國立政治大學資訊科學系

Department of Computer Science

National Chengchi University

碩士論文

Master's Thesis

基於橢圓曲線之非互動及指定驗證者零知識值域證明

Non-Interactive and Designated Verifier Zero-Knowledge Range Proof Based on Elliptic Curve

研究生: 陳庭軒 撰

指導教授：左瑞麟 博士

中華民國 一百一十年 七月

July, 2021

DOI:10.6814/NCCU202101402

謝辭

首先非常感謝指導教授左瑞麟教授這段時間的教導，左老師給予我很好的研究方向以及研究目標，很慶幸自己能夠在左老師的教導下順利完成碩士學位，也很感謝口試委員陳恭教授、曾一凡教授、王紹睿教授以及王智弘教授提供寶貴建議，尤其特別感謝陳恭教授及曾一凡教授給予我在學期間很多研究上的幫助，也很感謝大學時期的專題指導教授廖峻鋒教授，讓我學習如何做研究。另外也非常感謝實驗室的博士班學長劉子源學長很有耐心地督導我做研究以及許仁傑學長提供寶貴的建議，最後感謝吳映函同學總是提醒我各項重要日程，以讓我能兩年完成碩士學位。

再次感謝一路上曾經幫助過我的人，謝謝你們！

DOI:10.6814/NCCU202101402

摘要

零知識值域證明（zero-knowledge range proof，ZKRP）是一種特殊的零知識證明（zero-knowledge proof，ZKP），此種證明可以使得證明者（prover）說服驗證者（verifier），一個特定的秘密數值介於某一個範圍內，但不會洩漏該秘密數值，即驗證者無法得知此秘密數值實際之大小。本篇提出了一種有效率的非交互式零知識值域證明方案。透過橢圓曲線的應用，本篇方案在相同等級的安全強度下具有較短的執行時間、較小的金鑰長度和較小的證明大小，若將本篇 ZKRP 方案應用至區塊鏈，可降低區塊鏈上加密貨幣的交易成本。此外，本篇基於原先的零知識值域證明方案提出了一種指定驗證者（designated verifier）的零知識值域證明方案和另一種強指定驗證者（strong designated verifier）的零知識值域證明方案，此兩種方案在產生證明的過程中不需額外增加任何的計算步驟。其中，指定驗證者的方案僅被指定的驗證者能夠驗證此種方案產生的證明，且該驗證者無法說服任何第三方驗證之結果；而強指定驗證者的方案則是可以令任何第三方皆無法驗證此種方案產生的證明。上述的零知識值域證明方案皆可靈活運用，換言之，可以根據秘密值的機密性來選擇合適的方案。另外，本篇提出的方案協定亦通過嚴謹且完整的安全性證明，不失其應有的安全性。

關鍵字：區塊鏈、零知識值域證明、橢圓曲線、承諾方案、指定驗證者證明。

DOI:10.6814/NCCU202101402

Abstract

Zero-knowledge range proof (ZKRP) is a kind of particular zero-knowledge proof which allows a prover to convince a verifier that a secret value is in a specified range without revealing the actual value. In this thesis, we propose an efficient non-interactive ZKRP scheme based on elliptic curve. By applying the elliptic curve, our scheme has a shorter execution time, a smaller key size and a smaller proof size at the same level of the security strength compared to existing ZKRP schemes. If we apply our ZKRP scheme to the blockchain, the transaction cost of the cryptocurrency on the blockchain can be reduced. In addition, we propose a designated verifier ZKRP scheme and a strong designated verifier ZKRP scheme based on original ZKRP scheme without adding any extra computation steps during producing proofs. The designated verifier ZKRP scheme allows the only designated verifier to be able to verify the proof, and the verifier cannot convince any other third party of the verification result; the strong designated verifier ZKRP scheme makes any third party cannot verify the proof. Besides, these ZKRP schemes can be optional and flexible: we can choose a suitable scheme to produce a ZKRP proof according to the confidentiality of the secret value. Furthermore, we argue the security proofs of our schemes completely and rigorously so that our schemes can satisfy necessary security properties.

Keywords: Blockchain, Zero-knowledge range proof, Elliptic curve, Commitment scheme, Designated verifier proof

DOI:10.6814/NCCU202101402

CONTENTS

DOI:10.6814/NCCU202101402

DOI:10.6814/NCCU202101402

LIST OF FIGURES

DOI:10.6814/NCCU202101402

LIST OF TABLES

DOI:10.6814/NCCU202101402

1 Introduction

1.1 Background

Blockchain was created by Satoshi Nakamoto [22] in 2008 to serve as the public ledger of the cryptocurrency, Bitcoin, which is the first decentralized cryptocurrency that solves the double-spending problem without a trusted authority or central server. Since the release of Bitcoin, many other decentralized cryptocurrencies have also been created, such as Ether [2], Monero [27], and Zerocoin [20].

Zero-knowledge proof (ZKP) [11] is a method by which a user can prove to other people that he/she knows a secret without revealing any information of the secret. In cryptocurrency, ZKP can provide users with a higher level of privacy during transactions. For example, users on blockchain can use ZKP to verify the transactions but keep the identities of sender and receiver secret [29][30]; users use ZKP to enforce the correctness of the smart contract execution [16][19].

Zero-knowledge range proof (ZKRP) is a kind of particular ZKP, which allows a prover to convince a verifier that a secret value is in a specified range without revealing the actual value. For example, a buyer can prove that something is affordable for him/her without revealing the balance amount [14]; a payer can prove that a payment amount is in the limited range without revealing the exact amount [14]; a user can prove that he/she is exactly a country without revealing the exact location [14].

Many ZKRP schemes have been proposed in the literatures: the first ZKRP scheme was proposed by Boudot [1] in 2001, which is based on the Fujisaki-Okamoto commitment scheme [9] and is constructed with two proofs: the proof that two commitments hide the same secret (EL proof) and the proof that a committed number is a square (SQR proof). The EL proof can convince other people that two commitment hide the same secret value without revealing the secret value, while SQR proof can convince other people that a commitment hides a square number $y = x^2$, where $x \in \mathbb{Z}$, without revealing x or y. Pang et al. [24] applied the batch proof and verification to construct a more efficient ZKRP scheme in 2010. Chaabouni et al. [6] replaced the random oracle model with a common reference string (CRS) model to construct a new non-interactive ZKRP scheme in 2012. Bünz et al. [3] proposed a non-interactive ZKRP scheme without a trusted setup in 2017: the Bulletproofs. Koens et al. [14] improved

DOI:10.6814/NCCU202101402

the scheme proposed by Peng et al. [24] to construct a non-interactive ZKRP scheme and applied it to the smart contract in Ethereum in 2017. Tsai et al. [28] improved the scheme proposed by Boudot [1] and the scheme proposed by Pang et al. [24] to construct a new non-interactive ZKRP scheme in 2019.

1.2 Motivation

We analyze the scheme proposed by Boudot [1], the scheme proposed by Pang et al. [24], and the scheme proposed by Tsai et al. [28]. Under 1024-bit security parameter size, the proof sizes of them are approximately 896 bytes, 1280 bytes, and 2560 bytes respectively. If we use these schemes in practice but require the higher security strength, the security parameter size must be set at least 2048 bits or more, as a result, the proof sizes produced by these schemes also increase.

At the same level of the security strength, elliptic-curve cryptography (ECC) has a smaller key size compared to integer-factorization cryptography (IFC) according to NIST [4], e.g., to meet the 112-bit security strength, IFC has to be set 2048-bit key size while ECC only needs to be set 224 bits. Therefore, to reduce the proof size or even shorten the execution time, we apply the elliptic curve to the ZKRP scheme proposed by Tasi et al. [28].

In addition, we consider that the prover may not want to let everyone except the designated verifier knows the range of the secret value. That is, the proof produced by the prover cannot convince any other third parties. Thus, we require a designated verifier ZKRP scheme. Even if the designated verifier reveals the important information of the proof, any other third party cannot trust the verification result.

1.3 Contributions

As mentioned in previous section, ECC has a smaller key size compared to IFC according to NIST [4]. To reduce the proof size or even shorten the execution time, we apply the elliptic curve to the ZKRP scheme proposed by Tasi et al. [28] and construct a more efficient non-interactive ZKRP scheme, the elliptic-curve ZKRP (EC-ZKRP). To apply the elliptic curve, we replace the Fujisaki-Okamoto commitment scheme [9] with the Pedersen commitment scheme [23] so that our ZKRP scheme has a shorter execution time, a smaller key size and a smaller proof size at the same level of the

DOI:10.6814/NCCU202101402

security strength compared to existing ZKRP schemes, leading to the transaction cost can be reduced by applying our scheme in cryptocurrency.

In addition, if the prover does not want to let everyone except the designated verifier knows the range of the secret value, it means that the proof produced by the prover cannot convince any other third parties. Therefore, we propose the designated verifier ZKRP and the strong designated verifier ZKRP by applying the trapdoor commitment scheme [13] and the elliptic-curve Diffie-Hellman key exchange (ECDH) [5]. The trapdoor commitment scheme is computed through the public key of the designated verifier. Although any other third parties can open the commitment by receiving the public key of the designated verifier, they could cannot trust the commitment because they could think that the prover and verifier cheat together. Moreover, if the trapdoor commitment is computed through the shared key produced by ECDH, any other third party cannot open the commitment. As a result, by using the trapdoor commitment scheme [13] and ECDH [5], we propose a designated verifier ZKRP scheme and a strong designated verifier ZKRP scheme based on EC-ZKRP without adding any extra computation steps during producing proofs. The designated verifier ZKRP scheme allows the only designated verifier to be able to verify the proof, and the verifier cannot convince any other third party of the verification result; the strong designated verifier ZKRP scheme makes any third party cannot verify the proof. Besides, these ZKRP schemes can be optional and flexible: we can choose a suitable scheme to produce a ZKRP proof according to the confidentiality of the secret value. Furthermore, we argue the security proofs of our schemes completely and rigorously so that our schemes can satisfy necessary security properties, e.g., correctness, soundness, zero-knowledge, designated verifier and strong designated verifier. Finally, we provide the efficiency analysis compared to other existing ZKRP schemes and list some application scenarios that uses ZKRP schemes.

1.4 Organization

We start by describing some preliminaries that are used in our schemes in **Chapter 2**; we introduce the definitions, protocol, and security properties of the non-interactive EC-ZKRP scheme in **Chapter 3**; we describe the definitions, protocol, and security properties of our designated verifier ZKRP scheme and strong designated verifier ZKRP scheme based on EC-ZKRP in **Chapter 4**; in **Chapter 5**, we evaluate the efficiency of our ZKRP scheme and make a comparison of our ZKRP scheme and other existing ZKRP schemes; we describe some ZKRP application scenarios in **Chapter 6**;

DOI:10.6814/NCCU202101402

finally, we draw a conclusion in **Chapter 7**.

DOI:10.6814/NCCU202101402

2 Preliminaries

In this chapter, we introduce some notations and particular proofs that are used in our ZKRP schemes.

2.1 Notations

The notations are shown in **Table 1**.

Notation	Description
$\mathbb{N}$	the set of natural numbers
$\mathbb{Z}$	the set of integers
p	a safe prime, $p = 2p' + 1$, where p' is also a prime
$E(\mathbb{F}_p)$	an elliptic curve over finite field $\mathbb{F}_p$
q	the order of the points on $E(\mathbb{F}_p)$
$\mathbb{Z}_q^*$	the set of integers less than q
G	a generator point on $E(\mathbb{F}_p)$
H	another point on $E(\mathbb{F}_p)$, $H = s \cdot G$, where $s \in \mathbb{Z}_q^*$
$[a, b]$	the range between a and b, where $a < b$, $a, b \in \mathbb{Z}_q^*$
$str1 \| \| str2$	concatenate two strings $str1$ and $str2$
$X \xleftarrow{\$} Y$	randomly choose the value X from the space Y
$Hash(str)$	input a string str and output its hash value
λ	the security parameter
$\#E(\mathbb{F}_p)$	the cardinality of $E(\mathbb{F}_p)$

Table 1: Notations and Descriptions

The equation of elliptic curve over finite field that is used in our schemes is

$$E(\mathbb{F}_p) : y^2 = x^3 + Ax + B \pmod{p}.$$

Here p is a safe prime, which means that $p = 2p' + 1$ and p' is also a prime, and the discriminant

$$\Delta = 4A^3 + 27B^2 \pmod{p}$$

DOI:10.6814/NCCU202101402

must not equal to 0. In addition, the order q is a prime factor of the cardinality $\#E(\mathbb{F}_p)$, which is the number of points on the curve $E(\mathbb{F}_p)$. Many elliptic curves used in practice have been defined in the literatures, e.g., Secp256k1 [25], NIST224p [10], NIST256p [10].

2.2 Hardness Assumptions

In this section, we describe some definitions of hardness assumptions in detail.

Definition 1 (Elliptic-curve discrete logarithm problem). Let an elliptic curve $E(\mathbb{F}_p)$ over finite field $\mathbb{F}_p$, a generator point G and another point H which are on the curve $E(\mathbb{F}_p)$ have the order q. There does not exist any algorithm in probabilistic-polynomial time to find $s \in \mathbb{Z}_q^*$ such that $H = s \cdot G$.

Definition 2 (Elliptic-curve Diffie-Hellman assumption). Let an elliptic curve $E(\mathbb{F}_p)$ over finite field $\mathbb{F}_p$, and a generator point G which is on the curve $E(\mathbb{F}_p)$ has the order q. There does not exist any algorithm in probabilistic-polynomial time to determine whether $cG = abG$ by given (G, aG, bG, cG) with non-negligible probability, where $a, b, c \in \mathbb{Z}_q^*$.

2.3 Elliptic-curve Pedersen Commitment Scheme

A commitment scheme allows a person to commit to a secret value while keeping the value hidden, and the person can reveal the committed value later. Before the person reveal the secret value, no one can derive the secret value from the commitment (the hiding property). On the other hand, and the person cannot change the committed value (the binding property).

Pedersen [23] proposed a commitment scheme that based on the discrete logarithm problem in 1991. In our schemes, we use the elliptic-curve Pedersen commitment scheme (EC-Pedersen commitment scheme) [8][21], which also satisfies the hiding property and the binding property through the elliptic-curve discrete logarithm problem [12].

Definition 3 (Commitment scheme). A commitment scheme is composed by three functions $(Setup, Commit, Open)$.

DOI:10.6814/NCCU202101402

(1) $pp \xleftarrow{\$} Setup(\lambda)$: by inputting a secure parameter λ, the polynomial-time function $Setup$ outputs the public parameters pp.

(2) $C \xleftarrow{\$} Com(m, r)$: by inputting the secret value m and a random number r, the polynomial-time function Com outputs the commitment C.

(3) $b \leftarrow Open(C, m, r)$: by inputting the commitment C, the secret value m and the random number r, the polynomial-time function $Open$ outputs a result $b \in \{0,1\}$. The commitment C is accepted if b is equal to 1. Otherwise, it is rejected.

Below, we describe the protocol of EC-Pedersen commitment scheme in detail.

(1) $Setup$: first, choose a safe prime p. The elliptic curve E over $\mathbb{F}_p$ that is defined by an equation: $y^2 = x^3 + Ax + B \pmod p$, where $A, B \in \mathbb{F}_p$ (its discriminant $4A^3 + 27B^2 \not\equiv 0 \pmod p$). Secondly, randomly choose a generator point G on $E(\mathbb{F}_p)$, and then randomly choose $s \in \mathbb{Z}_q^*$ to compute the point $H = s \cdot G$. The points G and H have the order q. Lastly, publish the public parameters

$$pp = \{p, A, B, q, G, H\}.$$

(2) $Commit$: To commit to the secret value $m \in \mathbb{Z}_q^*$, the sender randomly chooses $r \in \mathbb{Z}_q^*$ to compute

$$C = m \cdot G + r \cdot H$$

and publishes the commitment C.

(3) $Open$: To open the commitment C, the sender reveals m and r. The receiver verifies

$$C \stackrel{?}{=} m \cdot G + r \cdot H$$

and accepts the commitment if and only if $C = m \cdot G + r \cdot H$. Otherwise, the commitment is rejected.

In what follows, we describe the definitions of hiding property and binding property in detail.

DOI:10.6814/NCCU202101402

Definition 4 (Hiding property). There does not exist any algorithm in probabilistic-polynomial time to compute the committed value through a commitment. More precisely, the probability

$$\left| \Pr\left[b' = b : \begin{array}{c} pp \leftarrow Setup(\lambda); \\ m_0 \neq m_1; \\ b \leftarrow \{0,1\}; \\ C_b \leftarrow Com(m_b, r); \\ b' \leftarrow \mathcal{A}(C_b) \end{array} \right] - \frac{1}{2} \right| \leq negl(\lambda)$$

for all probabilistic-polynomial time adversaries $\mathcal{A}$ and security parameter λ. If the probability of $\mathcal{A}$ guessing b is exactly equal to $\frac{1}{2}$, the commitment scheme satisfies perfect hiding.

Definition 5 (Binding property). There does not exist any algorithm in probabilistic-polynomial time to find $m_1, m_2, r_1, r_2 \in \mathbb{Z}_q^*$ such that $C = m_1 \cdot G + r_1 \cdot H = m_2 \cdot G + r_2 \cdot H$, where $(m_1, r_1) \neq (m_2, r_2)$. More precisely, the probability

$$\Pr\left[\begin{array}{c} Com(m_1, r_1) = Com(m_2, r_2); \\ (m_1, m_2, r_1, r_2) \leftarrow \mathcal{A}(pp) \end{array} : \begin{array}{c} pp \leftarrow Setup(\lambda); \\ m_1 \neq m_2 \end{array} \right] \leq negl(\lambda)$$

for all probabilistic-polynomial time adversaries $\mathcal{A}$ and security parameter λ. If the probability of $\mathcal{A}$ finding m_1, m_2, r_1, r_2 is exactly equal to 0, the commitment scheme satisfies perfect binding.

2.4 Elliptic-curve Diffie–Hellman Key Exchange (ECDH)

An elliptic-curve Diffie–Hellman (ECDH) [5] allows two parties to create a shared key over a public channel. It is a variant of the Diffie–Hellman protocol by applying the elliptic-curve computation. In our scheme, we use the shared key that created followed ECDH to fulfill a strong designated verifier ZKRP scheme.

We describe how a shared key is created by the following example. Assume that Alice and Bob want to create a shared key.

(1) Alice and Bob agree the curve $E(\mathbb{F}_p) : y^2 = x^3 + Ax + B \pmod{p}$, the

DOI:10.6814/NCCU202101402

discriminant of which $4A^3 + 27B^2 \not\equiv 0 \pmod{p}$. The public parameter is

$$pp = \{p, A, B, q, G\},$$

where G is a generator point on $E(\mathbb{F}_p)$, which has the order q.

(2) Alice and Bob randomly choose their private key $X_A \in \mathbb{Z}_q^*$ and $X_B \in \mathbb{Z}_q^*$ and compute their public key Y_A and Y_B.

$$Y_A = X_A \cdot G,$$
$$Y_B = X_B \cdot G.$$

Therefore, Alice's key pair is (X_A, Y_A) and Bob's key pair is (X_B, Y_B). They send their public key to each other.

(3) Both Alice and Bob can get the shared key S by computing

$$S = X_A \cdot Y_B = Y_A \cdot X_B = X_A \cdot X_B \cdot G.$$

There does not exist any adversary that can compute Alice's or Bob's private key through their public key, unless the adversary can solve the elliptic-curve discrete logarithm problem. In addition, there does not exist any adversary that can compute the shared key, unless the adversary can solve the elliptic-curve Diffie–Hellman problem.

2.5 Trapdoor Commitment Scheme

A trapdoor commitment scheme is proposed by Jakobsson et al. [13] in 1996, which is based on the Pedersen commitment scheme [23]. In our scheme, we use the EC-trapdoor Pedersen commitment scheme to fulfill the designated verifier ZKRP scheme. The difference between the original Pedersen commitment and trapdoor commitment is that the trapdoor commitment is computed through a user's public key. For example, Alice commits a commitment through Bob's public key. Since Bob knows his private key, the commitment cannot be trusted by everyone except Alice and Bob. We decribe the definition and the protocol of the trapdoor commitment and expain the reason why the commitment cannot be trusted by any third party in detail below.

Definition 6 (Trapdoor commitment scheme). A trapdoor commitment scheme is composed by three functions $(Setup, Commit, Open)$.

(1) $pp \xleftarrow{\$} Setup(\lambda)$: by inputting a secure parameter λ, the polynomial-time

DOI:10.6814/NCCU202101402

function $Setup$ outputs the public parameters pp.

(2) $C \xleftarrow{\$} Com(m, r, pk)$: by inputting the secret value m, a random number r, and a public key pk, the polynomial-time function Com outputs the trapdoor commitment C.

(3) $b \leftarrow Open(C, m, r, pk)$: by inputting the trapdoor commitment C, the secret value m, the random number r, and the public key pk, the polynomial-time function $Open$ outputs a result $b \in \{0,1\}$. The commitment C is accepted if b is equal to 1. Otherwise, it is rejected.

We describe the protocol of the EC-trapdoor Pedersen commitment scheme by the following example.

(1) Alice and Bob agree the curve $E(\mathbb{F}_p): y^2 = x^3 + Ax + B \pmod p$, the discriminant of which $4A^3 + 27B^2 \not\equiv 0 \pmod p$. The public parameter is

$$pp = \{p, A, B, q, G\},$$

where G is a generator point on $E(\mathbb{F}_p)$, which has the order q. After that, Alice and Bob randomly choose their private key $X_A \in \mathbb{Z}_q^*$ and $X_B \in \mathbb{Z}_q^*$ and compute their public key Y_A and Y_B.

$$Y_A = X_A \cdot G,$$
$$Y_B = X_B \cdot G.$$

Therefore, Alice's key pair is (X_A, Y_A) and Bob's key pair is (X_B, Y_B). Then they publish their public keys.

(2) Suppose that Alice wants to commit to the secret value $m \in \mathbb{Z}_q^*$. Alice randomly chooses $r \in \mathbb{Z}_q^*$ and computes

$$Com(m, r): C = m \cdot G + r \cdot Y_B$$

by using Bob's public key Y_B. After that, Alice publishes the commitment C.

(3) To open the commitment C, Alice reveals (m, r). Bob verifies

$$C \stackrel{?}{=} m \cdot G + r \cdot Y_B$$

and accepts the commitment if and only if $C = m \cdot G + r \cdot Y_B$.

DOI:10.6814/NCCU202101402

Since Bob knows his secret key X_B, which is the relation of G and Y_B, i.e., the verifier can find $m_1, m_2, r_1, r_1 \in \mathbb{Z}_q^*$ easily such that

$$C = m_1 \cdot G + r_1 \cdot Y_B = m_2 \cdot G + r_2 \cdot Y_B.$$

For any other third parties, the commitment does not satisfy the binding property. Therefore, they could think the prover and the designated verifier cheat together and cannot trust the commitment.

2.6 Elliptic-curve EL Proof

The EL proof was proposed by Boudot [1] in 2001. For short, we follow Tsai et al. [28] and call it the EL proof. This proof is a kind of zero-knowledge proof such that the verifier can verify that two commitments hide the same value without leaking the actual committed values of the two commitments. Since we use the elliptic-curve Pedersen commitment scheme as described in **Section 2.3**, we also use the elliptic-curve EL proof (EC-EL proof) in our schemes.

2.6.1 Definitions and Security Models

Definition 7 (EL proof). EL proof is composed by three functions $(Setup, \text{EL}, \text{vEL})$.

(1) $pp \xleftarrow{\$} Setup(\lambda)$: by inputting a secure parameter λ, the polynomial-time function $Setup$ outputs the public parameters pp.

(2) $EL \xleftarrow{\$} \text{EL}(pp, C_1, C_2)$: by inputting the public parameters pp and two commitments C_1, C_2, the polynomial-time function EL outputs the proof EL.

(3) $b \leftarrow \text{vEL}(pp, EL)$: by inputting the public parameters pp and the proof EL, the polynomial-time function vEL outputs a result $b \in \{0,1\}$. The proof EL is accepted if b is equal to 1, which means that two commitments C_1 and C_2 hide the same value. Otherwise, it is rejected.

The EL proof satisfies the correctness property: if the proof produced by two commitments which hide the same value, the proof must pass the verification. Therefore, the verifier can confirm that the two commitments hide the same value.

DOI:10.6814/NCCU202101402

Definition 8 (Correctness of EL proof). If the two commitments C_1 and C_2 hide the same value m and the scheme $\mathrm{EL} = (Setup, \mathrm{EL}, \mathrm{vEL})$ satisfies the correctness property, the probability

$$\left| \Pr\left[\mathrm{vEL}(pp, EL) = 1 : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ C_1 \leftarrow Com(m, r_1); \\ C_2 \leftarrow Com(m, r_2); \\ EL \xleftarrow{\$} \mathrm{EL}(pp, C_1, C_2) \end{array} \right] - 1 \right| \leq negl(\lambda)$$

for all security parameter λ.

The EL proof satisfies the soundness property: if the proof produced by two commitments hide different value, the proof cannot pass the verification. In other words, if the proof can pass the verification, the two commitment must hide the same value.

Definition 9 (Soundness of EL proof). If the two commitments C_1 and C_2 hide different value m_1, m_2 and the scheme $\mathrm{EL} = (Setup, \mathrm{EL}, \mathrm{vEL})$ satisfies the soundness property, the probability

$$\Pr\left[\mathrm{vEL}(pp, EL) = 1 : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ m_1 \neq m_2; \\ C_1 \leftarrow Com(m_1, r_1); \\ C_2 \leftarrow Com(m_2, r_2); \\ EL \xleftarrow{\$} \mathrm{EL}(pp, C_1, C_2) \end{array} \right] \leq negl(\lambda)$$

for all security parameter λ.

The EL proof satisfies the zero-knowledge property: the verifier can only confirm whether the two commitments hide the same value, but cannot know the exact committed value through the EL proof produced by the prover. More precisely, there does not exist any algorithm in probabilistic-polynomial time that can distinguish the real proof from the ideal proof produced by a simulator, which does not contain any information about the committed values.

Definition 10 (Zero-knowledge of EL proof). Given a polynomial-time simulator $\mathcal{ELS}$ that can produce a proof without inputting the secret m. If the scheme $\mathrm{EL} = (Setup, \mathrm{EL}, \mathrm{vEL})$ satisfies the zero-knowledge property, the probability

DOI:10.6814/NCCU202101402

$$\left| \Pr\left[b' = b : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ EL_1 \xleftarrow{\$} \mathrm{EL}(pp, C_1, C_2); \\ EL_2 \xleftarrow{\$} \mathcal{ELS}(pp); \\ b = \{0,1\}; \\ b' \leftarrow \mathcal{A}(pp, EL_b) \end{array} \right] - \frac{1}{2} \right| \leq negl(\lambda)$$

for all probabilistic-polynomial time adversaries $\mathcal{A}$ and security parameter λ.

2.6.2 EC-EL Proof Protocol

In the following, we describe the protocol of the EC-EL proof in detail: the prover knows the secret value $m \in \mathbb{Z}_q^*$ and two commitments

$$A = m \cdot G_1 + s \cdot H_1,$$
$$B = m \cdot G_2 + r \cdot H_2,$$

computed by two sets of public parameters (G_1, H_1) and (G_2, H_2), where G_1, G_2 are two generator points on $E(\mathbb{F}_p)$ and H_1, H_2 are two points on $E(\mathbb{F}_p)$. The two commitments A and B hide the same secret value m.

To produce the EL proof for A and B, the prover runs the EL proof function:

$$EL \xleftarrow{\$} \mathrm{EL}(m, s, r, G_1, H_1, G_2, H_2).$$

(1) The prover randomly chooses $\mu, v_1, v_2 \in \mathbb{Z}_q^*$ and computes

$$C_1 = \mu \cdot G_1 + v_1 \cdot H_1,$$
$$C_2 = \mu \cdot G_2 + v_2 \cdot H_2.$$

(2) The prover computes $h = Hash(C_1 || C_2)$.

(3) The prover computes

$$x = \mu + hm,$$
$$x_1 = v_1 + hs,$$
$$x_2 = v_2 + hr.$$

DOI:10.6814/NCCU202101402

(4) Finally, the prover produces the EL proof

$$EL = (h, x, x_1, x_2).$$

To verify the EL proof, the verifier runs the EL verification function:

$$\text{vEL}(EL, G1, H1, G2, H2, A, B).$$

(1) The verifier computes

$$C_1' = x \cdot G_1 + x_1 \cdot H_1 + (-h) \cdot A,$$
$$C_2' = x \cdot G_2 + x_2 \cdot H_2 + (-h) \cdot B.$$

(2) The verifier can be convinced that the two commitments A and B hide the same value if and only if

$$h = Hash(C_1' || C_2').$$

2.6.3 Security Descriptions

In this section, we describe the security properties of the EC-EL proof: correctness, soundness, zero-knowledge.

2.6.3.1 Correctness of EC-EL Proof

To verify the EC-EL proof, the verifier computes

$$C_1' = x \cdot G_1 + x_1 \cdot H_1 + (-h) \cdot A,$$
$$C_2' = x \cdot G_2 + x_2 \cdot H_2 + (-h) \cdot B.$$

In the detail, if the prover is honest and follows the EC-EL proof to produce $EL = (h, x, x_1, x_2)$, the verifier can expand C_1' and C_2':

$$\begin{aligned} C_1' &= x \cdot G_1 + x_1 \cdot H_1 + (-h) \cdot A \\ &= (\mu + hm) \cdot G_1 + (v_1 + hs) \cdot H_1 + (-h) \cdot (m \cdot G_1 + s \cdot H_1) \\ &= \mu \cdot G_1 + hm \cdot G_1 + v_1 \cdot H_1 + hs \cdot H_1 - hm \cdot G_1 - hs \cdot H_1 \\ &= \mu \cdot G_1 + v_1 \cdot H_1 \end{aligned}$$

DOI:10.6814/NCCU202101402

$$= C_1,$$
$$C_2' = x \cdot G_2 + x_2 \cdot H_2 + (-h) \cdot B$$
$$= (\mu + hm) \cdot G_2 + (v_2 + hr) \cdot H_2 + (-h) \cdot (m \cdot G_2 + r \cdot H_2)$$
$$= \mu \cdot G_2 + hm \cdot G_2 + v_2 \cdot H_1 + hr \cdot H_2 - hm \cdot G_2 - hr \cdot H_2$$
$$= \mu \cdot G_2 + v_2 \cdot H_2$$
$$= C_2.$$

Therefore, $C_1' = C_1$ and $C_2' = C_2$, i.e.,

$$h = Hash(C_1' || C_2') = Hash(C_1 || C_2).$$

Finally, since the honest prover follows the EC-EL proof to produce the proof which can pass the verification, the EC-EL proof satisfies the correctness property.

2.6.3.2 Soundness of EC-EL Proof

Lemma 1. The elliptic-curve Pedersen commitment scheme satisfies the binding property: it is difficult to find two different secret values that hidden by the same commitment. More precisely, there does not exist any algorithm in probabilistic-polynomial time to find $m_1, m_2, r_1, r_2 \in \mathbb{Z}_q^*$ such that

$$C = m_1 \cdot G + r_1 \cdot H = m_2 \cdot G + r_2 \cdot H,$$

where $(m_1, r_1) \neq (m_2, r_2)$.

Proof. Assume that the order of two points G, H is q, and $H = s \cdot G$. Let $m_1, m_2, r_1, r_2 \in \mathbb{Z}_q^*$, such that

$$C = m_1 \cdot G + r_1 \cdot H = m_2 \cdot G + r_2 \cdot H$$
$$\therefore (m_1 - m_2) \cdot G = (r_2 - r_1) \cdot H$$
$$\therefore (m_1 - m_2) = s(r_2 - r_1) \bmod q.$$

The discussion is divided into two cases:

(1) $r_2 - r_1 = 0$: it means that $m_1 - m_2 \equiv 0 \bmod q$, i.e.,

$$m_1 \equiv m_2 \bmod q.$$

All values are over a finite field $\mathbb{F}_q$, so $0 \leq m_1, m_2 \leq q$ and $0 \leq r_1, r_2 \leq q$.

DOI:10.6814/NCCU202101402

In this case, m_1 must be equal to m_2 and r_1 must be equal to r_2. This is contradiction with **Lemma 1**: $(m_1, r_1) \neq (m_2, r_2)$.

(2) $r_2 - r_1 \neq 0$: We know that $s \equiv \frac{m_1 - m_2}{r_2 - r_1} \bmod q$. If we can compute s through m_1, m_2, r_1, r_2, it means that there exists an algorithm in probabilistic-polynomial time to solve the elliptic-curve discrete logarithm problem [12].

According to (1)(2), there does not exist any algorithm in probabilistic-polynomial time to find $m_1, m_2, r_1, r_2 \in \mathbb{Z}_q^*$ such that $C = m_1 \cdot G + r_1 \cdot H = m_2 \cdot G + r_2 \cdot H$, where $(m_1, r_1) \neq (m_2, r_2)$. Therefore, **Lemma 1** is proved.

Theorem 1. If the prover follows EC-EL proof to produce the proof $EL = (h, x, x_1, x_2)$ that can pass the verification with non-negligible probability, the two commitments used to produce the proof must hide the same value m.

Proof. Assume that the prover follows the EC-EL proof protocol to produce the proof EL which pass the verification but uses the two commitments A and B hide different values $m_1, m_2 \in \mathbb{Z}_q^*$ respectively, i.e., $m_1 \neq m_2$.
We can simplify C_1' and C_2':

$$\begin{aligned} C_1' &= x \cdot G_1 + x_1 \cdot H_1 + (-h) \cdot A \\ &= (x - hm_1) \cdot G_1 + (x_1 - hs) \cdot H_1, \\ C_2' &= x \cdot G_2 + x_2 \cdot H_1 + (-h) \cdot B \\ &= (x - hm_2) \cdot G_2 + (x_2 - hr) \cdot H_1. \end{aligned}$$

If the proof can pass the verification, $C_1' = C_1$ and $C_1' = C_2$. Considering $C_1 = \mu \cdot G_1 + v_1 \cdot H_1$ and $C_2 = \mu \cdot G_2 + v_2 \cdot H_2$ and the binding property of the EC-Pederesen commitment scheme (**Lemma 1**), i.e.,

$$\mu = x - hm_1 = x - hm_2.$$

We can obtain that $m_1 - m_2 \equiv 0 \bmod q$, and we have

$$m_1 \equiv m_2 \bmod q.$$

All values are over a finite field $\mathbb{F}_q$, so $0 < m_1, m_2 < q$ In this case, m_1 must be equal to m_2. This is contradiction with the assumption: $m_1 \neq m_2$. Thus, if the proof

DOI:10.6814/NCCU202101402

can pass the verification, the two commitments must hide the same value. Therefore, **Theorem 1** is proved. The EC-EL proof satisfies the soundness property.

2.6.3.3 Zero-knowledge of EC-EL Proof

Theorem 2. Assume that there exists a simulator $\mathcal{ELS}$ and an oracle $\mathcal{HO}$ follows the EC-EL proof but replaces secret values which only the prover knows with random numbers to produce the proof $\mathcal{ELP}$. For all adversaries, there does not exist any algorithm in probabilistic-polynomial time to distinguish between the real proof EL produced by the prover and the ideal proof $\mathcal{ELP}$ produced by the simulator $\mathcal{ELS}$.

Proof. Assume that the oracle $\mathcal{HO}$ and simulator $\mathcal{ELS}$ are defined and shown in **Table 2**.

Oracle: $\mathcal{HO}(str)$
(1) $h \overset{\$}{\leftarrow} Hash(str)$ (2) Return h
Simulator: $\mathcal{ELS}(G_1, H_1, G_2, H_2, A, B)$
(1) Randomly choose $h, x, x_1, x_2 \in \mathbb{Z}_q^*$ (2) Compute $C_1 = x \cdot G_1 + x_1 \cdot H_1 + (-h) \cdot A$, $C_2 = x \cdot G_2 + x_2 \cdot H_2 + (-h) \cdot B$ (3) Compute $h' = \mathcal{HO}(C_1 \| C_2)$ (4) Return $\mathcal{ELP} = (h', x, x_1, x_2)$

Table 2: Simulator $\mathcal{ELS}$ and Oracle $\mathcal{HO}$

$\mathcal{HO}(str)$: input a string str and return the hash value h of the string.
$\mathcal{ELS}(G_1, H_1, G_2, H_2, A, B)$: input two commitment A, B and their public parameters G_1, H_1, G_2, H_2. $\mathcal{ELS}$ simulates the EC-EL proof to produce a EL proof without any information of committed values of A, B.
First, randomly chooses $h, x, x_1, x_2 \in \mathbb{Z}_q^*$ and computes C_1 and C_2:

$$C_1 = x \cdot G_1 + x_1 \cdot H_1 + (-h) \cdot A,$$
$$C_2 = x \cdot G_2 + x_2 \cdot H_2 + (-h) \cdot B,$$

and then computes

$$h' = \mathcal{HO}(C_1 \| C_2).$$

Finally, outputs the simulation proof

DOI:10.6814/NCCU202101402

$$\mathcal{ELP} = (h', x, x_1, x_2).$$

Assume that a prover knows a secret value m and two commitments $A = m \cdot G_1 + s \cdot H_1$ and $B = m \cdot G_2 + r \cdot H_2$. The prover produces a real proof

$$EL = (h, x, x_1, x_2): \begin{cases} \mu, v_1, v_2 \leftarrow \mathbb{Z}_q^*; \\ C_1 = \mu \cdot G_1 + v_1 \cdot H_1; \\ C_2 = \mu \cdot G_2 + v_2 \cdot H_2; \\ h = Hash(C_1 || C_2); \\ x = \mu + hm; \\ x_1 = v_1 + hs; \\ x_2 = v_2 + hr \end{cases}$$

and the simulator $\mathcal{ELS}$ inputs two commitments $A' = m' \cdot G_1 + s' \cdot H_1$ and $B' = m'' \cdot G_2 + r' \cdot H_2$. $\mathcal{ELS}$ produces a ideal proof

$$\mathcal{ELP} = (h', x', x_1', x_2'): \begin{cases} h_1, x', x_1', x_2' \leftarrow \mathbb{Z}_q^*; \\ C_1 = x' \cdot G_1 + x_1' \cdot H_1 + (-h_1) \cdot A; \\ C_2 = x' \cdot G_2 + x_2' \cdot H_2 + (-h_1) \cdot B; \\ h' = \mathcal{HO}(C_1 || C_2) \end{cases}.$$

Let $\widehat{EL} = (\hat{h}, \hat{x}, \widehat{x_1}, \widehat{x_2})$ be a randomly chosen proof in the set of all valid proofs. The probability

$$\Pr[EL = \widehat{EL}] = \Pr\begin{bmatrix} \mu, v_1, v_2 \in \mathbb{Z}_q^*; \\ h = \hat{h}, x = \hat{x}, x_1 = \widehat{x_1}, x_2 = \widehat{x_2} \end{bmatrix} = \frac{1}{(q-1)^4}$$

and the probability

$$\Pr[\mathcal{ELP} = \widehat{EL}] = \Pr\begin{bmatrix} h_1, x', x_1', x_2' \in \mathbb{Z}_q^*; \\ h' = \hat{h}, x = \hat{x}, x_1 = \widehat{x_1}, x_2 = \widehat{x_2} \end{bmatrix} = \frac{1}{(q-1)^4}$$

are equal, i.e., EL and $\mathcal{ELP}$ are indistinguishable. Therefore, **Theorem 2** is proved. The EC-EL proof satisfies the zero-knowledge property.

2.7 Elliptic-curve SQR Proof

DOI:10.6814/NCCU202101402

The SQR proof was proposed by Boudot [1] in 2001. For short, we follow Tsai et al. [28] and call it the SQR proof. This proof is a kind of zero-knowledge proof such that the verifier can verify that a commitment hides a square number α^2, $\alpha \in \mathbb{Z}_q^*$, without leaking the value α or α^2. Since we use the elliptic-curve Pedersen commitment scheme as described in **Section 2.3**, we use the elliptic-curve SQR proof (EC-SQR proof) in our schemes.

2.7.1 Definitions and Security Models

Definition 11 (SQR proof). SQR proof is composed by three functions $(Setup, \text{SQR}, \text{vSQR})$.

(1) $pp \xleftarrow{\$} Setup(\lambda)$: by inputting a secure parameter λ, the polynomial-time function $Setup$ outputs the public parameters pp.

(2) $SQR \xleftarrow{\$} \text{SQR}(pp, C)$: by inputting the public parameters pp and a commitment C, the polynomial-time function SQR outputs the proof SQR.

(3) $b \leftarrow \text{vSQR}(pp, SQR)$: by inputting the public parameters pp and the proof SQR, the polynomial-time function vSQR outputs a result $b \in \{0,1\}$. The proof SQR is accepted if b is equal to 1, which means that the commitment C hides a square number. Otherwise, it is rejected.

The SQR proof satisfies the correctness property: if the proof produced by a commitment which hides a square number, the proof must pass the verification. Therefore, the verifier can confirm that the commitment hides the committed value is a square number.

Definition 12 (Correctness of SQR proof). If the commitment C hides a square number $y = x^2, x \in \mathbb{Z}$ and the scheme $\text{SQR} = (Setup, \text{SQR}, \text{vSQR})$ satisfies the correctness property, the probability

$$\left| \Pr\left[\text{vSQR}(pp, SQR) = 1 : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ y = x^2, x \in \mathbb{Z}; \\ C \leftarrow Com(y, r); \\ SQR \xleftarrow{\$} \text{SQR}(pp, C) \end{array} \right] - 1 \right| \leq negl(\lambda)$$

for all security parameter λ.

DOI:10.6814/NCCU202101402

The SQR proof satisfies the soundness property: if the proof produced by a commitment hides a value which is not a square number, it cannot pass the verification. In other words, if the proof can pass the verification, the commitment must hide a square number.

Definition 13 (Soundness of SQR proof). If the commitments C hides the value y which is not a square number and the scheme SQR$= (Setup, \text{SQR}, \text{vSQR})$ satisfies the soundness property, the probability

$$\Pr\left[\text{vSQR}(pp, SQR) = 1 : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ y \in \mathbb{Z}, \sqrt{y} \notin \mathbb{Z}; \\ C \leftarrow Com(y, r); \\ SQR \xleftarrow{\$} \text{SQR}(pp, C) \end{array}\right] \leq negl(\lambda)$$

for all security parameter λ.

The SQR proof satisfies the zero-knowledge property: the verifier can only confirm whether the commitment hides a square number, but cannot know the exact committed value through the SQR proof produced by the prover. More precisely, there does not exist any algorithm in probabilistic-polynomial time that can distinguish the real proof from the ideal proof produced by a simulator, which does not contain any information about the committed value.

Definition 14 (Zero-knowledge of SQR proof). Given a polynomial-time simulator $\mathcal{SQRS}$ that can produce a proof without inputting the secret m. If the scheme SQR$= (Setup, \text{SQR}, \text{vSQR})$ satisfies the zero-knowledge property, the probability

$$\left|\Pr\left[b' = b : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ SQR_1 \xleftarrow{\$} \text{SQR}(pp, C); \\ SQR_2 \xleftarrow{\$} \mathcal{SQRS}(pp); \\ b = \{0,1\}; \\ b' \leftarrow \mathcal{A}(pp, SQR_b) \end{array}\right] - \frac{1}{2}\right| \leq negl(\lambda)$$

for all probabilistic-polynomial time adversaries $\mathcal{A}$ and security parameter λ.

DOI:10.6814/NCCU202101402

2.7.2 EC-SQR Proof Protocol

In the following, we describe the protocol of the EC-SQR proof in detail: the prover knows the secret value α and the commitment

$$E = \alpha^2 \cdot G + r_1 \cdot H,$$

computed by a generator point G on $E(\mathbb{F}_p)$ and another point H on $E(\mathbb{F}_p)$, where G and H are public parameters. The commitment E hides value $\alpha^2 \in \mathbb{Z}_q^*$.

To produce the SQR proof for E, the prover runs the SQR proof function:

$$SQR \xleftarrow{\$} SQR(\alpha, r_1, G, H).$$

(1) The prover randomly chooses $r_2 \in \mathbb{Z}_q^*$ and computes

$$F = \alpha \cdot G + r_2 \cdot H.$$

(2) The prover computes

$$r_3 = r_1 - r_2\alpha.$$

(3) The prover computes

$$E' = \alpha \cdot F + r_3 \cdot H,$$

where E' must be equal to E.

(4) Since the two commitments F and E' hide the same value α, the prover runs the EC-EL proof as described in **Section 2.6** to produce the proof

$$EL = (h, x, x_1, x_2) = \mathrm{EL}(\alpha, r_2, r_3, G, H, F, H).$$

In the detail, first, the provers randomly chooses $\mu, v_1, v_2 \in \mathbb{Z}_q^*$ to compute

$$C_1 = \mu \cdot G + v_1 \cdot H,$$
$$C_2 = \mu \cdot F + v_2 \cdot H.$$

Then, the prover computes $h = Hash(C_1 || C_2)$ to compute

DOI:10.6814/NCCU202101402

$$x = \mu + h\alpha,$$
$$x_1 = v_1 + hr_2,$$
$$x_2 = v_2 + hr_3.$$

(5) Finally, the prover produces the SQR proof

$$SQR = (h, x, x_1, x_2, F).$$

To verify the SQR proof, the verifier runs the SQR verification function:

$$\text{vSQR}(SQR, G, H, E).$$

(1) The verifier computes

$$C_1' = x \cdot G + x_1 \cdot H + (-h) \cdot F$$
$$C_2' = x \cdot F + x_2 \cdot H + (-h) \cdot E.$$

(2) The verifier can be convinced that the commitment E hides a square number if and only if

$$h = Hash(C_1' || C_2').$$

2.7.3 Security Descriptions

In this section, we describe the security properties of the EC-SQR proof: correctness and soundness. Since the proof of the zero-knowledge property of EC-SQR proof is easily obtained from the properties of the EC-EL proof, the description of the zero-knowledge property is omitted from this section.

2.7.3.1 Correctness of EC-SQR Proof

Since we can expand

$$E' = \alpha \cdot F + r_3 \cdot H$$
$$= \alpha \cdot (\alpha \cdot G + r_2 \cdot H) + r_3 \cdot H$$
$$= \alpha^2 \cdot G + (\alpha r_2 + r_3) \cdot H$$
$$= \alpha^2 \cdot G + r_1 \cdot H$$

DOI:10.6814/NCCU202101402

$$= E,$$

E' must be equal to E. To verify the EC-SQR proof, the verifier computes

$$C_1' = x \cdot G + x_1 \cdot H + (-h) \cdot F$$
$$C_2' = x \cdot G + x_2 \cdot H + (-h) \cdot E.$$

In the detail, if the prover is honest and follows the EC-SQR proof to produce $SQR = (h, x, x_1, x_2, F)$, the verifier can expand C_1' and C_2':

$$\begin{aligned}
&C_1' = x \cdot G + x_1 \cdot H + (-h) \cdot F \\
&= (\mu + h\alpha) \cdot G + (v_1 + hr_2) \cdot H + (-h) \cdot (\alpha \cdot G + r_2 \cdot H) \\
&= \mu \cdot G + h\alpha \cdot G + v_1 \cdot H + hr_2 \cdot H - h\alpha \cdot G - hr_2 \cdot H \\
&= \mu \cdot G + v_1 \cdot H \\
&= C_1, \\
&C_2' = x \cdot F + x_2 \cdot H + (-h) \cdot E \\
&= (\mu + h\alpha) \cdot F + (v_2 + hr_3) \cdot H + (-h) \cdot (\alpha \cdot F + r_3 \cdot H) \\
&= \mu \cdot F + h\alpha \cdot F + v_2 \cdot H + hr_3 \cdot H - h\alpha \cdot F - hr_3 \cdot H \\
&= \mu \cdot F + v_2 \cdot H \\
&= C_2.
\end{aligned}$$

Therefore, $C_1' = C_1$ and $C_2' = C_2$, i.e.,

$$h = Hash(C_1' || C_2') = Hash(C_1 || C_2).$$

Finally, since the honest prover follows the EC-SQR proof to produce the proof which can pass the verification, the EC-SQR proof satisfies the correctness property.

2.7.3.2 Soundness of EC-SQR Proof

Theorem 3. If the prover follows EC-SQR proof to produce the proof $SQR = (h, x, x_1, x_2, F)$ that can pass the verification with non-negligible probability, the commitment used to produce the proof must hide a square number α^2, where $\alpha \in \mathbb{Z}$.

Proof. Assume that the prover follows the EC- SQR proof protocol to produce the proof $SQR = (h, x, x_1, x_2, F)$ which can pass the verification. According to the soundness of EC-EL proof (**Theorem 2**), we can ensure that the commitment E and F must hide the same value based on (F, H) and (G, H) respectively. Without loss of generality,

DOI:10.6814/NCCU202101402

let

$$E = \alpha' \cdot F + r_3' \cdot H,$$
$$F = \alpha' \cdot G + r_2' \cdot H.$$

Therefore, we can expand

$$E = \alpha' \cdot F + r_3' \cdot H = (\alpha')^2 \cdot G + (\alpha' r_2' + r_3') \cdot H.$$

Obviously, the commitment E must hide a square number $(\alpha')^2$. Thus, **Theorem 3** is proved, the EC-SQR proof satisfies the soundness property.

2.8 Zero-Knowledge Proof with Commitment Secret (ZKPCS)

In this section, we introduce a particular zero-knowledge proof: the zero-knowledge proof with commitment secret (ZKPCS), which can be used to convince the verifier that the prover knows the committed value of a commitment without revealing the secrets, i.e., the prover can commit another commitment by using the same committed value. The four papers [1], [18], [24] and [28] all use the similar ZKPCS, but all of which are used to prove the knowledge of the Fujisaki-Okamoto commitment scheme [9]. Therefore, we propose a new ZKPCS for the elliptic-curve Pedersen commitment scheme.

Definition 15 (Zero-knowledge proof with commitment secret). Zero-knowledge proof with commitment secret is composed by five functions $(Setup, Pro, Chal, Res, Ver)$.

(1) $pp \xleftarrow{\$} Setup(\lambda)$: by inputting a secure parameter λ, the polynomial-time function $Setup$ outputs the public parameters pp.

(2) $y, \alpha \xleftarrow{\$} Pro(pp, m)$: by inputting the public parameters pp and the secret value m, the polynomial-time function Pro outputs two commitments y and α.

(3) $s \xleftarrow{\$} Chal(pp)$: by inputting the public parameters pp, the polynomial-time function $Chal$ outputs a random number s.

DOI:10.6814/NCCU202101402

(4) $\mu, \nu \xleftarrow{\$} Res(pp, m, r, x, z, s)$: by inputting the public parameters pp, the screct value m, random numbers r, x, z, and the challenge s, the polynomial-time function Res outputs two responses μ and ν.

(5) $b \leftarrow Ver(pp, y, \alpha, \mu, \nu)$: by inputting the public parameters pp, the two commitments y, α, and two responses μ, ν, the polynomial-time function Ver outputs a result $b \in \{0,1\}$. The verification is accepted if b is equal to 1, which means that the two commitments y and α hide the secret value m. Otherwise, it is rejected.

In the following, we describe the protocol of ZKPCS in detail. The prover is denoted as P, and the verifier is denoted as V.

Assume that P knows a secret value $m \in \mathbb{Z}_q^*$.

(1) P computes the commitment

$$y = m \cdot G + r \cdot H$$

and randomly chooses $x, r \in \mathbb{Z}_q^*$ to compute another commitment

$$\alpha = x \cdot G + z \cdot H,$$

where G and H which are two points on $E(\mathbb{F}_p)$ are the public parameters. Then, P publishes y, α.

(2) V gives P a challenge $s \in \mathbb{Z}_q^*$.

(3) P returns two responses

$$\mu = x - sm,$$
$$\nu = z - sr.$$

(4) V computes $\mu \cdot G + \nu \cdot H + s \cdot y$ and can be convinced that P knows the committed value m if and only if

$$\alpha = \mu \cdot G + \nu \cdot H + s \cdot y.$$

In the detail,

$$\mu \cdot G + \nu \cdot H + s \cdot y$$
$$= (x - sm) \cdot G + (z - sr) \cdot H + s \cdot (m \cdot G + r \cdot H)$$

DOI:10.6814/NCCU202101402

$$= x \cdot G - sm \cdot G + z \cdot H - sr \cdot H + sm \cdot G + sr \cdot H$$
$$= x \cdot G + z \cdot H$$
$$= \alpha.$$

The verifier can be convinced that the commitment α committed by the prover is another commitment of m, i.e., the prover definitely knows m.

DOI:10.6814/NCCU202101402

3 Non-Interactive EC-ZKRP Scheme

In this chapter, we introduce the definitions, protocol, and security description of the non-interactive EC-ZKRP scheme in detail.

3.1 Definitions and Security Models

Definition 16 (Non-interactive zero-knowledge range proof). The non-interactive zero-knowledge range proof is composed by three functions $(Setup, Pro, Ver)$.

(1) $pp \xleftarrow{\$} Setup(\lambda)$: by inputting a secure parameter λ, the polynomial-time function $Setup$ outputs the public parameters pp.

(2) $\pi \xleftarrow{\$} Pro(pp, m, a, b)$: by inputting the public parameters pp, the secret value m, and the lower bound and upper bound of a range a, b, where, $a < b$, the polynomial-time function Pro outputs the proof π.

(3) $r \leftarrow Ver(pp, \pi, a, b)$: by inputting the public parameters pp, the proof π, and the lower bound and upper bound of a range a, b, the polynomial-time function Ver outputs a result $r \in \{0,1\}$. The proof π is accepted if r is equal to 1, which means that the secret value m is in the range $[a, b]$, i.e., $a \leq m \leq b$. Otherwise, it is rejected, i.e., $m \notin [a, b]$.

A ZKRP scheme satisfies the correctness property: if the secret value is exactly in the specified range, the proof must pass the verification. Therefore, the verifier can confirm that the secret value must be in the specified range.

Definition 17 (Correctness of ZKRP). If the secret value m is exactly in the range $[a, b]$ and the scheme ZKRP $= (Setup, Pro, Ver)$ satisfies the correctness property, the probability

$$\left| \Pr\left[Ver(pp, \pi) = 1 : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ m \in [a, b]; \\ \pi \xleftarrow{\$} Pro(pp, m, a, b) \end{array} \right] - 1 \right| \leq negl(\lambda)$$

for all security parameter λ.

DOI:10.6814/NCCU202101402

A ZKRP scheme satisfies the soundness property: if the prover uses a secret value that is not in the specified range to produce a proof, then it cannot pass the verification. In other words, if the proof produced by the prover can pass the verification, the secret value must be in the specified range.

Definition 18 (Soundness of ZKRP). If the secret value m is not in the range $[a, b]$ and the scheme ZKRP $= (Setup, Pro, Ver)$ satisfies the soundness property, the probability

$$\Pr\left[Ver(pp, \pi) = 1 : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ m \notin [a, b]; \\ \pi \xleftarrow{\$} Pro(pp, m, a, b) \end{array}\right] \leq negl(\lambda)$$

for all security parameter λ.

A ZKRP scheme satisfies the zero-knowledge property: the verifier can only confirm whether the secret value is within the specified range, but cannot know the exact secret value through the proof produced by the prover. More precisely, there does not exist any algorithm in probabilistic-polynomial time that can distinguish the real proof from the ideal proof produced by a simulator, which does not contain any information about the secret value.

Definition 19 (Zero-knowledge of EC-ZKRP). Given a polynomial-time simulator SIM that can produce a proof without inputting the secret m. If the scheme ZKRP $= (Setup, Pro, Ver)$ satisfies the zero-knowledge property, the probability

$$\left|\Pr\left[b' = b : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ \pi_0 \xleftarrow{\$} Pro(pp, m, a, b); \\ \pi_1 \xleftarrow{\$} SIM(pp); \\ b = \{0,1\}; \\ b' \leftarrow \mathcal{A}(pp, \pi_b) \end{array}\right] - \frac{1}{2}\right| \leq negl(\lambda)$$

for all probabilistic-polynomial time adversaries $\mathcal{A}$ and security parameter λ.

3.2 Non-Interactive EC-ZKRP Protocol

DOI:10.6814/NCCU202101402

In this section, we introduce the EC-ZKRP protocol. The scheme proposed by Tsai et al. [28] is constructed with the Fujisaki-Okamato commitment scheme [9]. To apply elliptic curve to the non-interactive ZKRP , we replace the commitment scheme with the elliptic curve Pedersen commitment scheme as described in **Section 2.3** and the EC-EL proof (**Section 2.6**) and EC-SQR proof (**Section 2.7**) both can be applied in our scheme. The core idea of our scheme is the same as the scheme proposed by Tsai et al. [28]. Assume that the prover knows the secret value m in the range $[a, b]$, where $a, b \in \mathbb{Z}_q^*$. Therefore, $a \leq m \leq b$, i.e., $m - a \geq 0$ and $b - m \leq 0$. Then we have $m - a + 1 > 0$ and $b - m + 1 > 0$, so $(m - a + 1)(b - m + 1)$ must be greater than 0. If we prove to the verifier that m is in the range $[a, b]$ by revealing $(m - a + 1)(b - m + 1) > 0$, it is not difficult to compute the secret value m through $(m - a + 1)(b - m + 1)$ when the verifier knows the range $[a, b]$. Thus, we use

$$\omega^2(m - a + 1)(b - m + 1) > 0$$

instead, where $\omega \overset{\$}{\leftarrow} \mathbb{Z}_q^*$. To prover $\omega^2(m - a + 1)(b - m + 1)$ is positive, the prover randomly chooses $M = \alpha^2 \in \mathbb{Z}_q^*$ such that

$$M + R = \omega^2(m - a + 1)(b - m + 1).$$

If the verifier verifies that M is a square number and $R > 0$, he/she can be convinced that $\omega^2(m - a + 1)(b - m + 1)$ is greater than 0, i.e., the secret value m must be in the range $[a, b]$.

We describe the non-interactive EC-ZKRP protocol in detail below: the prover knows the secret value m in the range $[a, b]$ and randomly chooses $r \in \mathbb{Z}_q^*$ to compute the commitment

$$C = m \cdot G + r \cdot H.$$

To produce the non-interactive EC-ZKRP:

(1) The prover randomly chooses $r' \in \mathbb{Z}_q^*$ to compute

$$C_1 = C - (a - 1) \cdot G,$$
$$C_2 = (b + 1) \cdot G - C,$$

DOI:10.6814/NCCU202101402

$$C' = (b - m + 1) \cdot C_1 + r' \cdot H.$$

(2) Since the two commitments

$$C_2 = (b + 1) \cdot G - C = (b - m + 1) \cdot G + (-r) \cdot H,$$
$$C' = (b - m + 1) \cdot C_1 + r' \cdot H$$

hide the same value $(b - m + 1)$, the prover produces the EL proof:

$$EL = \mathrm{EL}(b - m + 1, -r, r', G, H, C_1, H).$$

(3) The prover randomly chooses ω and r'' to compute

$$C'' = \omega^2 \cdot C' + r'' \cdot H.$$

(4) Since the commitment C'' hides a square number ω^2, the prover produces the SQR proof:

$$SQR_1 = SQR(\omega, r'', C', H).$$

(5) The prover randomly chooses $\alpha \in \mathbb{Z}_q^*$ to compute

$$M = \alpha^2.$$

If $M \geq \omega^2(m - a + 1)(b - m + 1)$, repeat (5).

(6) The prover computes

$$R = \omega^2(m - a + 1)(b - m + 1) - M,$$

which must be greater than 0.

(7) The prover randomly chooses $r_1 \in \mathbb{Z}_q^*$ to compute

$$r_2 = \omega^2\big((b - m + 1)r + r'\big) + r'' - r_1.$$

(8) The prover computes

$$C_1' = M \cdot G + r_1 \cdot H,$$
$$C_2' = r_2 \cdot H.$$

DOI:10.6814/NCCU202101402

(9) Since the commitment C_1' hides a square number M, the prover produces the SQR proof:

$$SQR_2 = SQR(\alpha, r_1, G, H).$$

(10) The prover produces the proof

$$\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}.$$

To verify the non-interactive EC-ZKRP proof π, the verifier runs the following steps:

(1) Compute $C_1 = C - (a-1) \cdot G$.
(2) Compute $C_2 = (b+1) \cdot G - C$.
(3) Verify vEL$(EL, G, H, C_1, H, C_2, C')$.
(4) Verify vSQR(SQR_1, C', H, C'').
(5) Verify $C'' = C_1' + C_2' + R \cdot G$.
(6) Verify vSQR(SQR_2, G, H, C_1').
(7) Verify $R > 0$.

The verifier can be convinced that the secret value m must be in the range $[a, b]$ if and only if the verification step 3 to step 7 are passed.

DOI:10.6814/NCCU202101402

$C = m \cdot G + r \cdot H$, where $r \xleftarrow{\$} \mathbb{Z}_q^*$
$C_1 = C - (a-1) \cdot G$
$C_2 = (b+1) \cdot G - C$
$C' = (b-m+1) \cdot C_1 + r' \cdot H$, where $r' \xleftarrow{\$} \mathbb{Z}_q^*$
$EL \xleftarrow{\$} \mathrm{EL}(b-m+1, -r, r', G, H, C_1, H)$
$C'' = \omega^2 \cdot C' + r'' \cdot H$, where $\omega, r'' \xleftarrow{\$} \mathbb{Z}_q^*$
$SQR_1 \xleftarrow{\$} \mathrm{SQR}(\omega, r'', C', H)$
$M = \alpha^2$, where $\alpha \xleftarrow{\$} \mathbb{Z}_q^*$
$R = \omega^2(m-a+1)(b-m+1) - M$
$r_1 + r_2 = \omega^2\big((b-m+1)r + r'\big) + r''$, where $r_2 \xleftarrow{\$} \mathbb{Z}_q^*$
$C_1' = M \cdot G + r_1 \cdot H$
$C_2' = r_2 \cdot H$
$SQR_2 \xleftarrow{\$} \mathrm{SQR}(\alpha, r_1, G, H)$
Produce the proof $\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}$

Figure 1: Protocol to Produce EC-ZKRP Proof

To verify the proof $\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}$

(1) Compute $C_1 = C - (a-1) \cdot G$.
(2) Compute $C_2 = (b+1) \cdot G - C$.
(3) Verify $\mathrm{vEL}(EL, G, H, C_1, H, C_2, C')$.
(4) Verify $\mathrm{vSQR}(SQR_1, C', H, C'')$
(5) Verify $C'' = C_1' + C_2' + R \cdot G$
(6) Verify $\mathrm{vSQR}(SQR_2, G, H, C_1')$
(7) Verify $R > 0$

Figure 2: Protocol to Verify EC-ZKRP Proof

DOI:10.6814/NCCU202101402

3.3 Security Descriptions

In this section, we describe the security properties of our EC-ZKRP scheme: correctness, soundness, zero-knowledge.

3.3.1 Correctness

Assume that an honest prover follows our protocol and uses the secret value m in the range $[a, b]$ to produce the proof $\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}$, the verifier can determine whether π passes the verification through the following steps:

(1) Compute $C_1 = C - (a-1) \cdot G$.

(2) Compute $C_2 = (b+1) \cdot G - C$.

(3) Verify vEL(EL).

(4) Verify vSQR(SQR_1).

(5) Verify $C'' = C_1' + C_2' + R \cdot G$.

(6) Verify vSQR(SQR_2).

(7) Verify $R > 0$.

We explain that our protocol satisfies the correctness property below:

If the prover is honest and follows our protocol to produce the proof π, then (1)(2)(7) must be correct.

By expanding C_2, we obtain

$$C_2 = (b+1) \cdot G - C = (b-m+1) \cdot G + (-r) \cdot H,$$

its committed value is $(b-m+1)$, which is same as the committed value of C'. Therefore, if the EC-EL proof is correct, then (3) must be correct.

Considering $C'' = \omega^2 \cdot C' + r'' \cdot H$, the committed value of C'' is ω^2, which is a square number. Therefore, if the EC-SQL proof is correct, then (4) must be correct.

Since we know

$$\omega^2(m-a+1)(b-m+1) = M + R,$$
$$\omega^2\big((b-m+1)r + r'\big) + r'' = r_1 + r_2$$

according to our ZKRP protocol. By expanding C'' and $C_1' + C_2' + R \cdot G$, we obtain

$$\begin{aligned} C'' &= \omega^2 \cdot C' + r'' \cdot H \\ &= \omega^2(m-a+1)(b-m+1) \cdot G + \omega^2\big((b-m+1)r + r'\big) + r'' \cdot H \\ &= (M+R) \cdot G + (r_1 + r_2) \cdot H \end{aligned}$$

DOI:10.6814/NCCU202101402

and

$$C_1' + C_2' + R \cdot G$$
$$= M \cdot G + r_1 \cdot H + r_2 \cdot H + R \cdot G$$
$$= (M + R) \cdot G + (r_1 + r_2) \cdot H.$$

The two are equal, so (5) must be correct.

Considering $C_1' = M \cdot C' + r_1 \cdot H$, the committed value of C_1' is M, which is a square number. Therefore, if the EC-SQL proof is correct, then (6) must be correct.

Because (3)(4)(5)(6)(7) are all correct and π passes the verification, the verifier can confirm that $\omega^2(m - a + 1)(b - m + 1)$ must be greater than 0. As a result, the verifier can be convinced that the secret value m must be in the range $[a, b]$.

The honest prover follows our protocol to produce the proof π that can pass the verification such that the verifier can confirm that the secret value m must be in the range $[a, b]$. Therefore, our protocol satisfies the correctness property.

3.3.2 Soundness

Theorem 4. If the prover follows our protocol to produce the proof $\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}$ that can pass the verification with non-negligible probability, the secret value m must be in the range $[a, b]$. In addition, the committed value of C must be m.

Lemma 2. If the proof $\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}$ can pass the verification with non-negligible probability, the prover must know all the secret values that are used to produce the proof π, such as M, r_1. In other words, when the prover does not know any of secret values, the prover cannot produce a proof π that can pass the verification.

Proof. We take M, r_1 as example. Use one of the steps from our protocol: $C_1' = M \cdot G + r_1 \cdot H$ to run the ZKPCS (**Section 2.8**) with two different challenges ς and ς':
In the following, the prover is denoted as P, and the verifier is denoted as V.

(1) P randomly chooses $r, s \in \mathbb{Z}_q^*$ and computes $C = r \cdot G + s \cdot H$. Then, P publishes C.
(2) V randomly chooses two different challenges $\varsigma, \varsigma' \in \mathbb{Z}_q^*$ and sends them to P.
(3) P computes two responses

$$(u = r - \varsigma M, v = s - \varsigma r_1),$$

DOI:10.6814/NCCU202101402

$$(u' = r - \varsigma' M, v' = s - \varsigma' r_1)$$

and publishes them.

(4) V computes C by the two responses:

$$(u = r - \varsigma M, v = s - \varsigma r_1) \rightarrow C = u \cdot G + v \cdot H + \varsigma \cdot C_1',$$
$$(u' = r - \varsigma' M, v' = s - \varsigma' r_1) \rightarrow C = u' \cdot G + v' \cdot H + \varsigma' \cdot C_1'.$$

We subtract the two equations, and we obtain

$$0 = (u - u')G + (v - v')H + (\varsigma - \varsigma')C_1'.$$

Because $\varsigma \neq \varsigma'$, $\varsigma - \varsigma' \neq 0$. Therefore,

$$C_1' = \frac{(u - u')}{(\varsigma' - \varsigma)} \cdot G + \frac{(v - v')}{(\varsigma' - \varsigma)} \cdot H.$$

And then we know $C_1' = M \cdot G + r_1 \cdot H$, so

$$M = \frac{(u - u')}{(\varsigma' - \varsigma)} \bmod q,$$
$$r_1 = \frac{(u - u')}{(\varsigma' - \varsigma)} \bmod q.$$

In addition, since q which is the order of the points G and H is a prime number,

$$GCD(q, \varsigma' - \varsigma) = 1.$$

As a result, the inverse of $\varsigma' - \varsigma$, $\frac{1}{\varsigma' - \varsigma}$, must exist. Therefore, M, r_1 must exist, and the prover must know them. Similarly, the prover must know all the committed values that are used to produce the proof π and these values must exist, so **Lemma 2** is proved.

Lemma 3. If the proof $\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}$ can pass the verification with non-negligible probability, the committed value of C'', $\omega^2(m - a + 1)(b - m + 1)$, must greater than 0.

DOI:10.6814/NCCU202101402

Proof. According to **Lemma 2**, if the proof π can pass the verification, the prover must know M, r_1, r_2. According to our protocol, $C_1' = M \cdot G + r_1 \cdot H$, $C_2' = r_2 \cdot H$. In addition, the verifier can compute $C'' = C_1' + C_2' + R \cdot G$ through C_1', C_2', and R. We know $C'' = (M + R) \cdot G + (r_1 + r_2) \cdot H$. If π pass the verification step 6 and step 7, it means that M is a square number and $R > 0$. Obviously, the committed value of C'' is greater than 0, i.e., $\omega^2(m - a + 1)(b - m + 1) = M + R > 0$, so **Lemma 3** is proved.

According to the binding property of the EC-Pederesen commitment scheme (**Lemma 1**), **Lemma 2**, and **Lemma 3**, If the proof π can pass the verification with non-negligible probability, the committed value of C'' must be $\omega^2(m - a + 1)(b - m + 1)$, and $\omega^2(m - a + 1)(b - m + 1)$ must be greater than 0. In other words, when the proof π pass the verification, the verifier can confirm that the secret value m must be in the range $[a, b]$. As a result, **Theorem 4** always holds, our protocol satisfies the soundness property.

3.3.3 Zero-knowledge

Theorem 5. Assume that there exists a simulator S follows our protocol, but replaces all secret values that only the prover knows (e.g., m, r) with random numbers to produce the proof π', and π' can be verified. For all adversaries, there does not exist any algorithm in probabilistic-polynomial time to distinguish between the proof π produced by the prover and the proof π' produced by the simulator S.

Proof. To prove **Theorem 5**, we use game hopping. First, we define two games: the first game $\mathcal{G}_1$ (real game) follows our protocol to produce the proof. Secondly, the game $\mathcal{G}_2$ (ideal game) is to change the steps of $\mathcal{G}_1$ so that the proof produced by $\mathcal{G}_2$ does not contain any information of secret values. Lastly, we argue the computationally indistinguishable of the two games $\mathcal{G}_1$ and $\mathcal{G}_2$, to prove that **Theorem 5** holds.

According to the zero-knowledge property of EC-EL proof (**Theorem 2**), no adversary can distinguish the proof produced by simulator $\mathcal{ELS}$ from the real EC-EL proof.

We describe the two games $\mathcal{G}_1$ and $\mathcal{G}_2$ and explain their differences in detail. The complete game description is shown in **Figure 3**, where "chg" means to change $\mathcal{G}_1$ to a new instruction, and "del" means to delete the instruction of $\mathcal{G}_1$.

DOI:10.6814/NCCU202101402

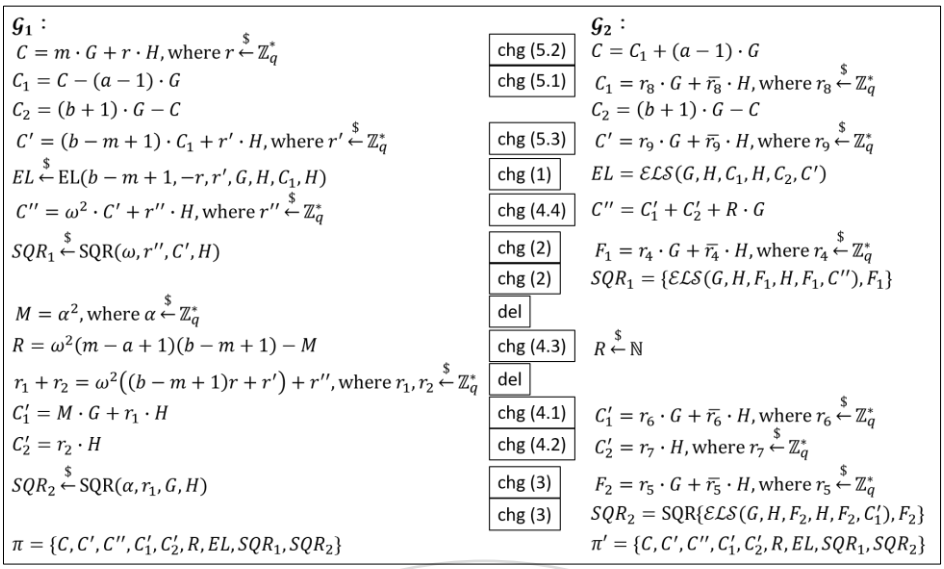

Figure 3: Proof Games $\mathcal{G}_1, \mathcal{G}_2$ of **Theorem 5**

$\mathcal{G}_1$: This game is the real model of our ZKRP protocol.

$\mathcal{G}_2$: This game is the ideal model. In this game, the EC-EL proof is replaced by the simulator $\mathcal{ELS}$ to produce the EL proof and the SQR proof, and any values that related to the secret value is replaced by a random number, so the proof produced by $\mathcal{G}_2$ does not contain any information of secret values.

In the following, we explain the computationally indistinguishable of the two games $\mathcal{G}_1$ and $\mathcal{G}_2$.

(1) In "chg 1", $\mathcal{G}_2$ replaces the proof produced by the EC-EL proof with the proof produced by the simulator $\mathcal{ELS}$. If there exists a distinguisher $\mathcal{D}$ in probabilistic-polynomial time such that the adversary can distinguish between the two proofs EL produced by $\mathcal{G}_1$ and $\mathcal{G}_2$, it breaks the zero-knowledge property of EC-EL proof (**Theorem 2**). Therefore, if the adversary follows the verification steps to verify the two proofs produced by $\mathcal{G}_1$ and $\mathcal{G}_2$, both can pass the verification step 3 to step 7. Thus, they cannot distinguish between the two proofs.

(2) In "chg 2", the adversary follows $\mathcal{G}_1$ to produce the proof SQR_1. First, the adversary computes

$$F_1 = \omega \cdot G + r_2' \cdot H \ , \ r_2' \xleftarrow{\$} \mathbb{Z}_q^*$$

and then computes

DOI:10.6814/NCCU202101402

$$E' = \omega \cdot F_1 + r_3' \cdot H.$$

After that, the adversary produces the proof

$$EL \xleftarrow{\$} \mathrm{EL}(\omega, r_2', r_3', G, H, F_1, \mathrm{H})$$

through F_1 and E'. Finally, the adversary produces the proof

$$SQR_1 = \{\mathrm{EL}(\omega, r_2', r_3', G, H, F_1, \mathrm{H}), F_1\}.$$

If there exists a distinguisher $\mathcal{D}$ in probabilistic-polynomial time such that the adversary can distinguish between the two proofs SQR_1 produced by $\mathcal{G}_1$ and $\mathcal{G}_2$, it means that $\mathcal{D}$ can distinguish between the EL proofs in the two SQR_1. This breaks the zero-knowledge property of EC-EL proof (**Theorem 2**). In addition, it can be seen that "chg 3" is the same as "chg 2". Therefore, if the adversary follows the verification steps to verify the two proofs produced by $\mathcal{G}_1$ and $\mathcal{G}_2$, both can pass the verification step 3 to step 7. Thus, they cannot distinguish between the two proofs.

(3) In “chg 4.1”, the adversary follows $\mathcal{G}_1$ to compute

$$C_1' = M \cdot G + r_1 \cdot H,$$

and then follows $\mathcal{G}_2$ to compute

$$C_1' = r_6 \cdot G + \overline{r_6} \cdot H \text{ , } r_6 \xleftarrow{\$} \mathbb{Z}_q^*.$$

If there exists a distinguisher $\mathcal{D}$ in probabilistic-polynomial time such that the adversary can distinguish between the two C_1' in $\mathcal{G}_1$ and $\mathcal{G}_2$, it means that the adversary can distinguish between the two committed values M and r_6. This breaks the hiding property of the elliptic-curve Pedersen commitment scheme.

In “chg 4.2”, the adversary follows $\mathcal{G}_1$ to compute

$$C_2' = r_2 \cdot H,$$

and then follows $\mathcal{G}_2$ to compute

DOI:10.6814/NCCU202101402

$$C_2' = r_7 \cdot H \text{ , } r_7 \overset{\$}{\leftarrow} \mathbb{Z}_q^*.$$

If there exists a distinguisher $\mathcal{D}$ in probabilistic-polynomial time such that the adversary can distinguish between the two C_2' in $\mathcal{G}_1$ and $\mathcal{G}_2$, it means that the adversary can compute r_2, r_7 through the two C_2'. That is, $\mathcal{D}$ can solve the elliptic-curve discrete logarithm problem [12] in probabilistic-polynomial time.

In "chg 4.3", The value of R in $\mathcal{G}_2$ must be positive, so the proof π produced by $\mathcal{G}_2$ can pass the verification step 7. Therefore, the adversary cannot distinguish the two R in $\mathcal{G}_1$ and $\mathcal{G}_2$.

In "chg 4.4", the adversary follows $\mathcal{G}_2$ to compute

$$C'' = C_1' + C_2' + R \cdot G,$$

so the proof π produced by $\mathcal{G}_2$ can pass the verification step 5. Therefore, the adversary cannot distinguish the two C'' in $\mathcal{G}_1$ and $\mathcal{G}_2$. If the adversary follows the verification steps to verify the two proofs produced by $\mathcal{G}_1$ and $\mathcal{G}_2$, both can pass the verification step 3 to step 7. Thus, they cannot distinguish between the two proofs.

(4) In "chg 5.1", the adversary follows $\mathcal{G}_1$ to compute

$$C_1 = C - (a-1) \cdot G = (m-a+1) \cdot G + r \cdot H,$$

and then follows $\mathcal{G}_2$ to compute

$$C_1 = r_8 \cdot G + \overline{r_8} \cdot H \text{ , } r_8 \overset{\$}{\leftarrow} \mathbb{Z}_q^*.$$

If there exists a distinguisher $\mathcal{D}$ in probabilistic-polynomial time such that the adversary can distinguish between the two C_1 in $\mathcal{G}_1$ and $\mathcal{G}_2$, it means that the adversary can distinguish between the two committed values $(m-a+1)$ and r_8. This breaks the hiding property of the elliptic-curve Pedersen commitment scheme.

In "chg 5.2", the adversary follows $\mathcal{G}_1$ to compute

$$C = m \cdot G + r \cdot H,$$

DOI:10.6814/NCCU202101402

and then follows $\mathcal{G}_2$ to compute

$$C = C_1 + (a-1)\cdot G = (r_8 + a - 1)\cdot G + r\cdot H.$$

This is the same as "chg 5.1".
In "chg 5.2", the adversary follows $\mathcal{G}_1$ to compute

$$\begin{aligned} C' &= (b-m+1)\cdot C_1 + r'\cdot H \\ &= (m-a+1)(b-m+1)\cdot G + \big((b-m+1)r + r'\big)\cdot H, \end{aligned}$$

and then follows $\mathcal{G}_2$ to compute

$$C' = r_9\cdot G + \overline{r_9}\cdot H \text{ , } r_9 \xleftarrow{\$} \mathbb{Z}_q^*.$$

This is the same as "chg 5.1". Therefore, if the adversary follows the verification steps to verify the two proofs produced by $\mathcal{G}_1$ and $\mathcal{G}_2$, both can pass the verification step 3 to step 7. Thus, they cannot distinguish between the two proofs.

According to (1)(2)(3)(4), there does not exist any algorithm in probabilistic-polynomial time to distinguish the two proofs produced by the game $\mathcal{G}_1$ and $\mathcal{G}_2$. That is $\mathcal{G}_1$ and $\mathcal{G}_2$ are computationally indistinguishable, so **Theorem 5** is proved.

According to **Theorem 5**, no adversary can distinguish between the two proofs produced by $\mathcal{G}_1$ and $\mathcal{G}_2$. It means that the adversary cannot determine whether the proof π produced by our protocol contains the secret value m. That is, the adversary cannot compute the secret value m through the proof π. Therefore, our protocol satisfies the zero-knowledge property.

DOI:10.6814/NCCU202101402

4 Non-Interactive Designated Verifier EC-ZKRP Protocol

In addition, we consider that the prover may not want to let everyone except the designated verifier knows the range of the secret value. That is, the proof produced by the prover cannot convince any other third parties. Thus, we require a designated verifier ZKRP scheme. Even if the designated verifier reveals the important information of the proof, any other third party cannot trust the verification result or verify the proof. There are several related works about the designated verifier non-interactive zero-knowledge proof (DV-NIZK) [7][14][17], but a DV-ZKRP scheme has not been proposed yet.

In this chapter, we introduce the non-interactive designated verifier EC-ZKRP scheme (DV-EC-ZKRP) and the non-interactive strong designated verifier EC-ZKRP scheme (SDV-EC-ZKRP) which are based on our EC-ZKRP scheme as described in **Chapter 3**, but the commitment scheme is replaced by the trapdoor commitment scheme (**Section 2.5**). The definitions of the designated verifier and strong designated verifier in this thesis follow by the definitions proposed by Jakobsson et al. [13], i.e., the designated verifier proof cannot be trusted by any third party, while the strong designated verifier proof cannot be verifier by any third party. However, the designated verifier is strictly defined in the schemes [7][14][17] mentioned above, which is different from our definitions: the designated verifier proof cannot be verified by any third party. Therefore, our definition of the strong designated verifier can correspond to the definition of the designated verifier in the schemes [7][14][17]. In the following, we describe the definitions and the protocols in detail.

4.1 Designated Verifier EC-ZKRP Scheme

In this section, we introduce the definitions, protocol, and security description of the DV-EC-ZKRP scheme in detail.

4.1.1 Definitions and Security Models

A scheme satisfies the designated verifier property: the prover follows the scheme to produce the proof which can only convince the designated verifier, and any other

DOI:10.6814/NCCU202101402

third party cannot be convince by the proof. More precisely, there does not exist any algorithm in probabilistic-polynomial time to distinguish between the proof produced by the prover and the proof produced by the verifier. Since any third parties can think the proof is produced by the designated verifier, they cannot trust the proof.

Definition 20 (Designated verifier). Let $P(A,B)$ is a designated verifier protocol which allows a prover A to produce a proof π to prove to B that a statement θ is true, and there is another protocol $P'(B,C)$ such that B can prove the truth of θ to C. The probability

$$\left|\Pr\left[b' = b : \begin{matrix}\pi_0 \leftarrow P(A,B);\\ \pi_1 \leftarrow P'(B,C);\\ b \in \{0,1\};\\ b' \leftarrow \mathcal{A}(\pi_b)\end{matrix}\right] - \frac{1}{2}\right| \leq negl(\lambda)$$

for all probabilistic-polynomial time adversaries $\mathcal{A}$ and security parameter λ.

Definition 21 (Non-interactive designated verifier zero-knowledge range proof, DV-ZKRP). DV-ZKRP is composed by three functions $(Setup, Pro, Ver)$.

(1) $pp \xleftarrow{\$} Setup(\lambda)$: by inputting a secure parameter λ, the polynomial-time function $Setup$ outputs the public parameters pp.

(2) $\pi \xleftarrow{\$} Pro(pp, pk_v, m, a, b)$: by inputting the public parameters pp, the public key of the designated verifier pk_v, the secret value m, and the lower bound and upper bound of a range a, b, where $a < b$, the polynomial-time function Pro outputs the proof π.

(3) $r \leftarrow Ver(pp, pk_v, \pi, a, b)$: by inputting the public parameters pp, the public key of the designated verifier pk_v, the proof π, and the lower bound and upper bound of a range a, b, the polynomial-time function Ver outputs a result $r \in \{0,1\}$. The proof π is accepted if r is equal to 1, which means that the secret value m is in the range $[a,b]$, i.e., $a \leq m \leq b$. Otherwise, it is rejected, i.e., $m \notin [a,b]$.

A DV-ZKRP scheme satisfies the designated verifier property: there does not exist any algorithm in probabilistic-polynomial time to distinguish between the proof produced by the prover and the proof produced by the designated verifier.

DOI:10.6814/NCCU202101402

Definition 22 (Designated verifier of DV-ZKRP). Given a prover A and a designated verifier B. The probability

$$\left|\Pr\left[b' = b : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ \pi_0 \leftarrow A\big(Pro(pp, pk_B, m, a, b)\big); \\ \pi_1 \leftarrow B\big(Pro(pp, pk_B, m, a, b)\big); \\ b \in \{0,1\}; \\ b' \leftarrow \mathcal{A}(pp, \pi_b) \end{array}\right] - \frac{1}{2}\right| \leq negl(\lambda)$$

for all probabilistic-polynomial time adversaries $\mathcal{A}$ and security parameter λ.

4.1.2 Designated Verifier EC-ZKRP Protocol

We describe the DV-EC-ZKRP protocol in detail below: the prover knows the secret value m in the range $[a, b]$, and the prover wants to prove to a designated verifier that m is in the range $[a, b]$ without revealing m. Given that the private key of the designated verifier is X and the public key is Y, where $Y = X \cdot G$. G and Y are two points on a curve $E(\mathbb{F}_p)$ have the order q, and $X \in \mathbb{Z}_q^*$. The prover randomly chooses $r \in \mathbb{Z}_q^*$ to compute the commitment

$$C = m \cdot G + r \cdot Y.$$

by using the public key of the verifier Y.

To produce the non-interactive DV-EC-ZKRP:

(1) The prover randomly chooses $r' \in \mathbb{Z}_q^*$ to compute

$$\begin{aligned} C_1 &= C - (a-1) \cdot G, \\ C_2 &= (b+1) \cdot G - C, \\ C' &= (b-m+1) \cdot C_1 + r' \cdot Y. \end{aligned}$$

(2) Since the two commitments

$$\begin{aligned} C_2 &= (b+1) \cdot G - C = (b-m+1) \cdot G + (-r) \cdot Y, \\ C' &= (b-m+1) \cdot C_1 + r' \cdot Y \end{aligned}$$

hide the same value $(b-m+1)$, the prover produces the EL proof:

DOI:10.6814/NCCU202101402

$$EL = \mathrm{EL}(b - m + 1, -r, r', G, Y, C_1, Y).$$

(3) The prover randomly chooses ω and r'' to compute

$$C'' = \omega^2 \cdot C' + r'' \cdot Y.$$

(4) Since the commitment C'' hides a square number ω^2, the prover produces the SQR proof:

$$SQR_1 = SQR(\omega, r'', C', Y).$$

(5) The prover randomly chooses $\alpha \in \mathbb{Z}_q^*$ to compute

$$M = \alpha^2.$$

If $M \geq \omega^2(m - a + 1)(b - m + 1)$, repeat (5).

(6) The prover computes

$$R = \omega^2(m - a + 1)(b - m + 1) - M,$$

which must be greater than 0.

(7) The prover randomly chooses $r_1 \in \mathbb{Z}_q^*$ to compute

$$r_2 = \omega^2\big((b - m + 1)r + r'\big) + r'' - r_1.$$

(8) The prover computes

$$C_1' = M \cdot G + r_1 \cdot Y,$$
$$C_2' = r_2 \cdot Y.$$

(9) Since the commitment C_1' hides a square number M, the prover produces the SQR proof:

$$SQR_2 = SQR(\alpha, r_1, G, Y).$$

(10) The prover produces the proof

$$\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}.$$

DOI:10.6814/NCCU202101402

To verify the non-interactive DV-EC-ZKRP proof π, the designated verifier runs the following steps:

(1) Compute $C_1 = C - (a-1) \cdot Y$.

(2) Compute $C_2 = (b+1) \cdot G - C$.

(3) Verify $\text{vEL}(EL, G, Y, C_1, Y, C_2, C')$.

(4) Verify $\text{vSQR}(SQR_1, C', Y, C'')$.

(5) Verify $C'' = C_1' + C_2' + R \cdot G$.

(6) Verify $\text{vSQR}(SQR_2, G, Y, C_1')$.

(7) Verify $R > 0$.

The designated verifier can be convinced that the secret value m must be in the range $[a, b]$ if and only if the verification step 3 to step 7 are passed.

If any other third parties follow the verification steps to verify the proof, they cannot accept the verification result even if the step 3 to step 7 are passed, because they could think the prover and the designated verifier cheat together since the verifier knows X, which is the relation of G and Y, i.e., the verifier can find $m_1, m_2, r_1, r_1 \in \mathbb{Z}_q^*$ easily such that

$$C = m_1 \cdot G + r_1 \cdot Y = m_2 \cdot G + r_2 \cdot Y.$$

That is, for any other third parties, the commitments that are used to produce the proof does not satisfy the binding property. Therefore, the verification result can be only accepted by the designated verifier.

4.1.3 Security Description: Designated Verifier

Our DV-EC-ZKRP scheme satisfies the designated verifier property: there does not exist any algorithm in probabilistic-polynomial time to distinguish between the proof produced by the prover and the proof produced by the designated verifier. In other words, the designated verifier can find two different secret values to produce the same proof such that any third party cannot trust the proof.

Lemma 4. Anyone who knows the relation of the public parameters G, H can produce an EC-EL proof $EL = (h, x, x_1, x_2)$ with different inputs.

Proof. Let $\rho \in \mathbb{Z}_q^*$ be the the relation of the two sets of public parameters $(G_1, H_1), (G_2, H_2)$, i.e., $H_1 = \rho G_1, H_2 = \rho G_2$. Assume that an adversary $\mathcal{A}$ knows ρ

DOI:10.6814/NCCU202101402

and the two commitments

$$A = m \cdot G_1 + s \cdot H_1,$$
$$B = m \cdot G_2 + r \cdot H_2,$$

where A and B have the same committed value $m \in \mathbb{Z}_q^*$, $s, r \xleftarrow{\$} \mathbb{Z}_q^*$. The adversary $\mathcal{A}$ inputs $(m, r, s, G_1, H_1, G_2, H_2, A, B)$ and to produce

$$EL = (h, x, x_1, x_2): \begin{cases} \mu, v_1, v_2 \xleftarrow{\$} \mathbb{Z}_q^* \\ C_1 = \mu G_1 + v_1 H_1 \\ C_2 = \mu G_2 + v_2 H_2 \\ h = hash(C_1 || C_2) \cdot \\ x = \mu + hm \\ x_1 = v_1 + hs \\ x_2 = v_2 + hr \end{cases}$$

Then the adversary $\mathcal{A}$ can compute

$$s' = \frac{(m - m')}{\rho} + s \pmod q,$$
$$r' = \frac{(m - m')}{\rho} + r \pmod q$$

such that the two commitment

$$A = m \cdot G_1 + s \cdot H_1 = (m + \rho s) \cdot G_1 = (m' + \rho s') \cdot G_1 = m' \cdot G_1 + s' \cdot H_1,$$
$$B = m \cdot G_2 + r \cdot H_2 = (m + \rho r) \cdot G_2 = (m' + \rho r') \cdot G_2 = m' \cdot G_2 + r' \cdot H_2.$$

Finally, the adversary $\mathcal{A}$ can produce the same proof $EL = (h, x, x_1, x_2)$ with the different input $(m', r', s', G_1, H_1, G_2, H_2, A, B)$ through

$$\mu' = x - hm',$$
$$v_1' = x_1 - hs',$$
$$v_2' = x_2 - hr',$$

since

DOI:10.6814/NCCU202101402

$$
\begin{aligned}
&\mu' G_1 + v_1' H_1 \\
&= [(x - hm') + (x_1 - hs')\rho] G_1 \\
&= \big(\mu + h(m - m') + \big(v_1 + h(s - s')\big)\rho\big) G_1 \\
&= (\mu + v_1\rho) G_1 + h(m + \rho s) G_1 - h(m' + \rho s') G_1 \\
&= \mu G_1 + v_1 H_1 + A - A = C_1,
\end{aligned}
$$

and

$$
\begin{aligned}
&\mu' G_2 + v_2' H_2 \\
&= [(x - hm') + (x_2 - hr')\rho] G_2 \\
&= \big(\mu + h(m - m') + \big(v_2 + h(r - r')\big)\rho\big) G_2 \\
&= (\mu + v_2\rho) G_2 + h(m + \rho r) G_2 - h(m' + \rho r') G_2 \\
&= \mu G_2 + v_2 H_2 + B - B = C_2.
\end{aligned}
$$

Thus, the value $h = hash(C_1 || C_2) = hash\big((\mu' G_1 + v_1' H_1) || (\mu' G_2 + v_2' H_2)\big)$ does not change, i.e., the adversary $\mathcal{A}$ can produce the same EC-EL proof $EL = (h, x, x_1, x_2)$ with two different inputs

$$
\begin{aligned}
&(m, r, s, G_1, H_1, G_2, H_2, A, B), \\
&(m', r', s', G_1, H_1, G_2, H_2, A, B)
\end{aligned}
$$

by knowing ρ. Therefore, **Lemma 4** is proved.

Theorem 6. The designated verifier can find two different secret values m, m' to produce the same proof $\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}$ if the verifier has his/her private key X.

Proof. Let Y be the public key of the designated verifier, and $Y = X \cdot G$. Assume that the verifier knows a secret value $m \in [a, b]$ and produces a DV-EC-ZKRP proof

DOI:10.6814/NCCU202101402

$$\pi = \begin{pmatrix} C, C', C'', C_1', C_2', \\ R, EL, SQR_1, SQR_2 \end{pmatrix} : \begin{cases} r, r', r'', \omega, \alpha, r_2 \leftarrow \mathbb{Z}_q^*; \\ C = mG + rY; \\ C' = (b - m + 1)C + r'Y; \\ C'' = \omega^2 C' + r''Y; \\ C_1' = \alpha^2 G + \left(\omega^2\left((b - m + 1)r + r'\right) + r'' - r_2\right)Y; \\ C_2' = r_2 Y; \\ R = \omega^2(m - a + 1)(b - m + 1) - \alpha^2; \\ EL \overset{\$}{\leftarrow} \mathrm{EL}(b - m + 1, -r, r', G, Y, C - (a - 1)G, Y); \\ SQR_1 \overset{\$}{\leftarrow} \mathrm{SQR}(\omega, r'', C', Y); \\ SQR_2 \overset{\$}{\leftarrow} \mathrm{SQR}(\alpha, r_1, G, Y) \end{cases}$$

with his/her public key Y. Then the verifier sets $\widehat{\omega} = 0$ and computes

$$\hat{r} = \frac{(m - m')}{X} + r \pmod q,$$

$$\widehat{r'} = \frac{(m' - m)(m + rX)}{X} + r' \pmod q,$$

$$\widehat{r''} = \frac{(\omega^2 - \widehat{\omega}^2)\left((b - m + 1)(m - a + 1 + rX) + r'X\right)}{X} + r'' \pmod q,$$

$$\widehat{r_2} = r_2 \pmod q,$$

$$\widehat{r_1} = \widehat{\omega}^2\left((b - m' + 1)\hat{r} + \widehat{r'}\right) + \widehat{r''} - \widehat{r_2} \pmod q,$$

$$\widehat{M} = M + (r_1 - \widehat{r_1})X \pmod q, \text{where } M = \alpha^2$$

such that

$$C = mG + rY = m'G + \hat{r}Y,$$
$$C' = (b - m + 1)C + r'Y = (b - m' + 1)C + \widehat{r'}Y,$$
$$C'' = \omega^2 C' + r''Y = \widehat{\omega}^2 C' + \widehat{r''}Y,$$
$$C_1' = MG + r_1 Y = \widehat{M}G + \widehat{r_1}Y,$$
$$C_2' = r_2 Y = \widehat{r_2}Y.$$

Furthermore, according to **Lemma 4**, the verifier can produce the same EL proof EL with $(b - m' + 1)$ and the same SQR proof SQR_1, SQR_2 with $\widehat{\omega}, \widehat{M}$. Finally, the verifier can produce the same proof $\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}$ by using the different secret $m' \in \mathbb{Z}_q^*$ since

$$\widehat{\omega}^2(m' - a + 1)(b - m' + 1) - \widehat{M}$$

DOI:10.6814/NCCU202101402

$$
\begin{aligned}
&= 0 - \widehat{M} \\
&= (\widehat{r_1} - r_1)X - M \\
&= \left(\widehat{r''} - (r_1 + r_2)\right)X - M \\
&= \left(\omega^2\left((b - m + 1)(m - a + 1 + rX) + r'X\right) + r''X\right) \\
&\qquad - \left(\omega^2\left((b - m + 1)r + r'\right) + r''\right)X - M \\
&= \omega^2(b - m + 1)(m - a + 1) - M \\
&= R.
\end{aligned}
$$

Thus, it can be seen that the designated verifier can produce the same proof $\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}$ with two different secret m, m' if the verifier knows his/her private key X. Therefore, **Theorem 6** is proved. The DV-EC-ZKRP scheme satisfies the designated verifier property.

4.2 Strong Designated Verifier EC-ZKRP Scheme

Due to the confidentiality of the secret value, sometimes we would everyone except the designated verifier like to not only be unable to trust the proof, but also cannot verify the proof. Therefore, we require a strong designated verifier ZKRP scheme.

A strong designated verifier ZKRP scheme can make any third be unable to verify the proof, or trust the proof even if they receive the key used to produce the proof, while a designated verifier ZKRP scheme can only make any third party does not trust the proof, but they can still verify the proof.

In this section, we introduce the definitions, protocol, and security description of the SDV-EC-ZKRP scheme in detail.

4.2.1 Definitions and Security Models

A scheme satisfies the strong designated verifier property: the prover follows the scheme to produce the proof which can only convince the designated verifier, and any other third party cannot follow the scheme to verify the proof. More precisely, there does not exist any algorithm in probabilistic-polynomial time to distinguish between two proofs produced by different provers.

Definition 23 (Strong designated verifier). Let $P(A, B)$ is a designated verifier

DOI:10.6814/NCCU202101402

protocol which allows a prover A to produce a proof π to prove to B that a statement θ is true, and there is another protocol $P'(C,D)$ such that C can prove the truth of θ to D. The probability

$$\left|\Pr\left[b'=b:\begin{matrix}\pi_0\leftarrow P(A,B);\\ \pi_1\leftarrow P'(C,D);\\ b\in\{0,1\};\\ b'\leftarrow\mathcal{A}(\pi_b)\end{matrix}\right]-\frac{1}{2}\right|\leq negl(\lambda)$$

for all probabilistic-polynomial time adversaries $\mathcal{A}$ and security parameter λ.

Definition 24 (Non-interactive strong designated verifier zero-knowledge range proof, SDV-ZKRP). SDV-ZKRP is composed by three functions $(Setup, Pro, Ver)$.

(1) $pp \xleftarrow{\$} Setup(\lambda)$: by inputting a secure parameter λ, the polynomial-time function $Setup$ outputs the public parameters pp.

(2) $\pi \xleftarrow{\$} Pro(pp, S_{pv}, m, a, b)$: by inputting the public parameters pp, the shared key of the prover and the designated verifier S_{pv}, the secret value m, and the lower bound and upper bound of a range a, b, where $a < b$, the polynomial-time function Pro outputs the proof π.

(3) $r \leftarrow Ver(pp, S_{pv}, \pi, a, b)$: by inputting the public parameters pp, the shared key of the prover and the designated verifier S_{pv}, the proof π, and the lower bound and upper bound of a range a, b, the polynomial-time function Ver outputs a result $r \in \{0,1\}$. The proof π is accepted if r is equal to 1, which means that the secret value m is in the range $[a, b]$, i.e., $a \leq m \leq b$. Otherwise, it is rejected, i.e., $m \notin [a, b]$.

A SDV-ZKRP scheme satisfies the strong designated verifier property: there does not exist any algorithm in probabilistic-polynomial time to distinguish between two proofs produced by different provers.

Definition 25 (Strong designated verifier of SDV-EC-ZKRP). Given two provers A and C, and two designated verifiers B and D. The probability

DOI:10.6814/NCCU202101402

$$\left|\Pr\left[b' = b : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ \pi_0 \leftarrow A\big(Pro(pp, S_{AB}, m, a, b)\big); \\ \pi_1 \leftarrow C\big(Pro(pp, S_{CD}, m, a, b)\big); \\ b \in \{0,1\}; \\ b' \leftarrow \mathcal{A}(pp, \pi_b) \end{array}\right] - \frac{1}{2}\right| \leq negl(\lambda)$$

for all probabilistic-polynomial time adversaries $\mathcal{A}$ and security parameter λ.

4.2.2 Strong Designated Verifier EC-ZKRP Protocol

We describe the SDV-EC-ZKRP protocol in detail below: the prover knows the secret value m in the range $[a, b]$, and the prover wants to prove to a designated verifier that m is in the range $[a, b]$ without revealing m. The prover has the private key X_p and the public key $Y_p = X_p \cdot G$, and the designated verifier has the private key X_v and the public key $Y_v = X_v \cdot G$. They runs ECDH as described in **Section 2.4** to get the shared key

$$S = X_p \cdot Y_v = X_v \cdot Y_p = X_p \cdot X_v \cdot G.$$

By using the public key of the designated verifier, the prover computes the shared key $S = X_p \cdot Y_v$ and randomly chooses $r \in \mathbb{Z}_q$ to compute the commitment

$$C = m \cdot G + r \cdot S.$$

To produce the non-interactive SDV-EC-ZKRP:

(1) The prover randomly chooses $r' \in \mathbb{Z}_q^*$ to compute

$$C_1 = C - (a-1) \cdot G,$$
$$C_2 = (b+1) \cdot G - C,$$
$$C' = (b-m+1) \cdot C_1 + r' \cdot S.$$

(2) Since the two commitments

$$C_2 = (b+1) \cdot G - C = (b-m+1) \cdot G + (-r) \cdot S,$$
$$C' = (b-m+1) \cdot C_1 + r' \cdot S$$

hide the same value $(b-m+1)$, the prover produces the EL proof:

DOI:10.6814/NCCU202101402

$$EL = \mathrm{EL}(b - m + 1, -r, r', G, S, C_1, S).$$

(3) The prover randomly chooses ω and r'' to compute

$$C'' = \omega^2 \cdot C' + r'' \cdot S.$$

(4) Since the commitment C'' hides a square number ω^2, the prover produces the SQR proof:

$$SQR_1 = SQR(\omega, r'', C', S).$$

(5) The prover randomly chooses $\alpha \in \mathbb{Z}_q^*$ to compute

$$M = \alpha^2.$$

If $M \geq \omega^2(m - a + 1)(b - m + 1)$, repeat (5).

(6) The prover computes

$$R = \omega^2(m - a + 1)(b - m + 1) - M,$$

which must be greater than 0.

(7) The prover randomly chooses $r_1 \in \mathbb{Z}_q^*$ to compute

$$r_2 = \omega^2\big((b - m + 1)r + r'\big) + r'' - r_1.$$

(8) The prover computes

$$C_1' = M \cdot G + r_1 \cdot S,$$
$$C_2' = r_2 \cdot S.$$

(9) Since the commitment C_1' hides a square number M, the prover produces the SQR proof:

$$SQR_2 = SQR(\alpha, r_1, G, S).$$

(10) The prover produces the proof

$$\pi = \{C, C', C'', C_1', C_2', R, EL, SQR_1, SQR_2\}.$$

DOI:10.6814/NCCU202101402

To verify the non-interactive SDV-EC-ZKRP proof π, the designated verifier computes the shared key $S = X_v \cdot Y_p$ and runs the following steps:

(1) Compute $C_1 = C - (a-1) \cdot S$.

(2) Compute $C_2 = (b+1) \cdot G - C$.

(3) Verify $\text{vEL}(EL, G, S, C_1, S, C_2, C')$.

(4) Verify $\text{vSQR}(SQR_1, C', S, C'')$.

(5) Verify $C'' = C_1' + C_2' + R \cdot G$.

(6) Verify $\text{vSQR}(SQR_2, G, S, C_1')$.

(7) Verify $R > 0$.

The designated verifier can be convinced that the secret value m must be in the range $[a, b]$ if and only if the verification step 3 to step 7 are passed.

Any other third party cannot follow the verification steps to verify the proof because they are not able to compute the shared key S through Y_p and Y_v. Note that it is not helpful for any third party to compute the addition of two points Y_p and Y_v, since

$$Y_p + Y_v = (X_p + X_v)G \neq X_p \cdot X_v \cdot G.$$

However, if the designated verifier is corrupted and sends his/her private key X_v or the shared key S to a third party so that they can verify the proof, the third party still cannot accept the verification result, because the third party could think the prover and the designated verifier cheat together since they can exchange their private key to compute $X_p \cdot X_v$, which is the relation of G and S, i.e., the prover and the designated verifier can find $m_1, m_2, r_1, r_1 \in \mathbb{Z}_q^*$ easily such that

$$C = m_1 \cdot G + r_1 \cdot S = m_2 \cdot G + r_2 \cdot S.$$

That is, for any other third parties, the commitments that are used to produce the proof does not satisfy the binding property. Therefore, the verification result can be only accepted by the designated verifier and cannot be accepted by any other third party even if the third party knows the shared key S.

4.2.3 Security Description: Strong Designated Verifier

Our SDV-ZKRP scheme satisfies the strong designated verifier property: there does not exist any algorithm in probabilistic-polynomial time to verify the proof

DOI:10.6814/NCCU202101402

without using the correct shared key and there does not exist any algorithm in probabilistic-polynomial time to distinguish between two proofs produced by different provers.

Theorem 7. There does not exist any algorithm in probabilistic-polynomial time to verify the proof without using the correct shared key S.

Proof. We divide the discussion into two cases:

(1) Assume that an adversary verifies the proof through an incorrect shared key S': the probability that the adversary finds an incorrect shared key S' to verify the proof is

$$\Pr\left[Ver(pp, S', \pi, a, b) = 1 : \begin{array}{c} pp \xleftarrow{\$} Setup(\lambda); \\ \pi \leftarrow Pro(pp, S, m, a, b); \\ S' \leftarrow \mathcal{A}(pp, \pi); \\ S' \neq S \end{array}\right] \leq negl(\lambda) \approx 0.$$

Therefore, with such a negligible probability, we can ignore this case.

(2) Assume that an adversary verifies the proof through the correct shared key S: it means that the adversary can solve the elliptic-curve Diffie–Hellman problem in probabilistic-polynomial time.

According to (1)(2), there does not exist any algorithm in probabilistic-polynomial time to verify the proof without using the correct shared key S. Therefore, **Theorem 7** is proved.

Theorem 8. There does not exist any algorithm in probabilistic-polynomial time to distinguish from two proofs produced by different provers.

Proof. Assume that a prover knows a secret value $m \in [a, b]$ and produces a SDV-EC-ZKRP proof

DOI:10.6814/NCCU202101402

$$\pi=\begin{pmatrix}C,C',C'',C_1',C_2',\\ R,EL,SQR_1,SQR_2\end{pmatrix}:\begin{cases}r,r',r'',\omega,\alpha,r_2\leftarrow\mathbb{Z}_q^*;\\ C=mG+rS;\\ C'=(b-m+1)C+r'S;\\ C''=\omega^2C'+r''S;\\ C_1'=\alpha^2G+\big(\omega^2\big((b-m+1)r+r'\big)+r''-r_2\big)S;\\ C_2'=r_2S;\\ R=\omega^2(m-a+1)(b-m+1)-\alpha^2;\\ EL\overset{\$}{\leftarrow}\mathrm{EL}(b-m+1,-r,r',G,S,C-(a-1)G,S);\\ SQR_1\overset{\$}{\leftarrow}\mathrm{SQR}(\omega,r'',C',S);\\ SQR_2\overset{\$}{\leftarrow}\mathrm{SQR}(\alpha,r_1,G,S)\end{cases}$$

and another prover knows a secret value $m'\in[a,b]$ and uses another shared key S' to produce the proof

$$\pi'=\begin{pmatrix}c,c',c'',c_1',c_2',\\ r,el,sqr_1,sqr_2\end{pmatrix}:\begin{cases}s,s',s'',\psi,\beta,s_2\leftarrow\mathbb{Z}_q^*;\\ c=m'G+sS';\\ c'=(b-m'+1)c+s'S';\\ c''=\psi^2c'+s''S';\\ c_1'=\beta^2G+\big(\psi^2\big((b-m'+1)s+s'\big)+s''-s_2\big)S';\\ c_2'=s_2S';\\ r=\psi^2(m'-a+1)(b-m'+1)-\beta^2;\\ el\overset{\$}{\leftarrow}\mathrm{EL}(b-m'+1,-s,s',G,S',C-(a-1)G,S');\\ sqr_1\overset{\$}{\leftarrow}\mathrm{SQR}(\psi,s'',c',S');\\ sqr_2\overset{\$}{\leftarrow}\mathrm{SQR}(\beta,s_1,G,S')\end{cases}.$$

Let $\hat{\pi}=\big(\hat{C},\hat{C}',\hat{C}'',\hat{C}_1',\hat{C}_2',\hat{R},\widehat{EL},\widehat{SQR_1},\widehat{SQR_2}\big)$ be a randomly chosen proof in the set of all valid proofs. The probability

$$\Pr[\pi=\hat{\pi}]=\Pr\begin{bmatrix}r,r',r'',\omega,\alpha,r_2\in\mathbb{Z}_q^*;\\ C=\hat{C},C'=\hat{C}',C''=\hat{C}'',C_1'=\hat{C}_1',C_2'=\hat{C}_2',\\ R=\hat{R},EL=\widehat{EL},SQR_1=\widehat{SQR_1},SQR_2=\widehat{SQR_2}\end{bmatrix}=\frac{1}{(q-1)^6}$$

and the probability

$$\Pr[\pi'=\hat{\pi}]=\Pr\begin{bmatrix}s,s',s'',\psi,\beta,s_2\in\mathbb{Z}_q^*;\\ c=\hat{C},c'=\hat{C}',c''=\hat{C}'',c_1'=\hat{C}_1',c_2'=\hat{C}_2',\\ r=\hat{R},EL=\widehat{el},SQR_1=\widehat{sqr_1},sqr_2=\widehat{SQR_2}\end{bmatrix}=\frac{1}{(q-1)^6}$$

DOI:10.6814/NCCU202101402

are equal, i.e., π and π' are indistinguishable. Therefore, **Theorem 8** is proved. The SDV-EC-ZKRP scheme satisfies the strong designated verifier property.

DOI:10.6814/NCCU202101402

5 Efficiency Analysis

In this chapter, we evaluate the efficiency of our scheme and compare our scheme with other existing ZKRP schemes.

In **Table 3**, we compare our scheme with some related ZKRP schemes [1][24][28]. The comparison focuses on the provable range, the execution time and the proof size. The modular exponentiation is a type of exponentiation that performed over a modulus, i.e.,

$$y = x^i \bmod N.$$

In the schemes [1][24][28], since most of their steps are kind of the modular exponentiation, we count the times of the modular exponentiation to represent the execution time of these schemes; the execution time of our scheme is to count the times of the point multiplication, i.e.,

$$Y = i \cdot X,$$

where X and Y are the two points on an elliptic curve. In addition, we set 1024 bits as the security parameter size to estimate the proof size produced by the schemes. According to the analysis of the National Institute of Standards and Technology (NIST) [4], since our scheme is based on the elliptic-curve cryptography (ECC), we set 160 bits as the security parameter size to estimate the proof size produced by the schemes so that the security strength of our scheme can be the same as other schemes which are based on integer-factorization cryptography (IFC).

DOI:10.6814/NCCU202101402

Scheme	Cryptography	Security Parameter Size (bit)	Provable Range	Computation Times	Proof Size (byte)
[1]	IFC[1]	1024	Arbitrary: $[0,2^{1024}]$	40	896
[24]	IFC[1]	1024	Arbitrary: $[0,2^{1024}]$	33	1280
[28]	IFC[1]	1024	Arbitrary: $[0,2^{1024}]$	30	2560
Our scheme	ECC[2]	160	Arbitrary: $[0,2^{160}]$	30	400

Table 3: Comparison of Our ZKRP Scheme with Other Related Schemes

[1] IFC: integer-factorization cryptography

[2] ECC: elliptic-curve cryptography

It can be seen from **Table 3** that the execution time of our scheme is the same as the scheme proposed by Tsai et al. [28], but only a 160-bit security parameter size is required to meet the same security strength. At this level of the security strength, the proof produced by [28] is about 6.4 times different from the proof produced by our scheme.

Besides, these four schemes have an arbitrary provable range: the prover can determine the secret range by themselves but the range cannot be outside the security parameter. Since our scheme is constructed with the elliptic-curve cryptography, the number of points on a curve can be counted by the Schoof's algorithm [26]. If the range of the secret value that the prover wants to prove is over the number of elements, the proof produced followed our scheme cannot convinces the verifier that the secret value is in the specified range even if the proof can pass the verification. However, the length of 160 bits is about 1.46×10^{48}. Taking a 160-bit length as the range is quite sufficient when using the ZKRP scheme in practice, such as the cryptocurrency transactions and the application scenarios as described in **Chapter 6**.

More importantly, if the ZKRP schemes that are used in practice requires the higher security strength, the security parameter size must be set at least 2048 bits or more for the schemes [1][24][28], but only need to set a 224-bit security parameter size for our scheme to meet the same level of the security strength, and the 224-bit length is about 2.7×10^{67}. In addition, at this level of the security strength, the size of the proof produced by our scheme is about 560 bytes. The gap of the proof size can be quite wide:

DOI:10.6814/NCCU202101402

it is about 9.14 times different from the scheme proposed by Tsai et al. [28].

Therefore, our ZKRP scheme has a shorter execution time, a smaller security parameter, and a smaller proof size among the four schemes. By applying our scheme to the cryptocurrency, the transaction cost can be reduced effectively.

DOI:10.6814/NCCU202101402

6 Application Scenarios

Our non-interactive EC-ZKRP scheme can be applied to not only the cryptocurrency on the blockchain to hide the transaction amount, but also other scenarios in practice. In this chapter, we describe some application scenarios, the procedure of which is shown in **Figure 4**.

To produce the ZKRP proof, the prover starts by applying for the ZKRP proof documents to the corresponding issuer. When applying, it is necessary to send the documents that can prove one's identity and the range of the secret value for producing the proof. After the issuer confirms the identity, the issuer produces the ZKRP proof document and sends it to the applicant. Then the prover sends the document to the verifier to determine whether the verification is passed or not. By following this procedure, the prover can convince the verifier that the secret value is in the specified range without providing the exact secret value. We describe some practical examples as below.

Scenario 1: Alice wants to apply to a company for a new job. The company specifies that the language test score must reach a certain score. Alice knows that his test score has reached the certain score, but she does not want to provide the exact test score to the company. Therefore, Alice applies to the test center for the ZKRP proof document of the score, and then the test center produced the document and sends to Alice. As a result, instead of providing the exact test score to the company, Alice only needs to send the ZKRP proof document so that the company can be convinced that her test score has reached the certain score.

Scenario 2: Bob wants to donate blood. However, Bob wants to know whether his physical condition is suitable for blood donation, so Bob starts by going to a health facility to make a blood test report. If the health facility approves that Bob can donate blood, then Bob can apply to the health facility for the ZKRP proof of the blood draw report. Therefore, by sending the ZKRP proof of the blood draw report, Bob can directly answer some important questions from the collecting agency instead of sending the complete blood draw report.

Scenario 3: Cindy is a student whose family is a low-income family. She wants to apply to her school for the tuition and miscellaneous fees exemption, so Cindy starts by applying to the social welfare organization for the ZKRP proof document of the family

DOI:10.6814/NCCU202101402

income. Then all students are required by the school to submit the document. Therefore, the school cannot know the family income of all students, but it can still determine which students meet the condition.

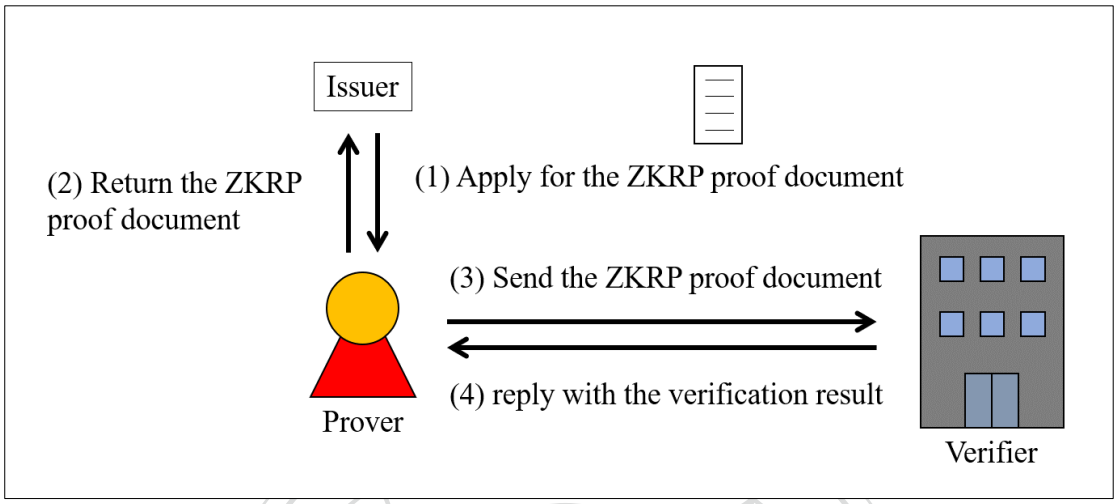

Figure 4: Schematic Diagram of ZKRP Application Scenarios

DOI:10.6814/NCCU202101402

7 Conclusions

We propose the EC-ZKRP scheme. By applying the elliptic curve, our scheme has a shorter execution time, a smaller key size and a smaller proof size at the same level of the security strength compared to existing ZKRP schemes such that the transaction cost can be reduced. In addition, by using the trapdoor commitment scheme [13] and ECDH [5], we propose a designated verifier ZKRP scheme and a strong designated verifier ZKRP scheme based on EC-ZKRP without adding any extra computation steps during producing proofs. The designated verifier ZKRP scheme allows the only designated verifier to be able to verify the proof, and the verifier cannot convince any other third party of the verification result; the strong designated verifier ZKRP scheme makes any third party cannot verify the proof. Besides, these ZKRP schemes can be optional and flexible: we can choose a suitable scheme to produce a ZKRP proof according to the confidentiality of the secret value. Furthermore, we argue the security proofs of our schemes completely and rigorously so that our schemes can satisfy necessary security properties, e.g., correctness, soundness, zero-knowledge, designated verifier and strong designated verifier. Finally, we provide the efficiency analysis compared to other existing ZKRP schemes and list some application scenarios that uses ZKRP schemes. Our ZKRP schemes can be applied to not only the cryptocurrency on the blockchain, but also other scenarios in practice. By applying ZKRP widely, our privacy can be more protected.

DOI:10.6814/NCCU202101402

Reference

[1] F. Boudot. Efficient proofs that a committed number lies in an interval. In International Conference on the Theory and Applications of Cryptographic Techniques, pages 431–444. Springer, 2000.

[2] V. Buterin. Ethereum white paper. In GitHub repository, 2013.

[3] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Efficient range proofs for confidential transactions. Technical report, Cryptology ePrint Archive, Report 2017/1066, 2017. https://eprint. iacr. org/2017/1066, 2017.

[4] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. Recommendation for key management part 1: General (revision 3). In NIST Special Publication 800-57, pages 1–147. July, 2012.

[5] E. Barker, D. Johnson, and M. Smid. Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography. In Special Publication 800-56A, National Institute of Standards and Technology, Gaithersburg, MD, March, 2007.

[6] R. Chaabouni, H. Lipmaa, and B. Zhang. A non-interactive range proof with constant communication. In Financial Cryptography and Data Security, A. D. Keromytis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, pages 179–199. 2012.

[7] P. Chaidos, and G. Couteau. Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In Annual International Conference on the Theory and Applications of Cryptographic Techniques pages 193–221. Springer, Cham, April, 2018.

[8] F. Christian and G. Johann. Efficient Implementation of Pedersen Commitments Using Twisted Edwards Curves. In Mobile, Secure, and Programmable Networking - Third International Conference, MSPN 2017, pages 1–17, 2017.

[9] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Annual International Cryptology Conference, pages 16–30. Springer, 1997.

[10] P. Gallagher. Digital signature standard (DSS). In Federal Information Processing Standards Publications, volume FIPS, 186, 2013.

[11] O. Goldreich, Y. Oren. Definitions and properties of zero-knowledge proof systems. In J. Cryptology 7, pages 1–32, 1994.

[12] D. Hankerson, A. Menezes. Elliptic Curve Discrete Logarithm Problem. In van Tilborg H.C.A., Jajodia S. (eds) Encyclopedia of Cryptography and Security, 2011.

[13] M. Jakobsson, K. Sako, R. Impagliazzo. Designated Verifier Proofs and their

DOI:10.6814/NCCU202101402

Applications. In Eurocrypt'96, Springer LNCS Vol. 1070, pages 142–154, 1996.
[14] S. Katsumata, R. Nishimaki, S. Yamada, and T. Yamakawa. Designated verifier/prover and preprocessing NIZKs from Diffie-Hellman assumptions. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 622–651. Springer, Cham, May, 2019.
[15] T. Koens, C. Ramaekers and C. van Wijk. Efficient Zero-Knowledge Range Proofs in Ethereum. In ING media.
[16] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In 2016 IEEE symposium on security and privacy (SP), pages 839–858, 2016.
[17] B. Libert, A. Passelègue, H. Wee, and D. Wu. New constructions of statistical NIZKs: dual-mode DV-NIZKs and more. In Eurocrypt 2020-39th Annual International Conference on the Theory and Applications of Cryptographic Techniques. May, 2020.
[18] H. Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In International Conference on the Theory and Application of Cryptology and Information Security, pages 398–415, Springer, 2003.
[19] P. McCorry, S. Shahandashti, and F. Hao. A smart contract for boardroom voting with maximum voter privacy. In International Conference on Financial Cryptography and Data Security, pages 357–375. Springer, 2017.
[20] I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In 2013 IEEE Symposium on Security and Privacy, pages 397–411, IEEE, May, 2013.
[21] E. Morais, T. Koens, C. Wijk, and A. Koren. A survey on zero knowledge range proofs and applications. In Nature Switzerland AG 2019, Springer, 2019.
[22] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. In Decentralized Business Review, 2008.
[23] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In CRYPTO, volume 576 of Lecture Notes in Computer Science, pages 129–140, 1991.
[24] K. Peng and F. Bao. Batch range proof for practical small ranges. In International Conference on Cryptology in Africa, pages 114–130, Springer, 2010.
[25] M. Qu. Sec 2: Recommended elliptic curve domain parameters. In Certicom Res., Mississauga, ON, Canada, Tech. Rep. SEC2-Ver-0.6, 1999.
[26] R. Schoof. Elliptic Curves over Finite Fields and the Computation of Square Roots mod p. In Mathematics of Computation Vol. 44, No. 170, pages 483–494, April, 1985.
[27] N. Van Saberhagen. CryptoNote v 2.0, 2013.

DOI:10.6814/NCCU202101402

[28] Y. Tsai, R. Tso, Z. Liu, and K. Chen. An improved non-interactive zero-knowledge range proof for decentralized applications. In 2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON), pages 129–134, April 2019.

[29] Y. Wang and A. Kogan. Designing confidentiality-preserving blockchain-based transaction processing systems. In International Journal of Accounting Information Systems, vol. 30, pages 1–18, 2018.

[30] L. Xu, N. Shah, L. Chen, N. Diallo, Z. Gao, Y. Lu, and W. Shi. Enabling the sharing economy: Privacy respecting contract based on public blockchain. In Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, pages 15–21, 2017.

DOI:10.6814/NCCU202101402