entropy

Article

Cuproof: Range Proof with Constant Size

Cong Deng [1], Lin You [2,*], Xianghong Tang [1], Gengran Hu [2] and Shuhong Gao [3]

[1] School of Communication Engineering, Hangzhou Dianzi University, Hangzhou 310018, China; mrdengcong@gmail.com (C.D.); tangxh@hdu.edu.cn (X.T.)
[2] School of Cyberspace Security, Hangzhou Dianzi University, Hangzhou 310018, China; grhu@hdu.edu.cn
[3] Department of Mathematical Sciences, Clemson University, Clemson, SC 29634, USA; sgao@clemson.edu
* Correspondence: mryoulin@gmail.com

Abstract: Zero-Knowledge Proof is widely used in blockchains. For example, zk-SNARK is used in Zcash as its core technology to identifying transactions without the exposure of the actual transaction values. Up to now, various range proofs have been proposed, and their efficiency and range-flexibility have also been improved. Bootle et al. used the inner product method and recursion to construct an efficient Zero-Knowledge Proof in 2016. Later, Benediky Bünz et al. proposed an efficient range proof scheme called Bulletproofs, which can convince the verifier that a secret number lies in $[0, 2^{\kappa} - 1]$ with κ being a positive integer. By combining the inner-product and Lagrange's four-square theorem, we propose a range proof scheme called Cuproof. Our Cuproof can make a range proof to show that a secret number v lies in an interval $[a, b]$ with no exposure of the real value v or other extra information leakage about v. It is a good and practical method to protect privacy and information security. In Bulletproofs, the communication cost is $6 + 2\log\kappa$, while in our Cuproof, all the communication cost, the proving time and the verification time are of constant sizes.

Keywords: zero-knowledge proof; range proof; inner-product; Bulletproofs; blockchain

check for updates

Citation: Deng, C.; You, L.; Tang, X.; Hu, G.; Gao, S. Cuproof: Range Proof with Constant Size. *Entropy* **2022**, *24*, 334. https://doi.org/10.3390/e24030334

Academic Editor: Zouheir Rezki

Received: 27 January 2022
Accepted: 22 February 2022
Published: 25 February 2022

1. Introduction

The blockchain technology is the most well-known decentralized and tamper-proof information technology, and it can be applied to construct many different digital service systems or application platforms, such as digital currencies, supply systems and so on. Wu et al. [1] elaborated the intellectual cores of the blockchain-Internet of Things (BIoT). Fedorov et al. [2] stated how to apply blockchain technology to 5G. Cryptocurrencies were the first to bring the concept of blockchain into the world. The blockchain-based cryptocurrencies enable peer-to-peer transactions and make sure that the transactions are valid. In the Bitcoin [3] system, all the transactions are recorded in a public ledger, and everyone can check whether the transactions in the ledger are valid. The hash function used in the blockchains ensures that the transaction data cannot be tampered with. However, every coin has two sides. Despite its advantage, the transparency in Bitcoin also has a disadvantage. In a transaction of Bitcoin, the transaction data, the addresses of the senders and the receivers are almost transparent, and it means that Bitcoin cannot achieve anonymity and cannot provide the same level of privacy as paper cash.

In order to offset the disadvantages that exist in Bitcoin, people have start to think about using zero-knowledge proof to protect the privacy of blockchain users, because a zero-knowledge proof is a cryptographic protocol that has strong privacy protection function. In [4], Sun et al. showed how zero-knowledge proof technology is applied to the blockchain. There are lots of blockchain-based cryptocurrencies using range proofs [5,6] or zk-SNARKs [7–10] such as Zcash [11]. The transactions between the shielded addresses are what makes Zcash special. In these transactions, although the traders' addresses and the amount of the transactions are all covert, the validity of these transactions can still be checked because zk-SNARKs have been applied. According to the property of protecting anonymity, more and more cryptocurrencies apply range proof as a tool to avoid the disclosure of users' information.

In 2018, Bünz et al. proposed a type of range proof that is called Bulletproofs [5]. The efficiency of Bulletproofs is particularly well suited for the blockchains. However, its communication cost, which is $6 + 2\log\kappa$, grows with larger κ. In this paper, we combine the Lagrange's four-square theorem with Bulletproofs [5] to construct a range proof for arbitrary interval $[a, b]$. In our scheme, the communication cost is 4 elements of $\mathbb{G}$ and 18 elements of $\mathbb{Z}$. Our Cuproof is a good method to protect uers' privacy and information security. For example, we can use the Cuproof scheme to declare that our age v lies in some interval. Because of the RSA assumption and discrete logarithm problems, it is hard for the verifier to get the secret v but still believe that v is in this interval.

1.1. Related Work

Nowadays, information security or privacy protection has become more and more important for each of us. A number of works on information security or privacy protect have been published. For example, Dong et al. [12] elaborated how overconfidence affects information security investment and information security performance. Range proof technology, a kind of zero-knowledge proof protocol, is a good method for protecting information security or privacy. There have been lots of research works on range proof since the first relevant algorithm of range proof was proposed. Brickel et al. [6] first stated the correlative algorithm of range proof in 1987. Its purpose was to send reliable values to other participants, which can allow a user with a discrete logarithm to disclose one 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. [13] showed how to use the algorithm given in [6] to verify the non-negative transaction amount and they also enhanced the algorithm in [6]. Their improved proof method was called CTF proof. In 2000, Boudot [14] used the square numbers to build an effective range proof which was based on CTF.

By using the Lagrange's four-square theorem [15], that is, any non-negative integer can be represented as the sum of squares of four integers, Lipmaa [16] proposed a proof of any range for the first time. In 2005, Groth [17] pointed out that if y is a non-negative integer, then $4y + 1$ could be represented as the sum of the squares of three integers. Using Boneh-Boyen signature [18], Teranishi et al. [19] proposed many anonymous authentication methods in 2006. In 2008, Camenisch et al. [20] used signature method that relies on the security of the q-Strong Diffie-Hellman assumptions to construct a range proof. In 2014, Belenkiy [21] designed a scheme to extend the u-proof cryptographic specification [22] 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 some other committed value, and therefore it can be used to construct a range proof.

Bootle et al. [23] made a step forward on the efficiency of space in Zero-Knowledge Proof based on discrete logarithms. They combined the inner product method and recursion to enhance the efficiency of Zero-Knowledge Proof. Based on this work, Bünz et al. [5] improved the inner product method for zero certificate range proof and proposed a more efficient Zero-Knowledge Proof scheme called Bulletproofs.

1.2. Contributions

Our scheme, called Cuproof for conveniency, is established on the techniques of Bulletproofs and Lagrange's three-square theorem given in [17]. Our protocol can be used to construct a range proof for arbitrary range. The argument of our scheme has low computation complexity. The main difference between Bulletproofs and ours is that Bulletproofs's communication cost [5] is logarithmic in κ, where κ is the exponent in the proving range $[0, 2^\kappa - 1]$, while the cost in our scheme is constant. The key is that we combine the following Theorem 2 with Bulletproofs. Our Cuproof satisfies the three security properties required for a secure Zero-Knowledge Proof: completeness, 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 that convinces the verifier that the prover knows the secret number v. In Section 3.2, we describe our range proof protocol Cuproof in detail. The performance comparisons among Bulletproofs, some other range proof protocols and Cuproof are shown in Section 4. Finally, the proof of Theorem 3 about our Cuproof will be given in Appendix A.

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. 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 $\mathbb{Z}$ denote the set of all integers. Let $\mathbb{Z}_n$ denote the ring of integers modulo n. 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 $\mathbb{Z}_n \setminus \{0\}$. Group elements which represent commitments are capitalized. For example, $C = g^a h^\alpha$ is a Pedersen commitment to a for $g, h \in \mathbb{G}$. $x \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ 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 $\mathbf{a} \circ \mathbf{b} = (a_1 \cdot b_1, ..., a_j \cdot b_j) \in \mathbb{F}^j$ be the Hadamard product, respectively. We define vector polynomials $\mathbf{P}(x) = \sum_{i=0}^{d} \mathbf{p_i} \cdot x^i \in \mathbb{Z}^j[x]$ where each coefficient $\mathbf{p_i}$ is a vector in $\mathbb{Z}^j$. The inner product between two vector polynomials $\mathbf{l}(x)$ and $\mathbf{r}(x)$ is defined as

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

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 $t(x) = \langle \mathbf{l}(x), \mathbf{r}(x) \rangle$ 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 $C = \mathbf{g}^{\mathbf{a}} = \prod_{i=1}^{j} g_i^{a_i} \in \mathbb{G}$. We set $\vec{\mathbf{u}} = (1, 2, 3, ..., u) \in \mathbb{Z}^u$ for $u \geq 1$.

2.2. Assumptions

Groups of Unknown Order: In order to achieve 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 the 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 the same as the hardness of factoring n.

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

$$\mathrm{P}\left[\begin{array}{lcl} \mathbb{G} = \mathrm{Setup}\left(1^{\lambda}\right), & & \\ g_1, ..., g_j \xleftarrow{\$} \mathbb{G}; & : & \begin{array}{l} \exists a_i \neq 0, \\ \prod_{i=1}^{j} g_i^{a_i} = 1 \end{array} \\ a_1, ..., a_j \in \mathbb{Z}_{2^{\lambda} n} \leftarrow \mathcal{A}\left(g_1, ..., g_j\right) & & \end{array}\right] \leq \mu(\lambda).$$

As Bünz et al. [5] stated, $\prod_{i=1}^{j} g_i^{a_i} = 1$ is a non trivial discrete log relation among $g_1, ..., g_j$. The discrete log relation assumption makes sure that an adversary cannot 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). *For any efficient adversary $\mathcal{A}$ there exists a negligible function $\mu(\lambda)$ such that:*

$$\mathrm{P}\left[\begin{array}{cl} & \mathbb{G} \xleftarrow{\$} \mathrm{Setup}(\lambda), \\ g_1 \neq 1 \wedge g_1^{a_1} = 1 : & (g_1, a_1) \xleftarrow{\$} \mathcal{A}(\mathbb{G}), \\ & \text{where } a_1 \neq 0 \in \mathbb{Z}_{2^{\lambda} n}, \\ & \text{and } g_1 \in \mathbb{G} \end{array}\right] \leq \mu(\lambda).$$

Lemma 1. *A PPT adversary $\mathcal{A}$ breaking Order Assumption can also break Discrete Log Relation Assumption easily.*

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 $(a_1, a_2, ..., a_j) \in \mathbb{Z}_{2^{\lambda n}}^{j}$ such that $g_1^{a_1} \cdot g_2^{a_2} \cdots g_j^{a_j} = 1$ 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 output $g_j \in \mathbb{G}$ and $a_j \in \mathbb{Z}$ such that $g_j^{a_j} = 1$ for $j = 1, \ldots, n$. It follows that $\prod_{j=1}^{n} g_j^{a_j} = 1$. □

2.3. Commitments

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

Definition 2 (Pedersen Commitment). *Let $\mathrm{M}_{pp} = \mathbb{Z}_n, \mathrm{R}_{pp} = \mathbb{Z}_{2^{\lambda} n}$ and $\mathrm{C}_{pp} = (\mathbb{G}, *)$ be a multiplicative group, the commitment is generated as follows:*

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

Definition 3 (Pedersen Vector Commitment). *Let $\mathrm{M}_{pp} = \mathbb{Z}_n^j, \mathrm{R}_{pp} = \mathbb{Z}_{2^{\lambda} n}$ and $\mathrm{C}_{pp} = (\mathbb{G}, *)$ being a multiplicative group, the commitment is generated as follows:*

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

2.4. Zero-Knowledge Arguments of Knowledge

A Zero-Knowledge Argument consists 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 the verifier rejects, $b = 1$ if the verifier accepts.

Let $\mathcal{R}$ 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 4 (Argument of Knowledge)**.** (Setup, $\mathcal{P}$, $\mathcal{V}$) *is called an argument of knowledge for relation $\mathcal{R}$ if it satisfies both the Perfect Completeness and the Computational Soundness.*
Perfect Completeness:

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

Computational Soundness:

$$\mathrm{P}\left[\begin{array}{l|l} (\sigma) \leftarrow \mathrm{Setup}(1^{\check{}}); & u \notin \mathcal{L}_\sigma \text{ and} \\ u \leftarrow A(\sigma) & \langle \mathcal{A}, \mathcal{V}(\sigma, u)\rangle = 1 \end{array}\right] \approx 0.$$

Definition 5 (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 $\mathcal{A}_2$, we have*

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

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

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

Definition 6 (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* $\mathcal{R}_{\mathrm{Range}}$:

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

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 [15] and an algorithm for finding four such squares was provided in [16].

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 [17], and ref. [15] 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.

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 contains 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 and λ is the security parameter.

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:

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

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 $y \in \mathbb{Z}^*_{2^\lambda n}$ 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 $\mathbf{a} \in \mathbb{Z}_n^4$ satisfying $\langle \mathbf{a}, \mathbf{a} \rangle = v$. To construct this zero knowledge proof, $\mathcal{V}$ should randomly choose $z \in \mathbb{Z}_{2^\lambda n}$, and then the prover proves that

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

This equality can be re-written as:

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

The verifier can easily calculate that $\delta(y) = \langle \mathbf{y}, \mathbf{y} \rangle \in \mathbb{Z}$. Hence, the problem of proving that Equation (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 Equation (5), then the verifier could check Equation (5) itself by using the commitment V to v and be convinced that Equation (3) holds. However, these two vectors reveal the information of $\mathbf{a}$ and so the prover cannot send them to the verifier. To solve this problem, we use two additional blinding terms $\mathbf{s}_L, \mathbf{s}_R \in \mathbb{Z}^4_{2^\lambda n}$.

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

$$\begin{aligned}
&\mathcal{P} \text{ inputs } v, r \text{ and computes :} && \\
&\quad \mathbf{a} = [a_1, a_2, a_3, a_4] \in \mathbb{Z}_n^4 \text{ s.t.} \langle \mathbf{a}, \mathbf{a} \rangle = v && (6)\\
&\quad \alpha \xleftarrow{\$} \mathbb{Z}_{2^\lambda n} && (7)\\
&\quad A = h^\alpha \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{a}} \in \mathbb{G} && (8)\\
&\quad \mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_{2^\lambda n}^4 && (9)\\
&\quad \rho \xleftarrow{\$} \mathbb{Z}_{2^\lambda n} && (10)\\
&\quad S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G} && (11)\\
&\mathcal{P} \rightarrow \mathcal{V} : A, S && (12)\\
&\mathcal{V} : y', z' \xleftarrow{\$} \mathbb{Z}_{2^\lambda n}^* && (13)\\
&\mathcal{V} \text{ computes } : y = g^{y'}, z = g^{z'} \in \mathbb{G} && (14)\\
&\mathcal{V} \rightarrow \mathcal{P} : y, z && (15)
\end{aligned}$$

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

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

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

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

$$\begin{aligned}
&\mathcal{P} \text{ computes :} && \\
&\quad \tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_{2^\lambda n} && (16)\\
&\quad T_i = g^{t_i} h^{\tau_i} \in \mathbb{G},\ \ i \in \{1, 2\} && (17)\\
&\mathcal{P} \rightarrow \mathcal{V} : T_1, T_2 && (18)\\
&\mathcal{V} : x' \xleftarrow{\$} \mathbb{Z}_{2^\lambda n}^* && (19)\\
&\mathcal{V} \text{ computes} : x = g^{x'} \in \mathbb{G} && (20)\\
&\mathcal{V} \rightarrow \mathcal{P} : x && (21)\\
&\mathcal{P} \text{ computes :} && \\
&\quad \mathbf{l} = \mathbf{l}(x) = \mathbf{a}z - \mathbf{y} + \mathbf{s}_L x \ \in \mathbb{Z}^4 && (22)\\
&\quad \mathbf{r} = \mathbf{r}(x) = \mathbf{a}z + \mathbf{y} + \mathbf{s}_R x \ \in \mathbb{Z}^4 && (23)\\
&\quad \hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z} && (24)\\
&\quad \tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 r \ \in \mathbb{Z} && (25)\\
&\quad \mu = \alpha z + \rho x \ \in \mathbb{Z} && (26)\\
&\mathcal{P} \rightarrow \mathcal{V} : \tau_x, \mu, \hat{t}, \mathbf{l}, \mathbf{r} && (27)
\end{aligned}$$

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

$$P = A^z \cdot S^x \cdot \mathbf{g}^{-\mathbf{y}} \cdot \mathbf{h}^{\mathbf{y}} \in \mathbb{G} \tag{28}$$

$$P \overset{?}{=} h^{\mu} \cdot \mathbf{g}^{\mathbf{l}} \cdot \mathbf{h}^{\mathbf{r}} \in \mathbb{G} \tag{29}$$

$$g^{\hat{t}} h^{\tau_x} \overset{?}{=} V^{z^2} g^{-\delta(y)} \cdot T_1^x \cdot T_2^{x^2} \in \mathbb{G} \tag{30}$$

$$\hat{t} \overset{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z} \tag{31}$$

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 soundness.*

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 type 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 some modification to the protocol of zero-knowledge proof in Section 3.1. The relation that we will prove is as follows:

$$\begin{aligned}\{(g, h \in \mathbb{G}, \mathbf{V} \in \mathbb{G}^m; \mathbf{v} \in \mathbb{Z}_n^m, \mathbf{r} \in \mathbb{Z}_{2^{\lambda} n}^m) :\\ V_j = h^{r_j} g^{v_j} \text{ for all } j \in [m]\}.\end{aligned} \tag{32}$$

The prover does similar work as the prover does for a simple zero-knowledge proof in Section 3.1 except for the following modifications. First, we set $y \in \mathbb{Z}_{2^{\lambda} n}^*$, $\mathbf{y} = y \cdot \overrightarrow{\mathbf{4m}} \in \mathbb{Z}^{4m}$ and $|\overrightarrow{\mathbf{4m}}| = 4m$. As in Equation (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 $\mathbf{l}(x)$ and $\mathbf{r}(x)$ as follows:

$$\mathbf{l}(x) = \sum_{j=1}^{m} z \cdot j \Big(\mathbf{0}^{4(j-1)} \| \mathbf{a}_{[4(j-1):4j]} \| \mathbf{0}^{4(m-j)} \Big) - \mathbf{y} + \mathbf{s}_L \cdot x \tag{33}$$

$$\mathbf{r}(x) = \sum_{j=1}^{m} z \cdot j \Big(\mathbf{0}^{4(j-1)} \| \mathbf{a}_{[4(j-1):4j]} \| \mathbf{0}^{4(m-j)} \Big) + \mathbf{y} + \mathbf{s}_R \cdot x \tag{34}$$

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 Equation (30) needs to be adjusted to include all the V_j commitments as follows

$$g^{\hat{t}} h^{\tau_x} = \mathbf{V}^{(z^2 \cdot \vec{\mathbf{m}} \circ \vec{\mathbf{m}})} g^{-\delta(y)} T_1^x T_2^{x^2} \tag{35}$$

Finally, we change the definition of A as follows:

$$A = h^{\alpha} \prod_{j=1}^{m} \mathbf{g}_{[4(j-1):4j]}^{j \cdot \mathbf{a}_{[4(j-1):4j]}} \cdot \prod_{j=1}^{m} \mathbf{h}_{[4(j-1):4j]}^{j \cdot \mathbf{a}_{[4(j-1):4j]}} \tag{36}$$

Theorem 3 (Aggregate Logarithmic Proof)**.** *The Aggregate Logarithmic Proof presented in Section 3.2 has perfect completeness, perfect honest verifier zero-knowledge, and computational soundness.*

The proof for Theorem 3 is presented in Appendix A. This protocol can also be transformed into a NIZK protocol by using the Fiat-Shamir heuristic.

3.3. Our Protocol: Cuproof

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 is in $[a,b]$. Based on Theorem 2, We can find $a,b \in \mathbb{Z}_n$ and $\mathbf{d} = (d_1,\dots,d_6) \in \mathbb{Z}_n^6$ such that the following conditions hold:

$$\begin{cases} d_1^2 + d_2^2 + d_3^2 = 4v - 4a + 1 = v_1 \in \mathbb{Z}, \\ d_4^2 + d_5^2 + d_6^2 = 4b - 4v + 1 = v_2 \in \mathbb{Z}. \end{cases} \tag{37}$$

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:

$$\begin{gathered} \{(g,h \in \mathbb{G}, \mathbf{V} = (V_1,V_2) \in \mathbb{G}^2) : \\ V_j = h^{r_j} g^{v_j}\ \forall\, j \in \{1,2\}, V = g^v h^r \wedge v \in [a,b]\} \end{gathered} \tag{38}$$

The protocol is as follows:

$$\begin{aligned}
&\mathcal{P} \text{ inputs } v, r \text{ and computes :} \\
&\quad v_1 = 4v - 4a + 1, v_2 = 4b - 4v + 1\ \in \mathbb{Z}, && (39)\\
&\quad \text{Finds } \mathbf{d} = (d_1,\dots,d_6) \text{ satisfying (37)} && (40)\\
&\quad \alpha \xleftarrow{\$} \mathbb{Z}_{2^\lambda n} && (41)\\
&\quad A = h^\alpha \prod_{j=1}^{2} \mathbf{g}_{[3(j-1):3j]}^{j\cdot \mathbf{d}_{[3(j-1):3j]}} \cdot \prod_{j=1}^{2} \mathbf{h}_{[3(j-1):3j]}^{j\cdot \mathbf{d}_{[3(j-1):3j]}} \in \mathbb{G} && (42)\\
&\quad \mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_{2^\lambda n}^6 && (43)\\
&\quad \rho \xleftarrow{\$} \mathbb{Z}_{2^\lambda n} && (44)\\
&\quad S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G} && (45)\\
&\mathcal{P} \ \rightarrow \mathcal{V} : A, S && (46)\\
&\mathcal{V} \ : y', z' \xleftarrow{\$} \mathbb{Z}_{2^\lambda n}^* && (47)\\
&\mathcal{V} \text{ computes } : y = g^{y'}, z = g^{z'} \in \mathbb{G} && (48)\\
&\mathcal{V} \ \rightarrow \mathcal{P} : y, z && (49)
\end{aligned}$$

Here, as shown in Section 3.1, we have

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

$\mathcal{P}$ computes :

$$\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_{2^\lambda n} \tag{50}$$

$$T_i = g^{t_i} h^{\tau_i} \in \mathbb{G}, i \in \{1,2\} \tag{51}$$

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

$$\mathcal{P} \ \rightarrow \mathcal{V} : T_1, T_2 \tag{52}$$

$$\mathcal{V} \ : x' \xleftarrow{\$} \mathbb{Z}^*_{2^\lambda n} \tag{53}$$

$$\mathcal{V} \text{ computes} : x = g^{x'} \in \mathbb{G} \tag{54}$$

$$\mathcal{V} \ \rightarrow \mathcal{P} : x \tag{55}$$

$$\mathcal{P} \text{ computes} : \tag{56}$$

$$\mathbf{l} = z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{3(j-1)} \| \mathbf{d}_{[3(j-1):3j]} \| \mathbf{0}^{3(2-j)}) - \mathbf{y} + \mathbf{s}_L x \in \mathbb{Z}^6. \tag{57}$$

$$\mathbf{r} = z \cdot \sum_{j=1}^{2} j \cdot (\mathbf{0}^{3(j-1)} \| \mathbf{d}_{[3(j-1):3j]} \| \mathbf{0}^{3(2-j)}) + \mathbf{y} + \mathbf{s}_R x \in \mathbb{Z}^6. \tag{58}$$

$$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle = t_0 + t_1 \cdot x + t_2 \cdot x^2 \in \mathbb{Z} \tag{59}$$

$$r_1 = 4r, r_2 = -4r \ \in \mathbb{Z} \tag{60}$$

$$\tau_x = \tau_2 x^2 + \tau_1 x + z^2 \sum_{j=1}^{2} j^2 \cdot r_j \in \mathbb{Z} \tag{61}$$

$$\mu = \alpha z + \rho x \in \mathbb{Z} \tag{62}$$

$$\mathcal{P} \ \rightarrow \mathcal{V} : \tau_x, \mu, \hat{t}, \mathbf{l}, \mathbf{r} \tag{63}$$

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

$$V_1 = V^4 \cdot g^{-4a} \cdot g = g^{4v-4a+1} h^{4r} = g^{v_1} h^{r_1} \in \mathbb{G} \tag{64}$$

$$V_2 = g^{4b} \cdot V^{-4} \cdot g = g^{4b-4v+1} h^{-4r} = g^{v_2} h^{r_2} \in \mathbb{G} \tag{65}$$

$$\mathbf{V} = (V_1, V_2) \ \in \mathbb{G}^2 \tag{66}$$

$$P = A^z S^x \mathbf{g}^{-\mathbf{y}} \mathbf{h}^{\mathbf{y}} \ \in \mathbb{G} \tag{67}$$

$$P \overset{?}{=} h^\mu \mathbf{g}^{\mathbf{l}} \mathbf{h}^{\mathbf{r}} \ \in \mathbb{G} \tag{68}$$

$$g^{\hat{t}} h^{\tau_x} \overset{?}{=} \mathbf{V}^{z^2 \cdot (\vec{\mathbf{2}} \circ \vec{\mathbf{2}})} g^{-\delta(y)} T_1^x T_2^{x^2} \ \in \mathbb{G} \tag{69}$$

$$\hat{t} \overset{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \ \in \mathbb{Z} \tag{70}$$

Theorem 4. *The protocol for range proof presented here above has perfect completeness, perfect special honest verifier zero-knowledge, and computational soundness.*

Proof. The protocol for range proof is a special case of the Aggregated Logarithmic Proof in Section 3.2 with $m = 2$ and $\mathbf{a} \in \mathbb{Z}_n^6$. Hence, this theory is a direct corollary of our Theorem 3. □

In short, we call our given protocol for range proof *Cuproof*.

4. 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 1024 bits. The prover uses the algorithms of [15,16] 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. Table 1 shows the comparison of our Cuproof with Bulletproofs and the three range proofs put out by Boudot [14], Lipmaa [16] and Groth et al. [24], respectively. It states that the communication cost is const while Bulletproof's communication cost is sublinear in n. Moreover, Cuproof is more efficient than the three range proof schemes proposed by Boudot [14], Lipmaa [16] and Groth et al. [24], respectively. Table 2 shows the proving time, verification time, and the gates of the range proofs under the different ranges (the final data is the average of the data we obtained by doing 10,000 experiments). Figure 1 shows the line charts of the proving time and the verification time of the Four-Integer Zero-Knowledge Proofs (no including the witness generation) for the secret of the different sizes, 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 large the range is, the proving time is near 170 ms and the verification time is near 447 ms. Figure 3 shows the proof sizes in different intervals and it demonstrates that the proof size is near 5500 bytes. Table 3 shows the proof sizes, proving time and the verification time for the interval range proofs on the different sizes, respectively.

Table 1. The comparison of Cuproof with Bulletproofs and the three range proofs respectively proposed by Boudot [14], Lipmaa [16] and Groth [24] for arithmetic circuit satisfiability with d the maximum size of the committed polynomials, m wires, SRS (the structured reference string) and n gates. The computational costs are measured in terms of the number of group elements and ring elements. $m\mathbb{G}$ means m group elements in the RSA group, ℓEx means ℓ group exponentiations. ℓ is the number of the elements that the known circuit inputs.

Scheme	**Universal SRS**	**Circle SRS**	**Size**	**$\mathcal{P}'$s Computation**	**$\mathcal{V}'$s Computation**
Bulletproofs [5]	$\frac{n}{2}\mathbb{G}$	–	$2\log_2(n) + 6\mathbb{G} + 5\mathbb{Z}_p$	$8nEx$	$4nEx$
Boudot [14]	$16\mathbb{G}$	–	$6\mathbb{G} + 19\mathbb{Z}$	$36Ex$	$38Ex$
Lipmaa [16]	$14\mathbb{G}$	–	$12\mathbb{G} + 18\mathbb{Z}$	$36Ex$	$36Ex$
Groth et al. [24]	–	$3n + m\mathbb{G}$	$3\mathbb{G}$	$4n + m - \ell Ex$	$3P + \ell Ex$
This work	$14\mathbb{G}$	–	$7\mathbb{G} + 15\mathbb{Z}$	$28Ex$	$38Ex$

Table 2. Asymptotic efficiency comparison of zero-knowledge proofs for arithmetic circuits. Here n is the number of gates. A white rhombus for post-quantum security denotes that it is feasibly post-quantum secure. A black rhombus for untrusted setup denotes that the scheme is updatable. DL stands for discrete log.

Scheme	**PQ?**	**Universal**	**Untrusted Setup**	**Assumption**	**Runtime**	
					Prover	**Verifier**
Bulletproofs [5]	◊	♦	♦	DL	$\mathcal{O}(n\log(n))$	$\mathcal{O}(n\log(n))$
Boudot [14]	◊	♦	◊	DL	$\mathcal{O}(n\log(2n))$	$\mathcal{O}(n\log(n+2))$
Lipmaa [16]	◊	♦	◊	DL	$\mathcal{O}(n\log(2n+4))$	$\mathcal{O}(n\log(2n))$
This work	◊	♦	◊	RSA	$\mathcal{O}(6\log(n))$	$\mathcal{O}(6\log(n))$

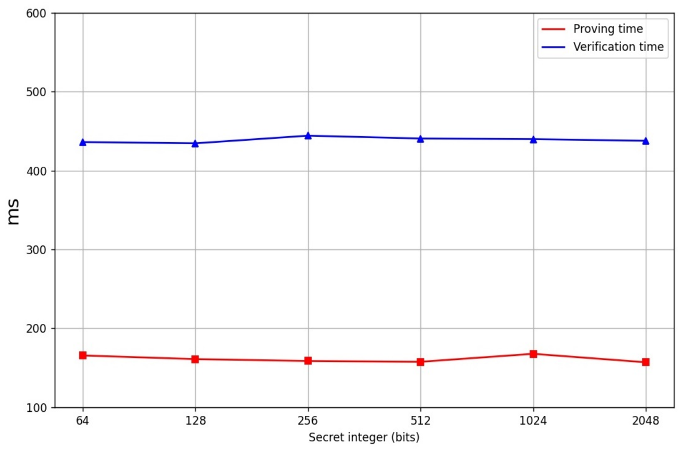

Figure 1. Four-integer zero-knowledge proof time.

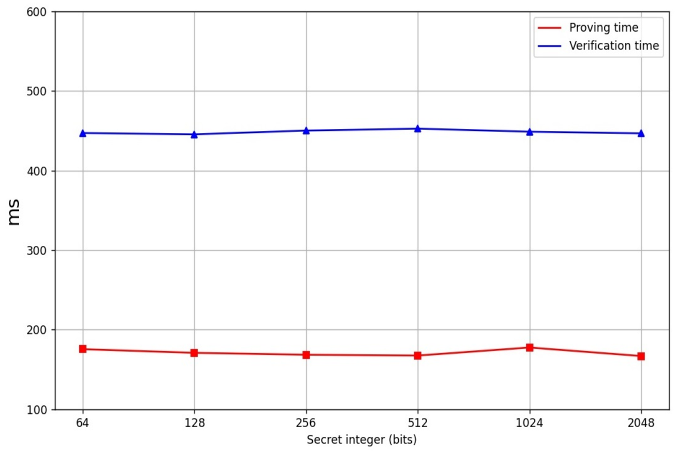

Figure 2. Range proof time.

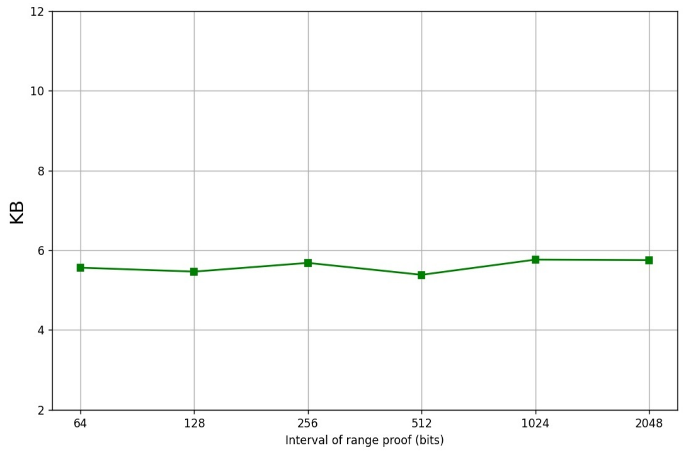

Figure 3. Sizes for range proofs.

Table 3. Our Cuproof's performances for the different sizes' range proofs.

Range Size	Gates	Proof Size	Timing (ms)	
		(Bytes)	Prove	Verify
64 bit	6	5561	175.4	446.2
128 bit	6	5462	170.8	444.6
256 bit	6	5681	168.4	452.3
512 bit	6	5382	167.4	450.7
1024 bit	6	5763	177.5	449.6
2048 bit	6	5751	166.8	447.8

5. Conclusions

In this paper, we construct a kind of range proof scheme *Cuproof*, which can prove $v \in [a, b]$ without revealing v's actual value. In our protocol, by combining Theorem 2 into Bulletproofs, we reduce the communication cost to the constant sizes, make the computation complexity lower, and enhance the efficiency of our range proof. Compared to the works [14,16], our zero-knowledge proof *Cuproof* is more efficient. The Cuproof can be applied to cryptocurrencies such as Monero [25] does and it can also be used for personal privacy protection. For example, in a biometric-based identity authentication system, we can use our Cuproof to prove that the Euclidean distance between the two biometric vectors respectively extracted during the registration phase and during authentication phase is within a preset threshold to identify a user's identity. Besides, we can also use Cuproof to prove that we are adults without exposing our true age. For instance, we can use Cuproof to prove that our age is lager than 18. However, a disadvantage of our range proof is that it still needs a trusted setup. Once the trusted setup is malicious, the secret number needs to be proved whether it has been leaked. In addition, because the security of Cuproof is based on the discrete logarithm problem, it is vulnerable to quantum attacks. Therefore, in our future work, we may use two groups to remove the trusted setting, one is a common group and the other is the verifier's secret group, that is, Equation (68) is checked in the common group and Equation (69) is checked in the verifier's secret group. In addition, in order to resist quantum attacks, we will consider to improve Cuproof based on an integer lattice. For example, we will use the elements in some integer lattice to replace the secret vectors of Cuproof.

Author Contributions: Writing, editing, original draft and software, C.D.; Reviewing, revising and innovative ideas, L.Y.; Methodology, X.T.; Reviewing and editing, G.H.; Formal analysis and Revising, S.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been partially supported by the Key Program of the Natural Science Foundation of Zhejiang Province of China (No. LZ17F020002) and the National Natural Science Foundation of China (No. 61772166).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We thank the anonymous referees for their valuable comments for the improvement of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. 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, that is,

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

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 does not change the zero-knowledge property of the overall protocol. The simulator is efficient because it runs in time $O(\mathcal{V} + \mathcal{P}_{\text{InnerProduct}})$. In the Aggregating Logarithmic Proofs, if the proof π passes successfully, then it means:

$$\langle \mathbf{a}_{[4(j-1):4j]}, \mathbf{a}_{[4(j-1):4j]} \rangle = v_j \text{ for all } j \in [m],$$
$$\xi(j,m) - \mathbf{y} + \mathbf{s}_L \cdot x = \mathbf{l}(x),$$
$$\xi(j,m) + \mathbf{y} + \mathbf{s}_R \cdot x = \mathbf{r}(x),$$
$$\tau_x = \tau_1 \cdot x + \tau_2 \cdot x^2 + z^2 \sum_{j=1}^{m} j^2 \cdot r_j,$$
$$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle,$$
$$\mu = \alpha z + \rho\, x.$$

Here, $\xi(j,m) = \sum_{j=1}^{m} z \cdot j \left(\mathbf{0}^{4(j-1)} \| \mathbf{a}_{[4(j-1):4j]} \| \mathbf{0}^{4(m-j)} \right)$.

If any of the above equations does not hold and the prover passes the verification as

$$A^z \cdot S^x \cdot \mathbf{g}^{-\mathbf{y}} \cdot \mathbf{h}^{\mathbf{y}} = h^{\mu} \cdot \mathbf{g}^{\mathbf{l}} \cdot \mathbf{h}^{\mathbf{r}},$$

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

then we have
$$h^{\alpha z + \rho x} \mathbf{g}^{\xi(j,m) - \mathbf{y} + \mathbf{s}_L \cdot x} \mathbf{h}^{\xi(j,m) + \mathbf{y} + \mathbf{s}_R \cdot x} = h^{\mu} \cdot \mathbf{g}^{\mathbf{l}} \cdot \mathbf{h}^{\mathbf{r}}$$

and
$$g^{\hat{t}} h^{\tau_x} = g^{(z^2 \sum_{j=1}^{m} j^2 v_j) - \delta(y) + x t_1 + x^2 t_2} h^{(z^2 \sum_{j=1}^{m} j^2 r_j) + \tau_1 x + \tau_2 x^2}.$$

By shifting the equations to one side we get:

$$g^{\hat{t} - (z^2 \sum_{j=1}^{m} j^2 v_j) - \delta(y) + x t_1 + x^2 t_2} h^{\tau_x - (z^2 \sum_{j=1}^{m} j^2 r_j) + \tau_1 x + \tau_2 x^2} = 1$$

and
$$h^{\alpha z + \rho x - \mu} \mathbf{g}^{\xi(j,m) - \mathbf{y} + \mathbf{s}_L \cdot x - \mathbf{l}} \mathbf{h}^{\xi(j,m) + \mathbf{y} + \mathbf{s}_R \cdot x - \mathbf{r}} = 1.$$

Because some of the above equations do not hold, one or more of the following situations must be encountered:

$$\hat{t} - (z^2 \sum_{j=1}^{m} j^2 v_j) - \delta(y) + x t_1 + x^2 t_2 \neq 0,$$

$$\tau_x - (z^2 \sum_{j=1}^{m} j^2 r_j) + \tau_1 x + \tau_2 x^2 \neq 0,$$

$$\alpha z + \rho x - \mu \neq 0,$$

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

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

This contradicts the Order Assumption and Discrete Log Relation Assumption. Therefore, our Cuproof has computational soundness. □

References

1. Tsang, Y.; Wu, C.; Ip, W.; Shiau, W.L. Exploring the intellectual cores of the blockchain–Internet of Things (BIoT). *J. Enterp. Inf. Manag.* **2021**, *34*, 1287–1317. [CrossRef]
2. Fedorov, I.R.; Pimenov, A.V.; Panin, G.A.; Bezzateev, S.V. Blockchain in 5G Networks: Perfomance Evaluation of Private Blockchain. In Proceedings of the 2021 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF), St. Petersburg, Russia, 31 May–4 June 2021; pp. 1–4. [CrossRef]
3. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: http://www.bitcoin.org/bitcoin.pdf (accessed on 21 February 2022).
4. Sun, X.; Yu, F.R.; Zhang, P.; Sun, Z.; Xie, W.; Peng, X. A survey on zero-knowledge proof in blockchain. *IEEE Netw.* **2021**, *35*, 198–205. [CrossRef]
5. Bünz, B.; Bootle, J.; Boneh, D.; Poelstra, A.; Wuille, P.; Maxwell, G. Bulletproofs: Short Proofs for Confidential Transactions and More. In Proceedings of the 2018 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 20–24 May 2018; pp. 315–334. [CrossRef]
6. Brickell, E.F.; Chaum, D.; Damgård, I.B.; van de Graaf, J. Gradual and Verifiable Release of a Secret (Extended Abstract). In *Advances in Cryptology—CRYPTO '87*; Pomerance, C., Ed.; Springer: Santa Barbara, CA, USA, 1987; pp. 156–166.
7. Gabizon, A.; Williamson, Z.J.; Ciobotaru, O. PLONK: Permutations over Lagrange-Bases for Oecumenical Noninteractive Arguments of Knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. Available online: https://eprint.iacr.org/2019/953 (accessed on 8 December 2021).
8. Maller, M.; Bowe, S.; Kohlweiss, M.; Meiklejohn, S. Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings. Cryptology ePrint Archive, Report 2019/099, 2019. Available online: https://eprint.iacr.org/2019/099 (accessed on 8 December 2021).
9. Setty, S.; Angel, S.; Lee, J. Verifiable state machines: Proofs that untrusted services operate correctly. *ACM SIGOPS Oper. Syst. Rev.* **2020**, *54*, 40–46. [CrossRef]
10. Ben-Sasson, E.; Bentov, I.; Horesh, Y.; Riabzev, M. Scalable, Transparent, and Post-Quantum Secure Computational Integrity. 2018. Available online: https://eprint.iacr.org/2018/046 (accessed on 15 August 2021).
11. Ben Sasson, E.; Chiesa, A.; Garman, C.; Green, M.; Miers, I.; Tromer, E.; Virza, M. Zerocash: Decentralized Anonymous Payments from Bitcoin. In Proceedings of the 2014 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 18–21 May 2014; pp. 459–474. [CrossRef]
12. Dong, K.; Lin, R.; Yin, X.; Xie, Z. How does overconfidence affect information security investment and information security performance? *J. Enterp. Inf. Syst.* **2021**, *15*, 474–491. [CrossRef]
13. Chan, A.; Frankel, Y.; Tsiounis, Y. Easy come—Easy go divisible cash. In *Advances in Cryptology—EUROCRYPT'98*; Nyberg, K., Ed.; Springer: Berlin/Heidelberg, Germany, 1998; pp. 561–575. [CrossRef]
14. Boudot, F. Efficient Proofs that a Committed Number Lies in an Interval. In *Advances in Cryptology—EUROCRYPT 2000*; Preneel, B., Ed.; Springer: Berlin/Heidelberg, Germany, 2000; pp. 431–444. [CrossRef]
15. Rabin, M.O.; Shallit, J.O. Randomized algorithms in number theory. *Commun. Pure Appl. Math.* **1986**, *39*, 239–256. [CrossRef]
16. Lipmaa, H. On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In *Advances in Cryptology—ASIACRYPT 2003*; Laih, C.S., Ed.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 398–415. [CrossRef]
17. Groth, J. Non-interactive Zero-Knowledge Arguments for Voting. In *Applied Cryptography and Network Security*; Ioannidis, J.; Keromytis, A.; Yung, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 467–482. [CrossRef]
18. Boneh, D.; Boyen, X. Short Signatures Without Random Oracles. In *Advances in Cryptology—EUROCRYPT 2004*; Cachin, C.; Camenisch, J.L., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 56–73. [CrossRef]
19. Teranishi, I.; Sako, K. k-Times Anonymous Authentication with a Constant Proving Cost. In *Public Key Cryptography—PKC 2006*; Yung, M.; Dodis, Y.; Kiayias, A.; Malkin, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 525–542. [CrossRef]
20. Camenisch, J.; Chaabouni, R.; Shelat, A. Efficient Protocols for Set Membership and Range Proofs. In *Advances in Cryptology—ASIACRYPT 2008*; Pieprzyk, J., Ed.; Springer : Berlin/Heidelberg, Germany, 2008; pp. 234–252. [CrossRef]
21. Belenkiy, M. U-Prove Range Proof Extension, 2014. Available online: https://www.microsoft.com/en-us/research/publication/u-prove-range-proof-extension/ (accessed on 5 September 2021).
22. Paquin, C.; Zaverucha, G. U-Prove Cryptographic Specification V1.1 (Revision 3), 2013. Available online: https://www.microsoft.com/en-us/research/publication/u-prove-cryptographic-specification-v1-1-revision-3/ (accessed on 8 December 2021).

23. Bootle, J.; Cerulli, A.; Chaidos, P.; Groth, J.; Petit, C. Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting. In *Advances in Cryptology—EUROCRYPT 2016*; Fischlin, M., Coron, J.S., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 327–357. [CrossRef]
24. Groth, J. On the Size of Pairing-Based Non-interactive Arguments. In *Advances in Cryptology—EUROCRYPT 2016*; Fischlin, M.; Coron, J.S., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 305–326. [CrossRef]
25. Li, K.; Yang, R.; Au, M.H.; Xu, Q. Practical range proof for cryptocurrency monero with provable security. In Proceedings of the International Conference on Information and Communications Security, Beijing, China, 6–8 December 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 255–262. [CrossRef]