Cuproof: Range Proof with Constant Size

Cong Deng[a], Xianghong Tang[a], Lin You[b,*], Gengran Hu[b], Shuhong Gao[c]

[a]*School of Communication Engineering, Hangzhou Dianzi University, Hangzhou, China*
[b]*School of Cyberspace Security, Hangzhou Dianzi University, Hangzhou, China*
[c]*Department of Mathematical Sciences, Clemson University, SC, 29634, USA*

Abstract

Zero-knowledge proof is widely used in blockchains. For example, zk-SNARK is used by Zcash as its core technology in identifying transactions. Up to now, various range proofs have been proposed, and their efficiency and range-flexibility are enhanced. Bootle et al. used inner product method and recursion to make an efficient zero-knowledge proof. Then, Benediky Bünz et al. came up with an efficient zero-knowledge proof scheme called Bulletproofs which can convince the verifier that a secret number lies in $[0, 2^{\kappa} - 1]$. 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, all the communication cost, the proving time and the verification time are constant. When the interval of the range proof is large, our Cuproof would show much better.

Keywords: Blockchain, Zero-Knowledge Proof, Range Proof, Inner-product, Bulletproofs

1. Introduction

The blockchain-based cryptocurrencies enable peer-to-peer transactions and make sure that the transactions are valid. In the Bitcoin [1] system, all the

*Corresponding author.
Email addresses: mrdengcong@gmail.com (Cong Deng), tangxh@hdu.edu.cn (Xianghong Tang), mryoulin@gmail.com (Lin You), grhu@hdu.edu.cn (Gengran Hu), sgao@clemson.edu (Shuhong Gao)

transactions are recorded in a common ledger. So every one can check whether the transactions in the ledger are valid. The used hash function also makes the transactions in the blockchains 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 the senders and the receivers are all transparent, and it means that the anonymities of Bitcoin are not so strong.

In order to offset the disadvantage of Bitcoin, other cryptocurrencies using NIZK like Zcash [2] were proposed. The transactions between the shielded addresses are the special parts of Zcash. In these kind transactions, although the traders' addresses and the amount of the transactions are all covert, the validity of these transactions can still be checked because a kind of zero knowledge proof so called zk-SNARK is applied. 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 security of the 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.

In this paper, we combine the Lagrange integers theorem to Bulletproofs [5] to construct an range proof for arbitrary interval. In our scheme, the communication costs are 4 elements of $\mathbb{G}$, 3 elements of $\mathbb{Z}_n$ and 15 elements of $\mathbb{Z}$. Compared to the communication costs of Bulletproofs [5] which are logarithmic in κ where the κ is the exponent of upper limit value of the proving range $[0, 2^{\kappa}-1]$, the costs of ours are constant. So when the interval of the range proof is large, our scheme has more advantage than Bulletproofs in the communication cost.

1.1. Related work

There are lots of research work on range proof from the day when 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, this can allow a user with a discrete logarithm time to disclose a bit of information to another user so that any other user can verify the equations as they receive each bit. In 1998, Chan et al. [7] showed how to use the algorithm of [6] to verify the non-negative transaction amount and enhanced the algorithm in [6]. This is so called CTF proof because its security depends on modulus. To keep completeness, the order of the used group must be unknown. In 2000, Boudot [8] used the square numbers to build an effective range proof which is based on CTF. Using the Lagrange's four-square theorem [9], that is, any non-negative integer can be represented as the sum of squares of some 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 the squares of some three integers. The protocol given in [11] needed a trusted setup to generate RSA modulus or needed a prohibitively large modulus. Using Boneh-Boyen signature [12], Teranishi et al. [13] proposed many anonymous authentication methods in 2006. In 2008, Camenisch et al. [14] used signature method which relies on the security of the q-Strong Diffie-Hellman assumptions to construct a range proof. In 2014, Belenkiy [15] designed a scheme to extend the u-proof cryptographic specification [16] by making use of the membership proof of a set, this scheme can be used twice to compare the size of one committed value with other committed value, therefore it can make a range proof. In 2019, Maller et al. [17] presented Sonic which is the first potentially practical zk-SNARK with fully succinct verification for general arithmetic circuits with SRS, but the verson of Sonic enabling fully succinct verification still requires relatively high proof construction overheads. Later, Gabizon et al. [18] presented PLONK 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, Bünz et al. [5] improved the inner product method for zero certificate range proof and came up with a more efficient zero knowledge proof scheme called Bulletproofs. It can not only be applied to the range of certificates and reshuffle and other applications, but also doesn't need any trusted set up.

1.2. Contributions

Our scheme, called Cuproof for conveniency, is established on the techniques of Bulletproofs and Lagrange's three-square theorem given in [11]. Our protocol can make a range proof for arbitrary range. The argument of ours has low computation complexity. The main difference between Bulletproofs and ours is that Bulletproofs's communication costs[5] are logarithmic in κ, where κ is the exponent of upper limit value of the proving range $[0, 2^{\kappa} - 1]$, while the our costs are constant. The key is that we combine our following Theorem 2 into Bulletproofs [5]. Our Cuproof can satisfy the three security properties required for a secure 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 our range proof protocol Cuproof in detail. Some performance comparisons between Bulletproofs and 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

Groups of Unknown Order: In order to keep the soundness of our range proof, we use the RSA group $\mathbb{G}$ where the order of the group is unknown. The RSA group is generated by a trusted setup.

RSA Group. In the multiplicative group $\mathbb{G}$ of integers modulo n where n is the product of the large primes p and q. The hardness of computing the order of the group $\mathbb{G}$ is as the same as the hardness of factoring n.

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

$$\mathrm{P}\left[\begin{array}{ll} \mathbb{G} = \mathrm{Setup}\left(1^{\lambda}\right), & \\ g_1, ..., g_j \xleftarrow{\$} \mathbb{G}; & : \begin{array}{l} \exists a_i \neq 0, \\ \prod_{i=1}^{j} g_i^{a_i} = 1 \end{array} \\ a_1, ..., a_j \in \mathbb{Z}_n \leftarrow \mathcal{A}\left(g_1, ..., g_j\right) & \end{array}\right] \leqslant \mu(\lambda)$$

As Bünz et al. [5] stated, $\prod_{i=1}^{j} g_i^{a_i} = 1$ is a non trivial discrete log relation between $g_1, ..., g_j$. 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 $j \geq 1$.

Assumption 2 (Order Assumption)**.** *The Order Assumption holds for* Setup *if for any efficient adversary $\mathcal{A}$ there exists a negligible function $\mu(\lambda)$ such that:*

$$\mathrm{P}\left[\begin{array}{ll} & \mathbb{G} \xleftarrow{\$} \mathrm{Setup}(\lambda), \\ g_1 \neq 1 \wedge g_1^{a_1} = 1 : & (g_1, a_1) \xleftarrow{\$} \mathcal{A}(\mathbb{G}), \\ & \text{where } a_1 \neq 0 \in \mathbb{Z}_n, \\ & \text{and } g_1 \in \mathbb{G} \end{array}\right] \leq \mu(\lambda)$$

Lemma 1. *The Order Assumption implies the Discrete Log Relation Assumption.*

Proof. We show that if an adversary $\mathcal{A}_{Ord}$ breaks the Order Assumption, then we can construct $\mathcal{A}_{DL}$ which breaks the Discrete Log Relation Assumption

with overwhelming probability. In order to get a vector $(g_1, g_2, ..., g_j) \in \mathbb{G}^j$ and a vector $(a_1, a_2, ..., a_j) \in \mathbb{Z}_n^j$ such that $g_1^{a_1} \cdot g_2^{a_2} \cdots g_n^{a_n} = 1$ where $g_i \neq 1, a_i \neq 0$ and $i \in \{1, 2, \ldots, j\}$, we run $\mathcal{A}_{Ord}$ for n times and it will outputs $g_j \in \mathbb{G}$ and $a_j \in \mathbb{Z}$ such that $g_j^{a_j} = 1$ for $j = 1, \ldots, n$. And it follows $\prod_{j=1}^{n} g_j^{a_j} = 1$.

□

2.2. Commitments

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

Definition 2 (Homomorphic Commitments)**.** *A homomorphic commitment scheme is a non-interactive commitment scheme such that* $(\text{M}_{pp}, *), (\text{R}_{pp}, +)$*, and* $(\text{C}_{pp}, +)$ *are all abelian groups, and for all* $x_1, x_2 \in \text{M}_{pp}, r_1, r_2 \in \text{R}_{pp}$*, we have*

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

Here $(\text{M}_{pp}, *)$ can be additive or multiplicative. For ease of notation we drop pp in the subindex.

Definition 3 (Hiding Commitment)**.** *A commitment scheme is said to be hiding if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\mu(\lambda)$ *such that*

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

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

Definition 4 (Binding Commitment)**.** *A commitment scheme is said to be binding if for every PPT adversary $\mathcal{A}$ there exists a negligible function μ such that.*

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

where the probability is over Setup *and* $\mathcal{A}$*. If $\mu(\lambda) = 0$ then we say that 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 of these groups is implicitly dependent on the security parameter.

Definition 5 (Pedersen Commitment)**.** $\mathrm{M}_{pp}, \mathrm{R}_{pp} = \mathbb{Z}_n$ and $\mathrm{C}_{pp} = (\mathbb{G}, *)$ *being a multiplicative group.*

$$\mathrm{Setup} : g, h \xleftarrow{\$} \mathbb{G},$$
$$\mathrm{Com}(x; r) = (g^x h^r).$$

Definition 6 (Pedersen Vector Commitment)**.** $\mathrm{M}_{pp} = \mathbb{Z}_n^j, \mathrm{R}_{pp} = \mathbb{Z}_n$ and $\mathrm{C}_{pp} = (\mathbb{G}, *)$ *being a multiplicative group.*

$$\mathrm{Setup} : \mathbf{g} = (g_1, ..., g_j), h \xleftarrow{\$} \mathbb{G},$$
$$\mathrm{Com}(\mathbf{x} = (x_1, ..., x_j); r) = h^r \mathbf{g}^{\mathbf{x}} = h^r \prod_i g_i^{x_i} \in \mathbb{G}.$$

Under the discrete logarithm assumption, for the group $\mathbb{G}$, the Pedersen vector commitment is perfectly hiding and computationally binding. In the definition, r is picked at random.

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 hold without revealing any information of the knowledge. An argument is a proof which holds when the prover is computationally bounded and certain computational hardness assumptions hold. The formal definitions as follows.

Zero-knowledge arguments consist of three interactive algorithms (Setup, $\mathcal{P}$, $\mathcal{V}$) which run in probabilistic polynomial time. Setup is the common reference string generator, $\mathcal{P}$ is the prover and $\mathcal{V}$ is the verifier. The algorithm Setup produces a common reference string σ on inputting 1^λ. The transcript produced by $\mathcal{P}$ and $\mathcal{V}$ is denoted by $tr \leftarrow\!< \mathcal{P}(s), \mathcal{V}(t) >$ when they interact on the inputs s and t. We write $[\mathcal{P}(s), \mathcal{V}(t)] = b$ where $b = 0$ if verifier rejects, $b = 1$ if verifier accepts.

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

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

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

Definition 7 (Argument of Knowledge)**.** *The triple* (Setup, $\mathcal{P}$, $\mathcal{V}$) *is called an argument of knowledge for relation R if it satisfies both the Perfect Completeness and Computational Witness-Extended Emulation as defined in [5], respectively.*

Definition 8 (Perfect Special Honest-Verifier Zero-Knowledge)**.** *A public coin argument of knowledge* (Setup, $\mathcal{P}$, $\mathcal{V}$)*, as defined in [5], 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 every pair of interactive adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$:*

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

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

where ρ is the public coin randomness used by the verifier. The "transcript" can be simulated by S without knowing w.

In this definition the adversary chooses a distribution over statements and witnesses, but the adversary 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 the proofs that the prover knows an opening to a commitment in which the committed value is in a certain range. Range proofs can be used to state that an integer commitment is for a positive number or when two homomorphic commitments are added together, it will not overflow when they are taken modulo the given prime, and these two homomorphic commitments are the commitments to the elements in a prime field.

Definition 9 (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* $\mathcal{R}_{\text{Range}}$:

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

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

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

Theorem 2 (Lagrange's three-square theorem)**.** *If x is a positive integer, then $4x + 1$ can be written as the sum of three integer squares.*

The proof for Theorem 2 is given in [11], and [9] offered 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 $[N]$ denote the set $\{1, ..., N-1\}$. Let p and q denote two prime numbers. Let $\mathbb{G}$ denote the multiplicative group of integers modulo n, where n is the product of p and q, i.e. $\mathbb{G}$ is a RSA group. Let $\mathbb{Z}_n$ denote the ring of integers modulo n. Let $\mathbb{Z}$ denote the set of all integers. Let $\mathbb{G}^j$ and $\mathbb{Z}_n^j$ be vector spaces of dimension j over $\mathbb{G}$ and $\mathbb{Z}_n$, respectively. Let $\mathbb{Z}_n^*$ denote $\mathbb{Z}_n \backslash \{0\}$. Group elements which represent commitments are capitalized. For example, $C = g^a h^\alpha$ is a Pedersen commitment to a for $g, h \in \mathbb{G}$. $x \xleftarrow{\$} \mathbb{Z}_n^*$ means the uniform sampling of an element from $\mathbb{Z}_n^*$. In this paper, $\mathbf{a} \in \mathbb{F}^j$ is a vector with elements $a_1, ..., a_j \in \mathbb{F}$. For an element $c \in \mathbb{Z}_n$ and a vector $\mathbf{a} \in \mathbb{Z}_n^j$, we denote by $\mathbf{b} = c \cdot \mathbf{a} \in \mathbb{Z}_n^j$ the vector with $b_i = c \cdot a_i$. For the two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^j$, let $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{j} a_i \cdot b_i$ denote the inner product and $\mathbf{a} \circ \mathbf{b} = (a_1 \cdot b_1, ..., a_j \cdot b_j) \in \mathbb{F}^j$ the Hadamard product, respectively. We define vector polynomials $\mathscr{P}(x) = \sum_{i=0}^{d} \mathbf{p_i} \cdot x^i \in \mathbb{Z}^j[x]$ where each coefficient $\mathbf{p_i}$ is a vector in $\mathbb{Z}^j$. The inner product between two vector polynomials $l(x)$ and $r(x)$ is defined as

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

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

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

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

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 $x \in \mathbb{Z}_n$. For vectors $\mathbf{g} = (g_1, ..., g_j) \in \mathbb{G}^j$ and $\mathbf{a} \in \mathbb{Z}_n^j$ we write $C = \mathbf{g}^\mathbf{a} = \prod_{i=1}^{j} g_i^{a_i} \in \mathbb{G}$. For $1 \leq u$ we set $\vec{\mathbf{u}} = (1, 2, 3, ..., u) \in \mathbb{Z}^u$.

3. Efficient Range Proof Protocol

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

3.1. Four Integer Zero-Knowledge Proof

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

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

We let $v \in \mathbb{Z}_n$, and an element $V \in \mathbb{G}$ be a Pedersen commitment to v which uses a random number r. The proof system proves the following relation:

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

Choose $\mathbf{a} = (a_1, a_2, a_3, a_4) \in \mathbb{Z}_n^4$ such that

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

Let $y \in \mathbb{Z}_n^*$ and $\mathbf{y} = \vec{\mathbf{4}} \cdot y \in \mathbb{Z}^4$. The prover $\mathcal{P}$ uses an element in $\mathbb{G}$ to generate a commitment to the vector $\mathbf{a}$. To convince $\mathcal{V}$ that v be a positive number, the prover must prove that he knows an opening $\mathbf{a} \in \mathbb{Z}_n^4$ and $v, r \in \mathbb{Z}_n$ satisfying $V = h^r g^v$ and $\langle \mathbf{a}, \mathbf{a} \rangle = v$. To construct this zero knowledge proof, $\mathcal{V}$ should randomly choose $z \in \mathbb{Z}_n$, 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 $\delta(y) = \langle \mathbf{y}, \mathbf{y} \rangle \in \mathbb{Z}$. So the problem of proving Eq. (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 Eq. (5) then the verifier could check Eq. (5) itself by using the commitment V to v and be convinced that Eq. (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 $\mathbf{s}_L, \mathbf{s}_R \in \mathbb{Z}_n^4$ to blind these vectors.

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

$$
\begin{aligned}
&\mathcal{P} \text{ inputs } v, r \text{ and computes :} \\
&\quad \mathbf{a} = [a_1, a_2, a_3, a_4] \in \mathbb{Z}_n^4 \text{ s.t.} \langle \mathbf{a}, \mathbf{a} \rangle = v && (6) \\
&\quad \alpha \xleftarrow{\$} \mathbb{Z}_n && (7) \\
&\quad A = h^{\alpha} \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{a}} \in \mathbb{G} && (8) \\
&\quad \mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_n^4 && (9) \\
&\quad \rho \xleftarrow{\$} \mathbb{Z}_n && (10) \\
&\quad S = h^{\rho} \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G} && (11) \\
&\mathcal{P} \ \rightarrow \mathcal{V} : A, S && (12) \\
&\mathcal{V} \ : y, z \xleftarrow{\$} \mathbb{Z}_n^* && (13) \\
&\mathcal{V} \ \rightarrow \mathcal{P} : y, z && (14)
\end{aligned}
$$

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

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

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

$$t_0 = vz^2 - \delta(y)$$

$\mathcal{P}$ computes :

$$\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_n \tag{15}$$

$$T_i = g^{t_i} h^{\tau_i} \in \mathbb{G}, \quad i = \{1, 2\} \tag{16}$$

$$\mathcal{P} \ \rightarrow \mathcal{V} : T_1, T_2 \tag{17}$$

$$\mathcal{V} \ : x \xleftarrow{\$} \mathbb{Z}_n^* \tag{18}$$

$$\mathcal{V} \ \rightarrow \mathcal{P} : x \tag{19}$$

$\mathcal{P}$ computes :

$$\mathbf{l} = l(x) = \mathbf{a}z - \mathbf{y} + \mathbf{s}_L x \ \in \mathbb{Z}^4 \tag{20}$$

$$\mathbf{r} = r(x) = \mathbf{a}z + \mathbf{y} + \mathbf{s}_R x \ \in \mathbb{Z}^4 \tag{21}$$

$$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z} \tag{22}$$

$$\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 r \ \in \mathbb{Z} \tag{23}$$

$$\mu = \alpha z + \rho x \ \in \mathbb{Z} \tag{24}$$

$$\mathcal{P} \ \rightarrow \mathcal{V} : \tau_x, \mu, \hat{t}, \mathbf{l}, \mathbf{r} \tag{25}$$

$\mathcal{V}$ checks these equations and computes :

$$P = A^z \cdot S^x \cdot \mathbf{g}^{-\mathbf{y}} \cdot \mathbf{h}^{\mathbf{y}} \ \in \mathbb{G} \tag{26}$$

$$P \stackrel{?}{=} h^{\mu} \cdot \mathbf{g}^{\mathbf{l}} \cdot \mathbf{h}^{\mathbf{r}} \ \in \mathbb{G} \tag{27}$$

$$g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} V^{z^2} g^{-\delta(y)} \cdot T_1^x \cdot T_2^{x^2} \in \mathbb{G} \tag{28}$$

$$\hat{t} \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z} \tag{29}$$

Corollary 1 (Four-Integer Zero-Knowledge Proof)**.** *The Four-Integer 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-Integer Zero-Knowledge Proof is a special case of the aggregated logarithmic proof from the following Section 3.2 with $m = 1$, hence, it is a direct corollary of Theorem 3. □

3.2. Aggregating Logarithmic Proofs

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

$$\{(g, h \in \mathbb{G}, \mathbf{V} \in \mathbb{G}^m; \mathbf{v}, \mathbf{r} \in \mathbb{Z}_n^m) : V_j = h^{r_j} g^{v_j} \text{ for all } j \in [m]\} \tag{30}$$

The prover does the similar work as the prover does for a simple zero-knowledge proof in Section 3.1 with only the following slight modifications. First, we set $y \in \mathbb{Z}_n^*, \mathbf{y} = y \cdot \overrightarrow{\mathbf{4m}} \in \mathbb{Z}^{4m}$ and $|\overrightarrow{\mathbf{4m}}| = 4m$. In Eq. (6), the prover needs to find $\mathbf{a} \in \mathbb{Z}_n^{4m}$ so that

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

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

$$l(x) = \sum_{j=1}^{m} z \cdot j \left(\mathbf{0}^{4(j-1)} \| \mathbf{a}_{[4(j-1):4j]} \| \mathbf{0}^{4(m-j)} \right) - \mathbf{y} + \mathbf{s}_L \cdot x \tag{31}$$

$$r(x) = \sum_{j=1}^{m} z \cdot j \left(\mathbf{0}^{4(j-1)} \| \mathbf{a}_{[4(j-1):4j]} \| \mathbf{0}^{4(m-j)} \right) + \mathbf{y} + \mathbf{s}_R \cdot x \tag{32}$$

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

$$g^{\hat{t}} h^{\tau_x} = \mathbf{V}^{(z^2 \cdot \vec{\mathbf{m}} \circ \vec{\mathbf{m}})} g^{-\delta(y)} T_1^x T_2^{x^2} \tag{33}$$

Finally, we change the definition of A as follows:

$$A = h^{\alpha} \prod_{j=1}^{m} \mathbf{g}_{[4(j-1):4j]}^{j \cdot \mathbf{a}_{[4(j-1):4j]}} \cdot \prod_{j=1}^{m} \mathbf{h}_{[4(j-1):4j]}^{j \cdot \mathbf{a}_{[4(j-1):4j]}} \tag{34}$$

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. This protocol can also be transformed into a NIZK protocol by using the Fiat-Shamir heuristic.

3.3. Our Protocol

In this section, we will demonstrate how to prove that a secret number is within an arbitrary interval. The goal of our range proof protocol is to convince the verifier that the secret number v be in $[a, b]$. Based on Theorem 2, We can find $a, b \in \mathbb{Z}_n$ and $\mathbf{d} = (d_1, d_2, d_3, d_4, d_5, d_6) \in \mathbb{Z}_n^6$ such that the following conditions hold:

$$\begin{cases} d_1^2 + d_2^2 + d_3^2 = 4v - 4a + 1 = v_1 \in \mathbb{Z}, \\ d_4^2 + d_5^2 + d_6^2 = 4b - 4v + 1 = v_2 \in \mathbb{Z}. \end{cases} \tag{35}$$

The whole protocol is similar to the special case of the aggregating logarithmic proofs from Section 3.2 for $m = 2$ and $\mathbf{a} \in \mathbb{Z}_n^6$. In this protocol, we set $\delta(y) \in \mathbb{Z},\ \mathbf{y} \in \mathbb{Z}^6$. We will prove the following relations:

$$\{(g, h \in \mathbb{G}, \mathbf{V} \in \mathbb{G}^2) : V_j = h^{r_j} g^{v_j}\ \forall\ j \in \{1, 2\}, V = g^v h^r \wedge v \in [a, b]\} \tag{36}$$

The protocol is as follows:

$\mathcal{P}$ inputs $v, r,$ and computes :

$$v_1 = 4v - 4a + 1, v_2 = 4b - 4v + 1\ \in \mathbb{Z} \tag{37}$$

$$\mathbf{d} = (d_1, d_2, d_3, d_4, d_5, d_6)\ \in \mathbb{Z}_n^6 \text{ satisfying Eq. (35)} \tag{38}$$

$$\alpha \xleftarrow{\$} \mathbb{Z}_n \tag{39}$$

$$A = h^\alpha \prod_{j=1}^{2} \mathbf{g}_{[3(j-1):3j]}^{j \cdot \mathbf{d}_{[3(j-1):3j]}} \cdot \prod_{j=1}^{2} \mathbf{h}_{[3(j-1):3j]}^{j \cdot \mathbf{d}_{[3(j-1):3j]}} \in \mathbb{G} \tag{40}$$

$$\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_n^6 \tag{41}$$

$$\rho \xleftarrow{\$} \mathbb{Z}_n \tag{42}$$

$$S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G} \tag{43}$$

$$\mathcal{P}\ \rightarrow \mathcal{V} : A, S \tag{44}$$

$$\mathcal{V}\ : y, z \xleftarrow{\$} \mathbb{Z}_n^* \tag{45}$$

$$\mathcal{V}\ \rightarrow \mathcal{P} : y, z \tag{46}$$

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

$\mathcal{P}$ computes :

$$\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_n \tag{47}$$

$$r_1 = 4r, r_2 = -4r \ \in \mathbb{Z} \tag{48}$$

$$T_i = g^{t_i} h^{\tau_i} \in \mathbb{G}, i \in \{1, 2\} \tag{49}$$

(t_1, t_2 can be directly computed without knowing x)

$$\mathcal{P} \ \rightarrow \mathcal{V} : T_1, T_2 \tag{50}$$

$$\mathcal{V} \ : x \xleftarrow{\$} \mathbb{Z}_n^* \tag{51}$$

$$\mathcal{V} \ \rightarrow \mathcal{P} : x \tag{52}$$

$\mathcal{P}$ computes : (53)

$$\mathbf{l} = z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{3(j-1)} \| \mathbf{d}_{[3(j-1):3j]} \| \mathbf{0}^{3(2-j)}) - \mathbf{y} + \mathbf{s}_L x \tag{54}$$

$$\mathbf{l} \in \mathbb{Z}^6 \tag{55}$$

$$\mathbf{r} = z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{3(j-1)} \| \mathbf{d}_{[3(j-1):3j]} \| \mathbf{0}^{3(2-j)}) + \mathbf{y} + \mathbf{s}_R x \tag{56}$$

$$\mathbf{r} \in \mathbb{Z}^6 \tag{57}$$

$$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle = t_0 + t_1 \cdot x + t_2 \cdot x^2 \in \mathbb{Z} \tag{58}$$

$$\tau_x = \tau_2 x^2 + \tau_1 x + z^2 \sum_{j=1}^{2} j^2 \cdot r_j \in \mathbb{Z} \tag{59}$$

$$\mu = \alpha z + \rho x \in \mathbb{Z} \tag{60}$$

$$\mathcal{P} \ \rightarrow \mathcal{V} : \tau_x, \mu, \hat{t}, \mathbf{l}, \mathbf{r} \tag{61}$$

$\mathcal{V}$ computes and checks these equations :

$$V_1 = V^4 \cdot g^{-4a} \cdot g = g^{4v-4a+1} h^{4r} = g^{v_1} h^{r_1} \in \mathbb{G} \tag{62}$$

$$V_2 = g^{4b} \cdot V^{-4} \cdot g = g^{4b-4v+1} h^{-4r} = g^{v_2} h^{r_2} \in \mathbb{G} \tag{63}$$

$$\mathbf{V} = (V_1, V_2) \ \in \mathbb{G}^2 \tag{64}$$

$$P = A^z S^x \mathbf{g}^{-\mathbf{y}} \mathbf{h}^{\mathbf{y}} \ \in \mathbb{G} \tag{65}$$

$$P \stackrel{?}{=} h^{\mu} \mathbf{g}^{\mathbf{l}} \mathbf{h}^{\mathbf{r}} \ \in \mathbb{G} \tag{66}$$

$$g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} \mathbf{V}^{z^2 \cdot (\vec{\mathbf{z}} \circ \vec{\mathbf{2}})} g^{-\delta(y)} T_1^x T_2^{x^2} \ \in \mathbb{G} \tag{67}$$

$$\hat{t} \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \ \in \mathbb{Z} \tag{68}$$

Corollary 2 (Range Proof)**.** *The Range Proof presented in Section 3.3 has perfect completeness, perfect special honest verifier zero-knowledge, and computational witness extended emulation.*

Proof. The Range Proof is a special case of aggregated logarithmic proof from Section 3.2 with $m = 2,\ \mathbf{a} \in \mathbb{Z}_n^6$, hence it is a direct corollary of Theorem 3. □

4. Performance

4.1. Theoretical Performance

Table 1 shows the communication cost of zero-knowledge proof of our Cuproof. Table 2 shows the communication costs of Bulletproofs [5] and Cuproof, respectively. Table 2 states the numbers of the elements of the group, ring and set applied in the range proof protocols. We set $m = 2$ in the range proof protocol of Bulletproofs to achieve a range proof that can prove $v \in [a, b]$. We compare our range proof Cuproof to Bulletproofs. Table 2 indicates that our range proof protocol has some advantages over Bulletproofs when κ is large, where κ is the exponent of the upper limit value of the proving range $[0, 2^{\kappa} - 1]$ in Bulletproofs. The proof size of Bulletproofs [5] grows by an additive logarithmic factor while ours remains constant.

Table 1: the number of elements

Protocol	$\mathbb{G}$	$\mathbb{Z}_p$	$\mathbb{Z}$
This scheme	4	3	11

Table 2: the number of elements

Protocol	$\mathbb{G}$	$\mathbb{Z}_p$	$\mathbb{Z}$	$\mathbb{Z}_n$
Bulletproofs[5]	$6 + 2\log\kappa$	5	0	0
Cupoof	4	0	15	3

4.2. Practical Performance

In order to evaluate the practical performance of our Cuproof, we provide a reference implementation in Python. We set that the sizes of the two primes p and q are 128 bits. The prover uses the algorithms of [9] and [10] to generate the witnesses $\mathbf{a}$ and $\mathbf{d}$, and compute the $\mathbf{l}$ and $\mathbf{r}$. A Pedersen hash function over an RSA group whose modulo $n = p * q$ is benchmarked. We performed our experiments on our computer with an Intel i5-7500 CPU@3.4 GHZ and we used a single thread. The performance is as practicable as we expect. Table 3 shows the proving time, verification time and the gates of the range proofs under the different range (The final data is the average of the data we obtained by doing 10000 experiments). Figure 1 shows the line charts of the proving time and the verification time of the Four-Integer Zero-Knowledge Proofs (no including witness generation) respectively. Figure 2 shows the line charts of the proving time and the verification time of the Range Proofs (no including witness generation) respectively. No matter how big the range is, the proving time is near 22.5 ms and the verification time is near 16.5 ms. Figure 3 shows the sizes of different interval range proofs.

5. Conclusions

In this paper, we construct an efficient range proof scheme, called Cuproof, which can prove $v \in [a, b]$ without revealing v's actual value. Our scheme is based on the Bünz et al.'s work on Bulletproofs. In our protocol, by combining Theorem 2 into Bulletproofs, we reduce the communication cost to the constant sizes and make the computation complexity lower and enhance the efficiency of our zero-knowledge range proof. Our range proof has unconditional soundness

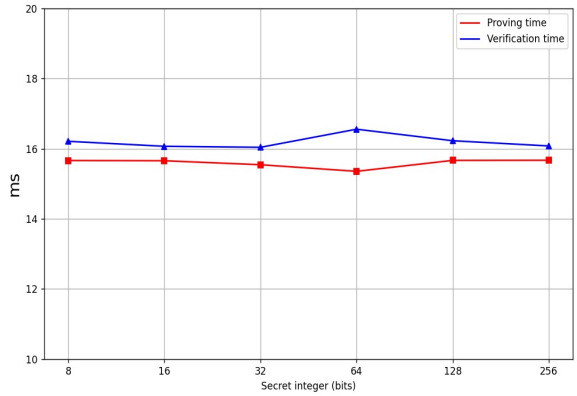

Figure 1: Four-Integer Zero-Knowledge Proof Time

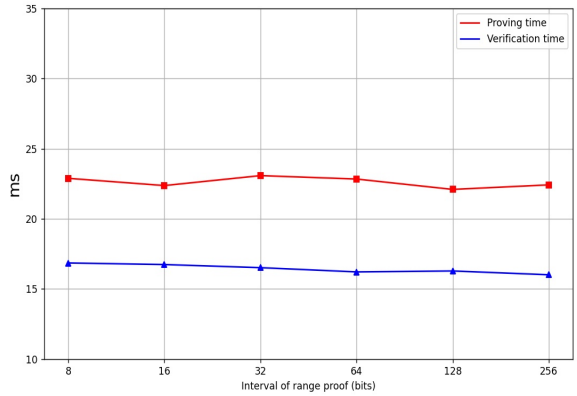

Figure 2: Range Proof Time

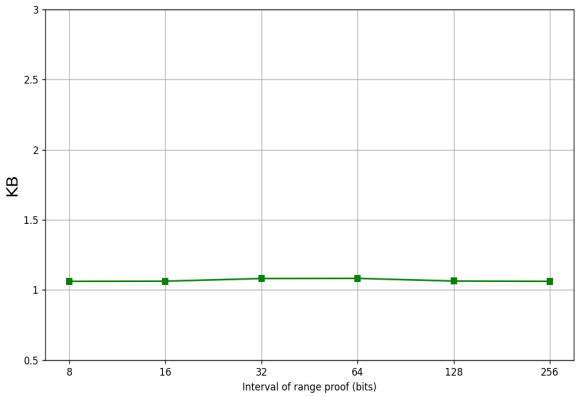

Figure 3: Sizes for range proofs

Table 3: Range proofs: performance

Range size	Gates	Proof Size	Timing (ms)	
		(bytes)	Prove	Verify
8 bit	6	1062	22.8	16.8
16 bit	6	1063	22.3	16.7
32 bit	6	1082	23.0	16.5
64 bit	6	1081	22.8	16.2
128 bit	6	1063	22.0	16.2
256 bit	6	1061	22.4	16.0

and perfect zero-knowledge. The inadequacy of our range proof is that it needs a trusted setup.

Acknowledgements

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

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Tech. rep., Manubot (2019).

[2] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, M. Virza, Zerocash: Decentralized anonymous payments from bitcoin, in: 2014 IEEE Symposium on Security and Privacy, 2014, pp. 459–474. doi:10.1109/SP.2014.36.

[3] S. Setty, S. Angel, J. Lee, Verifiable state machines: Proofs that untrusted services operate correctly, ACM SIGOPS Operating Systems Review 54 (1) (2020) 40–46.

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

[5] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, Bulletproofs: Short proofs for confidential transactions and more, in: 2018 IEEE Symposium on Security and Privacy, 2018, pp. 315–334. doi:10.1109/SP.2018.00020.

[6] E. F. Brickell, D. Chaum, I. B. Damgård, J. van de Graaf, Gradual and verifiable release of a secret (extended abstract), in: C. Pomerance (Ed.),

Advances in Cryptology — CRYPTO '87, Springer Berlin Heidelberg, 1988, pp. 156–166.

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

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

[9] M. O. Rabin, J. O. Shallit, Randomized algorithms in number theory, Communications on Pure and Applied Mathematics 39 (S1) 239–256.

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

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

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

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

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

[15] M. Belenkiy, U-prove range proof extension (June 2014).

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

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

[18] A. Gabizon, Z. J. Williamson, O. Ciobotaru, Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge, Cryptology ePrint Archive, Report 2019/953, https://eprint.iacr.org/2019/953 (2019).

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

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 $k_i\ (\geq 1) \in \mathbb{Z}$ for $1 \leq i \leq \mu$. Consider there are $\prod_{i=1}^{\mu} k_i$ accepting transcripts with challenges in the following tree format. The tree has $\prod_{i=1}^{\mu} k_i$ leaves and depth μ. The root of the tree is labeled with the statement. Each node of depth $i < \mu$ has exactly k_i children nodes, each child node is labeled with a distinct value of the ith challenge x_i.

The above structure can be referred to a $(k_1, ..., k_\mu)$-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 with $\mu = 1$ and $k_1 = 2$. Combined with Theorem 3, this shows that the protocols have a 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 $\mathbb{Z}_n$ with $|n| = \lambda$, but any sufficiently large challenge space is enough. 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. Therefore, if $\prod_{i=1}^{\mu} k_i$ is negligible in 2^λ, then a cheating prover can create a proof that the verifier accepts with only negligible probability.

Theorem 4 (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* $(k_1, ..., k_k)$*-tree of accepting transcripts in probabilistic polynomial time. Assume that* $\prod_{i=1}^{k} k_i$ *is bounded above by a polynomial in the security parameter* λ*. Then* $(Setup, \mathcal{P}, \mathcal{V})$ *has witness-extended emulation.*

Like Bulletproofs [5], Theorem 4 is slightly different from the lemma 1 of [19]. We allow the extractor χ to fail with a negligible probability. Whenever this happens, the emulator ε as defined by Definition 9 also simply fails. Even with this slight modification, this 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 that produces a distribution of proofs for a given statement $(g, h \in \mathbb{G},\ \mathbf{g}, \mathbf{h} \in \mathbb{G}^{4 \cdot m}, \mathbf{V} \in \mathbb{G}^m)$ which is indistinguishable from valid proofs produced by an honest prover interacting with an honest verifier. All the proof elements and the challenges according to the randomness supplied

by the adversary from their respective domains are chosen by the simulator or directly computed by the simulator. S and T_1 are computed according to the verification equations, i.e.:

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

According to the simulated witness $(\mathbf{l}, \mathbf{r})$ and the verifier's randomness, the simulator runs the inner-product argument. In the zero-knowledge 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 the leaking information about witness or revealing it does not change the zero-knowledge property of the overall protocol. The simulator is efficient because it runs in time $O(\mathcal{V} + \mathcal{P}_{\text{InnerProduct}})$.

We construct an extractor χ to prove computational witness extended emulation. The extractor χ uses $4m$ different values of y, $m+2$ different values of z and 3 different values of the challenge x to run prover algorithm. It additionally 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 the witnesses $\mathbf{l}$ and $\mathbf{r}$ 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. In order to compute $\alpha, \rho, \mathbf{a}, \mathbf{s}_L$ and $\mathbf{s}_R$ such that $A = h^{\alpha} \prod_{j=1}^{m} \mathbf{g}_{[4(j-1):4j]}^{j \cdot \mathbf{a}_{[4(j-1):4j]}} \cdot \prod_{j=1}^{m} \mathbf{h}_{[4(j-1):4j]}^{j \cdot \mathbf{a}_{[4(j-1):4j]}}$ and $S = h^{\rho}\mathbf{g}^{\mathbf{s}_L}\mathbf{h}^{\mathbf{s}_R}$, we can compute the linear combinations of the Eq. (27) by using two valid transcripts and extract the inner product argument witnesses for different x challenges.

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

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

challenges x, y, and z, we have

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

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

Given y and z, we takes 3 transcripts for different x and use linear combinations of Eq. (33) to compute τ_1, τ_2, t_1 and 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 set of v and 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 $g^{v_j} h^{r_j} = V_j, \ \forall \ j \in [m]$. 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 the y, z challenges and the 3 distinct challenges $X = x_j, j \in \{1, 2, 3\}$:

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

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

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

challenges:

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

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

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

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