國立政治大學資訊科學系

碩士學位論文

基於 ECDSA 之部分盲簽章及其在比特幣上應用之研究
A Study on Partially Blind ECDSA and Its Application on Bitcoin

指導教授：左瑞麟　博士

研究生：黃泓遜　撰

中 華 民 國 110 年 1 月

摘要

盲簽章是一種能夠不讓簽名者知道自己所簽訊息的數位簽章。然而在實際應用中，簽名者往往需要記錄一些與簽名相關的額外訊息。爲了解決這個問題，部分盲簽章的概念被提出。除了具有盲簽章的性質外，部分盲簽章可以讓簽名者能從所簽訊息中獲取到所需的相關的資訊。部分盲簽章在被提出至今有不少成果被提出，但這些成果都需要花費較多的運算時間，或是不易應用到實際應用中。除此之外，隨著數位貨幣（如：比特幣）的興起，愈來愈多消費者會購買數位貨幣。但目前的購買方式無法隱藏消費者的電子錢包位置，因此一些研究將重點放在基於橢圓曲線簽章算法（Elliptic Curve Digital Signature Algorithm，ECDSA）的盲簽章的研究上。然而由於盲簽章簽名者完全不知道所簽訊息的特性，使得這些基於 ECDSA 的盲簽章難以靈活地運用在數位貨幣系統上。因此，我們提出了提出了三個部分盲簽章。我們的第一個簽章是到目前爲止的研究是效能最好的部分盲簽章。另外，爲了與比特幣系統更加契合，我們提出了兩種改版之 ECDSA 及其在通用群模型（Generic Group Model）下的安全性證明，並基於此提出了兩種首次與現行比特幣系統相契合的 ECDSA 部分盲簽章。我們爲上述的部分盲簽章都提供了安全性證明及效能分析。最後我們提出了我們的部分盲簽章在購買比特幣時的應用方式。

關鍵字：ECDSA、部分盲簽章、比特幣

Abstract

Blind signatures allow a user to obtain a signature without revealing message information to the signer. However, in many cases, the signer must record additional information relevant to the signature. Therefore, a partially blind signature was proposed to enable the signer to obtain some information from the signed message. Although many partially blind signature schemes have been proposed, they are time intensive and impractical. Additionally, with the development of blockchain technology, users increasingly use Bitcoin for purchasing and trading with coin providers. Some studies have indicated that elliptic curve digital signature algorithm (ECDSA)-based blind signatures are compatible with Bitcoin because they prevent the linking of sensitive information due to the untamability of Bitcoin. However, these approaches are not sufficiently flexible because blind signatures do not allow the signer to obtain any information. In this thesis, we proposed three partially blind signature schemes. To the best of our knowledge, compared with other state-of-the-art schemes, our first scheme is the most practical partially blind signature. Additionally, to be more compatible with the current Bitcoin protocol, we introduced two variants of ECDSA with their security proofs under generic group model. Based on these two variants of ECDSA we proposed two partially blind signature schemes. Security proofs are provided to demonstrate that all proposed schemes have satisfactory unforgeability and blindness. At last we describe a application of bitcoin purchasing based on proposed schemes.

Keywords: ECDSA, Partially Blind Signature, Bitcoin

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In financial and commercial systems, it is often necessary to sign or seal documents to provide authenticity. To sign digital documents in digital way, the concept of digital signature [28] appeared. The digital signature adopts the public key cryptography (PKC) [13] technology. Every signer will get a public key that published and a secret key that kept by signer himself. The private key is used to sign the file while the public key is published for verifier to check the validity of the digital signature. And Digital signature should be unforgeable just like paper signature and seal. The *unforgeability* of digital signatures means that no adversary can forge the a valid signature that never seen before except knowing the secret key.

However, in practical, there are many scenarios for where common digital signatures are not suitable, such as electronic cash [8] and electronic voting systems. For instance, in the electronic cash system, the consumer purchase electronic cash from the electronic cash issuer (for example, the bank). If the bank completes the entire process of signature of issuing electronic cash, then the bank clearly knows the signed message. On the contrary, consumers only have the right to use the cash. After the consumer's electronic cash is used, the bank stores the consuming record in its database. Therefore, the bank knows when the electronic cash was used and who used it. Since banks know the identity in real life of consumers, consuming records can easily be associated with consumers. Banks can easily grasp the consumption of all consumers. Therefore, it is necessary for consumers to hide the connection between these records and consumers.

In order to solve the above problem, the concept of blind signature [7] was proposed. The signature process of blind signature requires the joint participation of the signer (that is, the role of the bank in electronic cash system) and the user (that is, the role of the consumer in electronic cash system). User can decide what the signed message and signer signs the message without knowing what it is. Compared to the unforgeability of common digital signatures, blind signatures need to meet the the property called *blindness* additionally. We now discuss some further details about *blindness*. During the signing process, user blinds the message, so that signer does not know what he/she signs. After signer signing the message, user unblinds the result provided by signer and obtains the signature of original message. Therefore, signer can not get any information about the message or the final published signature during the signing process. This means that signer does not know when he signed that message and who he signed the message for after user publishing the original message and the signature (such as consuming electronic cash, in electronic cash system). The *blindness* property leads to the impossibility of linking the records and the user(consumer) identities since the information is not enough for the bank to link the records in database with the signing processes.

However, in practical, blind signatures face a problem : signer can only control the public key and the attributes related to it. For the remaining attributes, signer has no ability in control. This means that signer can only affect the signature by replacing the public key. For example, in the electronic cash system, the bank does not know the signed message, and can only set the rule that one signature provides one coin to the consumer, which leads to low efficiency. If the bank wants to provide more than one coin in an signature, it needs to use different public keys to represent different amounts in each signature. Therefore, consumers and electronic cash receivers (usually merchants) need to record all possible public keys in their electronic wallets. However, their electronic wallets are usually smart cards, which has no way to afford such amount of data.

Another problem is double-spending. If the signature of electronic cash is valid forever, the holder of the electronic cash may use the electronic cash twice, which is called double-spending. To prevent double-spending, the bank has to record every consumption of the cash to ensure that the cash can only be consumed once. This will makes overflow of database, since the number of these records increases over time.

In order to solve this problem, the concept of partially blind signature was first proposed by Abe and Fujisaki [1] as an extension of blind signature in 1996. Abe and Okamoto [2] further introduced a concrete blind signature protocol along with a concrete security proof. Partial blind signatures can provide more convenience for signature issuing, and also provide more flexibility for application under different scenarios. For example, in the electronic cash system, if the bank want to add the amount of coins to a signature, it is no longer necessary to prepare a large number of public keys. Instead, bank signs the amount into the signature. Moreover, the bank may sign the date into the signature, then the bank will know whether each electronic cash is due or not. Thereby the bank can delete the expired message so that the size of the database is controlled.

1.2 Motivation

With the rise of digital currencies (such as Bitcoin [33]), more and more users purchase digital currencies. Purchasers often purchase digital currency through some digital currency trading platforms. These platforms provide services to convert real currency into digital currency. When purchase digital currencies, purchasers have to reveal their true identity. Since they release their electronic wallet address sometime after, the platform can easily associate their true identity with the wallet address. Therefore, if partially blind signature can be applied in purchasing of digital currency, the connection between the true identity and the address of the electronic wallet can be hidden, thereby ensuring the anonymity.

Most current digital currencies use signatures that are based on elliptic curve digital signature algorithm (ECDSA) [22], which is an elliptic curve version of DSA (Digital Signature Algorithm) [24] that has a shorter length public key size than DSA, and therefore used in many systems as digital signature scheme. However, at present, partially blind signatures based on ECDSA are rare. For example, the partially blind signature proposed by Li in 2004 [56]. In Li's scheme, only signer has the right to put common information into signature but user can not. Moreover, the final signature does not contain the common information, so verifier has no way to check whether signer used the correct common information or signer changed the common information halfway. In 2004, An proposed a blind signature based on pure ECDSA [3] (that is, the signature generated by the blind signature can be directly verified by the ECDSA verification algorithm). Later Ladd [25] proposed an application of blind signature on Bitcoin based on An's research, but due to the need to combine the cut-and-choose [8] method, the efficiency is not that good. In 2019, Yi *et al.* [52] modified An's scheme and fixed its security problem. However, since the usage of Yi's blind ECDSA restricted by the constraint that it only transfer one bitcoin with one signature, the practical efficiency of Yi's blind ECDSA is still limited. Therefore we hope to raise the efficiency of blind signature in Bitcoin system by transfer blind signature into partially blind signature.

1.3 Contributions

To solve the aforementioned problems, we propose three partially blind signatures. Our first scheme is a practical partially blind signature scheme. The experimental results demonstrated that compared with other state-of-the-art schemes, our first scheme is the most efficient in terms of computational cost.

As mentioned in previous section, although Yi [52]'s blind ECDSA is fully compatible with standard ECDSA, this scheme transfers only one bitcoin with one signature, which means that the computation is expensive. If there exists a partially blind ECDSA scheme which can transfer arbitrary amount of bitcoins, the efficiency may be significantly improved. The common information is one of most important components of partially blind signature, since common information contains information that signer desires. However the transfer from blind signature to partially blind signature, which implies the participation of common information actually, needs explicit or implicit verification on common information. Thereby it is impractical to transfer Yi's blind ECDSA into partially blind ECDSA by directly adding the common information since a verification of the common information is also needed. Therefore, we introduced two variants of ECDSA which will be used to construct our partially blind signatures that are compatible with ECDSA. Two variants of ECDSA are called variant-ECDSA-1 and variant-ECDSA-2, followed by their security proofs under generic group model. Subsequently, based on variant-ECDSA-1 and 2, and the blind ECDSA of Yi [52], we proposed two ECDSA-based partially blind signatures (*i.e.,* the signature generated by the blind signature can be directly verified by the proposed variants of ECDSA verification algorithm). Because the published signature is a valid variant ECDSA signature, our second and third scheme are compatible with current Bitcoin protocol in accordance with the description in [52]. Moreover, we provide proofs of security to demonstrate that our schemes satisfy the unforgeability of partially blind signatures under adaptive chosen-message attacks and partial blindness. Finally, we describe a application of bitcoin purchasing based on proposed schemes.

1.4 Organization

We organize the rest of the manuscript as follows : our Chapter 2 mentions some previous research results. Some concrete schemes Chapter are introduced in 3 while preliminary knowledge is described in Chapter 4. In Chapter 5 we introduce the generic group model[5]. We then make the definition of variant-ECDSA in Chapter 6. Two Concrete schemes of variant-ECDSA are proposed in Chapter 7 and Chapter 8 with their security proofs under generic group model. Chapter 9 provides the definition and security models of partially blind signatures. Chapter 10, Chapter 11 and Chapter 12 we proposed three partially blind signature schemes. Chapter 13 provides the correctness and security proofs. Chapter 14 compares our proposed scheme with other state-of-the-art schemes while in Chapter 15 we show the performance of our proposed schemes. In Chapter 16 we discuss some further details about our scheme 2. Chapter 17 we describe an application to show how do our proposed schemes work during bitcoin trading, and make some further discussion. Finally, Chapter 18 summarizes this thesis.

Chapter 2

Related work

The blind signature was first proposed by Chaum in 1983. In blind signature, the message was blinded and invisible to signer during signing process. The user can unblind the result provided by signer and obtain the final digital signature for the original message, and then make it public. Blind signature can protect the anonymity of user, and can play a good role in scenarios like Bitcoin and electronic cash system in practical. Later, Pointcheval and Stern [37] defined the security of blind signatures in 1996, and introduced a blind signature scheme that can be proved secure.

There are many directions of study to the blind signature. For example, due to the blindness property of blind signature, it may be used for crime. Therefore, some studies have focused on solving this problem. Such as the blind signature proposed by Stadler and Camenisch [45] in 1995, the "indirect discourse certificate" proposed by Frankel *et al.* [16] in 1996, and the "magic potion" signature proposed by Zhang *et al.* [51] in 2003, all can prevent blind signatures from being used for crime.

Besides, it is also a common research direction to apply blind signature to identity-based signature (IBS), which was first proposed by Shamir [42]. The characteristic of this signature is that the public key of the signer is calculated by the identity of the signer. The identity of the signer is a string that can represent the signer, such as his mailbox. By combining Menezes et al.'s [31] bilinear pairing scheme, Zhang and Kim [53] proposed a blind signature based on the property of IBS in 2002, so that the scheme has a wider range of application.

In addition to applying blind signature to IBS, there are many other types of digital signatures that have been applied to blind signature. For instance, proxy blind signatures proposed by Lal and Awasthi [26], Safavi-Naini et al. [54] and Zuo-Wen et al. [47]. Proxy blind signature applies property of proxy signature into blind signautre, which means that signer can let the agent sign on behalf of himself/herself, thereby reducing the computation load of original signer. Forward-secure blind signature rises the security level of common blind signature. In common blind signature, even the private key is leaked, the signature is still valid. But in forward-secure blind signature, the signature will be invalid. There are several schemes proposed by Sherman et al. [11] and Duc et al. [14]. However, it is a pity that Liu and Cao [27] proved that Duc et al.'s [14] scheme is unsafe later. To meet the need for more than one person to jointly sign a blind signature, another type of signature is proposed, including the blind multisignature proposed by Chen et al. [9], the group blind signatures proposed by Lysyanskaya, Ramzan [29], and proposed by Kim et al. [23], the blind threshold signature proposed by Vo et al. [49], and the blind threshold ring signature proposed by Chan et al. [6].

Since the application of elliptic curve cryptography (ECC) in digital signature can significantly decrease the key size, Debasish et al. [21] and Morteza et al. [35] conducted some researches on the blind signature that based on the elliptic curve discrete logarithm problem (ECDLP) in 2007 and 2009, respectively. Zhang and Kim's [53] blind signature is based on the IBS, in the way of combining bilinear pairing, but the computation cost of bilinear pairing is not low enough. Until Debiao [18] proposed a identity-based blind signature without bilinear pairing in 2011, which only apply the scalar point multiplication in elliptic curve, greatly improves the efficiency.

One problem of blind signature is that the signed message remains unknown to signer during signing phase, and signer can only control the public key. To solve this problem, Abe and Fujisaki [1] proposed the concept of partially blind signature in 1996, as an extension of blind signature. The difference between partially blind signature and blind signature (also called the fully blind signature) is that: the signing process requires another public message called common information. The common information is generated in the negotiation phase before signing, and (usually) should be added to the signature by both signer and user. Depending on the scenario, the common information can be produced in the negotiation phase or just be determined by one of the participants. Therefore partially blind signature provides more convenience for signer. Compare to the fully blind signature, partially blind signature ensures that some part of the signed message is reliable.

Since the first partially blind signature was proposed, many studies [10, 12, 30, 36, 55] followed. For example, in 2001, Chien et al. [10] proposed a partially blind signature protocol which has lower computation cost, making partially blind signature easier to implement on machines with weaker computing power. Partially blind signature scheme using bilinear pairing proposed by Zhang [55] in 2003 laid the foundation for other bilinear-pairing-based blind signatures. In 2002, Greg and Colin [30] proposed the first partially blind signature with restrictive property. The restrictive property means user can only choose the message in the set that signer agreed. In 2005, Sherman et al. [12] proposed a threshold partially blind signature based on bilinear pairings so that the partially blind signature can only be valid when enough signers participate in the signing phase. Almost all the blind signatures or partially blind signatures aforementioned are proved security in random oracle model. In 2006, Okamoto [36] proposed a blind signature scheme that no longer relies on random oracle. This scheme has a higher security level and improved the performance.

Chapter 3

Background

3.1 Digital Signature

The digital signature [28], which use the public-key Cryptography (PKC) [13] technology, is a mechanism that can imitate the behavior of signing or stamping paper documents in reality and “sign” digital documents. Digital signature requires the use of two keys, a public key and a private key. Everyone who needs to sign a file will have their own private key and public key. The private key is kept on its own without being known to others, and is used to sign the document. The public key is made public to be used in verifying whether the signature is valid or not. Digital signature has property called unforgability: The unforgeability of digital signature means that except for the legal signer, no one can forge a valid signature that never seen before without knowing the secret key.

In 1968, the U.S. federal government issued the Federal Information Processing Standards (FIPS) in order to formulate open standards for all government agencies and government contractors except U.S. military agencies, and gradually supplemented this Standard since then. Today’s standards mainly include data encoding standards, such as country area code, and encryption standards, such as data encryption standard and digital signature standard. In standard of digital signature, the U.S. federal government adopted a variant of digital signature proposed by Schnorr [41] in 1989 (we describe it in 3.2) and named it DSA [24] (Digital Signature Algorithm) as a part of the standard. DSA is not only used under the government system, many private institutions and private systems also use DSA as their digital signature scheme. However, the minimum length of the key of DSA specified by the Federal Information

Processing Standard is 2048 bits, which is a little too long in practical.

Elliptic curve cryptography [32] (ECC) is a kind of public key cryptography that relies on the mathematical structure of elliptic curves. Compared to non-elliptic-curve-based cryptography, elliptic curve cryptography requires shorter key size to provide the same security level. Therefore, after the emergence of elliptic curve cryptography, the elliptic curve digital signature algorithm [22] (ECDSA), which combines DSA and elliptic curve cryptography, came into being. ECDSA can achieve the same security level as that of the 2048-bit key size DSA with only 224 bits key size, which is much shorter than DSA. ECDSA was also included in the Federal Information Processing Standard, and has been widely used by federal agencies and private institutions. For example, the digital signature used by Bitcoin is ECDSA. We describe its scheme in section 3.4.

3.2 Schnorr Blind Signature

In 1989, Schnorr [41] proposed a digital signature based on discrete logarithm problem (DLP). Schnorr signature is the foundation of many later digital signature schemes. The blind version of Schnorr signature is proposed later [38, 39]. In Schnorr blind signature, public parameter is $(p, q, g, H(\cdot))$, where p, q are two large primes which satifsy $q|p-1$, g is an element in ${Z_p}^*$ of prime order q, and $H(\cdot)$ is a secure cryptography hash function agreed by both signer and user. The secret key is x, while public key is $y = g^x$. Schnorr Blind Signature process can be described as follows:

1. Signer randomly picks $k \in Z_q$, computes $r = g^k \pmod p$ and sends r to user.

2. User randomly chooses α, $\beta \in Z_q$ and calculates $r' = rg^{-\alpha}y^{-\beta} \pmod p$. User then compute $e' = H(m, r')$ and $e = e' + \beta \pmod q$, and send e back.

3. Signer calculates $s = k - ex \pmod q$ and send it to user.

4. User computes $s' = s - \alpha$. The final signature is (e', s').

5. Anyone who knows public key y can verify the sign (e', s') by checking whether $e' =$

$H(m, g^{s'} y^{e'} \pmod{p})$ holds:

$$\begin{aligned}
&H(m, g^{s'} y^{e'} \pmod{p}) \\
&= H(m, g^{s-\alpha} y^{e-\beta} \pmod{p}) \\
&= H(m, g^{k-ex-\alpha \pmod{q}} y^{e-\beta} \pmod{p}) \\
&= H(m, g^{k} g^{-ex} g^{-\alpha} y^{e-\beta} \pmod{p}) \\
&= H(m, r y^{-e} g^{-\alpha} y^{e-\beta} \pmod{p}) \\
&= H(m, r y^{-e} g^{-\alpha} y^{e} y^{-\beta} \pmod{p}) \\
&= H(m, r g^{-\alpha} y^{-\beta} \pmod{p}) \\
&= H(m, r') \\
&= e'
\end{aligned}$$

3.3 Elliptic Curve Discrete Logarithm Problem

The security of our proposed scheme is based on the elliptic curve discrete logarithm problem (ECDLP) [44], which is a special case of the discrete logarithm problem defined as follows:

Definition 1. Suppose $\mathcal{E}$ is an elliptic curve over $\mathbb{Z}/p\mathbb{Z}$, where p is a prime. Given an elliptic curve point tuple (P, Q), where $P \in \mathcal{E}(\mathbb{Z}/p\mathbb{Z})$ and Q is a multiple of P, the ECDLP is to find $a \in \mathbb{Z}_p^*$ such that $Q = aP$.

3.4 ECDSA

The ECDSA [22] is a digital signature scheme consisting of four algorithms:

- **Setup:** An elliptic curve $\mathcal{E}$ is randomly selected by this algorithm, following by a group generator G of prime order q over $\mathcal{E}$. Besides, algorithm randomly selects a hash function $H(\cdot)$. Finally, this algorithm outputs a set of public parameters $pp = (\mathcal{E}, G, q, H)$.
- **KeyGen:** The Signer randomly selects their private key $d \in [2, q-1]$ and then computes their public key $Q = d \cdot G$.

- **Sign:** For a message m, the signer randomly selects a integer k from 2 to $q-1$ and computes $(K_x, K_y) = kG$, $t = K_x \bmod q$, and $s = k^{-1}(H(m) + t \cdot d) \bmod q$. Finally, the signer outputs a signature $\sigma = (t, s)$.

- **Verify:** Anyone who knows the signer's public key can validate the signature $\sigma = (t, s)$. First, a verifier can compute $u = s^{-1}H(m) \bmod q$, $v = s^{-1}t \bmod q$, then let $(K'_x, K'_y) = uG + v \cdot Q$. Finally, the verifier output 1 if $t' = K'_x \bmod q = t$. Otherwise, the verifier outputs $\perp$.

3.5 Yi's Blind ECDSA

We first describe the modified Paillier encryption in Yi *et al.* [52]. This encryption consists of the following three algorithms:

- **KeyGen:** The user randomly selects two large different primes p and k that satisfy $gcd(p-1, q) = 1$ and $gcd(k-1, q) = 1$, then calculate $N = pqk$ and $g = (1+N)^{pk} \bmod N^2$, where (p, k) remains secret and (N, g) is released.

- **Enc:** Any one who holds (N, g) can encrypt a message $m \in \mathbb{Z}_q$ into ciphertext. The encryptor randomly select $r \in \mathbb{Z}^*_{N^2}$ and output a ciphertext by computing $C = g^m r^N \bmod N^2$.

- **Dec:** The decryptor can use secret key (p, k) to decrypt ciphertext by calculating $D = C^{(p-1)(q-1)(k-1)} \bmod N^2$ and $m = [\frac{D-1}{Npk}][(p-1)(q-1)(k-1)]^{-1} \bmod q$.

Based on the modified Paillier encryption, Yi's blind ECDSA scheme was proposed, as shown in Fig. 1.

Furthermore, based on the proposed blind ECDSA scheme, Yi *et al.* introduced a application which achieve anonymity during buying bitcoins. The process is described as follows:

- Bitcoin provider $\mathcal{B}$ and purchaser $\mathcal{P}$ start the process of buying bitcoins. $\mathcal{P}$ first use fiat currency to pay for bitcoins in credit card payment, cash or some other ways.

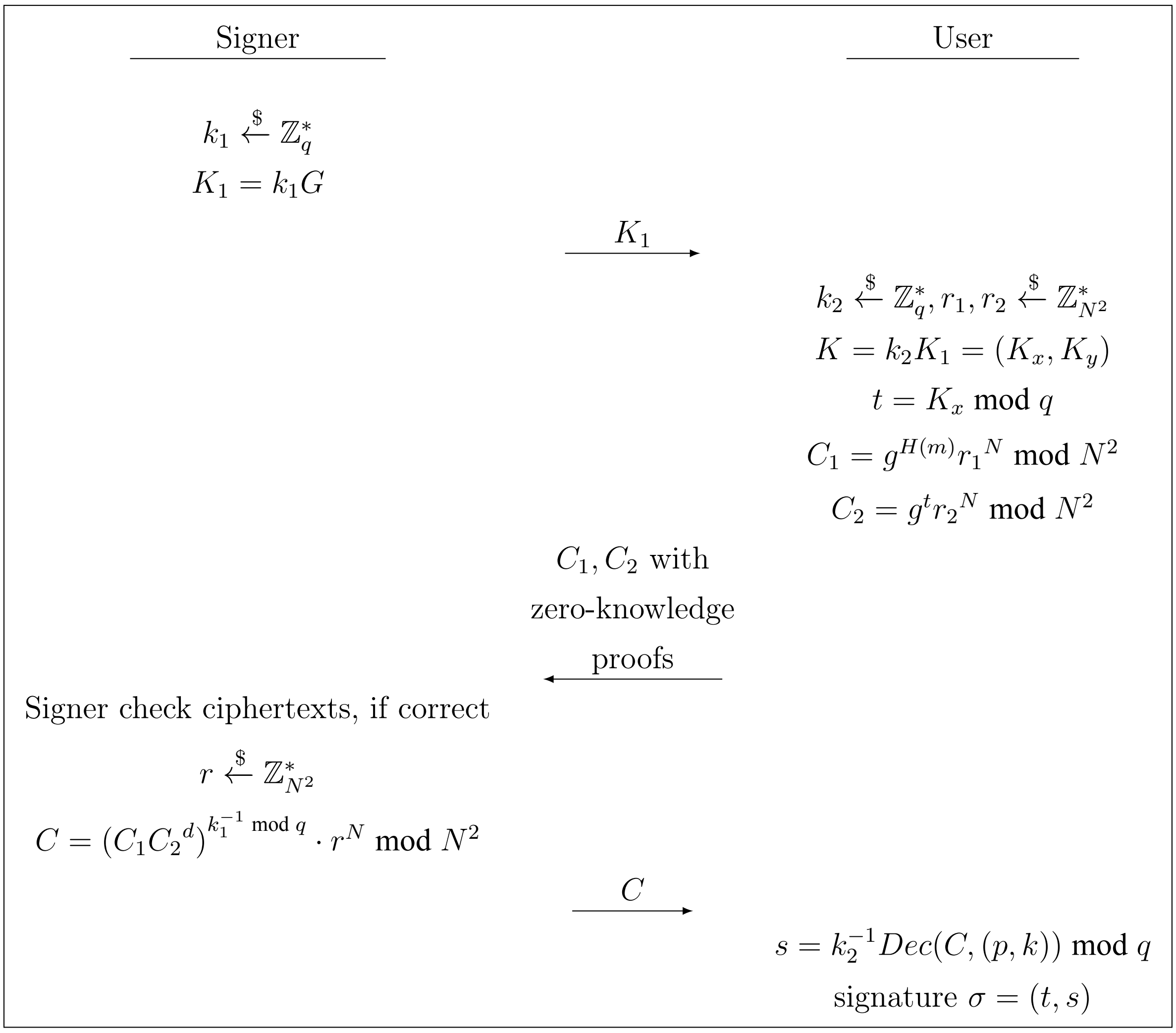
Signer
User
$k_1 \xleftarrow{\$} \mathbb{Z}_q^*$
$K_1 = k_1 G$
$\xrightarrow{K_1}$
$k_2 \xleftarrow{\$} \mathbb{Z}_q^*, r_1, r_2 \xleftarrow{\$} \mathbb{Z}_{N^2}^*$
$K = k_2 K_1 = (K_x, K_y)$
$t = K_x \bmod q$
$C_1 = g^{H(m)} {r_1}^N \bmod N^2$
$C_2 = g^t {r_2}^N \bmod N^2$
C_1, C_2 with zero-knowledge proofs $\longleftarrow$
Signer check ciphertexts, if correct
$r \xleftarrow{\$} \mathbb{Z}_{N^2}^*$
$C = (C_1 {C_2}^d)^{k_1^{-1} \bmod q} \cdot r^N \bmod N^2$
$\xrightarrow{C}$
$s = k_2^{-1} Dec(C, (p, k)) \bmod q$
signature $\sigma = (t, s)$

Figure 1: Yi's blind ECDSA scheme

- Next, Bitcoin provider $\mathcal{B}$ and purchaser $\mathcal{P}$ interact following Yi's blind ECDSA scheme. That is: purchaser $\mathcal{P}$ generate his/her public key PK_P which is the message m to be signed under ECDSA. Purchaser $\mathcal{P}$ then run the blind ECDSA with Bitcoin provider $\mathcal{B}$ to obtain the signature (t, s) of his/her public key PK_P on the basis of $\mathcal{B}$'s public key PK_B. One signature implies to transfer one bitcoin to purchaser $\mathcal{P}$.

- Purchaser $\mathcal{P}$ later broadcast its transaction including its public key PK_P and the corresponding signature in the Bitcoin system. Some miners or nodes may verify this signature. Since the signature generated from Yi's blind ECDSA is a standard ECDSA signature which can be verified by ECDSA verification of Bitcoin, this signature can be verified by miners or nodes in Bitcoin system directly.

Purchaser $\mathcal{P}$ later broadcast its transaction including its public key PK_P and the corresponding signature in the Bitcoin system. Some miners or nodes may verify this signature. Since the signature generated from Yi's blind ECDSA is a standard ECDSA signature which can be verified by ECDSA verification of Bitcoin, this signature can be verified by miners or nodes in Bitcoin system directly.

This application make sure that Bitcoin provider $\mathcal{B}$ only knows that $\mathcal{P}$ is one of the purchasers after the release of the transaction even though he/she holds the identity of purchaser $\mathcal{P}$ in real live. This is guaranteed by the blindness property of blind signature guarantee. However Yi's application only transfers one bitcoin with one signature, in practical it means expensive computational cost for large amount transfer.

Chapter 4

Preliminary

In this Chapter we introduce some preliminary components which we use in the following Chapters.

4.1 Unforgeability

A signature scheme consists of three algorithms: *key generation* for the generation of public key and secret key, *signature generation* for the generation of signature on the input message, using the secret key, and *signature verification* for the verification of the input signature, message and public key. Briefly speaking, forger F is an algorithm try to forge a valid signature without knowing the secret key. There are two different kinds of forger: existential forger and selective forger.

4.1.1 Existential Forger

Let (Q, d) be the public key and the private key. An $(\epsilon_F, \tau_F, q_F)$-forger F of a signature scheme is a probabilistic algorithm with input Q and running time at most τ_F. The forger F is allowed to select a sequence of messages M_i for $1 \leq i \leq q_F$ (for mo-message attack, $i = 0$) and invoke the signing protocol to obtain signatures on these messages under the private key d. F can decide the choices of M_i depending on the previous obtained signatures. Finally F outputs a message M (different from the M_i's in signing protocols) and a candidate signature σ on M under public key Q. The forger F succeeds if (M, σ) pass the verification algorithm under public key Q. The forger's probability of success is at least ϵ_F, where the probability is assessed over random choices of the key generation algorithm, the forger F, and the signing oracle.

4.1.2 Selective Forger

The only difference between a $\mathcal{S}$-selective $(\epsilon_F, \tau_F, q_F)$-forger F and the existential forger mentioned above is that F has an additional input: the message M forged by F should be drawn at random from $\mathcal{S}$.

4.2 Property of Hash Function

We now describe some properties of hash function needed in our following schemes. As shown in Brown's research [5], the hash function used in ECDSA, usually SHA-1 in practice, matches the following required properties.

4.2.1 One-Wayness (Preimage-Resistance)

(ϵ_I, τ_I)-inverter I_h of hash function h is an probabilistic algorithm which has the input $e \in_R \mathcal{H}$ and the output message $M \in \{0,1\}^*$ within running-time at most τ_I. Under the random choices of both e and I_h, the probability of $h(M) = e$ is at least ϵ_I. If no such I_h exists, then h is one-way or preimage-resistant of strength (ϵ_I, τ_I).

4.2.2 Second-Preimage-Resistance

Let $\mathcal{S} \subseteq \{0,1\}^*$. $(\epsilon_S, \tau_S, \mathcal{S})$-second-preimage-finder S_h for hash function h is a probabilistic algorithm which has the input $M \in_R \mathcal{S}$ and the output message $M' \in \{0,1\}^*$ within running-time at most τ_S. Under the random choices of both M and S_h, the probability of $M \neq M'$ but $h(M) = h(M')$ is at least ϵ_S. If no such S_h exists, then h is second-preimage-resistant of strength $(S, \tau_S, \mathcal{S})$.

4.2.3 Zero-Finder-Resistance

(ϵ_Z, τ_Z)-zero-finder Z_h for hash function h is a probabilistic algorithm which has the output message $M \in \{0,1\}^*$ within running-time at most τ_Z. Under the random choices of Z_h, the probability of $h(M) = 0$ is at least τ_Z. If no such Z_h is known, then h is zero-finder-resistant of strength (Z, τ_Z).

Chapter 5

Generic Group Model

So far ECDSA has been formally proved secure only by Brown [5] under generic group model, which was first introduced by Nechaev [34] and improved by Shoup [43]. In the generic group model for secure group $\mathbb{A}_n$, where a secure group $\mathbb{A}_n$ is defined as a group that have an intractable discrete logarithm problem, the group operation is assumed that can only be performed through an oracle. Moreover, the oracle of the group operation is assumed to be "random" subject to the constraint of giving valid group operations.

In order to prove the security of ECDSA and DSA, Brown describes a variant of the generic group model. The generic group oracle is shown in Table 1.

We now describe the oracle in detail. There are three commands that oracle takes: push, subtract and hint. The push and subtract commands are the same as Brown's but we modify the hint command a little. A forger can make push commands and subtract commands, but can not access to hint commands directly. Notice that in the first oracle hint command is never used, where forger can only make push, subtract command and verifying query, and in the second oracle hint command is invoked to response to the signing query. Each command appends an element pair (A_{m+1}, z_{m+1}) to the internal state, and the oracle maintain a the list of element pairs. The element pair consist of a public element $A_{m+1} \in \mathbb{A}_n$ and a private element $z_{m+1} \in \mathbb{Z}_n$. A_{m+1} and z_{m+1} both depends on commands that forger makes and random values that the oracle picks. We now discuss some further details.

The argument of push command is an arbitrary element $A \in \mathbb{A}_n$ and its output is $A_{m+1} = A$. If A_{m+1} does not equal to some A_i in $\{A_1, \cdots, A_m\}$, then oracle randomly selects a private value z_{m+1}. Otherwise, let z_{m+1} equals the previous same private value z_i.

Table 1: Generic group oracle proposed by Brown

$A_{m+1} \leftarrow$ Push ($A \in \mathbb{A}_n$).	
	1.Let $A_{m+1} = A$.
	2.If $A_{m+1} = A_i$ for some $i \in \{1, \cdots, m\}$,
	let $z_{m+1} = z_i$.
	3.If $A_{m+1} \notin \{A_1, \cdots, A_m\}$, choose
	$z_{m+1} \in_R \mathbb{Z}_n \backslash \{z_1, \cdots, z_m\}$.
$A_{m+1} \leftarrow$ Subtract (No argument)	
	1.Let $z_{m+1} = (z_{m-1} - z_m) \bmod n$.
	2.If $z_{m+1} = z_i$ for some $i \in \{1, \cdots, m\}$,
	let $A_{m+1} = A$.
	3.If $z_{m+1} \notin \{z_1, \cdots, z_m\}$, choose
	$A_{m+1} \in_R \mathbb{A}_n \backslash \{A_1, \cdots, A_m\}$.
$(A_{m+1}, s_{m+1}) \leftarrow$ Hint ($h_{m+1} \in \mathbb{Z}_n \backslash \{0\}$)	
	1.Randomly choose $z_{m+1} \in \mathbb{Z}_n$ except
	elements in $\{z_1, \cdots, z_m\}$.
	2.Randomly choose $A_{m+1} \in \mathbb{A}_n$ except
	elements in $\{A_1, \cdots, A_m\}$.
	3.$s_{m+1} = {z_{m+1}}^{-1}(h_{m+1}z_1 + f(A_{m+1})z_2) \bmod n$
	where $f : \mathbb{A}_n \rightarrow \mathbb{Z}_n$ is a certain fixed function.

Table 2: Notations in Brown's [5]

B	The set of the index of each basic pair
$C_{ij} \in \mathbb{Z}_n, i \in \{1, \cdots, m\}, j \in B$	Coefficients of derived private elements
$A = (A_1, \cdots, A_m)$	The vector of public elements
C	The coefficient matrix (C_{ij}) over $\mathbb{Z}_n$, where $i \in \{1, \cdots, m\}$ and $j \in B$
S	The response to the hint commands
$z_B = (z_j)_{j \in B}$	The basic vector
$z = (z_i)_{1 \leq i \leq m}$	The private vector
$\Delta_i = (C_{ij})_{j \in B}$	The derivation of (A_i, z_i) (row vector)

We assume that two push commands are made at first with arguments A_1 and A_2 that randomly selected by forger. Besides, the base generator $G = A_1$ and the public key $Q = A_2$. The only condition is that $A_1 \neq A_2$. Forger can subsequently submit arbitrary elements in $\mathbb{A}_n$ through push command.

Subtract command subtract the two previous element pairs. The output is the public value A_{m+1}. Actually the oracle subtract previous private values z_{m-1} and z_m to obtain $z_{m+1} = z_{m-1} - z_m$ first, ant then determine whether there is a previous z_i equals z_{m+1} or not. If not, the public value A_{m+1} will be a random element chose from $\mathbb{A}_n \backslash \{A_1, \cdots, A_m\}$.

The input of hint command is h_{m+1} and the output is (A_{m+1}, s_{m+1}), where the extra information is included in the s_{m+1}. s_{m+1} will later be a part of the signature for message $M_i || R_{m+1}$ (for scheme 3), where M_i is the input of signing query and R_{m+1} is a random element.

The hint command can be regarded as a part of signing query, since when forger makes a signing query the oracle actually invokes the hint command to obtain required objects to return to forger as the response for the signing query, which is a signature $(f(A_{m+1}), s_{m+1}, R_{m+1})$ for message $M_i || R_{m+1}$ (for scheme 3). Notice that hint command is not used in the oracle for our variant-ECDSA-1.

Additionally, we call the pair (A_i, z_i) generated from a push command and different from any previous pair the *basic* pair. And we call all the rest pairs *derived* pair. We can present every derived private element z_i as a $\mathbb{Z}_n$-linear combination of previous basic private element.

We recall some further notations in Brown's [5], as shown in Table 2.

Table 3: Derivations

```
Push
        1.If $A_{m+1} = A_i$ for some $i \in \{1, \cdots, m\}$ then:
          (a) Let $i$ be the least such index.
          (b) Let $C_{(m+1)j} = C_{ij}$ for all $j \in B$.
        2.If $A_{m+1} \notin \{A_1, \cdots, A_m\}$ then:
          (a) Add index $m+1$ to the set $B$.
          (b) Let $C_{i(m+1)} = 0$ for all $i \in \{1, \cdots, m\}$.
          (c) Let $C_{(m+1)(m+1)} = 1$.
Subtract
        1.Let $C_{(m+1)j} = C_{mj} - C_{(m-1)j}$ for all $j \in B$.
Hint
        1.Let $C_{(m+1)1} = s_{m+1}^{-1} h_{m+1}$.
        2.Let $C_{(m+1)2} = s_{m+1}^{-1} f(A_{m+1})$.
        3.Let $C_{(m+1)j} = 0$ for all $j \in B \backslash \{1, 2\}$.
```

From above notations we have $z = Cz_B$. Besides, the derivation of a basic pair (A_j, z_j) only has one non-zero coefficient which is in the j position. Furthermore, if there exist two pairs with same public elements ($A_i = A_j$) but different derivation ($\Delta_i \neq \Delta_j$) for some $1 \leq j < i \leq m$, we say there has occurred a coincidence. Coincidence-free means that there are no coincidences at any j . We show the generation of coefficients and derivations in push, subtract and hint command in Table 3.

Chapter 6

Variant-ECDSA

We recall that Yi *et al.* [52]'s blind ECDSA only transfers one bitcoin with one signature, which means it has expensive computational cost in practical. Hence a partially blind ECDSA scheme which can transfer arbitrary amount of bitcoins by release the amount, may have a significantly improvement on computational efficiency. The common information is necessary in partially blind signature, however it is impractical to transfer Yi's blind ECDSA into partially blind ECDSA by directly adding the common information. Therefore, we introduced two variants of ECDSA: variant-ECDSA-1 and variant-ECDSA-2, which will be used to construct partially blind signatures that are compatible with ECDSA.

We define the variant-ECDSA in this Chapter. The concrete schemes are described in Chapter 7 and Chapter 8.

6.1 Definition of Variant-ECDSA

- **Setup** algorithm takes a security parameter λ as input and outputs public parameter $pp = (\mathbb{E}, G, q, H)$.
- **KeyGen** algorithm takes the public parameter pp as input and outputs public key Q and secret key d.
- **Sign** algorithm takes the message to be signed m and the secret key d as inputs and outputs a signature $\sigma = (t, s, R)$ on message m, based on the public key Q.
- **Verify** algorithm takes the message m, the signature σ and the corresponding public key

Q as inputs and outputs 1 if the signature pass the verification.

6.2 Unforgeability of variant-ECDSA

We make two different security definition here, existential unforgeability against no-message attacks and selective unforgeability against adaptive chosen-message attacks.

6.2.1 Existential Unforgeability Against No-Message Attacks

To model the existential unforgeability against no-message attacks, we define Game-1 played by challenger $\mathcal{C}$ and adversary $\mathcal{A}$, where adversary $\mathcal{A}$ is actually the existential forger that we mentioned in Chapter 4.

Game - 1:

- **Setup:** Challenger $\mathcal{C}$ takes a security parameter λ and generates public parameters $params$. $\mathcal{C}$ subsequently sends λ and $params$ to adversary $\mathcal{A}$.
- **Response:** Adversary $\mathcal{A}$ outputs a signatures (m, σ).

Definition 2. A variant-ECDSA is existentially unforgeable against no-message attacks if no adversary wins the Game-1 with non-negligible probability.

6.2.2 Selective Unforgeability Against Adaptive Chosen-Message Attacks

To model the selective unforgeability against adaptive chosen-message attacks, we define Game-2 played by challenger $\mathcal{C}$ and adversary $\mathcal{A}$, where adversary $\mathcal{A}$ is actually the selective forger that we mentioned in Chapter 4.

Game - 2:

- **Setup:** Challenger $\mathcal{C}$ takes a security parameter λ and generates public parameters $params$. $\mathcal{C}$ subsequently sends λ and $params$ to adversary $\mathcal{A}$.
- **Attack:** Adversary $\mathcal{A}$ makes the signing queries to obtain signatures from challenger $\mathcal{C}$. The inputs of signing queries are messages $m_1, \cdots, m_q$ where q is the number of queries. Besides, the next input message of query can be decided after obtaining the previous response. Adversary $\mathcal{A}$ finally obtain signatures $(\sigma_1, \cdots, \sigma_q)$

- **Response:** Adversary $\mathcal{A}$ outputs a list of signatures $(m_1, \sigma_1), \cdots, (m_q, \sigma_q)$.

Definition 3. A variant-ECDSA is selectively unforgeable against adaptive chosen-message attacks if no adversary wins the Game-A with non-negligible probability.

Chapter 7

Variant-ECDSA-1

7.1 Scheme of Variant-ECDSA-1

- **Setup:** An elliptic curve $\mathbb{E}$ is randomly selected by this algorithm, following by a group generator G of prime order q over $\mathbb{E}$. It then randomly selects a hash function $H(\cdot)$. Finally, this algorithm outputs a set of public parameters $pp = (\mathbb{E}, G, q, H)$.
- **KeyGen:** The Signer randomly selects their private key $d \in [2, q-1]$ and then computes their public key $Q = d \cdot G$.
- **Sign:** For a message m, the signer randomly selects a integer k from 2 to $q - 1$ and computes $(K_x, K_y) = kG$, $t = K_x \bmod q$. Besides, signer random a R and compute $s = k^{-1}(H(M) + R + t \cdot d) \bmod q$. Finally, the signer outputs a signature $\sigma = (t, s, R)$.
- **Verify:** Anyone who knows the signer's public key can validate the signature $\sigma = (t, s, R)$. First, a verifier can compute $u = s^{-1}(H(M) + R) \bmod q$, $v = s^{-1}t \bmod q$, then let $(K'_x, K'_y) = uG + v \cdot Q$. Finally, the verifier output 1 if $t' = K'_x \bmod q = t$; otherwise, the verifier outputs $\perp$.

7.2 Generic Group Oracle for Variant-ECDSA-1

The oracle for variant-ECDSA-1 under generic group model is shown in Table 4. Compared to Brown's oracle, there is no hint command in this oracle but some special operations for special case is provided.

Table 4: Generic group oracle for variant-ECDSA-1

```
$A_{m+1} \leftarrow$ Push ($A \in \mathbb{A}_n$).
    1.Let $A_{m+1} = A$.
    2.If $A_{m+1} = A_i$ for some $i \in \{1, \cdots, m\}$,
    let $z_{m+1} = z_i$.
    3.If $A_{m+1} \notin \{A_1, \cdots, A_m\}$, choose
    $z_{m+1} \in_R \mathbb{Z}_n \backslash \{z_1, \cdots, z_m\}$.
$A_{m+1} \leftarrow$ Subtract (No argument)
    1.Let $z_{m+1} = (z_{m-1} - z_m)$ mod $n$.
    2.If $z_{m+1} = z_i$ for some $i \in \{1, \cdots, m\}$,
    let $A_{m+1} = A$.
    3.If $z_{m+1} \notin \{z_1, \cdots, z_m\}$, choose
    $A_{m+1} \in_R \mathbb{A}_n \backslash \{A_1, \cdots, A_m\}$.
Special Case :
    1.Randomly choose a time value $t \in [0, \tau_F]$.
    2.Run the above part of oracle to which forger $F$
      makes arbitrary commands and final verification
      with arbitrary arguments with one exception :
        The first subtract command, say command $i$,
        requested after time $t$, whose derivation $\Delta_i$
        only have two nonzero coefficients ($C_{i1}$ and
        $C_{i2}$) and the private element $z_i$ is
        distinct from all previous private elements.
    3.For this specail case, oracle chooss $A_i$ at random
      from $f^{-1}(eC_{i1}^{-1}C_{i2})$ by using the invertibility of the
      conversion function $f$, if possible (halt and admit
      failure otherwise).
```

7.3 Security Proof of Variant-ECDSA-1

Brown referred some security requirements (such as secure ECDSA group, second-preimage-resistant hash function, etc.) for security proof under generic group model when proved ECDSA is secure, since our variant-ECDSA-1 and variant-ECDSA-2 and our partially blind signatures scheme 2 and scheme 3 use the same parameters and hash function as ECDSA, our two variant-ECDSA and partially blind signatures schemes satisfy those security requirements. To proving our variant ECDSA schemes, we slightly modifed Brown's oracle. We show our oracles for variant-ECDSA-1 in Table 4 and variant-ECDSA-2 in next Chapter.

Theorem 1. If there exists an $(\epsilon_F, \tau_F, 0)$-forger F_h of variant-ECDSA-1 in the generic group model for $\mathbb{A}_n$ with hash function h and with almost-invertible conversion function f, then there exists an (ϵ_I, τ_I)-inverter I_h and (ϵ_Z, τ_Z)-zero-finder Z_h, where:

$$\epsilon_I + \frac{\epsilon_Z}{10\tau_F} \geq \frac{\epsilon_F}{10\tau_F} - \frac{\tau_F}{20n} \tag{7.1}$$

Proof. We call the special command in our oracle for variant-ECDSA-1 *qualified* command. Besides, the derivation Δ_i in *qualified* command is called exceptional derivation. Suppose F is an $(\epsilon_F, \tau_F, q_F)$-forger in the generic group model. We assume that there is an (ϵ_I, τ_I)-inverter I_h of the hash function h that invokes F as subroutine. Here is the further analysis :

We assume that F succeeds with probability ϵ_F. During the verification phase, forger call the oracle to verify the signature $(M, (t, s, R))$ he/she forged. The oracle then follows the variant-ECDSA-1 scheme to compute $V = s^{-1}(H(M) + R)G + s^{-1}tQ$ and compare $t' = f(V) \mod n$ to t. Hence the last response of oracle is $A_m = V$ for some m. Since $A_m = V = s^{-1}(H(M) + R)G + s^{-1}tQ$, $G = A_1$ and $Q = A_2$, the derivation $\Delta_m = (s^{-1}(H(M) + R), s^{-1}t, 0, \cdots, 0)$.

If $H(M) = 0$, a zero has been found, hence we proved the existence of the zero-finder Z_h. If $H(M) \neq 0$, since $t \neq 0$ as well, Δ_i is a derived derivation (more then one non-zero coefficients). There are three possible cases for Δ_m:

- Derivation Δ_m is a new derivation distinct from previous derivations but private element z_m equals to some previous private elements z_i.

- Derivation Δ_m is a new derivation distinct from previous derivations and private element z_m is a new element.

- Derivation $\Delta_m = \Delta_i$ where Δ_i is a previous derivation.

In the first case, there occurred a coincidence. Brown [5] shows that the probability of occurring a coincidence is at most $\binom{m}{2}/n$.

In the second case, since z_m is new and Δ_m is a new derivation, Δ_m is non-exceptional. Therefore A_m is a element randomly selected from $\mathbb{A}_n$ except $\{A_1, ..., A_{m-1}\}$ by the oracle and probability that $f(A_m)$ equals t is at most $10/(n-m)$ since t is determined before the random A_m is chosen.

In the third case, Δ_m equals some previous Δ_i, which has probability at least $1/\tau_F$ of being exceptional. Assume that Δ_i is a $exceptional derivation$ for $i \leq m$. In the oracle, $A_i \in f^{-1}(eC_{i1}^{-1}C_{i2})$, thereby it is obviously that $t = f(A_m) = f(A_i) = eC_{i_1}{}^{-1}C_{i_2} = eC_{m_1}{}^{-1}C_{m_2} = e(H(M)+R)^{-1}t$, where $t \neq 0$ since the forged signature is valid. Hence we have $e = H(M) + R$. Since R is a random number, we have $H(M) = e - R = e'$. It is obviously that there exists a inverter I_h.

According to Brown [5], there is a probability at least $1/10$ that f can be inverted successfully in the exceptional derivation. Thus, we have:

$$\epsilon_I \geq \frac{\epsilon_F - \epsilon_Z - \frac{\binom{m}{2}}{n} - \frac{10}{n-m}}{10\tau_F} \tag{7.2}$$

Assume that $m \leq \tau_F$ then we obtain the inequality in our theorem.

□

Chapter 8

Variant-ECDSA-2

8.1 Scheme of Variant-ECDSA-2

- **Setup:** An elliptic curve $\mathbb{E}$ is randomly selected by this algorithm, following by a group generator G of prime order q over $\mathbb{E}$. It then randomly selects a hash function $H(\cdot)$. Finally, this algorithm outputs a set of public parameters $pp = (\mathbb{E}, G, q, H)$.
- **KeyGen:** The Signer randomly selects their private key $d \in [2, q-1]$ and then computes their public key $Q = d \cdot G$.
- **Sign:** For a message m, the signer randomly selects a integer k from 2 to $q-1$ and computes $(K_x, K_y) = kG$, $t = K_x$ mod q. Besides, signer random a R and compute $s = k^{-1}(H(m||R) + (t+R) \cdot d)$ mod q. Finally, the signer outputs a signature $\sigma = (t, s, R)$.
- **Verify:** Anyone who knows the signer's public key can validate the signature $\sigma = (t, s, R)$. First, a verifier can compute $u = s^{-1}H(m||R)$ mod q, $v = s^{-1}(t+R)$ mod q, then let $(K'_x, K'_y) = uG + v \cdot Q$. Finally, the verifier output 1 if $t' = K'_x$ mod $q = t$; otherwise, the verifier outputs $\perp$.

8.2 Generic Group Oracle for Variant-ECDSA-2

The oracle for variant-ECDSA-2 under generic group model is shown in Table 5. The hint command is invoked to response to the signing query in this oracle.

Table 5: Generic group oracle for variant-ECDSA-2

$A_{m+1} \leftarrow$ Push $(A \in \mathbb{A}_n)$.	
	1.Let $A_{m+1} = A$. 2.If $A_{m+1} = A_i$ for some $i \in \{1, \cdots, m\}$, let $z_{m+1} = z_i$. 3.If $A_{m+1} \notin \{A_1, \cdots, A_m\}$, choose $z_{m+1} \in_R \mathbb{Z}_n \backslash \{z_1, \cdots, z_m\}$.
$A_{m+1} \leftarrow$ Subtract (No argument)	
	1.Let $z_{m+1} = (z_{m-1} - z_m) \bmod n$. 2.If $z_{m+1} = z_i$ for some $i \in \{1, \cdots, m\}$, let $A_{m+1} = A_i$. 3.If $z_{m+1} \notin \{z_1, \cdots, z_m\}$, choose $A_{m+1} \in_R \mathbb{A}_n \backslash \{A_1, \cdots, A_m\}$.
$(f(A_{m+1}), s_{m+1}, R_{m+1}) \leftarrow$ Signing query (M_i).	

	1.Random $R_{m+1} \in \mathbb{Z}_n$.
	2.Compute $h_{m+1} = H(M_i \vert\vert R_{m+1})$.
	3.Invoke hint command:
	$(A_{m+1}, s_{m+1}) \leftarrow$ Hint $(h_{m+1} \in \mathbb{Z}_n \backslash \{0\})$.
	  1.Randomly choose $z_{m+1} \in \mathbb{Z}_n$ except
	  elements in $\{z_1, \cdots, z_m\}$.
	  2.Randomly choose $A_{m+1} \in \mathbb{A}_n$ except
	  elements in $\{A_1, \cdots, A_m\}$.
	  3.$s_{m+1} = {z_{m+1}}^{-1}(h_{m+1} z_1 + f(A_{m+1}) z_2) \bmod n$
	where $f : \mathbb{A}_n \to \mathbb{Z}_n$ is a certain fixed function.

8.3 Security Proof of variant-ECDSA-2

Theorem 2. If there exists an S-selective $(\epsilon_F, \tau_F, q_F)$-forger F_h of variant-ECDSA-2 in the generic group model for $\mathbb{A}_n$ with hash function h and with conversion function f almost-bijective of strength at least β, then there exists an (ϵ_S, τ_S, S)-second-preimage-finder S_h and (ϵ_Z, τ_Z)-zero-finder Z_h, where:

$$\epsilon_S \geq \epsilon_F - \frac{{\tau_F}^2}{2n} - (\frac{1}{\tau_F} - \frac{1}{n})^{-1} \beta \tag{8.1}$$

Proof. Suppose F is an $(\epsilon_F, \tau_F, q_F)$-forger in the generic group model for group $\mathbb{A}_n$. We assume that there is an (ϵ_S, τ_S, S)-second-preimage-finder S_h of the hash function h that invokes forger F as subroutine. Here is the further analysis :

We assume that F succeeds with probability ϵ_F. During the verification phase, forger call the oracle to verify the signature $(M, (t, s, R))$ he/she forged. The oracle then follows the variant-ECDSA-1 scheme to compute $V = s^{-1}H(M||R)G + s^{-1}(t+R)Q$ and compare $t' = f(V) \mod n$ to t. Hence the last response of oracle is $A_m = V$ for some m. Since $A_m = V = s^{-1}H(M||R)G + s^{-1}(t+R)Q$, where $G = A_1$ and $Q = A_2$, the derivation of A_m is $\Delta_m = (s^{-1}H(M||R), s^{-1}(t+R), 0, \cdots, 0)$. There are four possible cases for Δ_m:

- Derivation Δ_m is a new derivation distinct from previous derivations but private element z_m equals to some previous private elements z_i.
- Derivation Δ_m is a new derivation distinct from previous derivations and private element z_m is a new element.
- Derivation $\Delta_m = \Delta_i$ where Δ_i is a previous derivation and arise from a hint command.
- Derivation $\Delta_m = \Delta_i$ where Δ_i is a previous derivation which is not arising from a hint command.

In the first case, there occurred a coincidence. Brown [5] shows that the probability of occurring a coincidence is at most $\binom{m}{2}/n$.

In the second case, since z_m is new and Δ_m is a new derivation, Δ_m is non-exceptional. Therefore A_m is a element randomly selected from $\mathbb{A}_n$ except $\{A_1, ..., A_{m-1}\}$ by the oracle. Since $f(A_i) = f(A_m) = t = C_{m2}{C_{m1}}^{-1}h(M||R) = C_{i2}{C_{i1}}^{-1}h(M||R)$ is determined before the random A_i is chosen and f is almost-bijective, this has with probability at most $n(n-m)^{-1}\beta$ according to Brown [5].

In the third case, Δ_m equals some previous Δ_i, which means that $C_{m1} = C_{i1}$ and $C_{m2} = C_{i2}$. Therefore we have $s^{-1}H(M||R) = {s_i}^{-1}H(M_i||R_i)$ and $s^{-1}t = {s_i}^{-1}f(A_i)$. Since $t = f(V) = f(A_m) = f(V)$, we have $s = s_i$, as a result $H(M||R) = H(M_i||R_i)$. Obviously, forger has to forge a signature for a new message M other than arguments M_i of signing queries. Therefore $M||R \neq M_i||R_i$. It is obviously that there exists a second-preimage-finder S_h.

In the last case, if $h(M||R) = 0$, a zero has been found, thereby there exists zero-finder Z_h. If $h(M||R) \neq 0$, since $t \neq 0$ as well, Δ_i is a derived derivation (more then one non-zero coefficients). A_i is derived and is randomly selected by the oracle from $\mathbb{A}_n$ except $\{A_1, \cdots, A_{i-1}\}$. Since $f(A_i) = f(A_m) = t = C_{m2}{C_{m1}}^{-1}h(M||R) = C_{i2}{C_{i1}}^{-1}h(M||R)$ is determined before the random A_i is chosen and f is almost-bijective, this has with probability at most $n(n-m)^{-1}\beta$. And the probability is at most $mn(n-m)^{-1}\beta$ after summing all possible i.

Thus, we have:

$$\epsilon_S \geq \epsilon_F - \frac{mn\beta}{n-m} - \frac{\binom{m}{2}}{n} - \frac{n\beta}{n-m} \tag{8.2}$$

We assume that $\tau_F < m$ to obtain the desired inequality.

□

Chapter 9

Partially Blind Signature

In this Chapter, we show the definition and security requirements of a partially blind signature.

9.1 Definition of Partially Blind Signature

A partially blind signature scheme consists of four algorithms, namely **Setup**, **KeyGen**, **Issue**, and **Verify**, described as follows:

- **Setup** is a probabilistic polynomial-time algorithm that takes security parameter λ as input and outputs a set of public parameters $params$.
- **KeyGen** is a probabilistic polynomial-time algorithm that takes a public parameters $params$ as input and outputs a signer's signing key sk and its corresponding public verification key pk.
- **Issue** is an interactive protocol between signer and user, described as follows.
  - The **Agree** algorithm allows user and signer to generate common information $info$.
  - The **Blind** algorithm takes a random string r, a message m, and common information $info$ as input and outputs a string h for signer.
  - The **Sign** algorithm takes a string h and signer's signing key sk as input and outputs a blind signature $\bar{\sigma}$ which will be unblinded by user.

  - The **Unblind** algorithm takes a signature $\bar{\sigma}$ and the previously used random string r as input and outputs an unblinded signature σ.

- **Verify** is a deterministic polynomial-time algorithm that takes an unblinded signature σ, message m, negotiated information $info$, and signer's public verification key pk as input, and outputs "true" if σ is a valid signature signed by signer with the corresponding signing key sk on message m and common information $info$. Otherwise, it outputs "false".

9.2 Security Definitions of Partially Blind Signature

Partially blind signature has to meet two security requirements in usual: unforgeability and partial blindness. We now define them as below.

9.2.1 Unforgeability

To model the unforgeability security requirement, we define Game-A played by challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

Game - A:

- **Setup:** Challenger $\mathcal{C}$ takes a security parameter λ and generates public parameters $params$. $\mathcal{C}$ subsequently sends λ and $params$ to adversary $\mathcal{A}$.
- **Attack:** Adversary $\mathcal{A}$ engages in the signature issuing protocol with challenger $\mathcal{C}$ in a concurrent and interleaving way. For each $info$, the number of adversary $\mathcal{A}$ executing the signature issuing protocol with Challenger $\mathcal{C}$ until obtain the signature is l_{info}.
- **Response:** Adversary $\mathcal{A}$ outputs a $info$ and l_{info+1} signatures $(m_1, \sigma_1), \cdots, (m_{l_{info+1}}, \sigma_{l_{info+1}})$.

Definition 4. A partially blind scheme is existentially unforgeable against adaptive chosen-message attacks if no adversary wins the Game-A with non-negligible probability.

9.2.2 Partial Blindness

To model the partial blindness security requirement, we define Game-B played between challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

Game - B:

- **Setup:** Challenger $\mathcal{C}$ takes a security parameter λ and generates public parameters $params$. $\mathcal{C}$ subsequently sends λ and $params$ to adversary $\mathcal{A}$.
- **Preparation:** Adversary $\mathcal{A}$ selects two different messages m_0, m_1 and a common information $info$. $\mathcal{A}$ then sends $(m_0, m_1, info)$ to $\mathcal{C}$.
- **Challenge:** Challenger $\mathcal{C}$ selects a random bit b, then $\mathcal{A}$ sign both m_b with $info$ and m_{1-b} with $info$. After unblinding the response of $\mathcal{A}$, $\mathcal{C}$ sends $(m_0, m_1, info, \sigma_b, \sigma_{1-b})$ to $\mathcal{A}$.
- **Response:** Adversary $\mathcal{A}$ outputs a guess b'. If $b' = b$, then $\mathcal{A}$ wins.

Definition 5. A partially blind scheme has partial blindness if no adversary wins the Game-B with non-negligible probability.

Chapter 10

The Proposed Scheme 1

We proposed three partially blind signature schemes. In this Chapter we introduce a partially blind signature based on Schnorr blind signature, using random numbers as blinding factors. In Chapter 11 and Chapter 12 we proposed two partially blind signatures based on variant-ECDSA-1 and variant-ECDSA-2 respectively, using Paillier cryptosystem to blind messages.

The selection of public parameters is similar with ECDSA. Public parameters $params = (\mathcal{E}, G, q, H, H_0)$, where $H : \{0,1\}^* \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*$ and $H_0 : \{0,1\}^* \to \mathbb{Z}_q^*$. To generate the key pair, the signer randomly selects a $d \in [2, \cdots, q-1]$ as the secret key that remains secret. Subsequently, the signer publishes their public key $Q = dG$. To generate a signature, the signer and user interact as presented in Fig. 2. Anyone who knows the signer's public key can verify this signature by verifying $e = H(m \| info, R_x(eQ + sG + cQ)$ mod $q)$, where R_x denotes the obtained *x* coordinate of an elliptic curve point.

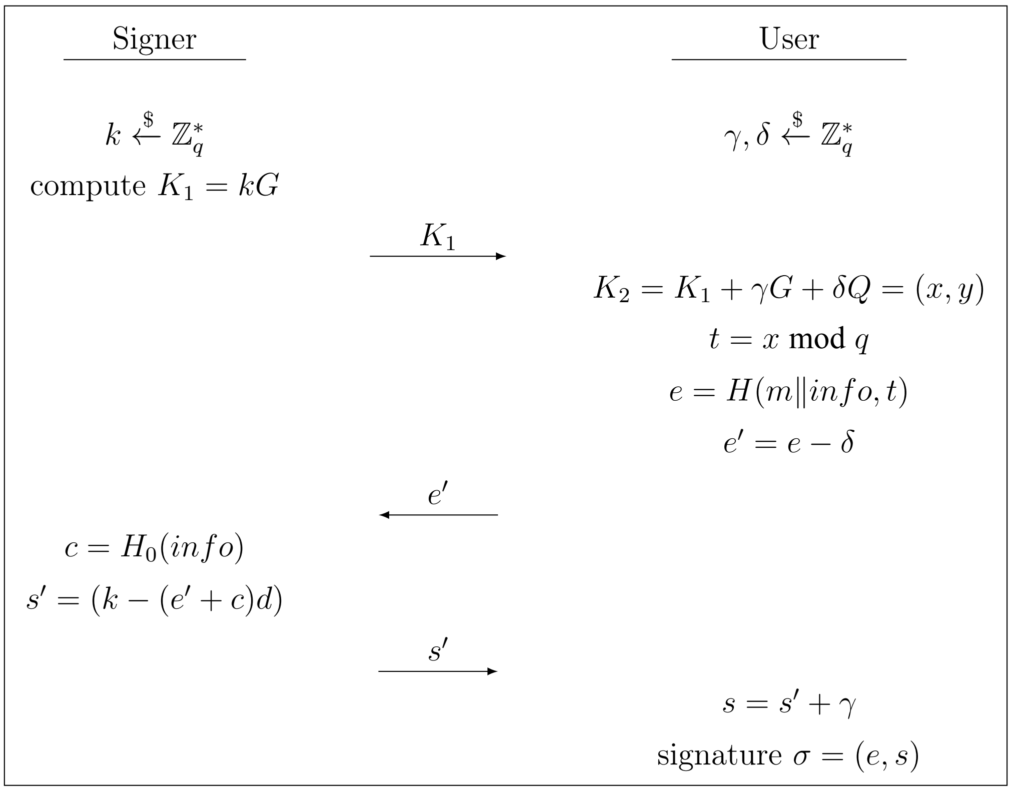

Figure 2: Our first scheme

The valid signature $\sigma = (e, s)$ can pass the verification because

$$
\begin{aligned}
e &= H(m \| info, t) \\
&= H(m \| info, R_x(k + \delta d + \gamma)G \bmod q) \\
&= H(m \| info, R_x(ed + (k - ed + \delta d - cd + \gamma) + cd)G \bmod q) \\
&= H(m \| info, R_x(ed + (k - e'd - cd + \gamma) + cd)G \bmod q) \\
&= H(m \| info, R_x(ed + (k - e''d) + \gamma + cd)G \bmod q) \\
&= H(m \| info, R_x(ed + s' + \gamma + cd)G \bmod q) \\
&= H(m \| info, R_x(eQ + sG + cQ) \bmod q)
\end{aligned}
$$

Chapter 11

The Proposed Scheme 2

In this Chapter, a partially blind signature scheme based on variant-ECDSA-1 that we mentioned in Chapter 7, is proposed.

The selection of public parameters in this scheme is the same as that in ECDSA (*i.e.,* $params = (\mathcal{E}, G, q, H)$, where $H : \{0, 1\}^* \to \mathbb{Z}_q^*$. To generate the key pair, the signer randomly selects a $d \in [2, \cdots, q-1]$ as the secret key that remains secret. Then, the signer publishes their public key $Q = dG$. To generate a signature, the signer and user interact as presented in Fig. 3. In particular, we adopt the modified Paillier encryption $(KeyGen, Enc, Dec)$ of Yi *et al.* [52]. More preciously, user needs to generate a modified Paillier encryption key pair $((N, g), (p, k))$ by using the $KeyGen$ algorithm and then to generate ciphertexts $C_1 = Enc(H(m)) = g^{H(m)} r_1^N$ (mod N^2) and $C_2 = Enc(t) = g^t r_2^N \pmod{N^2}$, where $r_1, r_2 \overset{\$}{\leftarrow} \mathbb{Z}_{N^2}^*$.

Moreover, we adopt a zero-knowledge proof into our protocol. The user must provide corresponding zero-knowledge proofs to convince the signer that C_1 and C_2 comply with the modified Paillier encryption with the form $g^m r^N \pmod{N^2}$, where m is the plaintext. We now described the process of interactive zero-knowledge proof. To prove that C_1 is a correct form, the user first randomly selects $x \in \mathbb{Z}_q, r' \in \mathbb{Z}_{N^2}^*$ and then sends $C_1' = g^x r'^N (\text{mod } N^2)$ to the signer. After receiving C_1', the signer selects $b \in \{0, 1\}$ and returns it to the user. The user then processes it on the basis of the value b. If $b = 0$, the user provides (x, r') to the signer. Otherwise, the user computes $x' = H(m) + x(\text{mod } q), r'' = rr'(\text{mod } N^2)$ and sends (x', r'') to the signer. Signer also processes based on the value of b. If $b = 0$, the signer evaluates whether $C_1' = g^x r'^N (\text{mod } N^2)$. Otherwise, the signer verifies whether $C_1 C_1' = g^{x'} r''^N (\text{mod}$ $N^2)$. After completing this process ℓ times, the user can convince the signer that the ciphertext

C_1 is generated with a probability of $1 - 1/2^\ell$ in accordance with the Paillier encryption scheme. In addition, the user must use the same method to convince the signer that C_2 is the correct form. This kind of zero-knowledge proof is called interactive, which can be transferred to non-interactive process [15].

Except sending C_1, C_2 along with zero-knowledge proofs, to make sure that user does not replace $info$ with other $info'$, we require he/she to run the following steps. User randomly selects $(l_1, \cdots, l_n) \leftarrow \mathbb{Z}^*_{\phi(N^2)}$, computes $F_i = g^{l_i I} l_i^N \bmod N^2$ for $i = 1, \cdots, n$, where $I = H(info)$, and sends $F_1, \cdots, F_n$ to signer. After accepting C_1, C_2 and their zero-knowledge proofs from user, signer randomly chooses u different numbers $i = (i_1, \cdots, i_u)$ from $[1, \cdots, n]$ and send the list i to user. User now starts the opening phase and discloses $L = (l_1, \cdots, l_n)$ except l_j where $j \in (i_1, \cdots, i_u)$. Signer can use L to check whether all the opening ciphertexts are correct. If there exists a ciphertext is not generated correctly, signer aborts it. Since user have no idea what indices signer will pick, he/she cannot replace $info$ with some other $info'$. We provide a detail discussion in Chapter 16.

After interacting with the signer, the user obtains a final signature (t, s, R), where $R = l_{i_1} + \cdots, l_{i_u} \bmod q$. Anyone who knows the public key of signer can verify this signature in the ECDSA by computing the following steps:

- Compute $u = s^{-1}(H(m) + RH(info)) \bmod q$, where $R = l_{i_1} + \cdots + l_{i_u}$, and $v = s^{-1}t \bmod q$.
- Compute $(K'_x, K'_y) = uG + vQ$ and $t' = K'_x \bmod q$.
- Verify whether $t' = t$.

Subsequently, we analyze how a valid signature $\sigma = (t, s, R)$ can pass the verification. First, because $C_1 = g^{H(m)} r_1^N \bmod N^2, C_2 = g^t r_2^N \bmod N^2$, $I = H(info)$ and $R = l_{i_1} + \cdots + l_{i_u} \bmod q$, we have

$$
\begin{aligned}
C &= (C_1 C_2^d F_{i_1} \cdots F_{i_u})^{k_1^{-1} \pmod q} \cdot r^N \pmod{N^2} \\
&= \left((g^{H(m)} r_1^N)(g^t r_2^N)^d (g^{l_{i_1} I} {l_{i_1}}^N) \cdots (g^{l_{i_u} I} {l_{i_u}}^N)\right)^{k_1^{-1} \pmod q} \\
&\quad \cdot r^N \pmod{N^2} \\
&= g^{k_1^{-1}(H(m)+RI+td) \pmod q} \\
&\quad \cdot \left(r_1^{k_1^{-1}} r_2^{k_1^{-1} d} (l_{i_1} \cdots l_{i_u})^{k_1^{-1}} r\right)^N \pmod{N^2} \\
&= Enc(k_1^{-1}(H(m) + RI + td)).
\end{aligned}
$$

Moreover, because $g^q = (1+N)^{pqk} = (1+N)^N = 1 \pmod{N^2}$, we have

$$
\begin{aligned}
s &= k_2^{-1} Dec(C, (p, k)) \\
&= k_2^{-1} k_1^{-1} (h + Rh' + td) \pmod q,
\end{aligned}
$$

where $h = H(m)$ and $h' = H(info)$, and

$$
\begin{aligned}
K &= (K_x, K_y) \\
&= k_2 K_1 \\
&= k_2 (k_1) G \\
&= k_1 k_2 G.
\end{aligned}
$$

Therefore, if the signature is valid, $t' = t$.

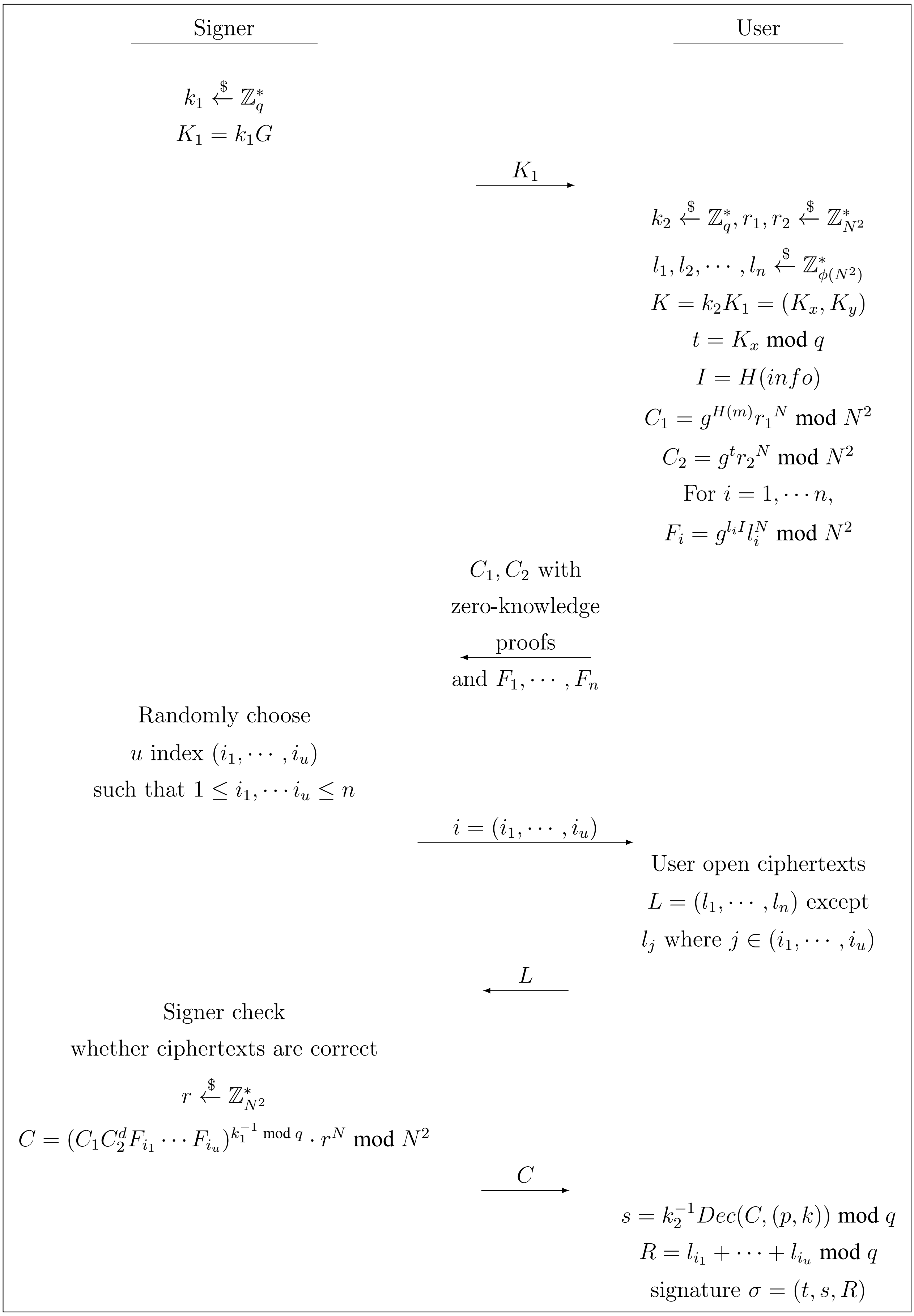
Signer
User
$k_1 \xleftarrow{\$} \mathbb{Z}_q^*$
$K_1 = k_1 G$
$\xrightarrow{K_1}$
$k_2 \xleftarrow{\$} \mathbb{Z}_q^*, r_1, r_2 \xleftarrow{\$} \mathbb{Z}_{N^2}^*$
$l_1, l_2, \cdots, l_n \xleftarrow{\$} \mathbb{Z}_{\phi(N^2)}^*$
$K = k_2 K_1 = (K_x, K_y)$
$t = K_x \bmod q$
$I = H(info)$
$C_1 = g^{H(m)} {r_1}^N \bmod N^2$
$C_2 = g^t {r_2}^N \bmod N^2$
For $i = 1, \cdots n$,
$F_i = g^{l_i I} l_i^N \bmod N^2$
C_1, C_2 with zero-knowledge proofs and $F_1, \cdots, F_n$
Randomly choose u index $(i_1, \cdots, i_u)$ such that $1 \leq i_1, \cdots i_u \leq n$
$\xrightarrow{i = (i_1, \cdots, i_u)}$
User open ciphertexts $L = (l_1, \cdots, l_n)$ except l_j where $j \in (i_1, \cdots, i_u)$
$\xleftarrow{L}$
Signer check whether ciphertexts are correct
$r \xleftarrow{\$} \mathbb{Z}_{N^2}^*$
$C = (C_1 C_2^d F_{i_1} \cdots F_{i_u})^{k_1^{-1} \bmod q} \cdot r^N \bmod N^2$
$\xrightarrow{C}$
$s = k_2^{-1} Dec(C, (p, k)) \bmod q$
$R = l_{i_1} + \cdots + l_{i_u} \bmod q$
signature $\sigma = (t, s, R)$

Figure 3: Our second scheme

Chapter 12

The Proposed Scheme 3

We proposed a partially blind signature scheme based on variant-ECDSA-2 that mentioned in Chapter 8, in this Chapter.

Both scheme 2 and scheme 3 are based on variants of ECDSA and adopt the modified Paillier encryption system we mentioned in Chapter 3.

The selection of public parameters in this scheme is the same as that in ECDSA (*i.e.,* $params = (\mathcal{E}, G, q, H, H_1)$, where $H : \{0,1\}^* \to \mathbb{Z}_q^*$ and H_1 is a function which map random string to a elliptic curve point. To generate the key pair, the signer randomly selects a $d \in [2, \cdots, q-1]$ as the secret key that remains secret. Then, the signer publishes their public key $Q = dG$. To generate a signature, the signer and user interact as presented in Fig. 4. The odified Paillier encryption $(KeyGen, Enc, Dec)$ of Yi *et al.* and the zero-knowledge proof are adopted there. User first generates a modified Paillier encryption key pair $((N, g), (p, k))$ by using the $KeyGen$ algorithm and then generates ciphertexts $C_1 = Enc(H(m, I)) = g^{H(m,I)} r_1^N$ (mod N^2) and $C_2 = Enc(t + \alpha_2) = g^{t+\alpha_2} r_2^N \pmod{N^2}$, where $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_{N^2}^*$.

In our third scheme, the user needs to provide corresponding zero-knowledge proofs of C_1 and C_2 well, with the process that we described in previous section. In additional, in this scheme, there is no need to generate additionally ciphertexts $F_i = g^{l_i I} l_i^N \bmod N^2$ other than C_1 and C_2. Our scheme 3 use a different method to prevent user from replacing the common information $info$ with other information $info'$ decided by user himself/herself. Precisely, we hash the common information $info$ into a elliptic curve point Z . Both signer and user will bind this point Z to a random number (α_1 for signer and α_2 for user). The random number α_1 will participate the generation of ciphertext C, which makes it harder for user to replace $info$.

After interacting with the signer, the user obtains a final signature (t, s, R). Anyone who knows the public key of signer can verify this signature in the ECDSA by computing the following steps:

- Compute $u = s^{-1}H(m) \bmod q$ and $v = s^{-1}(t + R) \bmod q$.
- Compute $(K'_x, K'_y) = uG + vQ$ and $t' = K'_x \bmod q$.
- Verify whether $t' = t$.

Subsequently, we analyze how a valid signature $\sigma = (t, s, R)$ can pass the verification. First, because $C_1 = g^{H(m)} r_1^N \bmod N^2, C_2 = g^{t+\alpha_2} r_2^N \bmod N^2$, we have

$$\begin{aligned}
C &= \left(C_1 (g^{\alpha_1} C_2)^d\right)^{k_1^{-1} \bmod q} \cdot r^N \bmod N^2 \\
&= \left((g^{H(m,I)} r_1^N)(g^{\alpha_1} g^{t+\alpha_2} r_2^N)^d\right)^{k_1^{-1} \pmod q} \\
&\quad \cdot r^N \pmod{N^2} \\
&= g^{k_1^{-1}(H(m,I)+(t+\alpha_1+\alpha_2)d) \pmod q} \\
&\quad \cdot \left(r_1^{k_1^{-1}} r_2^{k_1^{-1} d} r\right)^N \pmod{N^2} \\
&= g^{k_1^{-1}(H(m,I)+(t+R)d) \pmod q} \\
&\quad \cdot \left(r_1^{k_1^{-1}} r_2^{k_1^{-1} d} r\right)^N \pmod{N^2} \\
&= Enc(k_1^{-1}(H(m,I) + (t+R)d)).
\end{aligned}$$

Moreover, because $g^q = (1+N)^{pqk} = (1+N)^N = 1 \pmod{N^2}$, we have

$$\begin{aligned}
s &= k_2^{-1} Dec(C, (p, k)) \\
&= k_2^{-1} k_1^{-1}(h + (t+R)d) \pmod q,
\end{aligned}$$

where $h = H(m, I)$, and

$$\begin{aligned} K &= (K_x, K_y) \\ &= k_2 K_1 \\ &= k_2(k_1)G \\ &= k_1 k_2 G. \end{aligned}$$

Therefore, if the signature is valid, $t' = t$.

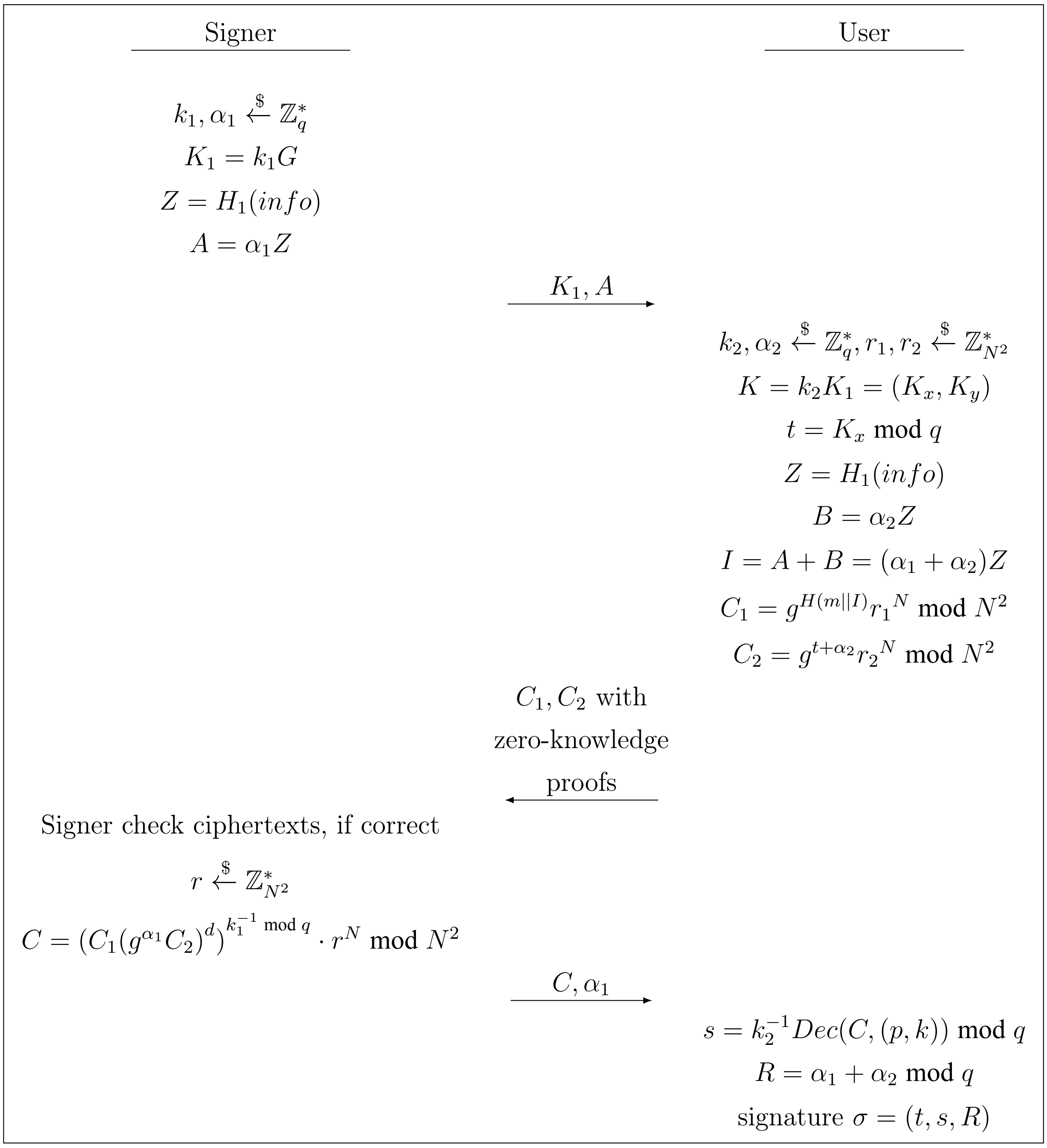
Signer
User
$k_1, \alpha_1 \xleftarrow{\$} \mathbb{Z}_q^*$
$K_1 = k_1 G$
$Z = H_1(info)$
$A = \alpha_1 Z$
$\xrightarrow{K_1, A}$
$k_2, \alpha_2 \xleftarrow{\$} \mathbb{Z}_q^*, r_1, r_2 \xleftarrow{\$} \mathbb{Z}_{N^2}^*$
$K = k_2 K_1 = (K_x, K_y)$
$t = K_x \bmod q$
$Z = H_1(info)$
$B = \alpha_2 Z$
$I = A + B = (\alpha_1 + \alpha_2) Z$
$C_1 = g^{H(m||I)} {r_1}^N \bmod N^2$
$C_2 = g^{t+\alpha_2} {r_2}^N \bmod N^2$
$\xleftarrow{C_1, C_2 \text{ with zero-knowledge proofs}}$
Signer check ciphertexts, if correct
$r \xleftarrow{\$} \mathbb{Z}_{N^2}^*$
$C = \left(C_1 (g^{\alpha_1} C_2)^d\right)^{k_1^{-1} \bmod q} \cdot r^N \bmod N^2$
$\xrightarrow{C, \alpha_1}$
$s = k_2^{-1} Dec(C, (p, k)) \bmod q$
$R = \alpha_1 + \alpha_2 \bmod q$
signature $\sigma = (t, s, R)$

Figure 4: Our third scheme

Chapter 13

Security Analysis

Theorem 3. Our first scheme satisfies unforgeability if ECDLP is hard.

Proof. Assume that adversary $\mathcal{A}$ can break the unforgeability of our first scheme with non-negligible probability. Then, a challenger $\mathcal{C}$ can make use of adversary $\mathcal{A}$ to solve ECDLP. At first, $\mathcal{C}$ is given an elliptic curve $\mathcal{E}$, a multiplicative group $\mathcal{G}$ with generator G and prime order q, and a group element $Q \in \mathcal{G}$.

- Setup: $\mathcal{C}$ selects hash functions $H : \{0,1\}^* \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*$ and $H_0 : \{0,1\}^* \to \mathbb{Z}_q^*$ which act as random oracles. Then, $\mathcal{C}$ initials two lists L_1, L_2, sets Q as the public key, and sends Q to $\mathcal{A}$.
- Attack: In this phase, $\mathcal{A}$ can adaptively makes the following requests:
  - H_0 request: When $\mathcal{A}$ requests the oracle with $info_i$, $\mathcal{C}$ first verifies whether $(info_i, c_i)$ is in list L_1. If it exists, $\mathcal{C}$ responds $\mathcal{A}$ with c_i; otherwise, $\mathcal{C}$ randomly selects a $c_i \in \mathbb{Z}_q^*$ as a response to $\mathcal{A}$ and stores $(info_i, c_i)$ in list L_1 for consistency and to avoid collision.
  - H request: Similarly, when $\mathcal{A}$ requests the oracle with $(m_i, info_i, t_i)$, $\mathcal{C}$ first verifies whether any $(m_i, info_i, t_i, e_i)$ pairs are in list L_2. If one exists, $\mathcal{C}$ responds $\mathcal{A}$ with e_i; otherwise, $\mathcal{C}$ randomly selects a $e_i \in \mathbb{Z}_q^*$ as a response to $\mathcal{A}$ and stores $(m_i, info_i, t_i, e_i)$ in list L_2 for consistency and to avoid collision.
  - Issue: When $\mathcal{A}$ requests the oracle with $(m_i, info_i)$, $\mathcal{C}$ selects $e_i \in \mathbb{Z}_q^*$ and $s_i \in \mathbb{Z}_q^*$ randomly and lets $t_i = R_x(eQ + sG + cQ) \bmod q$, where $R_x(\cdot)$ is the

x coordinate of the input value. When calculating t_i, $\mathcal{C}$ verifies whether any $(m_i, info_i, t_i)$ pairs are in L_2. If one is present, $\mathcal{C}$ selects a different s_i and repeats the aforementioned steps. Finally, $(m_i, info_i, t_i, e_i)$ is stored in list L_2. $\mathcal{C}$ then sends (e_i, s_i) to $\mathcal{A}$.

- Forgery: In this phase, $\mathcal{A}$ outputs a valid signature $\sigma^* = (e^*, s^*)$ for message m^* and agreed information $info^*$.

By using a forking lemma [39], $\mathcal{C}$ can construct a Las Vegas machine to produce another signature $\bar{\sigma} = (\bar{e}, \bar{s})$, where $\bar{e} \neq e^*$. $\mathcal{C}$ then solves the ECDLP by computing $(\bar{e} - e^*)^{-1}(s^* - \bar{s})$ and obtains d such that $dG = Q$ as follows (random numbers in two signatures are the same):

$$
\begin{aligned}
&(\bar{e} - e^*)^{-1}(s^* - \bar{s}) \\
&= (\bar{e} - e^*)^{-1}\big((k - (\bar{e}' + c)d + \gamma) - (k - (\bar{e}' + c)d + \gamma)\big) \\
&= (\bar{e} - e^*)^{-1}\big((k - ((e^* - \delta) + c)d + \gamma) \\
&\qquad - (k - ((\bar{e} - \delta) + c)d + \gamma)\big) \\
&= (\bar{e} - e^*)^{-1}\big((\bar{e} - \delta)d - (e^* - \delta)d\big) \\
&= (\bar{e} - e^*)^{-1}(\bar{e} - e^*)d \\
&= d
\end{aligned}
$$

□

Theorem 4. Our first scheme satisfies partial blindness.

Proof. Supposing adversary $\mathcal{A}$ is given a challenge tuple $(m_0, m_1, info, (e_0, s_0), (e_1, s_1))$. Moreover, the adversary's view of the signing process is $(K_1, (e'_0, s'_0), (e'_1, s'_1))$. Some blind random factors can map (K_1, e'_i, s'_i) to (e_j, s_j) for any $i, j \in \{0, 1\}$; therefore, $\mathcal{A}$ cannot distinguish between two message–signature pairs from their view of the signing process. Let $\gamma' = e - e'$, $\delta' = s - s'$, and $c = H_0(info)$. Because $s' = (k - (e' + c)d)$, we have:

$$
\begin{aligned}
e &= H(m\|info, R_x(eQ + sG + cQ) \bmod q) \\
&= H(m\|info, R_x((e' + \gamma')Q + (s' + \delta') + cQ) \bmod q) \\
&= H(m\|info, R_x(e'dG \\
&\qquad + (k - (e' + c)d)G + \gamma' dG + \delta' G + cdG) \bmod q) \\
&= H(m\|info, R_x(e'dG + kdG - e'dG - cdG \\
&\qquad + \gamma dG + \delta' G + cdG) \bmod q) \\
&= H(m\|info, R_x(kdG + \delta dG + \gamma G) \bmod q) \\
&= H(m\|info, R_x(K_1 + \gamma G + \delta Q) \bmod q) \\
&= H(m\|info, R_x(K_2) \bmod q) \\
&= H(m\|info, t)
\end{aligned}
$$

Hence, regardless of the values of $(K_1, {e'}_i, {s'}_i)$ and (e_j, s_j), some blind random factors always lead to the same relationship defined in the signing protocol.

Moreover, (e_0, s_0) has the same distribution as (e_1, s_1) where $(e_0, s_0), (e_1, s_1)$ are generated relative to the random numbers (γ, δ) chosen from $\mathbb{Z}_q^*$. Therefore, $\mathcal{A}$ cannot distinguish between two messages from the signature pairs in their view. □

Theorem 5. Our second scheme satisfies unforgeability if the vriant-ECDSA-1 is unforgeable.

Proof. Because adversary $\mathcal{A}$ must provide a ciphertext of $H(m)$ (i.e., C_1) and t (i.e., C_2) with their zero-knowledge proofs to challenger $\mathcal{C}$ to convince $\mathcal{C}$ that $\mathcal{A}$ produced the ciphertexts (C_1, C_2) in accordance with the Paillier cryptosystem honestly. $\mathcal{A}$ follows the Paillier encryption process, and challenger $\mathcal{C}$ responds with $C = (C_1 C_2^d F_{i_1} \cdots F_{i_u})^{k_1^{-1} \pmod q} r^N \pmod{N^2} = Enc(k_1^{-1}(H(m) + RH(info) + td)) \pmod q)$. Therefore, $\mathcal{A}$ can only obtain a signature (t, s, R) for message m and $info$, where $s = k_2^{-1}[k_1^{-1}(H(m) + RH(info) + td)] \pmod q$ and R is a sum of some random numbers. Note that if adversary $\mathcal{A}$ disobeys the scheme and sends integers other than the ciphertext of $H(m)$ and t, (t, s, R) will not be a valid signature for message m and $info$.

Moreover, the signature (t, s, R) that scheme 2 output is basically a valid variant-

ECDSA-1 signature. The only difference is: since our scheme 2 is a partially blind signature, verifier in our scheme 2 will calculate the hash value of $info$, say $H(info)$, and obtain $R' = RH(info)$. Verifier then verify the variant-ECDSA-1 signature (t, s, R') through the verification process of variant-ECDSA-1. Since common information $info$ is just a string generated from the negotiation between $\mathcal{A}$ and $\mathcal{C}$ and is known to both participants at the beginning, its presence does not affect the security. Therefore, if the variant-ECDSA-1 is unforgeable, as we proved in Chapter 7, given some partially blind signatures generated from scheme 2, $\mathcal{A}$ cannot forge one more partially blind signature. $\square$

Theorem 6. Our second scheme satisfies partial blindness if the underlying modified Paillier encryption is semantically secure.

Proof. Supposing adversary $\mathcal{A}$ is given a challenge tuple $(m_0, m_1, info, (t_0, s_0, R_0), (t_1, s_1, R_1))$. Besides, the adversary's view of the signing process is $(K_1, (C_1, C_2, F_1, \cdots, F_n, i, L, C)_0, (C_1, C_2, F_1, \cdots, F_n, i, L, C)_1)$. Here, $(C_1, C_2)_0$ and $(C_1, C_2)_1$ can be viewed as the ciphertexts that encrypt $(H(m_0), t_0)$ and $(H(m_1), t_1)$ by using the semantically secure modified Paillier encryption [52]. Therefore, $\mathcal{A}$ cannot obtain any information from the signing view without knowing the secret key. Moreover, although $\mathcal{A}$ can obtain $k_1^{-1}(H(m)+RI+td) \pmod q$ through the homomorphic property of Paillier cryptosystem, without knowing secret key, $\mathcal{A}$ cannot distinguish which message is encrypted, i.e., $m = m_0$ or $m = m_1$. Nevertheless, from $\mathcal{A}$'s view, he/she cannot distinguish between s_0, R_0 and s_1, R_1. In detail, since R is the sum of $l_{i_1}, \cdots, l_{i_u}$ and $\mathcal{A}$ only holds $F_{i_1}, \cdots, F_{i_u}$ (which are ciphertexts of $l_{i_1}I, \cdots, l_{i_u}I$, $I = H(info)$) from user, without the secret key of Paillier encryption, $\mathcal{A}$ cannot distinguish R with any other random number. Therefore, $\mathcal{A}$ cannot distinguish between two messages from the signature pairs and their view. $\square$

Theorem 7. Our third scheme satisfies unforgeability if the variant-ECDSA-2 is unforgeable.

Proof. Because adversary $\mathcal{A}$ must provide a ciphertext of $H(m)$ (i.e., C_1) and $t+\alpha_2$ (i.e., C_2) with their zero-knowledge proofs to challenger $\mathcal{C}$ to convince $\mathcal{C}$ that $\mathcal{A}$ produced the ciphertexts (C_1, C_2) in accordance with the Paillier cryptosystem honestly. $\mathcal{A}$ follows the Paillier encryption process, and challenger $\mathcal{C}$ responds with $C = (g^{\alpha_1} C_1 C_2^d)^{k_1^{-1} \pmod q} r^N \pmod{N^2} = Enc(k_1^{-1}(H(m||I) + (t+R)d)) \pmod q)$, where $I = RZ = RH_1(info)$, and

α_1. Therefore, $\mathcal{A}$ can only obtain a signature (t, s, R) for message m and $info$, where $s = k_2^{-1}[k_1^{-1}(H(m||I) + (t + R)d)] \pmod q$ and R is a random number. Note that, if adversary $\mathcal{A}$ disobeys the scheme and sends integersother than the ciphertext of $H(m||R)$ and $t + \alpha_2$, (t, s, R) will not be a valid signature for message m and $info$.

Nevertheless, the signature (t, s, R) that scheme 3 output is basically a valid variant-ECDSA-2 signature. Here is the only difference: since our scheme 3 is a partially blind signature, verifier in our scheme 3 will use hash function H_1 to calculate the $Z = H_1(info)$. In variant-ECDSA-2 verifier compute $u = s^{-1}H(m||R)$, while in our scheme 3 verifier compute $u = s^{-1}H(m||I)$ where $I = RZ = RH_1(info)$. Since common information $info$ is just a string generated from the negotiation between $\mathcal{A}$ and $\mathcal{C}$ and is known to both participants at the beginning, its presence does not affect the security. Therefore, if the variant-ECDSA-2 is unforgeable, as we proved in Chapter 8, given some partially blind signatures generated from scheme 3, $\mathcal{A}$ cannot forge one more partially blind signature. $\square$

Theorem 8. Our third scheme satisfies partial blindness if the underlying modified Paillier encryption is semantically secure.

Proof. Supposing adversary $\mathcal{A}$ is given a challenge tuple $(m_0, m_1, info, (t_0, s_0, R_0), (t_1, s_1, R_1))$. The adversary's view of the' signing process is $(K_1, (C_1, C_2, C)_0,\ (C_1, C_2, C)_1)$. Here, $(C_1, C_2)_0$ and $(C_1, C_2)_1$ can be viewed as the ciphertexts that encrypt $(H(m_0||I_0), t_0)$ and $(H(m_1||I_1), t_1)$, where $I_0 = ((\alpha_1)_0 + (\alpha_2)_0)H_1(info)$ and $I_1 = ((\alpha_1)_1 + (\alpha_2)_1)H_1(info)$, by using the semantically secure modified Paillier encryption [52]. Therefore, $\mathcal{A}$ cannot obtain any information from the signing view without knowing the secret key. Moreover, although adversary $\mathcal{A}$ can obtain $k_1^{-1}(H(m, I) + (t + R)d) \pmod q$ $(R = \alpha_1 + \alpha_2)$ through the homomorphic property of Paillier cryptosystem (in fact, ciphertext C only), without knowing secret key, $\mathcal{A}$ cannot distinguish which plaintext is encrypted, i.e., $(k_1)_0^{-1}(H(m_0, I_0)+(t_0+R_0)d) \pmod q$ or $(k_1)_1^{-1}(H(m_1, I_1)+(t_1+R_1)d) \pmod q$. Thereby, from $\mathcal{A}$'s view, $\mathcal{A}$ cannot distinguish between s_0, R_0, where $s_0 = (k_2)_0^{-1}(k_1)_0^{-1}(H(m_0, I_0)+ (t_0+R_0)d) \pmod q$ and s_1, R_1, where $s_1 = (k_2)_1^{-1}(k_1)_1^{-1}(H(m_1, I_1)+(t_1+R_1)d) \pmod q$).

Additionally, even though $\mathcal{A}$ try to hide a random integer β into the ciphertext C that he/she returns to $\mathcal{C}$ by $C = E(\beta)$ instead of $k_1^{-1}(H(m, I) + (t + R)d) \pmod q$, it can be detected because the signature (t, s, R), where $s = k_2^{-1}\beta$, is not a valid signature of $\mathcal{A}$. $\square$

Chapter 14

Efficiency Analysis

In this section, we compare our proposed scheme with other related partially blind signature schemes [4, 36, 48] in terms of computational cost and signature size. As presented in Table 6, T_{ML}, T_{EM}, T_{EX}, T_{IN}, T_{BP}, T_{MTP}, and T_{CH} represent the time required to calculate the modular multiplication of two integers, elliptic curve scalar point multiplication, modular exponentiation, modular inversion of an integer, bilinear pairing, map-to-point hash function, and chaotic function, respectively. Moreover, to facilitate result comparison, we use the least time-intensive operation of T_{ML} as the base operation to represent the other operations. According to [19, 20, 46], $T_{EM} \approx 29T_{ML}$, $T_{EX} \approx 240T_{ML}$, $T_{IN} \approx 11.6T_{ML}$, $T_{BP} \approx 87T_{ML}$, and $T_{MTP} \approx 29T_{ML}$.

The comparison results of computational costs are presented in Table 7 and 8. Although scheme [36] and scheme [4] are secure under the standard model, they require more than $20T_{EX}$

Table 6: Operation Notations

Notations	Operation	$\approx T_{ML}$
T_{ML}	Modular multiplication of two integers	1
T_{EM}	Elliptic curve scalar point multiplication	29
T_{EX}	Modular exponentiation	240
T_{IN}	Modular inversion of an integer	11.6
T_{BP}	Bilinear pairing	87
T_{MTP}	Map-to-point hash function	29

Table 7: Comparison-1 of Computational Efficiency

Schemes	Computational cost	
	Issue phase	Verification phase
[36]	$20T_{ML} + 17T_{EX} + 6T_{IN}$	$4T_{ML} + 3T_{EX} + 2T_{BP}$
[4]	$8T_{ML} + (11 + k_1 + k_2)T_{EX} + 5T_{BP}$	$3T_{EX} + 3T_{BP}$
[48]	$7T_{ML} + 6T_{EM} + T_{IN}$	$3T_{ML} + 3T_{EM}$
Scheme 1	$T_{ML} + 3T_{EM}$	$3T_{EM}$
Scheme 2	$(n + 2u + 3)T_{ML} + 2T_{EM}$ $+(2n + 2u + 4\ell + 5)T_{EX} + 2T_{IN}$	$2T_{ML} + 2T_{EM} + T_{IN}$
Scheme 3	$3T_{ML} + 4T_{EM} + (7 + 4\ell)T_{EX} + 2T_{IN} + 2T_{MTP}$	$2T_{ML} + 3T_{EM} + T_{IN} + T_{MTP}$

k_1, k_2: bit length of m and $m\|info$ in [4], respectively.
ℓ: number of interactive rounds in zero-knowledge proof.
n: the number of ciphertexts generated by user.
u: the number of ciphertexts chosen by signer.

Table 8: Comparison-2 of Computational Efficiency

Schemes	Total computational cost	$\approx T_{ML}$
[36]	$24T_{ML} + 20T_{EX} + 6T_{IN} + 2T_{BP}$	5415.6
[4]	$8T_{ML} + (14 + k_1 + k_2)T_{EX} + 8T_{BP}$	$4064 + 240(k_1 + k_2)$
[48]	$10T_{ML} + 9T_{EM} + T_{IN}$	300
Scheme 1	$T_{ML} + 6T_{EM}$	175
Scheme 2	$(n + 2u + 5)T_{ML} + 4T_{EM}$ $+(2n + 2u + 4\ell + 5)T_{EX} + 3T_{IN}$	$481n + 482u + 960\ell$ $+1355.8$
Scheme 3	$5T_{ML} + 7T_{EM} + (7 + 4\ell)T_{EX} + 3T_{IN} + 3T_{MTP}$	$960\ell + 2009.8$

k_1, k_2: bit length of m and $m\|info$ in [4], respectively.
ℓ: number of interactive rounds in zero-knowledge proof.
n: the number of ciphertexts generated by user.
u: the number of ciphertexts chosen by signer.

and several high-cost operations including T_{IN} and T_{BP} during the issuing and verification phases. Scheme [48] is inexpensive in terms of computational cost because it is based on ECC. It cost only $300T_{ML}$ without pairings and MTP functions. Subsequently, we discuss the computational cost of our scheme. Our scheme requires $(2n + 2u + 5)T_{EX}$ in the signature issuing phase, where n is the number of ciphertexts generated by user and u is the number of ciphertexts randomly chosen by signer. Although our scheme requires more computation than other schemes, the signature produced by the our scheme can be used in current ECDSA systems.

Table 9 compares the signature size of our scheme and those of [4, 36, 48]. To ensure a fair comparison, we adopted a similar security level [17] at 128-bit for all schemes.

Table 9: Comparison of Signature Size

Schemes	Complexity	Signature size (bits)
[36]	$\lvert\mathbb{G}\rvert + 2\lvert q\rvert$	672
[4]	$2\lvert\mathbb{G}\rvert$	448
[48]	$3\lvert q\rvert + \lvert\mathbb{G}\rvert$	896
Ours Scheme 1	$2\lvert q\rvert$	448
Ours Scheme 2	$3\lvert q\rvert$	672
Ours Scheme 3	$3\lvert q\rvert$	672

Chapter 15

Performance

Chapter 16

Discussion

In this section we focus on our scheme 2. We first discuss about if a malicious user who knows what indices that signer will pick, then he/she can make a signature for different $info'$. Then, we discuss the probability that malicious can successfully guess what indices that the signer will pick before sending C_1 to the signer.

First, if the user has known what indices that signer will pick, he/she can perform the following steps to replace $info$ with $info'$ and make a signature $\sigma = (t, s, R')$ for $(m, info')$ that can pass the verification. Suppose the user has known that signer will choose indices $(i_1, \cdots, i_u)$, he/she then produces $F_j = g^{l'_j I'} z_j^N \bmod N^2$, for $j = i_1, \cdots, i_u$, where $I' = H(info')$ and l'_j and z_j are decided by the user. Signer follows the protocol and computes $C = (C_1 C_2^d F_{i_1} \cdots F_{i_u})^{k_1^{-1} \pmod q} \cdot r^N \pmod{N^2}$. Then, the signer returns C to the user.

Since,

$$
\begin{aligned}
C &= (C_1 C_2^d F_{i_1} \cdots F_{i_u})^{k_1^{-1} \pmod q} \cdot r^N \pmod{N^2} \\
&= ((g^{H(m)} r_1^N)(g^t r_2^N)^d (g^{l'_{i_1} I'} z_{i_1}{}^N) \cdots (g^{l'_{i_u} I'} z_{i_u}{}^N))^{k_1^{-1} \pmod q} \\
&\quad \cdot r^N \pmod{N^2} \\
&= g^{k_1^{-1}(H(m)+R'I'+td) \pmod q} \\
&\quad \cdot (r_1^{k_1^{-1}} r_2^{k_1^{-1} d} (z_{i_1} \cdots z_{i_u})^{k_1^{-1}} r)^N \pmod{N^2} \\
&= Enc(k_1^{-1}(H(m) + R'I' + td)).
\end{aligned}
$$

where $R' = \sum_{j=1}^{u} l'_j \bmod q$. User then computes $s = k_2^{-1} Dec(C, (p, k))$, , and outputs $\sigma = (t, s, R')$ for message m and $info'$ instead of $info$. The following we show that such

σ can pass the verification: Any verifier calculates $u = s^{-1}(H(m) + R'H(info'))$ mod q and $v = s^{-1}t$ mod q. With (u, v), the verifier can process as the verify in ECDSA and pass the verification. Hence, the above shows that a malicious user who knows what indices signer will choose can produce valid signature for $info'$.

To avoid that user can easily guess the indices chosen by signer, the setting of n and u is important. The probability of a malicious user guessing all indices is

$$\tfrac{u}{n} \times \tfrac{u-1}{n-1} \times \tfrac{u-2}{n-2} \times \cdots \times \tfrac{1}{n-u+1},$$

as shown in Table 10.

Table 10: The Probability that User Guesses All the Indices Chosen by Signer under the Setting of n and u

u	5	10	15	20	25
n	10	20	30	40	50
Pro.	$3 * 10^{-3}$	$5 * 10^{-6}$	$6 * 10^{-9}$	$7 * 10^{-12}$	$7 * 10^{-15}$

Chapter 17

Application of Proposed Schemes

17.1 Application on Bitcoin

In this section, we describe the idea of how to apply our scheme 2/scheme 3 to Bitcoin to achieve anonymity. The basic purpose of our application is the same as that proposed by Ladd [25] and Yi [52] which is to prevent Bitcoin provider from keeping track of purchaser's consumption. Essentially it is to hide the connection between identity of purchaser and the transaction broadcast to Bitcoin provider, but not hide all the Bitcoin address and transfer amount to every nodes (this type of digital currency is called anonymous currency, such as Zerocash [40]). In summary, this application is mainly aimed at malicious Bitcoin providers.

Howerver Ladd's application is based on inefficient mechanism called cut-and-choose and Yi's application can only transfer one bitcoin in one transaction. Our application can accept arbitrary transfer amount basing on our partially blind ECDSA since we use partially blind signature. We regard the Bitcoin provider $\mathcal{B}$ as the role of signer and the purchaser $\mathcal{P}$ as the user in our partially blind ECDSA. The overall flow of our application can be divided into two parts, as demonstrated in Fig. 5 and Fig. 6.

In detail, our application can be describe as four stages:

- **Payment** Bitcoin provider $\mathcal{B}$ and purchaser $\mathcal{P}$ start the process of buying Bitcoin. $\mathcal{P}$ first use fiat currency to pay for Bitcoin in credit card payment, cash or some other ways.
- **Negotiation** Bitcoin provider $\mathcal{B}$ and purchaser $\mathcal{P}$ negotiate to decide the times of signing n and the common information $info_{i,1\leq i\leq n}$ to use. For example, purchaser $\mathcal{P}$ buy 10

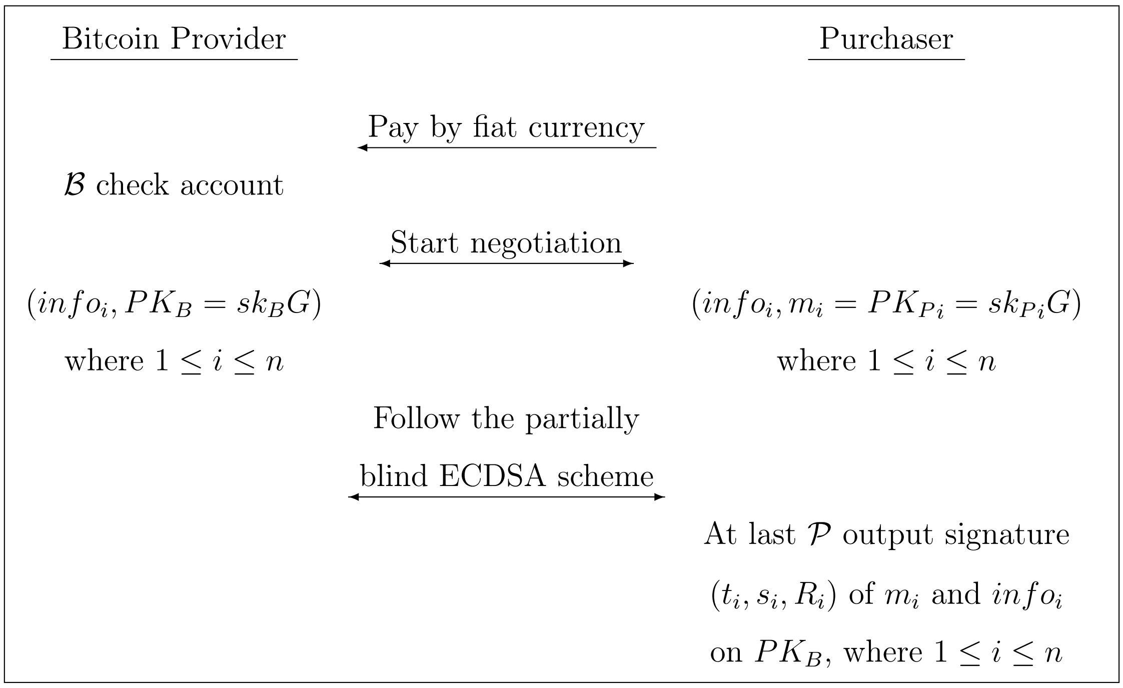
Bitcoin Provider
Purchaser
Pay by fiat currency
$\mathcal{B}$ check account
Start negotiation
$(info_i, PK_B = sk_B G)$
where $1 \leq i \leq n$
$(info_i, m_i = PK_{Pi} = sk_{Pi} G)$
where $1 \leq i \leq n$
Follow the partially
blind ECDSA scheme
At last $\mathcal{P}$ output signature
(t_i, s_i, R_i) of m_i and $info_i$
on PK_B, where $1 \leq i \leq n$

Figure 5: Application on Bitcoin

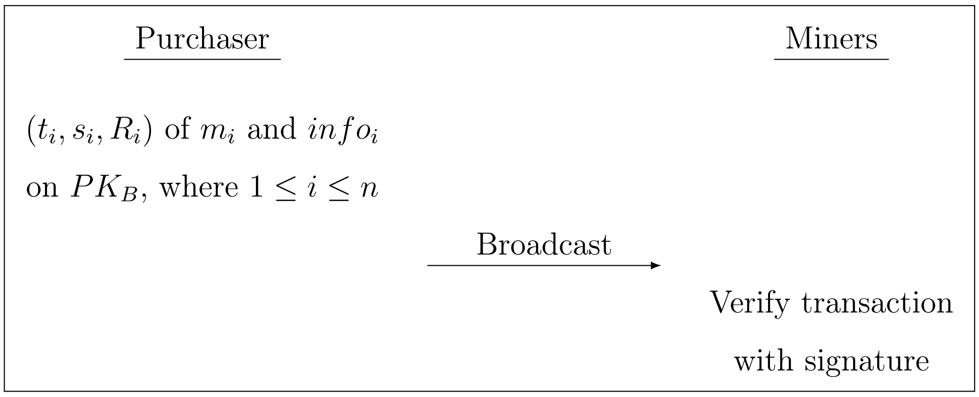
Purchaser
Miners
(t_i, s_i, R_i) of m_i and $info_i$
on PK_B, where $1 \leq i \leq n$
Broadcast
Verify transaction
with signature

Figure 6: Application on Bitcoin

bitcoins and want Bitcoin provider $\mathcal{B}$ to transfer to several public keys. Purchaser $\mathcal{P}$ can require Bitcoin provider $\mathcal{B}$ to sign one bitcoin ($info_i = 1$ where $1 \leq i \leq 10$) one signature or two bitcoins ($info_i = 2$ where $1 \leq i \leq 5$) one signature, even more only one transaction with 10 bitcoins ($info_1 = 10$), which is flexible. However, the less transaction $\mathcal{B}$ sign, the higher possibility for $\mathcal{B}$ to keep track of purchaser $\mathcal{P}$'s consumption. Since a transaction which transfers an exact total amount that purchaser $\mathcal{P}$ pay for, may be easy for $\mathcal{B}$ to figure out who hold this recipient public key. Here we recommended that Bitcoin provider $\mathcal{B}$ provide some fixed amount like fiat currency, say 1, 2, 10, 50 or larger amount. This may prevent lower efficiency and since $\mathcal{B}$ usually transfers a large amount of bitcoins to lots of purchasers every day, it is hard for $\mathcal{B}$ to distinguish between different transactions.

- **Partially Blind ECDSA** In this stage, Bitcoin provider $\mathcal{B}$ and purchaser $\mathcal{P}$ interact following proposed partially blind ECDSA scheme. Purchaser $\mathcal{P}$ generates public key PK_{Pi} where $1 \leq i \leq n$ under ECDSA, which is the message m to be signed and is basically $\mathcal{P}$'s Bitcoin address. They run partially blind ECDSA scheme n times and finally purchaser $\mathcal{P}$ obtain the signature (t_i, s_i, R_i) of public key PK_{Pi} and common information $info_i$ where $1 \leq i \leq n$. Each signature implies to transfer the amount in $info_i$ to purchaser $\mathcal{P}$.

- **Broadcast** Purchaser $\mathcal{P}$ can broadcast its transactions any time later in the Bitcoin system. The transaction includes public key PK_{Pi}, the common information $info_i$ and the corresponding signature (t_i, s_i, R_i) where $1 \leq i \leq n$.

- **Verify** Some miners or nodes may verify those signature. Since the verification of our scheme 2 and scheme 3 are slightly different from standard ECDSA in Bitcoin, this application may rely on new scripts which can add a few additional verifying operations on the standard ECDSA verification.

It is obviously that even Bitcoin provider $\mathcal{B}$ obtain the real-live identity of purchaser $\mathcal{P}$ when $\mathcal{P}$ pay fiat currency for Bitcoin, after $\mathcal{P}$ release the transaction he/she can only knows that $\mathcal{P}$ is one of the purchasers who has bought bitcoins. This is due to the blindness property of partially blind signature. Nevertheless, comparing to the application of blind signature proposed by Yi *et al.* [52], due to the common information $info$, our application can transfer an arbitrary amount or a fixed amount of bitcoins with one signature. It is much more flexible than application based on blind signature since latter can only transfer one bitcoin with one signature.

17.2 Further Discussion

In previous section we show idea of applying our ECDSA-compatible partially blind signature schemes on Bitcoin. However, there is still several details need to be discussed.

17.2.1 Fixed Denominations

First, in Yi's application all the transactions transfer one bitcoin to recipient, therefore all the transactions have the same transfer amount, which means that Bitcoin provider cannot link the recipient address in transaction and the identity of purchaser only by obtaining the transfer amount. In additional, the blind signature is used in Yi's application. The transfer amount does not participate in the blind signature, which ensures the unlinkability.

However in partially blind signature, the transfer amount is released. For example, the purchaser pay for Bitcoin provider for 7 bitcoin, and the result generated in the negotiation stage are signing only once where the common information (transfer amount) is 7 bitcoins. If there is no other transaction has the transfer amount equals 7 bitcoins, there is some probability that Bitcoin provider find the relevance between the identity of purchaser and the transaction. Therefore, with the release of the transfer amount, fixed denominations is important to reduce the probability that we mentioned.

We recommend using the same fixed denominations as fiat currency like 1 bitcoin, 5 bitcoins, 10 bitcoins, 50 bitcoins and so on. All transactions contains the transfer amount chosen from this fixed set. For example, a transaction with 7 bitcoins can be divided into one one with 5 bitcoins and two with 1 bitcoin, or seven with 1 bitcoin.

17.2.2 Different Address

The second issue is that using the same Bitcoin address to receive the bitcoin transfer may raise the probability that Bitcoin provider finding the identity of purchaser. For example, the purchaser release three transactions with 5 bitcoins, 1 bitcoins, 1 bitcoin, but all use the same Bitcoin address. It is not difficult for Bitcoin provider to find the identity of purchaser since provider can simply calculate the sum of all the transfer amounts, especially when the interval between purchaser broadcasting transactions is not long enough.

Thereby preparing different address to receive the bitcoins is necessary for purchaser to achieve anonymity. The best way is to generate a new Bitcoin address to receive the transfer bitcoins every time, but maintaining all the address is a big problem. One solution is that purchaser makes transactions at a later time to transfer bitcoins from these addresses to his/ her main wallet address and delete previous addresses, so these addresses are only temporary addresses. Another way is to generate several addresses for different transfer amount. Since we mention the fixed denominations, transfer amount can only be chosen from a fixed set. Therefore purchaser may use one address to receive the first denomination, say 1 bitcoin, one for the second, say 5 bitcoin, and so on. It is clear that different recipient addresses make the Bitcoin provider much more difficult to link the transaction to the identity of purchaser.

17.2.3 Amount of Daily Trades

As we mentioned before, if there is only one transaction being broadcast and being put on the Blockchain. It is easy for Bitcoin provider to know which purchaser broadcast this transaction. So Bitcoin provider with more trades every day may provide higher anonimity for purchaser since more trades means more transactions on Blockchain, which raise the difficulty to link the transaction to the identity of purchaser. Besides, purchaser can broadcast the transaction one day or two days later but not immediately, to wait for more transactions signed by provider putting on the Blockchain, if it is not urgent.

17.2.4 The Tradeoff

It is the tradeoff of anonimity and efficiency. If the purchaser and the provider decide to sign every transaction with 1 bitcoin, the anonimity is ensured while the efficiency is low; if the purchaser and the provider decide to sign transactions with least rounds, the anonimity is lower but the efficiency is much higher.

17.3 Applications beyond Bitcoin

Our partially blind ECDSA can be applied to any Bitcoin-liked cryptocurrency system. This kind of cryptocurrencies are usually based on UTXO-model [33] and often use ECDSA as signature scheme, just like Bitcoin. However, whether our application can be simply used in cryptocurrencies based on other model (like account-model in Ethereum [50]), is a question worth discussing in the future.

In addition to cryptocurrency, our application can be used in e-cash system. Since blind signature is born for e-cash and partially blind signature is born to make blind signature more flexible, it is clear that our partially blind ECDSA is compatible with e-cash system. Nevertheless, ECDSA has shorter public key size compared to other signature schemes which are not based on ECC, such as DSA. As mentioned before, our partially blind ECDSA makes it easy for the bank to control the database size, and can significantly reduce the storage for storing public key data.

Chapter 18

Conclusion

We proposed two variant-ECDSA schemes and three partially blind signature schemes. Compared with other schemes, our first scheme is the most practical in terms of computational cost. For the compatibility with Bitcoin system, we proposed two variants of ECDSA with their security proofs under generic group model. Subsequently, we proposed two ECDSA-based partially blind signatures based on variant-ECDSA-1 and 2. Our second and third schemes require more computation time but are compatible with Bitcoin system. Moreover, all our proposed schemes are unforgeable under adaptive chosen-message attacks and partially blind. In our future work, because our second and third scheme relies on a time-intensive zero-knowledge proof, we intend to develop an improved solution that reduces the computational cost.

Bibliography

[1] Masayuki Abe and Eiichiro Fujisaki. 1996. How to date blind signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 244–251.

[2] Masayuki Abe and Tatsuaki Okamoto. 2000. Provably secure partially blind signatures. In *Annual International Cryptology Conference*. Springer, 271–286.

[3] M An. 2004. Blind signatures with DSA/ECDSA? *Annual International Cryptology Conference* (2004), 271–286. http://lists.virus.org/cryptography-0404/msg00149.html

[4] Olivier Blazy, David Pointcheval, and Damien Vergnaud. 2012. Compact round-optimal partially-blind signatures. In *International Conference on Security and Cryptography for Networks*. Springer, 95–112.

[5] Daniel RL Brown. 2005. Generic groups, collision resistance, and ECDSA. *Designs, Codes and Cryptography* 35, 1 (2005), 119–152.

[6] Tony K Chan, Karyin Fung, Joseph K Liu, and Victor K Wei. 2004. Blind spontaneous anonymous group signatures for ad hoc groups. In *European Workshop on Security in Ad-hoc and Sensor Networks*. Springer, 82–94.

[7] David Chaum. 1983. Blind signatures for untraceable payments. In *Advances in cryptology*. Springer, 199–203.

[8] David Chaum, Amos Fiat, and Moni Naor. 1988. Untraceable electronic cash. In *Conference on the Theory and Application of Cryptography*. Springer, 319–327.

[9] Xiaofeng Chen, Fangguo Zhang, and Kwangjo Kim. 2003. ID-based multi-proxy signature and blind multisignature from bilinear pairings. *Proceedings of KIISC* 3 (2003), 11–19.

[10] Hung-Yu Chien, Jinn-Ke Jan, and Yuh-Min Tseng. 2001. RSA-based partially blind signature with low computation. In *Proceedings. Eighth International Conference on Parallel and Distributed Systems. ICPADS 2001*. IEEE, 385–389.

[11] Sherman SM Chow, Lucas CK Hui, Siu-Ming Yiu, and KP Chow. 2005. Forward-secure multisignature and blind signature schemes. *Appl. Math. Comput.* 168, 2 (2005), 895–908.

[12] Sherman SM Chow, Lucas CK Hui, Siu-Ming Yiu, and KP Chow. 2005. Two improved partially blind signature schemes from bilinear pairings. In *Australasian Conference on Information Security and Privacy*. Springer, 316–328.

[13] Whitfield Diffie and Martin Hellman. 1976. New directions in cryptography. *IEEE transactions on Information Theory* 22, 6 (1976), 644–654.

[14] Dang Nguyen Duc, Jung Hee Cheon, and Kwangjo Kim. 2003. A forward-secure blind signature scheme based on the strong RSA assumption. In *International Conference on Information and Communications Security*. Springer, 11–21.

[15] Amos Fiat and Adi Shamir. 1986. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*. Springer, 186–194.

[16] Yair Frankel, Yiannis Tsiounis, and Moti Yung. 1996. "Indirect discourse proofs": Achieving efficient fair off-line e-cash. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 286–300.

[17] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. 2004. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *International workshop on cryptographic hardware and embedded systems*. Springer, 119–132.

[18] Debiao He, Jianhua Chen, and Rui Zhang. 2011. An efficient identity-based blind signature scheme without bilinear pairings. *Computers & Electrical Engineering* 37, 4 (2011), 444–450.

[19] SK Hafizul Islam and GP Biswas. 2012. A pairing-free identity-based authenticated group key agreement protocol for imbalanced mobile networks. *Annals of télécommunications-annales des telecommunications* 67, 11-12 (2012), 547–558.

[20] SK Hafizul Islam and GP Biswas. 2013. Provably secure and pairing-free certificateless digital signature scheme using elliptic curve cryptography. *International Journal of Computer Mathematics* 90, 11 (2013), 2244–2258.

[21] Debasish Jena, Sanjay Kumar Jena, and Banshidhar Majhi. 2007. A novel untraceable blind signature based on elliptic curve discrete logarithm problem. (2007).

[22] Don Johnson, Alfred Menezes, and Scott Vanstone. 2001. The elliptic curve digital signature algorithm (ECDSA). *International journal of information security* 1, 1 (2001), 36–63.

[23] Jinho Kim, Kwangjo Kim, and Chulsoo Lee. 2001. An efficient and provably secure threshold blind signature. In *International Conference on Information Security and Cryptology*. Springer, 318–327.

[24] D W Kravitz. 1993. Washington, DC: U.S. Patent and Trademark Office. *U.S. Patent No. 5* 231 (1993), 668.

[25] Watsonn Ladd. 2012. Blind signatures for bitcoin transaction anonymity.

[26] Sunder Lal and Amit Kumar Awasthi. 2003. Proxy blind signature scheme. *Journal of Information Science and Engineering. Cryptology ePrint Archive, Report* 72 (2003).

[27] Lihua Liu and Zhengjun Cao. 2004. Universal Forgeability of a Forward-Secure Blind Signature Scheme Proposed by Duc et al. *IACR Cryptol. ePrint Arch.* 2004 (2004), 262.

[28] Anna Lysyanskaya. 2002. *Signature schemes and applications to cryptographic protocol design*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[29] Anna Lysyanskaya and Zulfikar Ramzan. 1998. Group blind digital signatures: A scalable solution to electronic cash. In *International Conference on Financial Cryptography*. Springer, 184–197.

[30] Greg Maitland and Colin Boyd. 2002. A provably secure restrictive partially blind signature scheme. In *International Workshop on Public Key Cryptography*. Springer, 99–114.

[31] Alfred J Menezes, Tatsuaki Okamoto, and Scott A Vanstone. 1993. Reducing elliptic curve logarithms to logarithms in a finite field. *iEEE Transactions on information Theory* 39, 5 (1993), 1639–1646.

[32] Victor S Miller. 1985. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*. Springer, 417–426.

[33] Satoshi Nakamoto. 2019. *Bitcoin: A peer-to-peer electronic cash system*. Technical Report. Manubot.

[34] Vassiliy Ilyich Nechaev. 1994. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes* 55, 2 (1994), 165–172.

[35] Morteza Nikooghadam and Ali Zakerolhosseini. 2009. An efficient blind signature scheme based on the elliptic curve discrete logarithm problem. *ISeCure-The ISC International Journal of Information Security* 1, 2 (2009), 125–131.

[36] Tatsuaki Okamoto. 2006. Efficient blind and partially blind signatures without random oracles. In *Theory of Cryptography Conference*. Springer, 80–99.

[37] David Pointcheval and Jacques Stern. 1996. Provably secure blind signature schemes. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 252–265.

[38] David Pointcheval and Jacques Stern. 1996. Provably secure blind signature schemes. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 252–265.

[39] David Pointcheval and Jacques Stern. 2000. Security arguments for digital signatures and blind signatures. *Journal of cryptology* 13, 3 (2000), 361–396.

[40] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*. IEEE, 459–474.

[41] Claus-Peter Schnorr. 1989. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*. Springer, 239–252.

[42] Adi Shamir. 1984. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*. Springer, 47–53.

[43] Victor Shoup. 1997. Lower bounds for discrete logarithms and related problems. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 256–266.

[44] Joseph H Silverman and Joe Suzuki. 1998. Elliptic Curve Discrete Logarithms and the Index Calculus. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 110–125.

[45] Markus Stadler, Jean-Marc Piveteau, and Jan Camenisch. 1995. Fair blind signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 209–219.

[46] Nedal Tahat, ES Ismail, and AK Alomari. 2018. PARTIALLY BLIND SIGNATURE SCHEME BASED ON CHAOTIC MAPS AND FACTORING PROBLEMS. *Italian Journal of Pure and Applied Mathematics* (2018), 165.

[47] Zuowen Tan, Zhuojun Liu, and Chunming Tang. 2002. Digital proxy blind signature schemes based on DLP and ECDLP. *MM Research Preprints* 21, 7 (2002), 212–217.

[48] Woei-Jiunn Tsaur, Jui-Heng Tsao, and Yuan-Heng Tsao. 2018. An Efficient and Secure ECC-based Partially Blind Signature Scheme with Multiple Banks Issuing E-cash Payment Applications. In *Proceedings of the International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE)*. The Steering Committee of The World Congress in Computer Science, Computer ⋯, 94–100.

[49] Duc Liem Vo, Fangguo Zhang, and Kwangjo Kim. 2003. A new threshold blind signature scheme from pairings. (2003).

[50] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

[51] Yan Xie, Fangguo Zhang, Xiaofeng Chen, and Kwangjo Kim. 2003. ID-based distributed 'Magic Ink' signature. In *Information and Communications Security, Fifth International Conference, ICICS*. 10–13.

[52] Xun Yi and Kwok-Yan Lam. 2019. A new blind ECDSA scheme for bitcoin transaction anonymity. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. 613–620.

[53] Fangguo Zhang and Kwangjo Kim. 2002. ID-based blind signature and ring signature from pairings. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 533–547.

[54] Fangguo Zhang, Reihaneh Safavi-Naini, and Chih-Yin Lin. 2003. New Proxy Signature, Proxy Blind Signature and Proxy Ring Signature Schemes from Bilinear Pairing. *IACR Cryptol. ePrint Arch.* 2003 (2003), 104.

[55] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. 2003. Efficient verifiably encrypted signature and partially blind signature from bilinear pairings. In *International Conference on Cryptology in India*. Springer, 191–204.

[56] 李鴻. 2004. 一種基於橢圓曲線的部分盲簽名方案. *宿州學院學報* 1 (2004), 89–91.