how "k-set splitting" works in the single vector argument (Section IV-A). For two vectors $\mathbf{a} \in \mathbb{Z}_p^n$ and $\mathbf{g} \in \mathbb{G}^n$, we first split $\mathbf{a}$ and $\mathbf{g}$ into k parts, $\mathbf{a} = [\mathbf{a}_1, \cdots \mathbf{a}_k]$, $\mathbf{g} = [\mathbf{g}_1, \cdots \mathbf{g}_k]$. Second, with a challenge x ($x^k = 1 \mod p$), we reduce $\mathbf{a}$ and $\mathbf{g}$ by:

$$\widehat{\mathbf{a}} = \sum_{i=1}^{k} x^i \mathbf{a}_i, \qquad \widehat{\mathbf{g}} = \prod_{i=1}^{k} \mathbf{g}_i^{x^{-i}}.$$

We use a matrix to represent the commitment of the reduced vector $\widehat{\mathbf{a}}$:

$$\begin{array}{c} \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_{k-1} \\ \mathbf{g}_k \end{array} \begin{array}{c} \begin{array}{ccccc} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_{k-1} & \mathbf{a}_k \end{array} \\ \begin{pmatrix} \mathbf{g}_1^{\mathbf{a}_1} & (\mathbf{g}_1^{\mathbf{a}_2})^{x} & \cdots & \cdots & (\mathbf{g}_1^{\mathbf{a}_k})^{x^{k-1}} \\ (\mathbf{g}_2^{\mathbf{a}_1})^{x^{-1}} & \mathbf{g}_2^{\mathbf{a}_2} & \ddots & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \cdots & \cdots & \ddots & \mathbf{g}_{k-1}^{\mathbf{a}_{k-1}} & (\mathbf{g}_{k-1}^{\mathbf{a}_k})^{x} \\ (\mathbf{g}_k^{\mathbf{a}_1})^{x^{-k+1}} & \cdots & \cdots & (\mathbf{g}_k^{\mathbf{a}_{k-1}})^{x^{-1}} & \mathbf{g}_k^{\mathbf{a}_k} \end{pmatrix} \end{array}$$

The new commitment $\widehat{\mathbf{g}}^{\widehat{\mathbf{a}}}$ is the product of all elements in the matrix. We denote elements that share the same exponent with the same color. Based on Equation (15), we have $x^{k-m} = x^{-m} \mod p$, which indicates we can further batch x^{k-m} and x^{-m} exponent parts. Therefore, the new commitment becomes

$$\begin{array}{c} \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_{k-1} \\ \mathbf{g}_k \end{array} \begin{array}{c} \begin{array}{ccccc} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_{k-1} & \mathbf{a}_k \end{array} \\ \begin{pmatrix} \mathbf{g}_1^{\mathbf{a}_1} & (\mathbf{g}_1^{\mathbf{a}_2})^{x} & \cdots & \cdots & (\mathbf{g}_1^{\mathbf{a}_k})^{x^{-1}} \\ (\mathbf{g}_2^{\mathbf{a}_1})^{x^{-1}} & \mathbf{g}_2^{\mathbf{a}_2} & \ddots & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \cdots & \cdots & \ddots & \mathbf{g}_{k-1}^{\mathbf{a}_{k-1}} & (\mathbf{g}_{k-1}^{\mathbf{a}_k})^{x} \\ (\mathbf{g}_k^{\mathbf{a}_1})^{x} & \cdots & \cdots & (\mathbf{g}_k^{\mathbf{a}_{k-1}})^{x^{-1}} & \mathbf{g}_k^{\mathbf{a}_k} \end{pmatrix} \end{array},$$

which can be formally expressed as

$$\widehat{\mathbf{g}}^{\widehat{\mathbf{a}}} = \prod_{i=0}^{k} A_i^{x^i}, \quad \text{where} \quad A_i = \prod_{t-s=i \mod k} \mathbf{g}_s^{\mathbf{a}_t} \quad \text{and} \quad A_0 = A.$$

Accordingly, the communication cost with "k-set splitting" is $(k-1)\log_k(n)$.

B. More Applications

Σ-protocol. Σ-protocols [41] (e.g. Schnorr protocol) for an n-size vector argument require the prover to send n elements in $\mathbb{Z}_p$ (the response to the challenge) and one group element (the commitment). Therefore, we can apply our vector compression technique in Section IV-A directly to Σ-protocols to reduce the communication size. Instead of sending an n-size response, the prover follows the protocol in Section IV-A to reduce the response vector to a single scalar with $\log(n)$ group elements. The communication cost of the compressed Schnorr's protocol is $\log(n) + 1$ group element and one $\mathbb{Z}_p$ element.

Aggregate transactions. In many cases, one confidential transaction may contain multiple parties and each party only knows some of the inputs and outputs to create range proofs for his own part. This technique has been widely used in CoinJoin transactions [42]. Bünz et al. introduce a secure multi-party computation (MPC) protocol to aggregate multiple range proofs into one based on the inner-product argument [18]. With our compression technique, we can further improve the performance of the MPC protocol by replacing the inner-product argument with our approach in Section IV-B.

Mimblewimble. Mimblewimble [19], [20] compresses the blockchain-based on two facts: (1) for valid transactions, the difference between outputs, inputs, and transaction fee should be 0; and (2) an ECDSA public key can be regarded as a commitment to 0. Therefore, Mimblewimble regards the public key as the signature of the difference, and thus reduces the transaction of scriptSig. Since it does not optimize the underlying range proof technique, SymmeProof can be used as a plug-in component to replace the range proofs.

Arithmetic circuit satisfiability. Bootle et al. [17] present an efficient zero-knowledge argument for arithmetic circuits satisfiability with $6\log(n) + 13$ elements by converting the Hadamard-product into a single inner-product relation. The further improvement, Bulletproofs [18], reduces the size to $2\log(n) + 13$ elements by converting the circuits satisfiability into an inner-product form and reducing the communication cost with the improved inner-product argument. Considering our inner-product argument technique in Section IV-B, we

can further reduce the communication cost to $\log(n)$. Thus, a protocol of argument for arithmetic circuits satisfiability with our inner-product argument technique only needs $\log(n) + 13$ elements.

IX. Conclusion

Range proofs have a wide application in today's blockchain-based cryptocurrencies. Previous techniques can be used in blockchain confidential transactions, but the communication cost of those techniques are prohibitive in practice. We propose SymmeProof, which significantly reduces the range proof size of Bulletproofs from $2\log(n) + 9$ to $\log(n) + 9$. Meanwhile, our technique can also be applied to other approaches such as the arithmetic circuit satisfiability argument to reduce the proof size. Evaluation results show that the proof size of our approach is the smallest among all approaches. Besides discrete logarithm implementations, we also wish our solution could provide some insights into lattice settings.

Acknowledgement

We would also like to thank the editor and reviewers of their constructive comments to improve our work. This paper is partially supported by HK PolyU ZVUE A0035279, HK RGC GRF PolyU 15216721/Q86A, and Guangdong Basic and Applied Basic Research Foundation 2020A1515111070.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

[2] G. Maxwell, "Confidential Transactions." https://github.com/lgrkvst/elementsproject.github.io/blob/master/confidential_values.md, 2015.

[3] S. Ma, Y. Deng, D. He, J. Zhang, and X. Xie, "An Efficient NIZK Scheme for Privacy-preserving Transactions over Account-model Blockchain," in *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2020.

[4] S. Gao, T. Zheng, Y. Guo, and B. Xiao, "Efficient and Post-Quantum Zero-Knowledge Proofs for Blockchain Confidential Transaction Protocols," *IACR Cryptology ePrint Archive*, 2021.

[5] T. Zheng, S. Gao, B. Xiao, and Y. Song, "Leaking Arbitrarily Many Secrets: Any-out-of-Many Proofs and Applications to RingCT Protocols," *IACR Cryptology ePrint Archive*, 2021.

[6] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge," in *Annual Cryptology Conference (CRYPTO)*, Springer, 2013.

[7] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic Span Programs and Succinct NIZKs without PCPs," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Springer, 2013.

[8] J. Groth, "On the Size of Pairing-based Non-interactive Arguments," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Springer, 2016.

[9] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers, "Updatable and Universal Common Reference Strings with Applications to zk-SNARKs," in *Annual Cryptology Conference (CRYPTO)*, Springer, 2018.

[10] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn, "Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings," *IACR Cryptology ePrint Archive*, 2019.

[11] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive Arguments of Knowledge," *IACR Cryptology ePrint Archive*, 2019.

[12] M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez, "Lunar: A Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions," in *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, Springer, 2021.

[13] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, Transparent, and Post-Quantum Secure Computational Integrity," in *IACR Cryptology ePrint Archive*, 2018.

[14] B. Bünz, B. Fisch, and A. Szepieniec, "Transparent SNARKs from DARK Compilers," *IACR Cryptology ePrint Archive*, 2019.

[15] J. Lee, "Dory: Efficient, Transparent Arguments for Generalised Inner Products and Polynomial Commitments," in *Theory of Cryptography Conference*, Springer, 2021.

[16] A. Chiesa, D. Ojha, and N. Spooner, "Fractal: Post-Quantum and Transparent Recursive Proofs from Holography," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Springer, 2020.

[17] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Springer, 2016.

[18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short Proofs for Confidential Transactions and More," in *Proc. of the IEEE Symposium onSecurity and Privacy (Oakland)*, IEEE, 2018.

[19] T. E. Jedusor, "Mimblewimble." https://github.com/mimblewimble/docs, 2022.

[20] A. Poelstra, "Mimblewimble." https://cyber.stanford.edu/sites/g/files/sbiybj9936/f/andrewpoelstra.pdf, 2016.

[21] C. Gentry and D. Wichs, "Separating Succinct Non-Interactive Arguments from all Falsifiable Assumptions," in *Proc. of the annual ACM Symposium on Theory of Computing (STOC)*, ACM, 2011.

[22] H. Chung, K. Han, C. Ju, M. Kim, and J. H. Seo, "Bulletproofs+: Shorter Proofs for Privacy-Enhanced Distributed Ledger," *IACR Cryptology ePrint Archive*, 2020.

[23] G. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh, "Provisions: Privacy-Preserving Proofs of Solvency for Bitcoin Exchanges," in *Proc. of the ACM Conference on Computer & Communications Security (CCS)*, ACM, 2015.

[24] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," in *Ethereum Project Yellow Paper*, 2014.

[25] Y. Lu, Q. Tang, and G. Wang, "Zebralancer: Private and Anonymous Crowdsourcing System Atop Open Blockchain," in *Proc. of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2018.

[26] Y. Lu, Q. Tang, and G. Wang, "Dragoon: Private Decentralized Hits Made Practical," in *Proc. of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2020.

[27] H. Feng and Q. Tang, "Witness Authenticating NIZKs and Applications," in *Annual Cryptology Conference (CRYPTO)*, Springer, 2021.

[28] K. Yang, P. Sarkar, C. Weng, and X. Wang, "QuickSilver: Efficient and Affordable Zero-knowledge Proofs for Circuits and Polynomials over any Field," in *Proc. of the ACM Conference on Computer & Communications Security (CCS)*, ACM, 2021.

[29] J. Zhang, T. Liu, W. Wang, Y. Zhang, D. Song, X. Xie, and Y. Zhang, "Doubly Efficient Interactive Proofs for General Arithmetic Circuits with Linear Prover Time," in *Proc. of the ACM Conference on Computer & Communications Security (CCS)*, ACM, 2021.

[30] Z. Wan, Y. Zhou, and K. Ren, "zk-AuthFeed: Protecting Data Feed to Smart Contracts with Authenticated Zero-Knowledge Proof," in *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2022.

[31] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts," in *Proc. of the IEEE Symposium onSecurity and Privacy (Oakland)*, IEEE, 2016.

[32] E. Ben-Sasson, A. Chiesa, A. Gabizon, M. Riabzev, and N. Spooner, "Interactive Oracle Proofs with Constant Rate and Query Complexity," in *Proc. of the International Colloquium on Automata, Languages, and Programming (ICALP)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

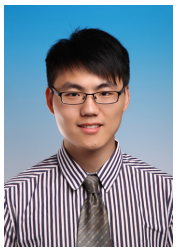
[33] H. Lipmaa, "On Diophantine Complexity and Statistical Zero-knowledge Arguments," in *Annual International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, Springer, 2003.

[34] A. R. Block, J. Holmgren, A. Rosen, R. D. Rothblum, and P. Soni, "Time- and Space-Efficient Arguments from Groups of Unknown Order," in *Annual Cryptology Conference (CRYPTO)*, Springer, 2021.

[35] B. Bünz and B. Fisch, "Schwartz-Zippel for Multilinear Polynomials mod N," in *IACR Cryptology ePrint Archive*, 2022.

[36] S. Bayer and J. Groth, "Efficient Zero-Knowledge Argument for Correctness of a Shuffle," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Springer, 2012.
[37] M. Bellare and P. Rogaway, "Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols," in *Proc. of the ACM Conference on Computer & Communications Security (CCS)*, ACM, 1995.
[38] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF Formulas on Ciphertexts," in *Theory of Cryptography Conference*, Springer, 2005.
[39] D. Boneh, K. Rubin, and A. Silverberg, "Finding Composite Order Ordinary Elliptic Curves Using the Cocks–Pinch Method," 2011.
[40] Gao, Shang, "SymmeProof Implementation." https://github.com/GoldSaintEagle/symmeproof_code/tree/master, 2022.
[41] R. Cramer and I. Damgård, "Zero-Knowledge Proofs for Finite Field Arithmetic, or: Can Zero-Knowledge be for Free?," in *Annual Cryptology Conference (CRYPTO)*, Springer, 1998.
[42] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, "Ligero: Lightweight Sublinear Arguments without a Trusted Setup," in *Proc. of the ACM Conference on Computer & Communications Security (CCS)*, ACM, 2017.

Shang Gao is currently a research assistant professor in the Department of Computing in the Hong Kong Polytechnic University. He received his B.S. degree from Hangzhou Dianzi University, China, in 2010, M.E. degree from Southeast University, China, in 2014, and Ph.D. degree from the Hong Kong Polytechnic University, Hong Kong, in 2019. After graduation, he worked in Microsoft China for one year. His research interests include information security, network security, software-defined networks, blockchain security, and applied cryptography. His work has been published in several top-tier conferences and journals, including CCS, INFOCOM, TON, etc.



Zhe Peng is currently a research assistant professor in the Department of Computer Science, Hong Kong Baptist University (HKBU). Before joining HKBU, he was a blockchain technical director at SF Technology in 2019. He received the Ph.D. degree in Computer Science from the Hong Kong Polytechnic University and the M.Sc. degree in Electronic Engineering and Information Science from University of Science and Technology of China in 2018 and 2013, respectively. In 2010, he received the B.Sc. degree in Communication Engineering from Northwestern Polytechnical University. During 2017, he was a visiting scholar in the Department of Electrical and Computer Engineering at Stony Brook University, supervised by Prof. Yuanyuan Yang. His primary research interests include blockchain system, mobile computing, data security and privacy. His work has been published in several top-tier journals and conferences, such as SIGMOD, TMC, TON, TASE, CCS, INFOCOM, etc.

Feng Tan is currently a senior researcher in Shanghai Artificial Intelligence Institute (SAIRI). Before joining SAIRI, he was a blockchain technical director at DianRong Fintech from 2016 to 2019. He received the Ph.D. degree in Computer Science from the Hong Kong Polytechnic University and the M.Sc. degree in Industrial Engineering from University of Electronic Science and Technology of China in 2016 and 2012, respectively. During 2013, he was a visiting scholar in the Department of Electrical and Computer Engineering at Darmstat University, supervised by Prof. Neeraj Suri. His primary research interests include blockchain system, Cyber-Physical System (CPS) and dependable distributed system. His work has been published in several top-tier journals and conferences, such as DSN, ICCPS, TCPS and TPDS, etc.

Yuanqing Zheng is an associate professor in the Department of Computing, the Hong Kong Polytechnic University. Previously, he was an assistant professor in the same department during 2014-2020. He received the Ph.D. degree in Computer Science from Nanyang Technological University, Singapore. He received the B.S. degree in Electrical Engineering and the M.E. degree in Communication and Information System both from Beijing Normal University, Beijing, China. Dr Zhengs research interests include human centered computing, mobile and network computing, wireless networks, and RFID systems. He has published several papers in premier journals including IEEE/ACM TON, IEEE TMC, ACM TOSN, and top conferences including ACM MobiCom, MobiSys, MobiHoc, SenSys, IEEE INFOCOM, ICNP, ICDCS, etc. He won the Best Demo Award in IEEE SECON 2014. Currently, he is on the editorial board of IEEE Transactions on Wireless Communications. He is a member of IEEE, ACM, and CCF.

Bin Xiao is a professor at Department of Computing, the Hong Kong Polytechnic University, Hong Kong. Prof. Xiao received the B.Sc and M.Sc degrees in Electronics Engineering from Fudan University, China, and Ph.D. degree in computer science from University of Texas at Dallas, USA. His research interests include AI and network security, data privacy, and blockchain systems. He published more than 180 technical papers in international top journals and conferences. Currently, he is the associate editor of IEEE IoTJ, IEEE TCC, and IEEE TNSE. He has been the associate editor of Elsevier JPDC from 2016 to 2021. He is the vice chair of IEEE ComSoc CISTC committee. He has been the track co-chair of IEEE ICDCS2022, the symposium track co-chair of IEEE ICC2020, ICC 2018 and Globecom 2017, and the general chair of IEEE SECON 2018. He is a senior member of IEEE, the member of ACM and CCF.

APPENDIX

A. Challenge Space

1) Generate Challenges: We show how to generate challenges in the challenge space $\mathbb{C} = \{x | x^2 = 1 \mod p\}$. Suppose the prime factorization of p is $p = \prod_{i=0}^{n} q_i^{e_i}$. We need to solve each quadratic equation

$$x^2 = 1 \mod q_i^{e_i}, \qquad i \in [0, n] \tag{17}$$

and get the results $\{x_{i,1}, x_{i,2}, \cdots, x_{i,k_i}\}$ which satisfy $x_{i,k_i} = r_{i,k_i} \mod q_i^{e_i}$. We choose one result in each equation, $x_{1,*}, \cdots, x_{n,*}$, and solve the module functions:

$$x = r_{i,*} \mod q_i^{e_i}, \qquad \forall i \in [0, n], \tag{18}$$

where "$*$" indicates any possible choice from 1 to k_i. The result is the one solution of Equation (6), which can be used as a challenge in $\mathbb{C}$. All results compose the challenge space.

2) Challenge Space Size: Similar to computing the challenge space, we can also use Equation (17) to derive the challenge space size. Suppose $x^2 = 1 \mod q_i^{e_i}$ has k_i solutions. The number of solutions to Equation (6) (i.e., the challenge space size) is $\prod_{i=1}^{n} k_i$.

3) Super-Poly Space: A simple approach to find a secure p with a super-poly (2^n) space is to find $(n-3)$ small odd primes $p_1, \cdots, p_{n-3}$ and a large secure prime p_0. Accordingly, p can be constructed as

$$p = 8 \cdot \prod_{i=0}^{n-3} p_i.$$

For instance, we can construct p with the 77-smallest odd primes (in $[3, 397]$) and the order of NIST P-256 curve, which has 2^{80} space and 128 bits security.

4) Small Challenge Space Size Attack: We show that the protocol is not secure when the challenge space size is small. Specifically, we use the argument of knowledge of one vector as an example to show the prover can pass the protocol without the secret.

Since the challenge space size is small, we suppose the prover guess a correct challenge x in any step of iteration. The prover first randomly generates $\widehat{\mathbf{a}}$ and computes $\widehat{\mathbf{g}} = \mathbf{g}_1 \circ \mathbf{g}_2^{x^{-1}}$. Then, he computes $X = (\widehat{\mathbf{g}}^{\widehat{\mathbf{a}}})^{x^{-1}} \cdot A^{-x^{-1}}$ and sends X to the verifier before receiving the challenge x. Since $\widehat{\mathbf{g}}^{\widehat{\mathbf{a}}} = A \cdot X^x$, the prover can use $\widehat{\mathbf{a}}$ in the following steps of verification to pass the protocol even without the secret $\mathbf{a}$.

B. Useful Lemmas in Composite-order Groups

Lemma 7. *Let $d = gcd(x_1 - x_2, p)$. When adopting the forking lemma, for two accepted transactions in one iteration of the single vector argument (Section IV-A), $(X, x_1, \widehat{\mathbf{a}}_1)$ and $(X, x_2, \widehat{\mathbf{a}}_2)$, $d | (\widehat{a}_{1,i} - \widehat{a}_{2,i})$ holds for all i's where $\widehat{a}_{1,i}$ and $\widehat{a}_{2,i}$ are the i-th elements of $\widehat{\mathbf{a}}_1$ and $\widehat{\mathbf{a}}_2$ respectively.*

Proof: Suppose $d \nmid (\widehat{a}_{1,i} - \widehat{a}_{2,i})$. Let $\widehat{a}_{1,i} - \widehat{a}_{2,i} = k_i \cdot d + r_i$ for some $k_i \in \mathbb{Z}$ and $r_i \in [1, d-1]$. We have

$$\begin{aligned} &\widehat{g}_{1,i}^{\widehat{a}_{1,i}} / \widehat{g}_{2,i}^{\widehat{a}_{2,i}} = (g_{1,i} \cdot g_{2,i}^{x_2})^{k_i \cdot d + r_i + \widehat{a}_{2,i}} / (g_{1,i} \cdot g_{2,i}^{x_2})^{\widehat{a}_{2,i}} \\ =& g_{1,i}^{(x_1 - x_2)\widehat{a}_{2,i}} \cdot (g_{1,i} \cdot g_{2,i}^{x_1})^{k_i \cdot d + r_i}. \end{aligned}$$

By multiplying all $n/2$ elements, we get

$$\prod_{i=0}^{n/2-1} \widehat{g}_{1,i}^{\widehat{a}_{1,i}} / \widehat{g}_{2,i}^{\widehat{a}_{2,i}} = \widehat{\mathbf{g}}_1^{\widehat{\mathbf{a}}_1} / \widehat{\mathbf{g}}_2^{\widehat{\mathbf{a}}_2} = (\widehat{\mathbf{g}}_1^{\widehat{\mathbf{a}}_1})^{x_1 - x_2} \cdot \prod_{i=0}^{n/2-1} \widehat{g}_{1,i}^{k_i \cdot d + r_i}.$$

Considering $\widehat{\mathbf{g}}_1^{\widehat{\mathbf{a}}_1} = A \cdot X^{x_1}$ and $\widehat{\mathbf{g}}_2^{\widehat{\mathbf{a}}_2} = A \cdot X^{x_2}$, we have

$$X^{x_1 - x_2} = (\widehat{\mathbf{g}}_1^{\widehat{\mathbf{a}}_1})^{x_1 - x_2} \cdot \prod_{i=0}^{n/2-1} \widehat{g}_{1,i}^{k_i \cdot d + r_i}. \tag{19}$$

Since we are working on a p-order group, the left-hand-side of Equation (19) indicates the exponents of the right-hand-side must contain the factor $x_1 - x_2$ after adding an integer time of q.[7] Furthermore, for each i, $(x_1 - x_2) | (k_i \cdot d + r_i + s_i \cdot p)$ must hold for some s_i, otherwise we can obtain nontrivial discrete logarithm relations between $\widehat{g}_{1,i}$'s. As $d = gcd(x_1 - x_2, p)$, we have $d | (x_1 - x_2)$ and $d | p$, which implies $d | (k_i \cdot d + r_i + s_i \cdot p)$. Thus, $d | r_i$ must hold. This contradicts with $r_i \in [1, d-1]$. Therefore, $d | (\widehat{a}_{1,i} - \widehat{a}_{2,i})$ holds for all i's. ∎

Lemma 8. *If $d | (\widehat{a}_{1,i} - \widehat{a}_{2,i})$ and $d | (x_1 - x_2)$, then $d | (x_1 \widehat{a}_{1,i} - x_2 \widehat{a}_{2,i})$.*

Proof: Let $\widehat{a}_{1,i} - \widehat{a}_{2,i} = r \cdot d$ and $x_1 - x_2 = s \cdot d$. Then $x_1 \widehat{a}_{1,i} - x_2 \widehat{a}_{2,i} = (s \cdot d + x_2)(r \cdot d + \widehat{a}_{2,i}) - x_2 \widehat{a}_{2,i} = d(s \cdot r \cdot d + s \cdot \widehat{a}_{2,i} + r \cdot x_2)$. Thus $d | (x_1 \widehat{a}_{1,i} - x_2 \widehat{a}_{2,i})$. ∎

C. Proof of Theorem 2

Proof: **Perfect completeness.** In one iteration, $\widehat{\mathbf{a}} = \mathbf{a}_1 + x\mathbf{a}_2$ and $\widehat{\mathbf{g}} = \mathbf{g}_1 \circ \mathbf{g}_2^{x^{-1}}$, we have

$$\widehat{\mathbf{g}}^{\widehat{\mathbf{a}}} = \mathbf{g}_1^{\mathbf{a}_1} \cdot \mathbf{g}_2^{\mathbf{a}_2} \cdot (\mathbf{g}_1^{\mathbf{a}_2})^{x} \cdot (\mathbf{g}_2^{\mathbf{a}_1})^{x^{-1}} = \mathbf{g}_1^{\mathbf{a}_1} \cdot \mathbf{g}_2^{\mathbf{a}_2} \cdot (\mathbf{g}_1^{\mathbf{a}_2} \cdot \mathbf{g}_2^{\mathbf{a}_1})^{x} = A \cdot X^x.$$

Therefore, the single vector argument is perfect complete.

Witness extended emulation. We only consider how to construct an extractor $\mathcal{E}$ in one iteration. The discrete logarithm relations or the witness can be iteratively generated by $\mathcal{E}$.

Based on the Pohlig-Hellman algorithm, the discrete logarithm problem on a composite-order curve is as hard as the problem on a prime-order curve where the prime order being the largest prime factor of the composite order. Thus, the discrete logarithm assumption holds on in this paper.

The extractor runs as the normal protocol till the prover gets X. Then, with three different challenges x_1, x_2, x_3, the extractor gets $\widehat{\mathbf{a}}_1$, $\widehat{\mathbf{a}}_2$, and $\widehat{\mathbf{a}}_3$ such that:

$$A \cdot X^{x_i} = (\mathbf{g}_1 \circ \mathbf{g}_2^{x_i^{-1}})^{\widehat{\mathbf{a}}_i}, \quad \forall i \in [1, 3]. \tag{20}$$

Since $x_i^{-1} = x_i \mod p$, we have $X^{x_1 - x_2} = \mathbf{g}_1^{\widehat{\mathbf{a}}_1 - \widehat{\mathbf{a}}_2} \circ \mathbf{g}_2^{x_1 \widehat{\mathbf{a}}_1 - x_2 \widehat{\mathbf{a}}_2}$ by subtracting the two equations of x_1 and x_2. Let $d = gcd(x_1 - x_2, p)$. Based on Lemma 7 and Lemma 8, we get $d | (x_1 - x_2)$ and $d | (x_1 \widehat{\mathbf{a}}_1 - x_2 \widehat{\mathbf{a}}_2)$. Thus, $X^{(x_1 - x_2)/d} = \mathbf{g}_1^{(\widehat{\mathbf{a}}_1 - \widehat{\mathbf{a}}_2)/d} \circ \mathbf{g}_2^{(x_1 \widehat{\mathbf{a}}_1 - x_2 \widehat{\mathbf{a}}_2)/d}$ on a p/d-order group. Since $d = gcd(x_1 - x_2, p)$, we have $gcd((x_1 - x_2)/d, p/d) = 1$. Thus, $(x_1 - x_2)/d$ is invertible on the p/d-order group, which

[7] Suppose $\widehat{\mathbf{g}}_1^{\widehat{\mathbf{a}}_1} = X^t$ and $\widehat{g}_{1,i} = X^{t_i}$ for some unknown t and t_i's, Equation (19) indicates $x_1 - x_2 = t(x_1 - x_2) + \sum_{i=0}^{n/2-1} t_i(k_i \cdot d + r_i) + s \cdot p$ for some s.

allows us to extract $\mathbf{a}_X$ such that $X = \mathbf{g}^{\mathbf{a}_X}$.[8]

Taking $\mathbf{a}_X$ into Equation (21), we can further derive $\mathbf{a}_A$ such that $A = \mathbf{g}^{\mathbf{a}_A}$. Set $\mathbf{a}_A = [\mathbf{a}_{A,1}, \mathbf{a}_{A,2}]$ and $\mathbf{a}_B = [\mathbf{a}_{B,1}, \mathbf{a}_{B,2}]$. When given the previously extracted witness $\mathbf{a}'$, we have

$$\begin{aligned} &A \cdot X^x = \mathbf{g}^{\mathbf{a}_A + x\mathbf{a}_X} = \mathbf{g}_1^{\mathbf{a}_{A,1} + x\mathbf{a}_{X,1}} \cdot \mathbf{g}_2^{\mathbf{a}_{A,2} + x\mathbf{a}_{X,2}} \\ =&(\mathbf{g}_1 \circ \mathbf{g}_2^{x^{-1}})^{\mathbf{a}'} = \mathbf{g}_1^{\mathbf{a}'} \cdot \mathbf{g}_2^{x^{-1}\mathbf{a}'} \\ \implies &\mathbf{a}' = \mathbf{a}_{A,1} + x\mathbf{a}_{X,1} \quad \wedge \quad x^{-1}\mathbf{a}' = \mathbf{a}_{A,2} + x\mathbf{a}_{X,2}. \end{aligned}$$

If the implication does not hold, we can directly obtain a nontrivial discrete logarithm relation between $\mathbf{g}_1$ and $\mathbf{g}_2$ (and iteratively obtain $g_1, \cdots, g_n$). Otherwise, we consider all six distinct challenges. To ensure the implication holds, we must have $\mathbf{a}_{A,1} = \mathbf{a}_{X,2}$ and $\mathbf{a}_{A,2} = \mathbf{a}_{X,1}$ ($x = x^{-1} \mod p$). Then we have

$$\mathbf{a}' = \mathbf{a}_{A,1} + x\mathbf{a}_{A,2}.$$

Therefore, the extractor $\mathcal{E}$ either extracts the discrete logarithm relations, or efficiently computes the witness $\mathbf{a}_A$. We can further conclude the protocol has witness-extended emulation with the forking lemma in Lemma 1. ∎

D. Proof of Theorem 3

The proof of Theorem 3 is similar to the proof of Theorem 2. Here we only briefly describe the proof of witness extended emulation.

Proof: After rewinding four times with $x_1, \cdots, x_4$ and computing $\mathbf{a}_{(1)}, \cdots, \mathbf{a}_{(4)}$ and $\mathbf{b}_{(1)}, \cdots, \mathbf{b}_{(4)}$, Equation (21) becomes:

$$P \cdot X^{x_i} = (\mathbf{g}_1 \circ \mathbf{g}_2^{x_i^{-1}})^{\mathbf{a}_{(i)}} \cdot (\mathbf{h}_1 \circ \mathbf{h}_2^{x_i})^{\mathbf{b}_{(i)}} \cdot u^{\langle \mathbf{a}_{(i)}, \mathbf{b}_{(i)} \rangle}, \ \forall i \in [1, 4].$$

We use x_1 and x_2 to derive η_1 and η_2 such that $\eta_1 + \eta_2 = 0$ and $x_1\eta_1 + x_2\eta_2 = 1$, so do η_1' and η_2' such that $\eta_1' + \eta_2' = 1$ and $x_1\eta_1' + x_2\eta_2' = 0$. Based on Lemma 7 and Lemma 8, we compute $\mathbf{a}_X$, $\mathbf{b}_X$, and c_X with η_1 and η_2; $\mathbf{a}_P$, $\mathbf{b}_P$, and c_P with η_1' and η_2' such that $X = \mathbf{g}^{\mathbf{a}_X}\mathbf{h}^{\mathbf{b}_X}u^{c_X}$ and $P = \mathbf{g}^{\mathbf{a}_P}\mathbf{h}^{\mathbf{b}_P}u^{c_P}$.

Let $\mathbf{a}_X = [\mathbf{a}_{X,1}, \mathbf{a}_{X,2}]$, $\mathbf{b}_X = [\mathbf{b}_{X,1}, \mathbf{b}_{X,2}]$, $\mathbf{a}_P = [\mathbf{a}_{P,1}, \mathbf{a}_{P,2}]$ and $\mathbf{b}_P = [\mathbf{b}_{P,1}, \mathbf{b}_{P,2}]$. Given the previously extracted witness $\mathbf{a}'$ and $\mathbf{b}'$, we have:

$$\begin{aligned} &P \cdot X^x = \mathbf{g}^{\mathbf{a}_P + x\mathbf{a}_X} \cdot \mathbf{h}^{\mathbf{b}_P + x\mathbf{b}_X} \cdot u^{c_P + xc_X} \\ =&(\mathbf{g}_1 \circ \mathbf{g}_2^{x^{-1}})^{\mathbf{a}'} \cdot (\mathbf{h}_1 \circ \mathbf{h}_2^{x})^{\mathbf{b}'} \cdot u^{\langle \mathbf{a}', \mathbf{b}' \rangle} \\ \implies &\mathbf{a}' = \mathbf{a}_{P,1} + x\mathbf{a}_{X,1} \quad \wedge \quad x^{-1}\mathbf{a}' = \mathbf{a}_{P,2} + x\mathbf{a}_{X,2} \quad \wedge \\ &\mathbf{b}' = \mathbf{b}_{P,1} + x\mathbf{b}_{X,1} \quad \wedge \quad x\mathbf{b}' = \mathbf{b}_{P,2} + x\mathbf{b}_{X,2} \quad \wedge \\ &\langle \mathbf{a}', \mathbf{b}' \rangle = c_P + xc_X. \end{aligned}$$

The implication must hold, otherwise we can directly obtain a nontrivial discrete logarithm relation between $\mathbf{g}_1$, $\mathbf{g}_2$, $\mathbf{h}_1$, $\mathbf{h}_2$ and u. Therefore, considering $x = x^{-1} \mod p$, we must have $\mathbf{a}_{P,1} = \mathbf{a}_{X,2}$, $\mathbf{a}_{P,2} = \mathbf{a}_{X,1}$, $\mathbf{b}_{P,1} = \mathbf{b}_{X,2}$, and $\mathbf{b}_{P,2} = \mathbf{b}_{X,1}$ to also satisfy the third challenge x_3, which make

$$\mathbf{a}' = \mathbf{a}_{P,1} + x\mathbf{a}_{P,2}, \quad \mathbf{b}' = \mathbf{b}_{P,1} + x^{-1}\mathbf{b}_{P,2}.$$

[8]When multiple $\mathbf{a}_X$'s are extracted, e.g. both $\mathbf{a}_X$ and $\mathbf{a}_X + p/d$ are acceptable solutions, we can efficiently use the equation of x_3 to eliminate unexpected ones.

Meanwhile, c_P and c_X are expected to have the relationship:

$$\begin{aligned} &c_P + xc_X = \langle \mathbf{a}', \mathbf{b}' \rangle \\ =&\langle \mathbf{a}_{P,1} + x\mathbf{a}_{P,2},\ \mathbf{b}_{P,1} + x^{-1}\mathbf{b}_{P,2} \rangle \\ =&\langle \mathbf{a}_{P,1}, \mathbf{b}_{P,1} \rangle + \langle \mathbf{a}_{P,2}, \mathbf{b}_{P,2} \rangle + x\left(\langle \mathbf{a}_{P,1}, \mathbf{b}_{P,2} \rangle + \langle \mathbf{a}_{P,2}, \mathbf{b}_{P,1} \rangle\right) \\ =&\langle \mathbf{a}_P, \mathbf{b}_P \rangle + x\left(\langle \mathbf{a}_{P,1}, \mathbf{b}_{P,2} \rangle + \langle \mathbf{a}_{P,2}, \mathbf{b}_{P,1} \rangle\right). \end{aligned}$$

To also satisfy x_4, we must have

$$\langle \mathbf{a}_P, \mathbf{b}_P \rangle = c_P.$$

Therefore, the extractor either extracts the discrete logarithm relations, or efficiently computes the witness $\mathbf{a}_P, \mathbf{b}_P$. With the forking lemma in Lemma 1, we can conclude the protocol has witness-extended emulation. ∎

E. Proof of Theorem 6

Proof: Since $t_0 = y^{2n} \cdot \langle (\mathbf{y}^{\mathbf{n}})^m, \mathbf{v} \rangle + \delta(y)$, we can directly prove perfect completeness of Theorem 6. For perfect honest-verifier zero-knowledge, we construct a simulator which can generate a distribution of proofs from the statement $(u, g \in \mathbb{G}, \mathbf{g}, \mathbf{h} \in \mathbb{G}^{nm}, \mathbf{V} \in \mathbb{G}^m)$ that is indistinguishable from valid proofs generated by an honest prover and an honest verifier. Specifically, the verifier can randomly choose the unknown elements in the conversation expect S and T_1, and use computes S and T_1 as follows:

$$\begin{aligned} S =& \Big(u^{-\mu} \cdot A \cdot \mathbf{g}^{-y^{nm} \cdot \mathbf{1}^{nm} - \mathbf{l}} \cdot (\mathbf{h}')^{y^{nm} \cdot \mathbf{y}^{nm} - \mathbf{r}} \\ &\cdot \prod_{j=1}^{m} (\mathbf{h}_j')^{y^{2nm+j-1} \cdot \mathbf{2}^n} \Big)^{-x^{-1}}; \\ T_1 =& (u^{-\tau_x} g^{\delta(y) - \tilde{t}} \cdot \mathbf{V}^{y^{2nm} \cdot \mathbf{y}^{nm}} \cdot T_2^{x^2})^{-x^{-1}}. \end{aligned} \tag{21}$$

As the simulator can run the inner-product argument with the simulated witness $(\mathbf{l}, \mathbf{r})$ and the verifier's randomness, all elements in the proof are either independently randomly distributed or their relationships are defined by Equation (21). Since the range proof part is zero-knowledge, the inner-product argument remains zero-knowledge with our simulated witness. The simulator runs in time $O(\mathcal{V} + \mathcal{P})$ and is thus efficient.

For computational special soundness, we construct an extractor $\mathcal{E}$ that runs the prover with $2nm + m$ different y and 3 different x, which result in $3m \cdot (2n + 1)$ valid transcripts. $\mathcal{E}$ first calls the inner-product extractor $\mathcal{E}_{\mathsf{IP}}$ defined in Appendix E to extract $\mathbf{l}$ and $\mathbf{r}$ such that $\tilde{t} = \langle \mathbf{l}, \mathbf{r} \rangle$ and $P = A \cdot S^x \cdot \mathbf{g}^{-\mathbf{y}^{nm}} \cdot (\mathbf{h}')^{y^{nm} \cdot \mathbf{y}^{nm}} \cdot \prod_{j=1}^{m} (\mathbf{h}_j')^{y^{2nm+j-1} \cdot \mathbf{2}^n} = \mathbf{g}^{\mathbf{l}} (\mathbf{h}')^{\mathbf{r}} u^{\mu}$. Furthermore, we use $A \cdot S^x \cdot \mathbf{g}^{-\mathbf{y}^{nm}} \cdot (\mathbf{h}')^{y^{nm} \cdot \mathbf{y}^{nm}} \prod_{j=1}^{m} (\mathbf{h}_j')^{y^{2nm+j-1} \cdot \mathbf{2}^n} = \mathbf{g}^{\mathbf{l}} (\mathbf{h}')^{\mathbf{r}} u^{\mu}$ relationship to compute $\alpha, \rho, \mathbf{a}_L, \mathbf{a}_R, \mathbf{s}_L, \mathbf{s}_R$ such that $A = u^{\alpha}\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R}$ and $S = u^{\rho}\mathbf{g}^{\mathbf{s}_L}\mathbf{h}^{\mathbf{s}_R}$ with a fixed y and 2 different x. If we can derive different A and S with different x, we then find a non-trivial discrete logarithm relation between $u, \mathbf{g}, \mathbf{h}$.

With $A, S, \mathbf{l}, \mathbf{r}$, we can find for all challenges x and y, $\mathbf{l} = \mathbf{a}_L - y^{nm} \cdot \mathbf{1}^{nm} + \mathbf{s}_L \cdot x$ and $\mathbf{r} = \mathbf{y}^{nm} \circ (\mathbf{a}_R + y^{nm} \cdot \mathbf{1}^{nm} + \mathbf{s}_R \cdot x) + \sum_{j=1}^{m} y^{2nm+j-1} \cdot \mathbf{2}_i^{nm}$ must hold, otherwise, we have two distinct representations of the same group element using a set

of independent generators, which yields a non-trivial discrete logarithm relation.

Furthermore, we can compute τ_1, τ_2, t_1, t_2 based on $g^{\tilde{t}} u^{\tau_x} = \mathbf{V}^{y^{2nm} \cdot \mathbf{y}^{nm}} \cdot g^{\delta(y)} \cdot T_1^x \cdot T_2^{x^2}$ with 3 different x and a fixed y such that

$$T_1 = g^{t_1} u^{\tau_1}, \quad T_2 = g^{t_2} u^{\tau_2}.$$

Meanwhile, we can compute v, γ such that $g^v u^\gamma = \prod_{j=1}^m V_j^{y^{2nm+j-1}}$. When repeating with m different y, we can derive $v_1, \cdots, v_m$ and $\gamma_1, \cdots, \gamma_m$ such that

$$g^{v_j} u^{\gamma_j} = V_j, \quad \forall j \in [1, m].$$

If $\tilde{t} = t_2 \cdot x^2 + t_1 \cdot x + \sum_{j=1}^m y^{2nm+j-1} \cdot v_j + \delta(y)$ does not hold, we can yield a non-trivial discrete logarithm relation between g, u. Otherwise, for all 3 different x, we have

$$t_0 + t_1 \cdot x_i + t_2 \cdot x_i^2 - \langle l(x_i), r(x_i) \rangle = 0, \quad i \in [1, 6].$$

The left side of the equation has a degree of 2, but the equation must satisfy 6 distinct challenges, it must be a zero polynomial. Thus, we must have

$$t_0 + t_1 \cdot X + t_2 \cdot X^2 = \langle l(X), r(X) \rangle, \tag{22}$$

Therefore, for $2nm + m$ different y challenges, the following equation must hold:

$$\begin{aligned}
&\sum_{j=1}^m y^{2nm+j-1} \cdot \langle \mathbf{v}_j, \mathbf{2}^n \rangle + \delta(y) \\
=&\langle \mathbf{a}_L, \mathbf{y}^{nm} \circ \mathbf{a}_R \rangle + y^{nm} \cdot \langle \mathbf{a}_L - \mathbf{a}_R, \mathbf{y}^{nm} \rangle \\
&+ \sum_{j=1}^m y^{2nm+j-1} \langle \mathbf{a}_{L,j}, \mathbf{2}^n \rangle \\
&- y^{2nm} \langle \mathbf{1}^{nm}, \mathbf{y}^{nm} \rangle - \sum_{j=1}^m y^{2nm-j} \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle,
\end{aligned} \tag{23}$$

To ensure Equation (23) hold, we mush have

$$\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^{nm} \wedge \mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{nm} \wedge \langle \mathbf{a}_{L,j}, \mathbf{2}^n \rangle = v_j, \forall j \in [1, m].$$

Therefore, extractor $\mathcal{E}$ either extracts the discrete logarithm relations, or efficiently computes the witness $\mathbf{a}_L$ and $\mathbf{a}_R$. Using the forking lemma in Lemma 1, we can conclude the protocol has witness-extended emulation. ■