Cuproof: A Novel Range Proof with Constant Size

Cong Deng, Xianghong Tang, Lin You*, Gengran Hu

Abstract—By combining inner-product and Lagrange's four-square theorem, we structure a range proof scheme which is called Cuproof. The scheme of Cuproof would make a range proof to prove that a secret number $v \in [a, b]$ without exposing redundant information of v. In Cuproof, the communication cost and the proof time is constant. Once the interval of range proof is large, the scheme of Cuproof would show better. Zero-knowledge proof is widely used in blockchain. For example, zk-SNARK is used by Zcash as its core technology in identifying transactions. Up to now, various range proofs have been proposed as well their efficiency and range-flexibility are enhanced. Bootle et al. firstly used inner product method and recursion to an efficient zero-knowledge proof. Then, Benediky Bünz et al. came up with an efficient zero-knowledge argument called Bulletproofs which convinces the verifier that a secret number lies in $[0, 2^n]$. The scheme of Cuproof is based on the scheme of Bulletproofs.

Index Terms—Blockchain, Zero-Knowledge proof, Range proof, Inner-product, Bulletproofs.

1 Introduction

The blockchain-based cryptocurrencies enable peer-to-peer transactions and make sure the transactions are valid. In the system of Bitcoin [1], all the transactions are recorded in a common ledger. So every one can check whether the transactions in the ledger are valid. The hash function also makes the transactions in blockchain can't be tampered. However, every coin has two sides. The transparency is both an advantage and a disadvantage of Bitcoin. In a transaction of Bitcoin, the data of transaction, the addresses of sender and receiver are all transparent which means the anonymities of Bitcoin is not so strong.

In order to offset the disadvantage of Bitcoin, other cryptocurrencies like Zcash [2] was proposed. The transactions between shielded addresses are the special parts of Zcash. In these kind transactions, although the traders' addresses and the amount are all covert, the validity of these transactions can still be checked because of a kind of zero knowledge proof so called zk-SNARK. However, zk-SNARK requires a trusted setup that means the trusted setup should be honest. The trusted setup is the weakness of zk-SNARK because once the setup organization is rantankerous, the safety of secret number in the proof can't be guaranteed. To avoid trusted setup, some other zk-SNARKs without trusted setup were proposed. For example, Srinath Setty [3] put forward a kind of zk-SNARK which doesn't need a trust setup. One could also avoid trusted setup by using STARK [4]. According to the property of protecting anonymity, more and more cryptocurrencies apply zero knowledge proof as a tool to avert the disclosure of users' information.

- *Cong Deng is with the School of Communication Engineering, Hangzhou Dianzi University, Hangzhou, China, 310018.*
  Email: mrdengcong@gmail.com
- *Xianghong Tang is with the School of Communication Engineering, Hangzhou Dianzi University, Hangzhou, China, 310018.*
  Email: tangxh@hdu.edu.cn
- *Lin You is with the School of Communication Engineering, Hangzhou Dianzi University, Hangzhou, China, 310018.*
  Email: mryoulin@gmail.com (Corresponding author)
- *Gengran Hu is with the School of Communication Engineering, Hangzhou Dianzi University, Hangzhou, China, 310018.*
  E-mail: grhu@hdu.edu.cn

Our work is based on Bulletproofs [5]. We combine the Lagrange integers theorem to Bulletproofs [5] to construct an range proof for arbitrary interval. In our scheme, the communication costs are 2 elements of $\mathbb{G}_p$, 4 elements of $\mathbb{Z}_p$, 2 elements of $\mathbb{G}_q$, 3 elements of $\mathbb{Z}_q$, and 12 elements of $\mathbb{Z}$. The 12 elements of $\mathbb{Z}$ are in interval $[-6p, 3p^2 + 6p]$. Comparing to the communication costs of Bulletproofs [5] which is logarithmic in n, the communication costs of ours is constant. So when the interval of range proof is large, our scheme has more advantages than Bulletproof in communication costs.

1.1 Related work

There are lots of researches on range proof from the day the first relevant algorithm of range proof was proposed. Brickel et al. [6] first stated the correlative algorithm of range proof in 1987. Its aim was to send reliable values to other participants gradually so that can allow a user with a discrete logarithm to disclose a bit of information to another user that any other user can verify as they receive each bit. In 1998, Chan et al. [7] showed how to use [6] to verify the non-negative transaction amount and enhanced the algorithm in [6]. This is so called CTF proof and because its security depends on modulus, to keep completeness, the order of group must be unknown. In 2000, Boudot [8] used the square numbers to build an effective range proof which is based on CFT. Using the Lagrange's four-square theorem [9], "any non-negative integer can be represented as the sum of square of four integers", Lipmaa [10] pushed out a proof of any range for first time. In 2005, Groth [11] pointed out that if y was a non-negative integer, then $4y + 1$ could be represented as the sum of square of three integers. The protocol of [11] needs a trusted setup to generate RSA modulus or needs a prohibitively large modulus. Using Boneh-Boyen signature based on bilinear pairings implementation (BBS) [12], Teranishi and Sako [13] proposed many anonymous authentication in 2006, based this work, in 2008, Camenisch et al. [14] used signature method which relies on the security of the q-Strong DiffieHellman assumptions to construct a range proof. In 2014, Mira Belenkiy [15] designed a scheme to extended the u-proof cryptographic specification [16] with the set member proof, this scheme can be used twice to compare the size of the committed value to make a range proof. In 2019, Maller et al. presented Sonic [17] which is the first potentially practical zk-SNARK with fully succinct verification for general arithmetic circuits with such an SRS, but the verson of Sonic enabling fully succinct verification still requires relatively high proof construction overheads. Then, Ariel Gabizon et al. presented Plonk [18] which is more efficient than Sonic.

Bootle et al. [19] made a step forward on the efficiency of space in zero-knowledge proof based on discrete logarithms. They used inner product method and recursion to enhance the efficiency of zero-knowledge proof. Based on these works, Benediky Bünz et al. [5] improved method of inner product certificate and came up with a more efficient zero knowledge proof scheme so called Bulletproofs. Not only can it be applied to the range of certificates and reshuffle and other applications, but also doesn't need trusted setup. Our works are based on the techniques of Bulletproofs [5].

1.2 Contributions

Our scheme is established on the techniques of [5] and the theorem of [11]. The protocol of ours can make a proof for arbitrary range. The argument of ours has small computation complexity. The main difference between the protocol of [5] and ours is that the cost of communication of [5] is logarithmic in n but the cost of ours is constant. The key is that we combine the Theorem 2 to the protocol of [5]. Our protocol satisfies three security properties of zero-knowledge proof: correctness, soundness and zero-knowledge.

1.3 Structure of the paper

In Section 2, some mathematical symbols, definitions and theorems are given. The framework and construction of our range proof protocol are stated in Section 3. In Section 3.1, we show how to construct a proof which convinces the verifier that the prover knows the secret number v. In Section 3.2, we describe the protocol of our range proof which proves that $v \in [a, b]$. Some comparisons with other referred works and the performance of cuproof are shown in Section 4. Finally, the Forking Lemma and the proof of Theorem 3 will be given in Appendix B.

2 Preliminaries

Before we state our protocol, we first state some of the underlying tools. In this paper, $\mathcal{A}$ is a PPT adversary which is a probabilistic interactive Turing Machine that runs in polynomial time in the security parameter λ.

2.1 Assumptions

Definition 1 (Discrete Log Relation). *For all PPT adversaries $\mathcal{A}$ and for all $n \geq 2$ there exists a negligible function $\mu(\lambda)$ such that*

$$P\left[\begin{array}{ll} \mathbb{G} = \operatorname{Setup}\left(1^{\lambda}\right), & \\ g_1, \ldots, g_n \xleftarrow{\$} \mathbb{G}; & : \begin{array}{l} \exists a_i \neq 0, \\ \prod_{i=1}^{n} g_i^{a_i} = 1 \end{array} \\ a_1, \ldots, a_n \in \mathbb{Z}_p \leftarrow \mathcal{A}\left(g_1, \ldots, g_n\right) & \end{array}\right] \leqslant \mu(\lambda)$$

As Benediky Bünz et al [5] states, $\prod_{i=1}^{n} g_i^{a_i} = 1$ is a non trivial discrete log relation between $g_1, ..., g_n$. The Discrete Log Relation assumption makes sure that an adversary can't find a non-trivial relation between randomly selected group elements. This assumption is equivalent to the discrete-log assumption when $n \geq 1$.

2.2 Commitments

Definition 2 (Commitments). *A non-interactive commitment scheme consists of a pair of probabilistic polynomial time algorithms* (Setup, Com). *The setup algorithm $pp \leftarrow \operatorname{Setup}(1^{\lambda})$ generates public parameters pp for the scheme, for security parameter λ. The commitment algorithm Com_{pp} defines a function $\mathrm{M}_{pp} \times \mathrm{R}_{pp} \to \mathrm{C}_{pp}$ for message space M_{pp}, randomness space R_{pp} and commitment space C_{pp} determined by pp. For a message $x \in \mathrm{M}_{pp}$, the algorithm draws $r \xleftarrow{\$} \mathrm{R}_{pp}$ uniformly at random, and computes commitment $\mathbf{com} = \mathrm{Com}_{pp}$.*

Definition 3 (Homomorphic Commitments). *A homomorphic commitment scheme is a non-interactive commitment scheme such that $\mathrm{M}_{pp}, \mathrm{R}_{pp}$ and C_{pp} are all abelian groups, and for all $x_1, x_2 \in \mathrm{M}_{pp}, r_1, r_2 \in \mathrm{R}_{pp}$, we have*

$$\mathrm{Com}(x_1; r_1) + \mathrm{Com}(x_2; r_2) = \mathrm{Com}(x_1 + x_2; r_1 + r_2).$$

For ease of notation we write $\mathrm{Com} = \mathrm{Com}_{pp}$.

Definition 4 (Hiding Commitment). *A commitment scheme is said to be hiding if for all PPT adversaries $\mathcal{A}$ there exists a negligible function such that.*

$$\left|\mathrm{P}\left[b = b' \;\middle|\; \begin{array}{l} pp \leftarrow \operatorname{Setup}(1^{\lambda}); \\ (x_0, x_1) \in \mathrm{M}_{pp}^2 \leftarrow \mathcal{A}(pp), b \xleftarrow{\$} (0,1), r \xleftarrow{\$} \mathrm{R}_{pp}, \\ \mathbf{com} = \mathrm{Com}(x_b; r), b' \leftarrow \mathcal{A}(pp, \mathbf{com}) \end{array} \right] - \frac{1}{2}\right| \leq \mu(\lambda)$$

where the probability is over b, r, Setup *and $\mathcal{A}$. If $\mu(\lambda) = 0$ then we say the scheme is* perfectly hiding.

Definition 5 (Binding Commitment). *A commitment scheme is said to be binding if for all PPT adversaries $\mathcal{A}$ there exists a negligible function μ such that.*

$$P\left[\begin{array}{l} \mathrm{Com}(x_0; r_0) = \mathrm{Com}(x_1; r_1), \\ x_0 \neq x_1 \end{array} \;\middle|\; \begin{array}{l} pp \leftarrow \operatorname{Setup}(1^{\lambda}), \\ x_0, x_1, r_0, r_1 \leftarrow \mathcal{A}(pp) \end{array} \right] \leq \mu(\lambda)$$

where the probability is over Setup *and $\mathcal{A}$. If $\mu(\lambda) = 0$ then we say the scheme is* perfectly binding.

In the following content, to make sure that discrete log in the groups we used is intractable for PPT adversaries, the order p, q of these groups is implicitly dependent on the security parameter.

Definition 6 (Pedersen Commitment). $\mathrm{M}_{pp}, \mathrm{R}_{pp} = \mathbb{Z}_p, \mathrm{C}_{pp} = \mathbb{G}$ *with* $\mathbb{G}$ *of order* p

$$\begin{aligned}&\text{Setup}: g', h' \xleftarrow{\$} \mathbb{G},\\ &\text{Com}(x; r) = (g'^x h'^r).\end{aligned}$$

Definition 7 (Pedersen Vector Commitment). $\mathrm{M}_{pp} = \mathbb{Z}_p^n, \mathrm{R}_{pp}, \mathrm{C}_{pp} = \mathbb{G}$ *with* $\mathbb{G}$ *of order* p

$$\begin{aligned}&\text{Setup}: \mathbf{g} = (g_1, ..., g_n), h' \xleftarrow{\$} \mathbb{G},\\ &\text{Com}(\mathbf{x} = (x_1, ..., x_n; r)) = h'^r \mathbf{g}^{\mathbf{x}} = h'^r \textstyle\prod_i g_i^{x_i} \in \mathbb{G}.\end{aligned}$$

Under the discrete logarithm assumption, the Pedersen vector commitment is perfectly hiding and computationally binding. If $r = 0$, then the commitment is binding but not hiding.

2.3 Zero-Knowledge Arguments of Knowledge

In our protocol, we construct zero-knowledge arguments of knowledge. A zero-knowledge proof of knowledge means a prover can convince a verifier that some statement holds without revealing any information about why it holds. An argument is a proof which holds when the prover is computationally bounded and certain computational hardness assumptions hold. The formal definitions as follows.

The arguments are consisted of three interactive algorithms (Setup, $\mathcal{P}$, $\mathcal{V}$), they are all run in probabilistic polynomial time. The three interactive algorithms are the common reference string generator Setup, the prover $\mathcal{P}$, and the verifier $\mathcal{V}$. The algorithm Setup produces a common reference string σ on input 1^λ. The transcript produced by $\mathcal{P}$ and $\mathcal{V}$ when interacting on inputs s and t is denoted by $tr \leftarrow < \mathcal{P}(s), \mathcal{V}(t) >$. We write $< \mathcal{P}(s), \mathcal{V}(t) >= b$, if verifier rejects then $b = 0$, if verifier accepts then $b = 1$.

We let $\mathcal{R} \subset \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ be a polynomial-time-decidable ternary relation. Given σ, the w is a witness for a statement u only if $(\sigma, x, w) \in \mathcal{R}$. We define the CRS-dependent language

$$\mathcal{L}_\sigma = \{x | \exists w : (\sigma, x, w) \in \mathcal{R}\}$$

as the set of statements x that have a witness w in the relation $\mathcal{R}$

Definition 8 (Argument of Knowledge). *The triple (*Setup, $\mathcal{P}$, $\mathcal{V}$*) is called an argument of knowledge for relation R if it satisfies the following two definitions.*

Definition 9 (Perfect completeness). *(*Setup, $\mathcal{P}$, $\mathcal{V}$*) has perfect completeness if for all non-uniform polynomial time adversaries* $\mathcal{A}$

$$\mathrm{P}\left[\begin{array}{l|l} (\sigma, u, w) \notin \mathcal{R} & \sigma \leftarrow \text{Setup}(1^\lambda) \\ \textit{or}\ \langle \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u)\rangle = 1 & (u, w) \leftarrow \mathcal{A}(\sigma) \end{array}\right] = 1.$$

Definition 10 (Computational Witness-Extended Emulation). *if for all deterministic polynomial time* $\mathcal{P}^*$ *there exists an expected polynomial time emulator* ε *such that for all pairs of interactive adversaries* $\mathcal{A}_1, \mathcal{A}_2$*, there exists a negligible function* $\mu(\lambda)$ *such that*

$$\left|\begin{array}{l} \mathrm{P}\left[\begin{array}{l|l} \mathcal{A}_1(tr = 1) & \begin{array}{l}\sigma \leftarrow \text{Setup}(1^\lambda),\\ (u, s) \leftarrow \mathcal{A}_2(\sigma),\\ tr \leftarrow \langle \mathcal{P}^*(\sigma, u, s),\\ \mathcal{V}(\sigma, u)\rangle\end{array}\end{array}\right] - \\ \mathrm{P}\left[\begin{array}{l|l} \begin{array}{l}\mathcal{A}_1(tr) = 1\\ \wedge(tr \text{ is accepting}\\ \implies (\sigma, u, w) \in \mathcal{R})\end{array} & \begin{array}{l}\sigma \leftarrow \text{Setup}(1^\lambda),\\ (u, s) \leftarrow \mathcal{A}_2(\sigma),\\ (tr, w) \leftarrow \varepsilon^{\mathcal{O}}(\sigma, u)\end{array}\end{array}\right]\end{array}\right| \leq \mu(\lambda)$$

where the oracle is given by $\mathcal{O} = \langle \mathcal{P}^*(\sigma, u, s), \mathcal{V}(\sigma, u)\rangle$*, and permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards, Then we say* (Setup, $\mathcal{P}$, $\mathcal{V}$) *has witness-extended emulation. Another way to define computational witness-extended emulation is restricting to non-uniform polynomial time adversaries* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*.*

In this paper, the witness-extended emulation is used by us to define knowledge soundness as used for example in [19] and defined in [20] [21]. If an adversary produces an argument which satisfies the verifier with some probability, then there exists an emulator which produces an identically distributed argument with the same probability, but also a witness. The s can be thought as the internal state of $\mathcal{P}^*$, including randomness. The emulator is allowed to rewind the interaction between the prover and the verifier to any action, and restore the prover with the same internal state, but with new randomness to the verifier. If $\mathcal{P}^*$ in state s makes a convincing argument, then ε can extract a witness, and therefor, we have an argument of knowledge of w such that $(\sigma, u, w) \in \mathcal{R}$.

Definition 11 (Public Coin). *An argument of knowledge* (Setup, $\mathcal{P}$, $\mathcal{V}$) *is called public coin if all messages sent from the verifier to the prover are chosen uniformly at random and independently of the prover's messages, i.e., the challenges correspond to the verifier's randomness* ρ*.*

If an argument of knowledge does not leak information about w apart from what can be deduced from the fact that $(\sigma, x, w) \in \mathcal{R}$, then it is zero knowledge. In this paper, we will prefer arguments of knowledge which have special honest-verifier zero-knowledge. This means that, given the challenge value of the verifier, the entire argument can be effectively simulated without the need for a witness.

Definition 12 (Perfect Special Honest-Verifier Zero-Knowledge). *A public coin argument of knowledge* (Setup, $\mathcal{P}$, $\mathcal{V}$) *is a perfect special honest verifier zero knowledge (SHVZK) argument of knowledge for* $\mathcal{R}$ *if there exists a probabilistic polynomial time simulator* $\mathcal{S}$ *such that for all pairs of interactive adversaries* $\mathcal{A}_1, \mathcal{A}_2$*:*

$$\mathrm{Pr}\left[\begin{array}{l|l} (\sigma, u, w) \in \mathcal{R} \text{ and } \mathcal{A}_1(tr) = 1 & \begin{array}{l}\sigma \leftarrow \text{Setup}(1^\lambda)\\ (u, w, \rho) \leftarrow \mathcal{A}_2(\sigma),\\ tr \leftarrow \langle \mathcal{P}(\sigma, u, w),\\ \mathcal{V}(\sigma, u; \rho)\rangle\end{array}\end{array}\right]$$

$$= \mathrm{Pr}\left[\begin{array}{l|l} (\sigma, u, w) \in \mathcal{R} \text{ and } \mathcal{A}_1(tr) = 1 & \begin{array}{l}\sigma \leftarrow \text{Setup}(1^\lambda),\\ (u, w, \rho) \leftarrow \mathcal{A}_2(\sigma),\\ tr \leftarrow \mathcal{S}(u, \rho)\end{array}\end{array}\right]$$

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

In this definition the adversary chooses a distribution over statements and witnesses but is still not able to distinguish between the simulated and the honestly generated transcripts for valid statements and witnesses.

Now we define range proofs. Range proofs are proofs that the prover knows an opening to a commitment, such that the committed value is in a certain range. Range proofs can be used to state that an integer commitment is to a positive number or that two homomorphic commitments to elements in a field of prime order will not overflow modulo the prime when they are added together.

Definition 13 (Zero-Knowledge Range Proof)**.** *Given a commitment scheme* (Setup, Com) *over a message space* M_{pp} *which is a set with a total ordering, a Zero-Knowledge Range Proof is a SHVZK argument of knowledge for the relation* $\mathcal{R}_{\mathrm{Range}}$*:*

$$\mathcal{R}_{\mathrm{Range}} : (pp, (\mathbf{com}, l, r), (x, \rho)) \in \mathcal{R}_{\mathrm{Range}} \leftrightarrow \mathbf{com} = \mathrm{Com}(x; \rho) \ \wedge \ (l \le x < r).$$

Theorem 1 (Lagrange's four-square theorem)**.** *Any non-negative integer can be composed of the square of four integers.*

The proof for Theorem 1 is given in [9], and [10] offers an algorithm for finding four such squares.

Theorem 2 (Positive Integer Decompose)**.** *If x is a positive integer, then $4x + 1$ can be written as the sum of three squares.*

The proof for Theorem 2 is given in [11], and [9] offers an efficient and simple algorithm for finding three such squares. Theorem 2 also means writing $4x + 1$ as the sum of three squares implies that x is non-negative.

2.4 Notation

Let p, q denote two prime numbers. Let $\mathbb{G}_p, \mathbb{G}_q$ denote cyclic groups of order p, q. Let $\mathbb{Z}_p, \mathbb{Z}_q$ denote the ring of integers modulo p, q. Let $\mathbb{Z}$ denote the set of integers. Let $\mathbb{N}$ denote the set of natural numbers. Let $\mathbb{G}_p^n$ and $\mathbb{Z}_p^n$ be vector spaces of dimension n over $\mathbb{G}_p$ and $\mathbb{Z}_p$ respectively. Let $\mathbb{Z}_p^*$ denote $\mathbb{Z}_p \setminus 0$. Generators of $\mathbb{G}_p$ are denoted by $g', h', v, u \in \mathbb{G}_p$. Generators of $\mathbb{G}_q$ are denoted by $g, h \in \mathbb{G}_q$. Group elements which represent commitments are capitalized and blinding factors are denoted by Greek letters, i.e. $C = g^a h^\alpha$ is a Pedersen commitment to a. In our protocol the $\mathbb{G}_p$ is known by $\mathcal{P}$ and $\mathcal{V}$, the $\mathbb{G}_q$ is $\mathcal{V}$'s secret cyclic group that $\mathcal{P}$ doesn't know at first, i.e. the $\mathcal{P}$ doesn't know q at first. $x \xleftarrow{\$} \mathbb{Z}_p^*$ means the uniform sampling of an element from $\mathbb{Z}_p^*$. In this paper, $\mathbf{a} \in \mathbb{F}^n$ is a vector with elements $a_1, ..., a_n \in \mathbb{F}$. For a number $c \in \mathbb{Z}_p$ and a vector $\mathbf{a} \in \mathbb{Z}_p^n$, we denote by $\mathbf{b} = c \cdot \mathbf{a} \in \mathbb{Z}_p^n$ the vector where $b_i = c \cdot a_i$. Let $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^n a_i \cdot b_i$ denote the inner product between two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$ and $\mathbf{a} \circ \mathbf{b} = (a_1 \cdot b_1, ..., a_n \cdot b_n) \in \mathbb{F}^n$ the Hadamard product or entry wise multiplication of two vectors. We define vector polynomials $p(x) = \sum_{i=0}^d \mathbf{p_i} \cdot x^i \in \mathbb{Z}_q^n[x]$ where each coefficient $\mathbf{p_i}$ is a vector in $\mathbb{Z}_q^n$. The inner product between two vector polynomials $l(x), r(x)$ is defined as

$$\langle l(x), r(x) \rangle = \sum_{i=0}^{d} \sum_{j=0}^{i} \langle \mathbf{l_i}, \mathbf{r_j} \rangle \cdot x^{i+j} \in \mathbb{Z}_q[x] \tag{1}$$

Let $\mathbf{a} \| \mathbf{b}$ denote the concatenation of two vectors: if $\mathbf{a} \in \mathbb{Z}_p^n$ and $\mathbf{b} \in \mathbb{Z}_p^m$ then $\mathbf{a} \| \mathbf{b} \in \mathbb{Z}_p^{n+m}$. For $0 \leqslant \ell \leqslant n$, we use Python notation to denote slices of vectors:

$$\mathbf{a}_{[:\ell]} = \mathbf{a}_{[0:\ell]} = (a_1, ..., a_\ell) \in \mathbb{F}^\ell,$$

$$\mathbf{a}_{[\ell:]} = \mathbf{a}_{[\ell:n]} = (a_{\ell+1}, ..., a_n) \in \mathbb{F}^{n-\ell}.$$

Let $t(x) = \langle \mathbf{l}(x), \mathbf{r}(x) \rangle$, then the inner product is defined such that $t(x) = \langle l(x), r(x) \rangle$ holds for all $x \in \mathbb{Z}_p$. For vectors $\mathbf{g} = (g_1, ..., g_n) \in \mathbb{G}_p^n$ and $\mathbf{a} \in \mathbb{Z}_p^n$ we write $C = \mathbf{g}^\mathbf{a} = \prod_{i=1}^n g_i^{a_i} \in \mathbb{G}_p$. For $m \in \mathbb{Z}_p^*$ we set $\vec{\mathbf{m}} = (1, 2, 3, ..., m) \in \mathbb{Z}_p^m$.

3 Efficient Range Proof Protocol

In this section, we will present our range proof protocol.

3.1 Four Integers Zero-Knowledge Proof

We now describe how to use the inner-product argument to construct a proof. The proof convinces the verifier that a commitment V contains a number v without revealing v.

In our proof, a Pedersen commitment V is an element in the same group $\mathbb{G}$ that is used to perform the inner product argument.

We let $v \in \mathbb{Z}_p$ and let $V \in \mathbb{G}_p$ be a Pedersen commitment to v which uses randomness r. The proof system proves the following relation:

$$\{(g, h, V \in \mathbb{G}_p, v, r \in \mathbb{Z}_p) : V = h^r g^v\} \tag{2}$$

Let $\mathbf{a} = (a_1, a_2, a_3, a_4) \in \mathbb{Z}_p^4$ be the vector containing the four numbers, $\mathbf{y} = \vec{\mathbf{4}} \cdot y$ these numbers satisfy the equation:

$$v = a_1^2 + a_2^2 + a_3^2 + a_4^2, \text{ so that } \langle \mathbf{a}, \mathbf{a} \rangle = v \tag{3}$$

The prover $\mathcal{P}$ uses a constant size vector commitment $A \in \mathbb{G}_p$ to commit to $\mathbf{a}$. To convince the verifier that v is a positive number, the prover must proves that he knows an opening $\mathbf{a} \in \mathbb{Z}_p^4$ of A and $v, r \in \mathbb{Z}_p$ such that $V = h^r g^v$ and $\langle \mathbf{a}, \mathbf{a} \rangle = v$. To construct this zero knowledge proof, the verifier should choose a random $z \in \mathbb{Z}_p$ and then the prover proves that

$$\langle \mathbf{a}, \mathbf{a} \rangle z^2 + \langle \mathbf{a} - \mathbf{a}, \mathbf{y} \rangle z = v z^2 \tag{4}$$

This equality can be re-written as:

$$\langle \mathbf{a} \cdot z - \mathbf{y}, \mathbf{a} \cdot z + \mathbf{y} \rangle = v z^2 - \delta(y) \tag{5}$$

The verifier can easily calculate that $\delta(y) = \langle \mathbf{y}, \mathbf{y} \rangle \in \mathbb{Z}_q$. So the problem of proving (3) holds is reduced to proving a single inner-product identity.

If the prover sends to the verifier the two vectors in the inner product in (5) then the verifier could check (5) itself by using the commitment V to v and be convinced that (3) holds. But these two vectors reveal the information of $\mathbf{a}$, so the prover cannot send them to verifier. To solve this problem, we use two additional blinding terms $\mathbf{s}_L, \mathbf{s}_R \in \mathbb{Z}_p^4$ to blind these vectors.

To prove the statement (2), $\mathcal{P}$ and $\mathcal{V}$ should obey the following protocol:

$$\mathcal{P} \text{ on input } v, r \text{ and uses the algorithm to compute :} \tag{6}$$
$$\mathbf{a} = [a_1, a_2, a_3, a_4] \text{ s.t.} \langle \mathbf{a}, \mathbf{a} \rangle = v \tag{7}$$
$$\alpha \xleftarrow{\$} \mathbb{Z}_p \tag{8}$$
$$A = h^\alpha \mathbf{g}^\mathbf{a} \mathbf{h}^\mathbf{a} \in \mathbb{G}_p \tag{9}$$
$$\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^4 \tag{10}$$
$$\rho \xleftarrow{\$} \mathbb{Z}_p \tag{11}$$
$$S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G}_p \tag{12}$$
$$\mathcal{P} \rightarrow \mathcal{V} : A, S \tag{13}$$
$$\mathcal{V} : y, z \xleftarrow{\$} \mathbb{Z}_p^* \tag{14}$$
$$\mathcal{V} \rightarrow \mathcal{P} : y, z \tag{15}$$

Here let us expound two linear vector polynomials $l(x), r(x)$ in $\mathbb{Z}_p^4[x]$, and a quadratic polynomial $t(x) \in \mathbb{Z}_p[x]$ as follows:

$$\begin{aligned} l(x) &= \mathbf{a}z - \mathbf{y} + \mathbf{s}_L x && \in \mathbb{Z}_p^4[x] \\ r(x) &= \mathbf{a}z + \mathbf{y} + \mathbf{s}_R x && \in \mathbb{Z}_p^4[x] \\ t(x) &= \langle l(x), r(x) \rangle = t_0 + t_1 \cdot x + t_2 \cdot x^2 && \in \mathbb{Z}_p[x] \end{aligned}$$

The constant terms of $l(x), r(x)$ are the inner product vectors in (5), and the blinding vectors $\mathbf{s}_R, \mathbf{s}_L$ make sure that the prover can publish $l(x), r(x)$ for one $x \in \mathbb{Z}_p^*$ and doesn't need to reveal any information of $\mathbf{a}$. The constant term of $t(x)$ which is denoted as t_0 is the result of the inner product in (5). The prover needs to convince the verifier that the following equation is true:

$$t_0 = v \cdot z^2 - \delta(y)$$

$$\mathcal{P} \text{ computes :} \tag{16}$$
$$\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_p \tag{17}$$
$$T_i = g^{t_i} h^{\tau_i} \in \mathbb{G}_p, \quad i = \{1, 2\} \tag{18}$$
$$\mathcal{P} \rightarrow \mathcal{V} : T_1, T_2 \tag{19}$$
$$\mathcal{V} : x \xleftarrow{\$} \mathbb{Z}_p^* \tag{20}$$
$$\mathcal{V} \rightarrow \mathcal{P} : x \tag{21}$$
$$\mathcal{P} \text{ computes :} \tag{22}$$
$$\mathbf{l} = l(x) = \mathbf{a}z - \mathbf{y} + \mathbf{s}_L x \in \mathbb{Z}_p^4 \tag{23}$$
$$\mathbf{r} = r(x) = \mathbf{a}z + \mathbf{y} + \mathbf{s}_R x \in \mathbb{Z}_p^4 \tag{24}$$
$$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p \tag{25}$$
$$\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 r \in \mathbb{Z}_p \tag{26}$$
$$\mu = \alpha z + \rho x \in \mathbb{Z}_p \tag{27}$$
$$\mathcal{P} \rightarrow \mathcal{V} : \tau_x, \mu, \hat{t}, \mathbf{l}, \mathbf{r} \tag{28}$$
$$\mathcal{V} \text{ check these equations and computes :} \tag{29}$$
$$P = A^z \cdot S^x \cdot \mathbf{g}^{-\mathbf{y}} \cdot \mathbf{h}^{\mathbf{y}} \in \mathbb{G}_p \tag{30}$$
$$P \stackrel{?}{=} h^\mu \cdot \mathbf{g}^\mathbf{l} \cdot \mathbf{h}^\mathbf{r} \in \mathbb{G}_p \tag{31}$$
$$g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} V^{z^2} g^{-\delta(y)} \cdot T_1^x \cdot T_2^{x^2} \in \mathbb{G}_p \tag{32}$$
$$\hat{t} \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p \tag{33}$$

Corollary 1 (Four-Integers zero-knowledge proof)**.** *The Four-Integers zero-knowledge proof presented in Section 3.1 has perfect completeness,perfect special honest verifier zero-knowledge, and computational witness extended emulation.*

Proof. The Four-Integers zero-knowledge proof is a special case of aggregated Logarithmic proof from Section 3.2 with $m = 1$. This is therefore a direct corollary of Theorem 3. □

3.2 Aggregating Logarithmic proofs

Benediky Bünz et al [5] stated a kind of proof for m values which is more efficient than conducting m individual range proofs, based on the protocol of it, we can also perform a proof for m values like [5] does. In this section, we show that this can be done with a slight modification to the protocol of proof in Section 3.1. The relation that we will proof as follows:

$$\{(g, h \in \mathbb{G}_p, \mathbf{V} \in \mathbb{G}_p^m; \mathbf{v}, \mathbf{r} \in \mathbb{Z}_p^m) : V_j = h^{r_j} g^{v_j} \ \forall\, j \in [1, m]\} \tag{34}$$

The prover is very similar to the prover for a simple zero-knowledge proof in Section 3.1, only with the following slight modifications. First, we set $\mathbf{y} = y \cdot \overrightarrow{\mathbf{M}}$, $|\overrightarrow{\mathbf{M}}| = 4 \cdot m$. In (6), the prover needs to compute $\mathbf{a} \in \mathbb{Z}_p^{4 \cdot m}$ so that:

$$\langle \mathbf{a}_{[(j-1)\cdot 4 : j \cdot 4]}, \mathbf{a}_{[(j-1)\cdot 4 : j \cdot 4]} \rangle = v_j \ \textit{for all } j \textit{ in } [1, m].$$

We accordingly change $l(x)$ and $r(x)$ as follows:

$$l(x) = \sum_{j=1}^{m} z \cdot j \left(\mathbf{0}^{(j-1)\cdot 4} \| \mathbf{a}_{[(j-1)\cdot 4 : 4j]} \| \mathbf{0}^{(m-j)\cdot 4} \right) - \mathbf{y} + \mathbf{s}_L \cdot x \tag{35}$$

$$r(x) = \sum_{j=1}^{m} z \cdot j \left(\mathbf{0}^{(j-1)\cdot 4} \| \mathbf{a}_{[(j-1)\cdot 4 : 4j]} \| \mathbf{0}^{(m-j)\cdot 4} \right) + \mathbf{y} + \mathbf{s}_R \cdot x \tag{36}$$

To compute τ_x, we adjust for the randomness of each commitment V_j, so that $\tau_x = \tau_1 \cdot x + \tau_2 \cdot x^2 + z^2 \sum_{j=1}^{m} j^2 \cdot r_j$. The verification check (28) needs to be adjusted to include all the V_j commitments.

$$g^{\hat{t}} h^{\tau_x} = \mathbf{V}^{z^2 \cdot \vec{\mathbf{m}} \circ \vec{\mathbf{m}}} g^{-\delta(y)} T_1^x T_2^{x^2} \tag{37}$$

Finally,we change the definition of A(8) as follows:

$$A = h^\alpha \prod_{j=1}^{m} \mathbf{g}_{[(j-1)4:4j]}^{j \cdot \mathbf{a}_{[(j-1)4:4j]}} \cdot \prod_{j=1}^{m} \mathbf{h}_{[(j-1)4:4j]}^{j \cdot \mathbf{a}_{[(j-1)4:4j]}} \tag{38}$$

Theorem 3 (Aggregate Logarithmic Proof)**.** *The aggregate logarithmic proof presented in Section 3.2 has perfect completeness, perfect honest verifier zero-knowledge and computational witness extended emulation.*

The proof for Theorem 3 is presented in Appendix B We can eliminate a trusted setup by using a hash function which needs to map from $\{0, 1\}^*$ to $\mathbb{G}_p \backslash 1$ to generate the public parameters $\mathbf{g}, \mathbf{h}, g, h$ from a small seed. [22] gives a way to built this kind of hash function. This protocol can also be transformed into a NIZK protocol by using the Fiat-Shamir heuristic.

3.3 Final Protocol

In this section, we will demonstrate how to proof that a secret number is within an arbitrary interval. The Theorem 2 is used in this section. The goal of our range proof protocol is convincing the verifier that the secret number v is in $[a, b]$. We will set $a, b \in \mathbb{Z}_p$, $\mathbf{d} = (d_1, d_2, d_3, d_4, d_5, d_6)$, the $d_1, d_2, d_3, d_4, d_5, d_6$ satisfy the following conditions:

$$d_1^2 + d_2^2 + d_3^2 = 4v - 4a + 1 = v_1,$$
$$d_4^2 + d_5^2 + d_6^2 = 4b - 4v + 1 = v_2 \in \mathbb{Z}_p$$

For the soundness of the final zero-knowledge protocol of this paper, the $V = g^v h^r$ dose't mod any integer because prover dose't know q at first. The whole protocol is similar to the special case of the aggregating logarithmic proofs from Section 3.2 when $m = 2, \mathbf{a} = \mathbb{Z}_p^6$ and we will proof the following relation:

$$\{(g, h \in \mathbb{G}_q, g' \in \mathbb{G}_p, \mathbf{V} \in \mathbb{G}_q^2) :$$
$$V_j = h^{r_j} g^{v_j} \ \forall\, j \in [1, 2], V = g^v h^r \wedge v \in [a, b]\}$$

The protocol is as follows:

$$\mathcal{P} \text{ on input } v \text{ computes :} \tag{39}$$
$$v_1 = 4v - 4a + 1, v_2 = 4b - 4v + 1 \ \in \mathbb{Z}_p \tag{40}$$
$$\mathbf{d} = (d_1, d_2, d_3, d_4, d_5, d_6) \ \in \mathbb{Z}_p^6 \tag{41}$$
$$\alpha \xleftarrow{\$} \mathbb{Z}_p \tag{42}$$
$$A = (h')^\alpha \prod_{j=1}^{2} \mathbf{g}_{[(j-1)3:3j]}^{j \cdot \mathbf{d}_{[(j-1)3:3j]}} \cdot \prod_{j=1}^{2} \mathbf{h}_{[(j-1)3:3j]}^{j \cdot \mathbf{d}_{[(j-1)3:3j]}} \in \mathbb{G}_p \tag{43}$$
$$\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^6 \tag{44}$$
$$\rho \xleftarrow{\$} \mathbb{Z}_p \tag{45}$$
$$S = (h')^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G}_p \tag{46}$$
$$\mathcal{P} \ \rightarrow \mathcal{V} : A, S \tag{47}$$
$$\mathcal{V} \ : y, z \xleftarrow{\$} \mathbb{Z}_p^* \tag{48}$$
$$\mathcal{V} \ \rightarrow \mathcal{P} : y, z, q \tag{49}$$

Here, as showed in Section 3.1, $t(x) = \langle l(x), r(x) \rangle = t_0 + t_1 \cdot x + t_2 \cdot x^2 \in \mathbb{Z}_q[x]$.

$$\mathcal{P} \text{ computes :} \tag{50}$$
$$r, \tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_q \tag{51}$$
$$r_1 = 4r, r_2 = -4r \ \in \mathbb{Z}_q \tag{52}$$
$$T_i = g^{t_i} h^{\tau_i} \in \mathbb{G}_q, i = \{1, 2\} \tag{53}$$
$$(t_1, t_2 \text{ can be computed directly without knowing } x)$$
$$\mathcal{P} \ \rightarrow \mathcal{V} : T_1, T_2 \tag{54}$$
$$\mathcal{V} \ : x \xleftarrow{\$} \mathbb{Z}_p^* \tag{55}$$
$$\mathcal{V} \ \rightarrow \mathcal{P} : x \tag{56}$$
$$\mathcal{P} \text{ computes :} \tag{57}$$
$$\mathbf{l} = z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{(j-1)3} \| \mathbf{d}_{[(j-1)3:3j]} \| \mathbf{0}^{(2-j)3}) - \mathbf{y} + \mathbf{s}_L x \tag{58}$$
$$\mathbf{l} \in \mathbb{Z}^6 \tag{59}$$
$$\mathbf{r} = z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{(j-1)3} \| \mathbf{d}_{[(j-1)3:3j]} \| \mathbf{0}^{(2-j)3}) + \mathbf{y} + \mathbf{s}_R x \tag{60}$$
$$\mathbf{r} \in \mathbb{Z}^6 \tag{61}$$
$$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle = t_0 + t_1 \cdot x + t_2 \cdot x^2 \in \mathbb{Z}_q \tag{62}$$
$$\tau_x = \tau_2 x^2 + \tau_1 x + z^2 \sum_{j=1}^{2} j^2 \cdot r_j \in \mathbb{Z}_q \tag{63}$$
$$\mu = \alpha z + \rho x \in \mathbb{Z}_p \tag{64}$$
$$\mathcal{P} \ \rightarrow \mathcal{V} : \tau_x, \mu, \hat{t}, \mathbf{l}, \mathbf{r} \tag{65}$$
$$\mathcal{V} \text{ computes and checks these equations :} \tag{66}$$
$$V_1 = \left(\frac{V}{g^a}\right)^4 \cdot g = \frac{V^4}{g^{4a-1}} = g^{4v-4a+1} h^{4r} = g^{v_1} h^{r_1} \in \mathbb{G}_q \tag{67}$$
$$V_2 = \left(\frac{g^b}{V}\right)^4 \cdot g = \frac{g^{4b+1}}{V^4} = g^{4b-4v+1} h^{-4r} = g^{v_2} h^{r_2} \in \mathbb{G}_q \tag{68}$$
$$\mathbf{V} = (V_1, V_2) \ \in \mathbb{G}_q^2 \tag{69}$$
$$P = A^z S^x \mathbf{g}^{-\mathbf{y}} \mathbf{h}^{\mathbf{y}} \ \in \mathbb{G}_p \tag{70}$$
$$P \overset{?}{=} (h')^\mu \mathbf{g}^{\mathbf{l}} \mathbf{h}^{\mathbf{r}} \ \in \mathbb{G}_p \tag{71}$$
$$g^{\hat{t}} h^{\tau_x} \overset{?}{=} \mathbf{V}^{z^2 \cdot (\vec{\mathbf{2}} \circ \vec{\mathbf{2}})} g^{-\delta(y)} T_1^x T_2^{x^2} \ \in \mathbb{G}_q \tag{72}$$
$$\hat{t} \overset{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \ \in \mathbb{Z}_q \tag{73}$$

Theorem 4 (Range Proof)**.** *The range proof presented in Section 3.3 has perfect completeness, perfect special honest verifier zero-knowledge, and computational witness extended emulation.*

The proof for Theorem 4 is presented in Appendix C.

4 Performance

4.1 Theoretical Performance

In Table 1 we show the communication cost of range proof protocols of Bulletproofs [5] and ours. The table 1 states the number of elements of group, ring, set which are applied in range proof protocol. We set $m = 2$ in the range proof protocols of Bulletproofs [5] to achieve a range proof that

TABLE 1
the number of elements

Protocol	$\mathbb{G}_p$	$\mathbb{Z}_p$	$\mathbb{G}_q$	$\mathbb{Z}_q$	$\mathbb{Z}$
Bulletproofs [5]	$6 + 2\log n$	5	0	0	0
Cupoof	2	4	2	3	12

proves $v \in [a, b]$. We compare our range proofs against Bulletproofs [5]. The table indicates that our range proof protocol has advantages when n is large. The proof size of Bulletproofs [5] grows by an additive logarithmic factor while ours is constant.

4.2 Realistic Performance

In order to evaluate the performance of our range proofs in practice, we provide a reference implementation in Python and employ the elliptic curve secp256k12 which has 128 bit security to structure cycle groups. We performed our experiments on an Intel i5-7500 CPU system throttled to 3.4 GHZ and we used a single thread. The performance is as practicable as we expectet. Table 2 shows the proof time, verify time and gates of range proofs under the different intervals. Figure 1 shows the line chart of proving time and the line chart of verification time (no including witness generation). Figure 2 shows the sizes of different interval range proofs.

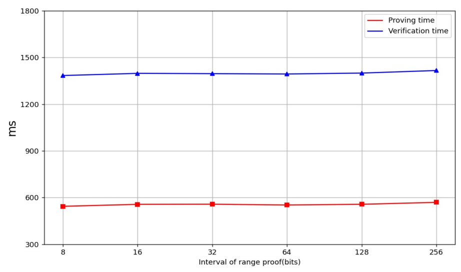

Fig. 1. Proof time

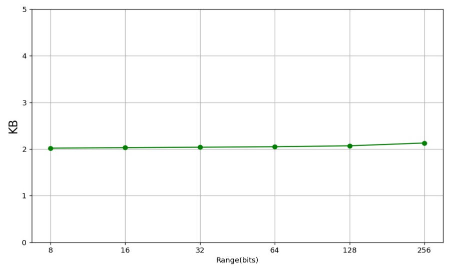

Fig. 2. Sizes for range proofs

5 Conclusion

In this paper, we construct an efficient range proof which proves that $v \in [a, b]$. Our scheme is based the work of Bulletproofs. In our protocol, by combining the Theorem 2 to Bulletproofs, we reduce the cost of communication to constants and make the computation complexity small which

TABLE 2
Range proofs:performance

Problem size	Gates	π Size (bytes)	Timing (ms) prove	verify
8 bit	6	2068	543	138.4
16 bit	6	2078	556	139.8
32 bit	6	2088	556	139.6
64 bit	6	2099	552	139.4
128 bit	6	2119	556	139.9
256 bit	6	2181	568	141.6

enhance the efficiency of zero-knowledge range proof. Our range proof has unconditional soundness and perfect zero-knowledge. The inadequacy of our range proof is that it is an interactive range proof. There are lots of room for improvement in our protocol. For example, we hope to make this range proof a non-interactive range proof in future.

Appendix A
A General Forking Lemma

Now we briefly describe the forking lemma of [19] that will be needed in the proofs.

Suppose that we have a $(2\mu + 1)$-move public-coin argument with μ challenges $x_1, ..., x_\mu$ in sequence. Let $n_i \geq 1$ for $1 \leq i \leq \mu$. Consider $\prod_{i=1}^{\mu} n_i$ accepting transcripts with challenges in the following tree format. The tree has $\prod_{i=1}^{\mu} n_i$ leaves and depth μ. The root of the tree is labeled with the statement. Each node of depth $i < \mu$ has exactly n_i children, each labeled with a distinct value of the ith challenge x_i.

This can be referred to as an $(n_1, ..., n_\mu)$-tree of accepting transcripts. Given a suitable tree of accepting transcripts, one can compute a valid witness for our inner-product argument, range proof, and argument for arithmetic circuit satisfiability. This is a natural generalization of special-soundness for Sigma-protocols, where $\mu = 1$ and $n = 2$. Combined with Theorem 3, this shows that the protocols have witness-extended emulation, and hence, the prover cannot produce an accepting transcript unless they know a witness. For simplicity in the following lemma, we assume that the challenges are chosen uniformly from $\mathbb{Z}_p$ where $|p| = \lambda$, but any sufficiently large challenge space would suffice. The success probability of a cheating prover scales inversely with the size of the challenge space and linearly with the number of accepting transcripts that an extractor needs. Therefor if $\prod_{i=1}^{\mu} n_i$ is negligible in 2^λ, then a cheating prover can create a proof that the verifier accepts with only negligible probability.

Theorem 5 (Forking Lemma, [19]). *Let $(\text{Setup}, \mathcal{P}, \mathcal{V})$ be a $(2k+1)$-move, public coin interactive protocol. Let χ be a witness extraction algorithm that succeeds with probability $1 - \mu(\lambda)$ for some negligible function $\mu(\lambda)$ in extracting a witness from an $(n_1, ..., n_k)$-tree of accepting transcripts in probabilistic polynomial time. Assume that $\prod_{i=1}^{k} n_i$ is bounded above by a polynomial in the security parameter λ. Then $(Setup, \mathcal{P}, \mathcal{V})$ has witness-extended emulation.*

We use the protocol of [5] to proof that our protocol has witness-extended emulation, so the theorem is slightly different than the one from [19]. We allow the extractor χ

to fail with negligible probability. Whenever this happens the Emulator ε as defined by Definition 10 also simply fails. Even with this slight modification this slightly stronger lemma still holds as ε overall still only fails with negligible probability.

Appendix B Proof of Theorem 3

Proof. Perfect completeness always holds as the fact that $t_0 = z^2 \cdot \langle \vec{\mathbf{m}} \circ \vec{\mathbf{m}}, \mathbf{v}\rangle - \delta(\mathbf{y}, \mathbf{y})$ for all valid witnesses. In order to prove perfect honest-verifier zero-knowledge, we construct a simulator which produces a distribution of proofs for a given statement $(g, h \in \mathbb{G}_p, \mathbf{g}, \mathbf{h} \in \mathbb{G}_p^{4\cdot m}, \mathbf{V} \in \mathbb{G}_p^m)$ which is indistinguishable from valid proofs produced by an honest prover interacting with an honest verifier. All proof elements and challenges according to the randomness supplied by the adversary from their respective domains were chosen by the simulator or directly computed by the simulator. S and T_1 are computed according to the verification equations, i.e.:

$$S = (h^{-\mu} \cdot \mathbf{g}^{-\mathbf{l}-\mathbf{y}} \cdot \mathbf{h}^{\mathbf{y}-\mathbf{r}} \cdot A^z)^{-x^{-1}}$$
$$T_1 = (g^{-\hat{t}-\delta(y)} \cdot h^{-\tau_x} \cdot \mathbf{V}^{z^2 \cdot \vec{\mathbf{m}} \circ \vec{\mathbf{m}}} \cdot T_2^{x^2})^{-x^{-1}}$$

Finally, according to the simulated witness $(\mathbf{l}, \mathbf{r})$ and the verifier's randomness, the simulator runs the inner-product argument. In the proof, all elements are either independently randomly distributed or their relationship is completely defined by the verification equation. Because we can successfully simulate the witness, the inner product argument remains zero knowledge. Thus leaking information about witness or revealing it does not change the zero-knowledge property of the overall protocol. The simulator which runs in time $O(\mathcal{V} + \mathcal{P}_{\text{InnerProduct}})$ is efficient.

We construct an extractor χ to prove computational witness extended emulation. The extractor χ uses $4 \cdot m$ different values of y, $m+2$ different values of z, and 3 different values of the challenge x to run prover. Additionally it invokes the extractor for the inner product argument on each of the transcripts. This results in $4 \cdot m \cdot (m+2) \cdot 3 \cdot O(1)$ valid proof transcripts.

For each transcript, in order to extract a witness $\mathbf{l}, \mathbf{r}$ to the inner product argument such that $h^{\mu}\mathbf{g}^{\mathbf{l}}\mathbf{h}^{\mathbf{r}} = P \wedge \langle \mathbf{l}, \mathbf{r}\rangle = \hat{t}$, the extractor χ first runs the extractor $\chi_{\text{InnerProduct}}$ for the inner-product argument. Using 2 valid transcripts and extracted inner product argument witnesses for different x challenges, we can compute linear combinations of (29) such that in order to compute $\alpha, \rho, \mathbf{a}, \mathbf{s}_L, \mathbf{s}_R$ such that $A = h^{\alpha} \prod_{j=1}^{m} \mathbf{g}_{[(j-1)4:4j]}^{j \cdot \mathbf{a}_{[(j-1)4:4j]}} \cdot \prod_{j=1}^{m} \mathbf{h}_{[(j-1)4:4j]}^{j \cdot \mathbf{a}_{[(j-1)4:4j]}}$, as well as $S = h^{\rho}\mathbf{g}^{\mathbf{s}_L}\mathbf{h}^{\mathbf{s}_R}$.

If the extractor can compute a different representation of A or S with any other set of challenges (x, y, z), then this yields a non-trivial discrete logarithm relation between independent generators $h, \mathbf{g}, \mathbf{h}$ which contradicts the discrete logarithm assumption.

Then using these representations of A and S and $\mathbf{l}$ and $\mathbf{r}$, we find that for all challenges x, y, and z

$$\mathbf{l} = l(x) = z \cdot \sum_{j=1}^{m} j \cdot (\mathbf{0}^{(j-1)4} \| \mathbf{a}_{[(j-1)4:4j]} \| \mathbf{0}^{(m-j)4}) - \mathbf{y} + \mathbf{s}_L x$$
$$\mathbf{r} = r(x) = z \cdot \sum_{j=1}^{m} j \cdot (\mathbf{0}^{(j-1)4} \| \mathbf{a}_{[(j-1)4:4j]} \| \mathbf{0}^{(m-j)4}) + \mathbf{y} + \mathbf{s}_R x$$

Once these equalities do not hold for all challenges and $\mathbf{l}, \mathbf{r}$ from the transcript, then we have two distinct representations of the same group element using a set of independent generators. This would be a non-trivial discrete logarithm relation.

With given y and z, we takes 3 transcripts for different x's and uses linear combinations of equation (33) to compute τ_1, τ_2, t_1, t_2 such that

$$T_1 = g^{t_1}h^{\tau_1} \ \wedge \ T_2 = g^{t_2}h^{\tau_2}.$$

Additionally we can compute a v, r such that $g^v h^r = \prod_{j=1}^{m} V_j^{z^2 \cdot j^2}$. Repeating this for m different z challenges, we can compute $(v_j, r_j)_{j=1}^{m}$ such that $g^{v_j}h^{r_j} = V_j \ \forall \ j \in [1, m]$. If there exists any transcript $\sum_{j=1}^{m} z^2 \cdot j^2 \cdot v_j - \delta(y) + t_1 \cdot x + t_2 \cdot x^2 \neq \hat{t}$ then this directly yields a discrete log relation between g and h, i.e. a violation of the binding property of Pedersen commitment. If not, then for all y, z challenges and 3 distinct challenges $X = x_j, j \in [1, 3]$:

$$\sum_{i=0}^{2} t_i X^i - p(X) = 0$$

with $t_0 = \sum_{j=1}^{m} z^2 \cdot j^2 \cdot v_j - \delta(y)$ and $p(X) = \sum_{i=0}^{2} p_i \cdot X^i = \langle l(X), r(X)\rangle$. Since the polynomial $t(X) - p(X)$ is of degree 2, but has at least 3 roots (each challenge x_j), it is necessarily the zero polynomial, i.e. $t(X) = \langle l(X), r(X)\rangle$.

Because this implies that $t_0 = p_0$,the following holds for all y, z challenges:

$$\begin{aligned}&\textstyle\sum_{j=1}^{m} z^2 \cdot j^2 \cdot v_j - \delta(y)\\ &=\\ &< z \cdot \sum_{j=1}^{m} j \cdot (\mathbf{0}^{(j-1)4} \| \mathbf{a}_{[(j-1)4:4j]} \| \mathbf{0}^{(m-j)4}),\\ &z \cdot \sum_{j=1}^{m} j \cdot (\mathbf{0}^{(j-1)4} \| \mathbf{a}_{[(j-1)4:4j]} \| \mathbf{0}^{(m-j)4}) >\\ &+ < z \cdot \sum_{j=1}^{m} j \cdot (\mathbf{0}^{(j-1)4} \| \mathbf{a}_{[(j-1)4:4j]} \| \mathbf{0}^{(m-j)4}), \mathbf{y} >\\ &- < \mathbf{y}, z \cdot \sum_{j=1}^{m} j \cdot (\mathbf{0}^{(j-1)4} \| \mathbf{a}_{[(j-1)4:4j]} \| \mathbf{0}^{(m-j)4}) + \mathbf{y} > \in \mathbb{Z}_p\end{aligned}$$

If this equality holds for $4 \cdot m$ distinct y challenges and 3 distinct z challenges, then we can infer the following:

$$v_j = \langle \mathbf{a}_j, \mathbf{a}_j \rangle \in \mathbb{Z}_p \ \forall \ j \in [1, m].$$

Because $g^{v_j}h^{r_j} = V_j \ \forall \ j \in [1, m]$, we have that $(\mathbf{v}, \mathbf{r})$ is valid witness for relation (30). The extractor rewinds the prover $3 \cdot (m+2) \cdot 4 \cdot m \cdot O(1)$ times. Extraction is efficient and the number of transcripts is polynomial in λ because $m = O(\lambda)$. Note that extraction either returns a valid witness or a discrete logarithm relation between independently chosen generators. In our definition, χ' is equal to χ but when χ extracts a discrete log relation, χ' fails. This would happen with at most negligible probability because of Discrete Log Relation assumption. Therefor, we can apply the forking lemma and see that computational witness emulation holds. $\square$

APPENDIX C PROOF OF THEOREM 4

Proof. Theorem 4 is the special case of Theorem 3. The proof for Theorem 4 is similar to the proof for Theorem 3, so the perfect completeness always holds as the fact that $t_0 = z^2 \cdot \langle \vec{\mathbf{2}} \circ \vec{\mathbf{2}}, \mathbf{v} \rangle - \delta(\mathbf{y}, \mathbf{y})$ for all valid witnesses. In order to prove perfect honest-verifier zero-knowledge, we construct a simulator which produces a distribution of proofs for a given statement $(g, h \in \mathbb{G}_q, h' \in \mathbb{G}_p, \mathbf{g}, \mathbf{h} \in \mathbb{G}_p^6, \mathbf{V} \in \mathbb{G}_q^2)$ which is indistinguishable from valid proofs produced by an honest prover interacting with an honest verifier. All operating procedures are the same as those in proof for Theorem 3. S and T_1 are computed according to the verification equations, i.e.:

$$S = ((h')^{-\mu} \cdot \mathbf{g}^{-\mathbf{l}-\mathbf{y}} \cdot \mathbf{h}^{\mathbf{y}-\mathbf{r}} \cdot A^z)^{-x^{-1}}$$
$$T_1 = (g^{-\hat{t}-\delta(y)} \cdot h^{-\tau_x} \cdot \mathbf{V}^{z^2 \cdot \vec{\mathbf{2}} \circ \vec{\mathbf{2}}} \cdot T_2^{x^2})^{-x^{-1}}$$

Finally, according to the simulated witness $(\mathbf{l}, \mathbf{r})$ and the verifier's randomness, the simulator runs the inner-product argument. Just as the proof for Theorem 3 shows, because we can successfully simulate the witness, the inner product argument remains zero knowledge. Thus leaking information about witness or revealing it does not change the zero-knowledge property of the overall protocol. The simulator which runs in time $O(\mathcal{V} + \mathcal{P}_{\text{InnerProduct}})$ is efficient. An extractor χ is used by us to prove computational witness extended emulation. The extractor χ uses 6 different values of y, 4 different values of z, and 3 different values of the challenge x to run prover. In addition, it calls the extractor for the inner product argument on each of the transcripts. This results in $72 \cdot O(1)$ valid proof transcripts.

For each transcript, in order to extract a witness $\mathbf{l}, \mathbf{r}$ to the inner product argument such that $(h')^{\mu}\mathbf{g}^{\mathbf{l}}\mathbf{h}^{\mathbf{r}} = P \wedge \langle \mathbf{l}, \mathbf{r} \rangle = \hat{t}$, the extractor χ first runs the extractor $\chi_{\text{InnerProduct}}$ for the inner-product argument. Using 2 valid transcripts and extracted inner product argument witnesses for different x challenges, we can compute linear combinations of (36) such that in order to compute $\alpha, \rho, \mathbf{d}, \mathbf{s}_L, \mathbf{s}_R$ such that $A = (h')^{\alpha} \prod_{j=1}^{2} \mathbf{g}_{[(j-1)3:3j]}^{j \cdot \mathbf{d}_{[(j-1)3:3j]}} \cdot \prod_{j=1}^{2} \mathbf{h}_{[(j-1)3:3j]}^{j \cdot \mathbf{d}_{[(j-1)3:3j]}}$, as well as $S = (h')^{\rho}\mathbf{g}^{\mathbf{s}_L}\mathbf{h}^{\mathbf{s}_R}$.

If the extractor can compute a different representation of A or S with any other set of challenges (x, y, z), then this yields a non-trivial discrete logarithm relation between independent generators $h', \mathbf{g}, \mathbf{h}$ which contradicts the discrete logarithm assumption.

Then using these representations of A and S and $\mathbf{l}$ and $\mathbf{r}$, we find that for all challenges x, y, and z

$$\begin{aligned} \mathbf{l} &= l(x) \\ &= z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{(j-1)3} \| \mathbf{d}_{[(j-1)3:3j]} \| \mathbf{0}^{(2-j)3}) - \mathbf{y} + \mathbf{s}_L x \\ \mathbf{r} &= r(x) \\ &= z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{(j-1)3} \| \mathbf{d}_{[(j-1)3:3j]} \| \mathbf{0}^{(2-j)3}) + \mathbf{y} + \mathbf{s}_R x \end{aligned}$$

Once these equalities do not hold for all challenges and $\mathbf{l}, \mathbf{r}$ from the transcript, then we have two distinct representations of the same group element using a set of independent generators. This would be a non-trivial discrete logarithm relation.

With given y and z, we takes 3 transcripts for different x's and uses linear combinations of equation (37) to compute τ_1, τ_2, t_1, t_2 such that

$$T_1 = g^{t_1} h^{\tau_1} \ \wedge \ T_2 = g^{t_2} h^{\tau_2}.$$

Also, we can compute a v, r such that $g^v h^r = \prod_{j=1}^{2} V_j^{z^2 \cdot j^2}$. Repeating this for 2 different z challenges, we can compute $(v_j, r_j)_{j=1}^{2}$ such that $g^{v_j} h^{r_j} = V_j \ \forall \ j \in [1, 2]$. If there exists any transcript $\sum_{j=1}^{2} z^2 \cdot j^2 \cdot v_j - \delta(y) + t_1 \cdot x + t_2 \cdot x^2 \neq \hat{t}$ then this directly yields a discrete log relation between g and h. If not, then for all y, z challenges and 3 distinct challenges $X = x_j, j \in [1, 3]$:

$$\sum_{i=0}^{2} t_i X^i - p(X) = 0$$

with $t_0 = \sum_{j=1}^{2} z^2 \cdot j^2 \cdot v_j - \delta(y)$ and $p(X) = \sum_{i=0}^{2} p_i \cdot X^i = \langle l(X), r(X) \rangle$. Since the polynomial $t(X) - p(X)$ is of degree 2, but has at least 3 roots (each challenge x_j), it is necessarily the zero polynomial, i.e. $t(X) = \langle l(X), r(X) \rangle$.

Because this implies that $t_0 = p_0$, the following holds for all y, z challenges:

$$\begin{aligned} &\sum_{j=1}^{2} z^2 \cdot j^2 \cdot v_j - \delta(y) \\ &= \\ &\langle z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{(j-1)3} \| \mathbf{d}_{[(j-1)3:3j]} \| \mathbf{0}^{(2-j)3}), \\ &z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{(j-1)3} \| \mathbf{d}_{[(j-1)3:3j]} \| \mathbf{0}^{(2-j)3}) \rangle \\ &+ \langle z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{(j-1)3} \| \mathbf{d}_{[(j-1)3:3j]} \| \mathbf{0}^{(2-j)3}), \mathbf{y} \rangle \\ &- \langle \mathbf{y}, z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{(j-1)3} \| \mathbf{d}_{[(j-1)3:3j]} \| \mathbf{0}^{(2-j)3}) + \mathbf{y} \rangle \in \mathbb{Z}_q \end{aligned}$$

If this equality holds for 6 distinct y challenges and 3 distinct z challenges, then we can infer the following:

$$v_1 = \langle \mathbf{d}_{[:3]}, \mathbf{d}_{[:3]} \rangle$$
$$v_2 = \langle \mathbf{d}_{[3:]}, \mathbf{d}_{[3:]} \rangle$$

The equations mean that $v_j \geq 0$ for all $j \in [1, 2]$. Because $g^{v_j} h^{r_j} = V_j \ \forall \ j \in [1, 2]$, we have that $(\mathbf{v}, \mathbf{r})$ is valid witness for relation (3). The extractor rewinds the prover $72 \cdot O(1)$

times. Extraction is efficient and the number of transcripts is polynomial in λ. Note that extraction either returns a valid witness or a discrete logarithm relation between independently chosen generators. Also as what we do in the proof for Theorem 3, χ' is equal to χ but when χ extracts a discrete log relation, χ' fails. This would happen with at most negligible probability because of Discrete Log Relation assumption. Therefor, we can apply the forking lemma and see that computational witness emulation holds. □

Acknowledgment

This research is partially supported by the Key Program of the Natural Science Foundation of China (No. 61772166) and the Natural Science Foundation of Zhejiang province of China (No. LZ17F020002).

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[2] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 459–474.

[3] S. Setty, S. Angel, and J. Lee, "Verifiable state machines: Proofs that untrusted services operate correctly," *SIGOPS Oper. Syst. Rev.*, vol. 54, no. 1, p. 40C46, Aug. 2020. [Online]. Available: https://doi.org/10.1145/3421473.3421479

[4] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," Cryptology ePrint Archive, Report 2018/046, 2018, https://eprint.iacr.org/2018/046.

[5] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 315–334.

[6] E. F. Brickell, D. Chaum, I. B. Damgård, and J. van de Graaf, "Gradual and verifiable release of a secret (extended abstract)," in *Advances in Cryptology — CRYPTO '87*, C. Pomerance, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 156–166.

[7] A. Chan, Y. Frankel, and Y. Tsiounis, "Easy come — easy go divisible cash," in *Advances in Cryptology — EUROCRYPT'98*, K. Nyberg, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 561–575.

[8] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Advances in Cryptology — EUROCRYPT 2000*, B. Preneel, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 431–444.

[9] M. O. Rabin and J. O. Shallit, "Randomized algorithms in number theory," *Communications on Pure and Applied Mathematics*, vol. 39, no. S1, pp. S239–S256, 1986. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160390713

[10] H. Lipmaa, "On diophantine complexity and statistical zero-knowledge arguments," in *Advances in Cryptology - ASIACRYPT 2003*, C.-S. Laih, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 398–415.

[11] J. Groth, "Non-interactive zero-knowledge arguments for voting," in *Applied Cryptography and Network Security*, J. Ioannidis, A. Keromytis, and M. Yung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 467–482.

[12] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 56–73.

[13] I. Teranishi and K. Sako, "k-times anonymous authentication with a constant proving cost," in *Public Key Cryptography - PKC 2006*, M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 525–542.

[14] J. Camenisch, R. Chaabouni, and a. shelat, "Efficient protocols for set membership and range proofs," in *Advances in Cryptology - ASIACRYPT 2008*, J. Pieprzyk, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 234–252.

[15] M. Belenkiy, "U-prove range proof extension," *Microsoft Research*, June 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/u-prove-range-proof-extension/

[16] C. Paquin and G. Zaverucha, "U-prove cryptographic specification v1.1 (revision 3)," *Microsoft*, December 2013. [Online]. Available: https://www.microsoft.com/en-us/research/publication/u-prove-cryptographic-specification-v1-1-revision-3/

[17] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn, "Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings," Cryptology ePrint Archive, Report 2019/099, 2019, https://eprint.iacr.org/2019/099.

[18] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 953, 2019.

[19] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *Advances in Cryptology – EUROCRYPT 2016*, M. Fischlin and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 327–357.

[20] J. Groth and Y. Ishai, "Sub-linear zero-knowledge argument for correctness of a shuffle," in *Advances in Cryptology – EUROCRYPT 2008*, N. Smart, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 379–396.

[21] Lindell, "Parallel coin-tossing and constant-round secure two-party computation," *Journal of Cryptology*, vol. 16, pp. 143–184, 03 2008.

[22] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology — ASIACRYPT 2001*, C. Boyd, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 514–532.