Effective Privacy Preservation and Fast Signature Verification in Bitcoin Transaction

Zhenhua Liu, Yuanyuan Li, Dong Yuan, and Yaohui Liu
(Corresponding author: Yuanyuan Li)

School of Mathematics and Statistics, Xidian University
Xi'an 710071, China
(Email: liyuanyuan4621@163.com)

(Received Jan. 29, 2018; Revised and Accepted July 20, 2018; First Online June 11, 2019)

Abstract

As a decentralized cryptocurrency, bitcoin has attracted considerable attentions. In the original bitcoin system, a transaction script is described as a plaintext and thus reveals the privacy. Furthermore, it takes at least one hour to confirm one transaction, which causes high latency. In view of these shortcomings, a new protocol is proposed to preserve the transaction privacy and speed the verification of transaction. Firstly, a modified homomorphic Paillier cryptosystem is used to preserve transaction privacy for our protocol. Moreover, we combine Zhu *et al.*'s interactive incontestable signature with Boneh *et al.*'s aggregate technique to present a new aggregate signature scheme, which can process a batch signature and greatly reduce the storage space. Then our aggregate signature scheme is applied to achieve fast verification for our protocol. Finally, our aggregate signature scheme is proved to be unforgeable in the random oracles, and performance analysis shows that our protocol has the property of privacy preserving and high efficiency.

Keywords: Aggregate Signature; Fast Verification; Paillier Cryptosystem; Privacy Preserving

1 Introduction

Bitcoin blockchain can be essentially known as a decentralized ledger system, which records the transactions among bitcoin addresses. The transaction is a central part of bitcoin blockchain, and the process of transaction is divided into generation, propagation in the network, proof of work, verification and record on the blockchain in the end. In Nakamoto's white paper, bitcoin is defined as a chain-type string of digital signature. The owner of bitcoin completes a transaction by making a digital signature of the previous transaction and the next owner's public key and attaching this signature to the transaction. Various signature algorithms, such as multi-signature [8], blind signature [9], proxy signature [10, 12, 14] and so on, can be utilized in the process of signature [22]. Generally, each transaction in bitcoin includes multiple inputs (one transaction can be sent by multiple individuals to a user) and outputs (one transaction can be transferred to multiple individuals). In addition, the input for each new transaction is the unspent output of a transaction (UTXO) and also needs to be signed by the private key corresponding to the previous output, and all nodes on the network verify the legitimacy of the new transaction via UTXO and the signature algorithm. However, there exist many problems and challenges with the development of bitcoin [13].

There is a serious problem that the bitcoin system only provides weak privacy protection. The unencrypted transaction amounts might leak unpredictably massive information during the daily trading. The disclosure of privacy is mainly due to the public amount of transactions, transaction metadata and distributed ledger, then the attacker can extract a lot of information about the identity of the user. Furthermore, the association between payment and receipt accounts allows the attacker to track the entire historical transaction path [20].

In order to enhance privacy preservation, various methods have been proposed to improve the anonymity of bitcoin. Bonneau *et al.* [4] proposed Mixcoin, which upsets the relationship between the payment account and the receive account, thereby increasing the anonymity of bitcoin system. Wijaya *et al.* [25] also presented an improved scheme to enhance the anonymity of bitcoin by lifting the relevance of the transaction. Bergen *et al.* [2] established an anonymous e-cash scheme CryptoNote by using ring signature and concealing address. Miers *et al.* [16] designed an extended bitcoin protocol Zerocoin based on zero-knowledge proof. Ben-Sasson *et al.* [21] proposed Zerocash based on the Zerocoin protocol by using the zk-SNARKs [1] to achieve an anonymous e-cash system and protect the transaction privacy, but Zerocash needs some strong trust assumptions, which deviates from the original trust intention. Ibrahim [11] constructed Securecoin to improve anonymous in bitcoin. Wang *et al.* [24] adopted Paillier Cryptosystem to hide the transaction amount and then improve the anonymity of the system, which is compatible with the bitcoin system.

Simultaneously, for legal digital currencies, there is another problem that the bitcoin system suppplies sluggish transaction speed. In the original bitcoin transaction [17], the miners need to spend 10 minutes to dig a block. It takes at least 1 hour to ensure the irreversible transaction. Therefore, it is very meaningful to study on improving the speed of transaction.

In order to solve this problem, some technologies such as expansion, lightning network and other programs are introduced. The expansion includes the isolation of witness and hard bifurcation block expansion [6, 23]. Lightning network is a side-chain technology, which reduces the burden of the main chain transaction significantly and expand more payment model [19]. In addition, Eyal *et al.* [7] introduced a new interest measurement method for quantifying the relationship between the security and efficiency of bitcoin-like blockchain protocols. Micali *et al.* [15] proposed an efficient public book agreement, a variant of Proof-of-Stake mechanism, that can solve the problem of bitcoin transaction delays, energy waste and bifurcation, which requires the number of attackers or the number of assets controlled by an attacker are less than 1/3 of the total amount. Zhu *et al.* [27] proposed an interactive incontestable signature scheme to achieve instant confirmation.

Although the above schemes can solve the corresponding problems about efficiency or privacy, they fail to balance efficiency and privacy issues in various bitcoin-like systems. Chang *et al.* [5] modified Ohta-Okamoto digital signature to achieve batch verification. Yuan *et al.* [26] applied aggregate signature technique to protect privacy and improve the performance of signature, but their scheme is not compatible with bitcoin system. There is still not a perfect scheme that can not only increase the speed of transaction confirmation but also protect users privacy about transaction amount until now. Therefore, it is a great deal to study a scheme that can both protect privacy and speed up signature verification.

Our Contributions. This paper mainly focuses on privacy preserving and fast confirmation in bitcoin system. We propose a new scheme that can not only protect users' privacy but also increase the speed of transaction confirmation. The main techniques and contributions are summarized as follows:

1) To preserve the transaction privacy, we modify Wang *et al.*'s scheme [24] to encrypt the apparent amounts of users, which doesn't undermine the consensus mechanism;
2) In the process of signature, we propose a new aggregate signature based on interactive incontestable signature and aggregate signature technology and prove its security, which can be applied to achieve fast verification for our protocol;
3) We combine the modified Wang *et al.*'s scheme [24] with new aggregate signature technique to propose a new protocol for bitcoin system, which achieves privacy preservation and fast verification of signature.

Organization. The rest of this paper is organized as follows: Section 2 introduces some preliminaries. Section 3 focuses on transaction privacy with Paillier cryptosystem. Section 4 mainly presents our aggregate signature and security proof. Section 5 proposes our new protocol, gives the comparisons between new protocol and related works in functionality, and analyzes security and efficiency. Finally, the conclusion is shown in Section 6.

2 Preliminaries

2.1 Paillier Cryptosystem

Paillier cryptosystem is a homomorphic encryption scheme, which is based on the composite residuosity class problem [18]. We review the encryption/decryption process simply as follows:

KeyGen. Set $n = pq$, compute $\lambda = \lambda(n) = \mathrm{lcm}(p-1, q-1)$, and select a base $g \in \mathbb{G}$ randomly satisfying $\gcd(L(g^{\lambda} \bmod n^2), n) = 1$, where p and q are large primes, $\mathbb{G}$ is a multiplicative group $\mathbb{G} = \{w | w \in \mathbb{Z}_{n^2}^*\}$ and $L(\theta) = \frac{\theta - 1}{n}$. The public key is $pk = (n, g)$ and secret key is $sk = \lambda$.

Encrypt. For a message $m < n$, choose a random number $r < n$, and compute the corresponding ciphertext

$$c = \mathrm{Enc}_{pk}(m) = g^m r^n \bmod n^2$$

Decrypt. Decrypt the ciphertext $c < n^2$ and obtain

$$m = \mathrm{Dec}_{sk}(m) = \frac{L(c^{\lambda} \bmod n^2)}{L(g^{\lambda} \bmod n^2)} \bmod n.$$

Paillier cryptosystem has an additive homomorphic property as:

$$\mathrm{Dec}(\mathrm{Enc}_{pk}(m_1) \cdot \mathrm{Enc}_{pk}(m_2) \bmod n^2) = (m_1 + m_2) \bmod n.$$

Paillier cryptosystem can perform efficiently both encryption and decryption and be convincingly secure under the chosen-plaintext attack in the standard model [18]. Due to the inherent additive homomorphic, the Paillier cryptosystem can be applied to various fields, such as the design of voting protocols, the threshold cryptosystem, etc. Furthermore, to preserve the transaction privacy in the bitcoin system, Wang *et al.* [24] utilized Paillier cryptosystem to hide the transaction amounts.

2.2 Bilinear Pairings

Let $\mathbb{G}_1, \mathbb{G}_T$ be two cyclic groups of prime order p_1. A bilinear map is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ with the following properties:

1) **Bilinearity**. For all $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_{p_1}$, then $e(u^a, v^b) = e(u, v)^{ab}$.

2) **Non-degeneration**. There exist $u, v \in \mathbb{G}_1$, $e(u, v) \neq 1$.

3) **Computability**. There is an algorithm to compute $e(u, v)$ for all $u, v \in \mathbb{G}_1$.

2.3 Complexity Assumptions

The security of our new aggregate signature will be reduced to the hardness of an extended Computational Bilinear Diffie-Hellman (eCBDH) [27]. We review the definition of the eCBDH problem briefly.

Definition 1. *Given $G, H, G^x, H^x, H^y \in \mathbb{G}_1$ for unknown $x, y \in \mathbb{Z}_{p_1}^*$, the eCBDH problem in $\mathbb{G}_1$ is to compute G^{xy}.*

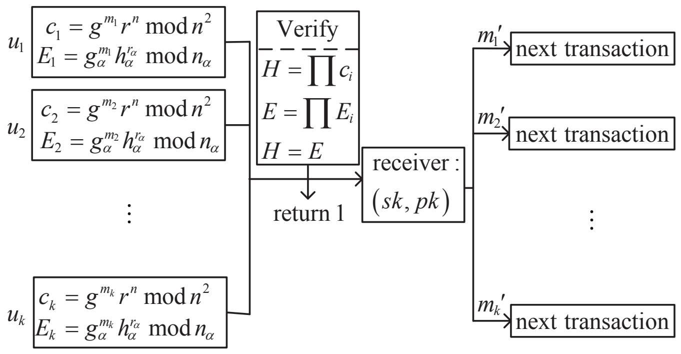
Definition 2. *We say that the (ϵ, t)-eCBDH assumption holds in a group $\mathbb{G}_1$ if no algorithm running in time at most t can solve the eCBDH problem in $\mathbb{G}_1$ with probability at least ϵ.*

3 Transaction Privacy with Paillier Cryptosystem

It is important for users to maintain their transaction privacy in bitcoin the system. Wang *et al.* [24] hide transaction amounts by using Paillier cryptosystem to preserve transaction privacy, where there is a sender who initiates many payments to multiple receivers. In this paper, we will consider the opposite situation that k senders separately send one payment to a receiver. To protect transaction privacy, we will modify Wang *et al.*'s scheme [24] to encrypt transaction amounts. The original amounts are replaced with the ciphertexts decrypted by only the receiver that owns the private key. Figure 1 shows the above process about transaction privacy with Paillier, where the definitions of letter symbols and parameters can be explained in Section 3.1.

3.1 Transaction Privacy Scheme with Paillier

KeyGen. For the specific receiver, select two large primes p, q, compute $n = pq$, $\lambda = \text{lcm}\,(p-1, q-1)$, and set $g \in \mathbb{Z}_{n^2}^*$. The public key is $pk = (n, g)$ and the secret key is $sk = \lambda$. Furthermore, generate public parameters (g_α, h_α) used in ***Verify*** phase and set $n_\alpha = n^2$, where $g_\alpha \in \mathbb{Z}_{n_\alpha}^*$ and h_α is an element of the group generated by g_α.



u_1
$c_1 = g^{m_1} r^n \bmod n^2$
$E_1 = g_\alpha^{m_1} h_\alpha^{r_\alpha} \bmod n_\alpha$
u_2
$c_2 = g^{m_2} r^n \bmod n^2$
$E_2 = g_\alpha^{m_2} h_\alpha^{r_\alpha} \bmod n_\alpha$
u_k
$c_k = g^{m_k} r^n \bmod n^2$
$E_k = g_\alpha^{m_k} h_\alpha^{r_\alpha} \bmod n_\alpha$
Verify
$H = \prod c_i$
$E = \prod E_i$
$H = E$
return 1
receiver : (sk, pk)
m_1'
m_2'
m_k'
next transaction
next transaction
next transaction

Figure 1: Transaction privacy with paillier cryptosystem

Encrypt. In our scheme, k senders initiate one payment separately to a receiver. To protect transaction privacy, Paillier cryptosystem is used to hide the transaction amounts m_i into ciphertexts c_i under the receiver's public key $pk = (n, g)$. Each sender u_i selects $r = h_\alpha$ and encrypts amounts as follows:

$$c_i = \text{Enc}_{pk}(m_i) = g^{m_i} r^n \bmod n^2$$

Meanwhile, each sender u_i makes a commitment $E_i = g_\alpha^{m_i} h_\alpha^{r_\alpha} \bmod n_\alpha$ for m_i, where $r_\alpha = n$.

Verify. In the ***Encrypt*** phase, the senders initiate bitcoins to the specific receiver under pk. Now the system will check the correctness of the transaction amounts in process as follows: whether the output-sum $\sum_{i=1}^{k} m_i'$ inside the cooperated cipher $\prod_{i=1}^{k} c_i$ is equal to input-sum $\sum_{i=1}^{k} m_i$ inside the cooperated commitment $\prod_{i=1}^{k} E_i$. The transaction will not be sent to the receiver unless the verification process are valid. The concrete details are divided into two steps:

Step 1. The system computes the cooperated ciphertexts and the cooperated commitments:

$$\begin{aligned} H &= \prod c_i = g^{\sum m_i'} r^n \bmod n^2 \\ E &= \prod E_i = g_\alpha^{\sum m_i} h_\alpha^{r_\alpha} \bmod n_\alpha \end{aligned}$$

Step 2. From ***KeyGen*** and ***Encrypt***, there are $n_\alpha = n^2$, $r = h_\alpha$, $r_\alpha = n$. The system checks whether H is equal to E. If yes, this shows that input-sum $\sum_{i=1}^{k} m_i$ is equal to output-sum $\sum_{i=1}^{k} m_i'$, then the system returns 1 for the next process.

Decrypt. As described in the previous process, the transaction will be sent to the specific receiver if the system returns 1. The receiver uses $sk = \lambda$ to decrypt $c_i\,(i = 1, \cdots, k)$:

$$m_i' = \text{Dec}\,(c_i) = \frac{L(c_i{}^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n.$$

A transaction is finished when the receiver assures the amounts are correct after decryption.

Broadcast. Finally, the transaction will be broadcast to P2P network.

3.2 Correctness of Decryption

To show the correctness of decryption, a few definitions and conclusions will be given firstly as follows [18]:

Definition 3. *For RSA modulus $n = pq$ where p and q are large primes, $g \in \mathbb{Z}^*_{n^2}$, and ε_g is defined as:*

$$(x, y) \to g^x \cdot y^n \bmod n^2$$

where $x \in \mathbb{Z}_n$, $y \in \mathbb{Z}^*_n$ and $c = g^x y^n \bmod n^2 \in \mathbb{Z}^*_{n^2}$.

Definition 4. *For ε_g and $c \in \mathbb{Z}^*_{n^2}$, the unique integer $x \in \mathbb{Z}_n$ is regarded as n-th residuosity class of w with respect to g for which there exists $y \in \mathbb{Z}^*_n$ such that $\varepsilon_g(x, y) = c$. The class of w is denoted by $[\![w]\!]_g$.*

Lemma 1. *For any $c_i \in \mathbb{Z}^*_{n^2}$, $L\left(c_i^\lambda \bmod n^2\right) = \lambda [\![c_i]\!]_{1+n}$.*

Next, we will give a brief correctness analysis of Paillier cryptosystem: Since $[\![g]\!]_{1+n} = [\![1+n]\!]_g^{-1}$ is revertible, which results in that $L(g^\lambda \bmod n^2)$ is revertible modulo n. Therefore, for any $g \in \mathbb{G}$ and $c_i \in \mathbb{Z}^*_{n^2}$, $i = 1, \cdots, k$, compute

$$\frac{L\left(c_i^\lambda \bmod n^2\right)}{L\left(g^\lambda \bmod n^2\right)} = \frac{\lambda [\![c_i]\!]_{1+n}}{\lambda [\![g]\!]_{1+n}} = \frac{[\![c_i]\!]_g [\![g]\!]_{1+n}}{[\![g]\!]_{1+n}} = m_i$$

the receiver decrypts c_i correctly to acquire the original amounts m_i.

3.3 Security Analysis

Our scheme can resist two major types of attacks: active attack and passive attack [24].

Active attack. Since each transaction in the bitcoin system is broadcast eventually to the P2P network and the attacker may destroy system deliberately (tampering attack) or forge transactions maliciously (Overlay attack) in different types. Our scheme in Section 3.1 can resist the active attacks as described above. We state security analysis:

Tampering attack. Our scheme uses the Paillier cryptosystem to encrypt transaction amounts, and only the receiver with private key can obtain the legal bitcoin. The receiver will not decrypt if the attacker tampers ciphertexts, which makes the transaction be discarded. Then our scheme can resist the information tampering attack.

Overlay attack. Overlay attack means that the attacker adds a forgery encrypted amount c_f to the original encrypted amount under the receiver's pk. The input-sum and the output-sum will be unequal if the attacker adds another amount to the transaction, which results in that the verification process will fail. Then our scheme can resist overlay attack.

Passive attack. The passive attacks usually contain information monitoring and traffic analysis. The attackers intend to extract secret information from the traders by monitoring communications between senders and a receiver or analyzing the traffic data of their transactions through internet, then the system may be unsecure due to some sensitive information in public. The Paillier cryptosystem is used to encrypt and protect the apparent amounts shown on the scripts, and what we can see is an unrecognized string which can only be readable to the receiver with private key. Then our scheme can resist passive attacks.

4 Aggregate Interactive Signature

Zhu *et al.* [27] addressed the problem of instant confirmation with incontestability in blockchain by adopting interactive signature. Aggregate signature technique proposed by Boneh *et al.* [3] can improve efficiency of signature verification. Next, we will combine interactive incontestable signature with aggregate signature technique to form a new aggregate signature scheme. The detail will be showed as follows.

4.1 The Proposed Signature Scheme

Setup. This algorithm firstly generates the bilinear groups $\mathbb{G}_1, \mathbb{G}_T$ of prime order p_1. Let g_1 be the generator of $\mathbb{G}_1$. This algorithm chooses a random element $h \in \mathbb{G}_1$ and outputs a master public key $mpk = (g_1, h)$. Meanwhile, there is a hash function H: $\{0,1\}^* \to \mathbb{G}_1$.

KeyGen. Each sender $u_i \, (i = 1, \cdots, k)$ runs this algorithm to generate private key $sk_i = x_i \in_R \mathbb{Z}^*_{p_1}$ and public key $pk_i = h^{x_i}$. The specific receiver runs this algorithm to generate private key $sk' = d \in_R \mathbb{Z}^*_{p_1}$ and public key $pk' = g_1^d$.

Sign. Each block contains multiple transactions in the bitcoin blockchain, and each transaction includes multiple inputs and outputs. Each sender $u_i \, (i = 1, \cdots, k)$ and the specific receiver interact separately as follows to generate a signature:

Step 1. The receiver selects $a \in_R \mathbb{Z}^*_{p_1}$, calculates $M_i = H(T_i)^a \in \mathbb{G}_1$ of transaction T_i, and transmits M_i to the corresponding sender, where transaction amounts in T_i are apparent. Meanwhile, the receiver outputs a witness $W = (wit_1, wit_2)$, where $wit_1 = g_1^a$, $wit_2 = (h^a)^{sk'} = (h^a)^d$;

Step 2. Each sender picks a number $r_i \in_R \mathbb{Z}^*_{p_1}$, computes

$$\sigma'_i = \left(g_1^{x_i H(ID_i)} \cdot M_i\right)^{r_i}$$

and returns σ'_i to the receiver, where ID_i is the identifier of transaction T_i;

Step 3. The receiver calculates

$$\sigma_i'' = (\sigma_i')^d = \left(g_1^{x_i H(ID_i)} \cdot M_i\right)^{r_i d}$$

with his private key $sk' = d$ and delivers σ_i'' to the corresponding sender;

Step 4. Finally, each sender computes

$$\sigma_i = (\sigma_i'')^{r_i^{-1}} = \left(g_1^{x_i H(ID_i)} \cdot M_i\right)^d$$

separately of T_i.

Aggregate. The signature σ_i of T_i is published in a block. The system selects the master node to calculate an aggregate signature $\sigma = \prod_{i=1}^{k} \sigma_i$.

Verify. The aggregate signature σ are given. In order to verify the aggregate signature σ, the verifier checks

$$e(\sigma, h) = \prod_{i=1}^{k} e\left((pk')^{H(ID_i)}, pk_i\right) \cdot e(H(T_i), wit_2)$$

then accepts σ if the above equation holds.

4.2 Correctness of Aggregate Signature

The correctness of an aggregate signature σ is proved as follows:

$$\begin{aligned}
&\prod_{i=1}^{k} e\left((pk')^{H(ID_1)}, pk_i\right) \cdot e(H(T_i), wit_2) \\
=& \prod_{i=1}^{k} e\left((g_1^d)^{H(ID_i)}, h^{x_i}\right) \cdot e\left(H(T_i), h^{ad}\right) \\
=& \prod_{i=1}^{k} e\left(\left(g_1^{x_i H(ID_i)} \cdot M_i\right)^d, h\right) \\
=& e\left(\prod_{i=1}^{k} \sigma_i, h\right)
\end{aligned}$$

4.3 Existential Unforgeability

Boneh *et al.* [3] set up the security model about aggregate signature at the first time. Aggregate signature means that k users separately signs k distinct messages, then k signatures are aggregated into a single signature. This single signature will convince the verifier that k users did indeed sign k messages. Therefore, the security model about aggregate signature of Boneh *et al.* [3] is applied to our aggregate interactive signature.

In our scheme, the adversary $\mathcal{A}$'s advantage, AdvAggSig$_{\mathcal{A}}$, is defined to be the probability of success in the following game [3]:

Setup. The adversary $\mathcal{A}$ is provided with a public key pk_1 generated at random.

Queries. Proceeding adaptively, $\mathcal{A}$ requests signatures with pk_1 on the messages of his choices.

Response. Finally, $\mathcal{A}$ outputs $k-1$ additional public keys $pk_2, pk_3, \cdots, pk_k$, here, k is at most N, a game parameter. These keys, along with the initial key pk_1, will be included in $\mathcal{A}$'s forged aggregate. $\mathcal{A}$ outputs transaction messages $T_1, T_2, \cdots T_k$ and an aggregate signature σ by k users.

The adversary $\mathcal{A}$ wins if the aggregate signature σ is a valid on messages $T_1, T_2, \cdots T_k$ under keys $pk_2, pk_3, \cdots, pk_k$, and σ is nontrivial, i.e., $\mathcal{A}$ did not request a signature on T_1 under pk_1.

Definition 5. *An aggregate forger $\mathcal{A}$ $(t, \varepsilon, q_h, q_s)$-breaks an N-user aggregate signature scheme in the aggregate chosen-key model if: $\mathcal{A}$ has advantage at least ε in the above game, runs in time at most t, and makes at most q_h queries to hash function, q_s queries to signing oracle. An aggregate signature scheme is $(t, \varepsilon, q_h, q_s)$-secure against existential forgery in the aggregate chosen-key model if no forger $(t, \varepsilon, q_h, q_s)$-breaks it.*

4.4 Security Proof

Theorem 1. *Our aggregate signature scheme is $(t, \varepsilon, q_h, q_s)$-secure against existential forgery in the aggregate chosen-key model, if no algorithm running in time at most t' can solve the eCBDH problem in $\mathbb{G}_1$ with probability at least ε', where*

$$t + c_{\mathbb{G}_1}(q_h + 2q_s + N + 5) + N + 1 \leq t'$$

$$\varepsilon' = \left(1 - \frac{1}{q_s + N}\right)^{q_s + N - 1} \cdot \frac{1}{q_s + N} \cdot \varepsilon$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that outputs a forged aggregate signature for new aggregate signature scheme with a non-negligible advantage ε. We can use the algorithm $\mathcal{A}$ to construct a PPT algorithm $\mathcal{C}$ that can break the eCBDH problem. □

Setup. The algorithm $\mathcal{C}$ is given $G, H, G^x, H^x, H^y \in \mathbb{G}_1$, where $x, y \in \mathbb{Z}_{p_1}^*$, and his goal is to calculate $G^{xy} \in \mathbb{G}_1$. $\mathcal{C}$ runs the aggregate interactive signature scheme to generate $mpk = (G, H) = (g_1, h)$ and starts as follows:

Algorithm $\mathcal{C}$ maintains a list of seven tuples $(T_i, pk_i, pk', W, \lambda_i, c, a)$. We refer to this list as the K-list, and the list is initially empty. When $\mathcal{A}$ performs the queries of the transaction T_i under the public keys, algorithm $\mathcal{C}$ responds as follows:

Step 1. If the query T_i already appears on the K-list in some tuple $(T_i, pk_i, pk', W, \lambda_i, c, a)$, then algorithm $\mathcal{C}$ responds with pk_i, pk', W.

Step 2. Otherwise, $\mathcal{C}$ generates a random coin $c \in \{0, 1\}$ so that $\Pr[c = 0] = 1/(q_s + N)$.

Step 3. Algorithm $\mathcal{C}$ picks $\lambda_i, a \in \mathbb{Z}_{p_1}^*$. If $c = 0$ holds, assuming that $sk_i = y\lambda_i$, $sk' = x/\lambda_i$, $\mathcal{C}$ computes

$$pk_i = (H^y)^{\lambda_i},\ pk' = G^{x/\lambda_i}$$

$$wit_1 = G^a = g_1^a, wit_2 = (H^x)^a = h^{ad}$$

$$W = (wit_1, wit_2)$$

If $c = 1$ holds, assuming that $sk_i = \alpha_i \in \mathbb{Z}_{p_1}^*$, $\mathcal{C}$ computes $pk_i = H = G^a = g_1^{\alpha_i}$, $pk' = G^{x/\lambda_i}$.

Step 4. Algorithm $\mathcal{C}$ adds the tuple $(T_i, pk_i, pk', W, \lambda_i, c, a)$ to the K-list and responds to $\mathcal{A}$ as pk_i, pk', W.

Note that, either way, pk_i, pk', W are uniform in $\mathbb{G}_1$ and are independent of $\mathcal{A}$'s current view as required.

Hash queries. Algorithm $\mathcal{A}$ can query the random oracle H to q_h times. When $\mathcal{A}$ queries $H(T_i)$ of T_i, algorithm $\mathcal{C}$ responds as $H(T_i) = G^{h(T_i)}$, where there is a map: $\{0,1\}^* \to \mathbb{Z}_{p_1}^*$ and $h(T_i) \in \mathbb{Z}_{p_1}^*$.

Signature queries. Algorithm $\mathcal{A}$ requests a signature on some transaction message T_i under the challenge key pk_1. Algorithm $\mathcal{C}$ responds to the query as follows:

1) Algorithm $\mathcal{C}$ runs the above algorithm to respond the hash queries on T_i, obtaining the corresponding tuple $(T_i, pk_i, pk', W, \lambda_i, c, a)$. If $c = 0$, then $\mathcal{C}$ reports failure and terminates.

2) Otherwise, algorithm $\mathcal{C}$ computes

$$\sigma_i = G^{xH(ID_i)+h(T_i)xa}$$

and returns σ_i to $\mathcal{A}$.

Output. Finally, $\mathcal{A}$ halts. It either concedes failure, in which case so does $\mathcal{A}$, or it returns a value k $(k \le N)$, $k-1$ public keys $pk_2, pk_3, \cdots, pk_k$, k transaction messages $T_1, \cdots, T_k$, and a forged aggregate signature σ. $\mathcal{A}$ must not have requested a signature on T_1. Algorithm $\mathcal{C}$ runs the above algorithms at each T_i and obtains k corresponding tuples $(T_i, pk_i, pk', W, \lambda_i, c, a)$, where $i = 1, 2, \cdots, k$.

Algorithm $\mathcal{C}$ now proceeds only if $c = 0$ when $i = 1$, and, for $2 \le i \le k$, $c = 1$; otherwise $\mathcal{C}$ declares failure and halts. The aggregate signature σ must satisfy the follow equation:

$$e(\sigma, h) = \prod_{i=1}^{k} e\left((pk')^{H(ID_i)}, pk_i\right) \cdot e(H(T_i), wit_2).$$

For each $i > 1$, $\mathcal{C}$ sets $\sigma_i = G^{xH(ID)_i + h(T_i)xa}$, then

$$\begin{aligned} & e(\sigma_i, h) \\ &= e\left(G^{xH(ID_i)+h(T_i)xa}, h\right) \\ &= e\left(G^{xH(ID_i)}, h\right) \cdot e\left(G^{h(T_i)}, h^{ax}\right) \\ &= e\left((pk')^{H(ID_i)}, h\right) \cdot e\left(G^{h(T_i)}, wit_2\right) \end{aligned}$$

So σ_i is a valid signature on T_i. Now $\mathcal{C}$ constructs a value $\sigma_1 : \sigma_1 \leftarrow \sigma \cdot \left(\prod_{i=2}^{k} \sigma_i\right)^{-1}$. Then

$$\begin{aligned} & e(\sigma_1, h) \\ &= e(\sigma, h) \cdot \prod_{i=2}^{k} e(\sigma_i, h)^{-1} \\ &= e\left((pk')^{H(ID_1)}, pk_1\right) \cdot e(H(T_1), wit_2) \end{aligned}$$

If the above equation holds, σ_1 is a valid signature on T_1. Then $\mathcal{C}$ can calculate and output his target value

$$G^{xy} = \left(\sigma_1 \big/ (G^x)^{ah(T_1)}\right)^{1/H(ID_1)}$$

The above steps complete the description of algorithm $\mathcal{C}$. It remains to show that $\mathcal{C}$ solves the eCBDH problem in $\mathbb{G}_1$ with probability at least ε'. To do so, we analyze the three events needed for $\mathcal{C}$ to succeed:

E_1: $\mathcal{C}$ does not abort as a result of any of $\mathcal{C}$'s signature queries.

E_2: $\mathcal{A}$ generates a valid, nontrivial aggregate signature forgery $(k, pk_1, \cdots, pk_k, T_1, \cdots, T_k)$.

E_3: Event E_2 occurs, and, in addition, $c = 0$ when $i = 1$, and, for $2 \le i \le k$.

$\mathcal{C}$ succeeds if all these events happen. The probability $\Pr[E_1 \wedge E_3]$ decomposes as

$$\Pr[E_1 \wedge E_3] = \Pr[E_1] \cdot \Pr[E_2|E_1] \cdot \Pr[E_3|E_1 \wedge E_2]$$

The following claims give a lower bound for each of these terms.

Claim 1. The probability that algorithm $\mathcal{C}$ does not abort as a result of $\mathcal{A}$'s aggregate signature queries are at least $(1 - 1/(q_s + N))^{q_s}$. Hence,

$$\Pr[E_1] \ge (1 - 1/(q_s + N))^{q_s}$$

Claim 2. If algorithm $\mathcal{C}$ does not abort as results of $\mathcal{A}$'s queries, then algorithm $\mathcal{A}$'s view is identical to its view in the real attack. Hence,

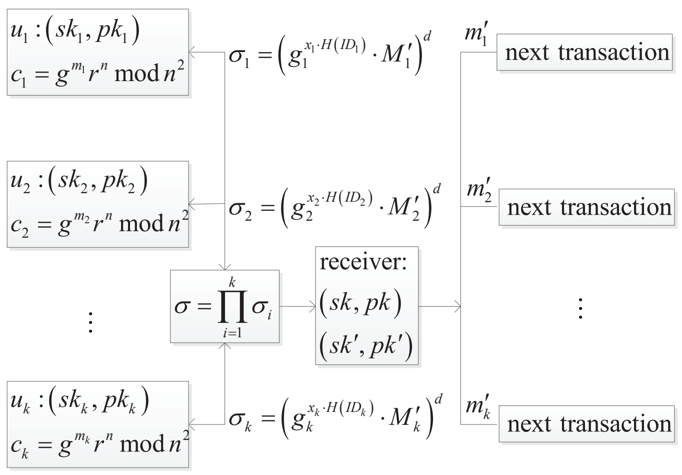
$$\Pr[E_2|E_1] \ge \varepsilon$$

Claim 3. The probability that algorithm $\mathcal{C}$ does not abort after $\mathcal{A}$ outputs a valid and nontrivial forgery is at least $(1 - 1/q_s + N)^{N-1} \cdot 1/(q_s + N)$. Hence,

$$\Pr[E_3|E_1 \wedge E_2] \ge (1 - 1/q_s + N)^{N-1} \cdot 1/q_s + N$$

Algorithm $\mathcal{C}$ produces the correct answer with probability at least

$$\varepsilon' = \left(1 - \frac{1}{q_s + N}\right)^{q_s + N - 1} \cdot \frac{1}{q_s + N} \cdot \varepsilon$$



$u_1 : (sk_1, pk_1)$
$c_1 = g^{m_1} r^n \bmod n^2$
$u_2 : (sk_2, pk_2)$
$c_2 = g^{m_2} r^n \bmod n^2$
$u_k : (sk_k, pk_k)$
$c_k = g^{m_k} r^n \bmod n^2$
$\sigma_1 = \left(g_1^{x_1 \cdot H(ID_1)} \cdot M_1'\right)^d$
$\sigma_2 = \left(g_2^{x_2 \cdot H(ID_2)} \cdot M_2'\right)^d$
$\sigma_k = \left(g_k^{x_k \cdot H(ID_k)} \cdot M_k'\right)^d$
$\sigma = \prod_{i=1}^{k} \sigma_i$
receiver:
(sk, pk)
(sk', pk')
m_1'
m_2'
m_k'
next transaction
next transaction
next transaction

Figure 2: Transaction privacy with paillier cryptosystem

Algorithm $\mathcal{C}$'s running time is the same as $\mathcal{A}$'s running time plus the time that is takes to respond to public-key queries hash queries and signature queries, and the time to transform $\mathcal{A}$'s final forgery into the eCBDH solution. Each hash query and signature query require an exponentiation in $\mathbb{G}_1$. The output phase requires at most N additional hash computations, three inversions, two exponentiations, and $N+1$ multiplications. We assume that exponentiation and inversion in $\mathbb{G}_1$ take time $c_{\mathbb{G}_1}$. Hence, the total running time is at most $t + c_{\mathbb{G}_1}(q_H + 2q_S + N + 5) + N + 1 \le t'$ as required. The above process complete the proof of **Theorem** 1.

5 New Protocol with Privacy Preserving and Fast Verification

5.1 The Proposed Protocol

A new protocol that can achieve privacy preservation and fast verification is proposed in this Section. Transaction privacy with Paillier in Section 3 and new aggregate signature in Section 4 are used to reach our goals. The sender signs a payment after transaction amounts are encrypted and verified by the system. Here, there exist k senders initiate k payments to a receiver. Figure 2 shows the structure of our protocol, where (sk_i, pk_i) is the signature key pair of each sender, (sk, pk) and (sk', pk') are separately the encryption key pair and the signature key pair of the receiver.

The concrete process of signature is shown as follows.

KeyGen. The system generates the bilinear group $\mathbb{G}_1, \mathbb{G}_T$ and chooses a random number $h \in \mathbb{G}_1$. Let g_1 be the generator of $\mathbb{G}_1$. Each sender u_i $(i = 1, \cdots, k)$ runs this algorithm to generate private key $sk_i = x_i \in_R \mathbb{Z}_{p_1}^*$ and public key $pk_i = h^{x_i}$. The specific receiver runs this algorithm to generate private key $sk' = d \in_R \mathbb{Z}_{p_1}^*$ and public key $pk' = g_1^d$.

Sign. Each block contains multiple transactions in the bitcoin blockchain. Then each sender and receiver interact separately as follows to generate a signature for a payment. Here, we denote the transaction as T_i', where each transaction amount m_i has been encrypted as c_i by Paillier cryptosystem.

Step 1. The receiver calculates the $M_i' = H(T_i')^a$ of transaction T_i' with $a \in_R \mathbb{Z}_{p_1}^*$, and then transmits M_i' to each sender. Meanwhile, the receiver outputs a witness $W = (wit_1, wit_2)$, where

$$wit_1 = g_1^a, wit_2 = (h^a)^{sk'} = (h^a)^d$$

Step 2. Each sender selects a number $r_i \in_R \mathbb{Z}_{p_1}^*$ and delivers

$$\sigma_i' = \left(g_1^{x_i H(ID_i)} \cdot M_i'\right)^{r_i}$$

to the receiver, where ID_i is the identifier of T_i';

Step 3. The receiver calculates

$$\sigma_i'' = (\sigma_i')^d = \left(g_1^{x_i H(ID_i)} \cdot M_i'\right)^{r_i d}$$

and returns σ_i'' to the corresponding sender;

Step 4. Finally, each sender computes

$$\sigma_i = (\sigma_i'')^{r_i^{-1}} = \left(g_1^{x_i H(ID_i)} \cdot M_i'\right)^d$$

of T_i'.

Aggregate. The system selects the master node to calculate aggregate signature $\sigma = \prod_{i=1}^{k} \sigma_i$.

Verify. An aggregate signature $\sigma \in \mathbb{G}_1$ is given. In order to verify the signature σ, the verifier computes

$$e(\sigma, h) = \prod_{i=1}^{k} e\left((pk')^{H(ID_i)}, pk_i\right) \cdot e\left(H(T_i'), wit_2\right)$$

then accepts σ if this equation holds. The transaction will be sent to the receiver if the above steps are performed correctly, where the transfer form of input is $c_i = g^{m_i} r^n \bmod n^2$.

Decrypt. When the receiver gets ciphertexts, he can use private key sk' to decrypt:

$$m_i = \mathrm{Dec}(c_i) = \frac{L(c_i{}^{\lambda} \bmod n^2)}{L(g^{\lambda} \bmod n^2)} \bmod n$$

A transaction is finished when the receiver assures the amounts are correct after decryption.

Broadcast. Finally, the transaction will be broadcast to P2P network.

5.2 Security Analysis

In Sections 3 and 4, the security of transaction privacy and aggregate signature have been revealed separately. Then security analysis about our protocol are showed as follows:

1) In the process of encrypting transaction amounts, our scheme can resist active attack (such as tampering attack and overlay attack) and passive attack (such as information monitoring) due to the use of Paillier cryptosystem. The details refer to Section 3.3.

2) In the process of signature, the new aggregate signature is proved to be unforgeable under eCBDH assumption, which ensures that the attacker cannot tamper with the aggregate signature and then any single signature of all senders is unforgeable. The details refer to Section 4.3.

Table 1: Functionality comparisons

	Transaction privacy	Fast verification	Compatible with bitcoin
[24]	√	×	√
[26]	√	√	×
[27]	×	×	√
Ours	√	√	√

Table 2: Complexity Analysis of transaction privacy

Algorithm	**Computation Costs**
KeyGen	$2\tau_m$
Encrypt	$k\tau_M + (k+1)\tau_E$
Verify	$(2k-2)\tau_m + k\tau_M + (k+1)\tau_E$
Decrypt	$(k+1)\tau_m + (k+1)\tau_E$

5.3 Functionality Comparisons

Features comparisons between our protocol and some recent schemes are listed in Table 1. As can be seen from the comparisons with some related works, our protocol can achieve more functionality, where √ means that the corresponding scheme achieves this functionality, and × means that the corresponding scheme doesn't achieve or mention this functionality.

5.4 Efficiency Analysis

Our protocol is split into two processes, the transaction amounts are fist encrypted, and then the signature is generated about the transaction. We set $\tau_m, \tau_M, \tau_E, \tau_B$ to represent separately multiplication operating time, modular multiplication time, modular exponentiation time and pairing operation time. Therefore, effective privacy preservation and fast signature verification for our protocol are showed as follows:

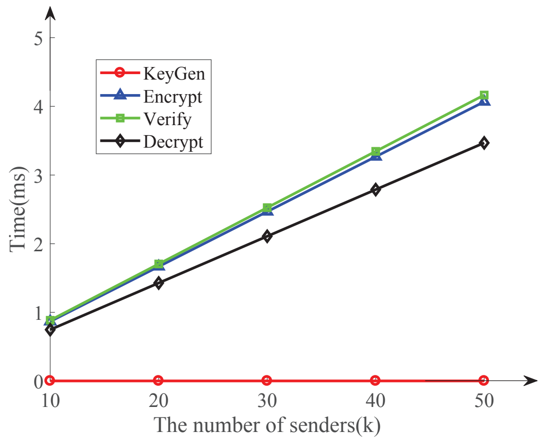


Figure 3: Computation time of transaction privacy

1) In the process of hiding transaction amounts, the complexity of preserving privacy is showed in Table 2. As for the actual performance analysis, we select the different number of senders separately to construct the experimental simulations, which proves that our protocol can achieve effective privacy preservation.

   As is described in Figure 3, we select the different number of senders: 10 senders, 20 senders, 30 senders, 40 senders, 50 senders. It is easy to see that the computation costs are all in milliseconds regardless of the number of senders. The computation time is so small that our protocol can preserve privacy effectively.

2) In the process of signature, the computational complexity comparisons between our protocol and Zhu *et al.*'s [27] scheme are given in Table 3.

As for the actual performance analysis, we assume that there are 10 senders in one bitcoin transaction. Then the experimental results is simulated in Figure 4. Apparently, we can see that our protocol has less time costs in the *verify* phase than Zhu *et al.*'s [27] scheme, which indicates that our protocol can achieve fast signature verification.

Above all, the proposed protocol can not only preserve privacy effectively but also fast confirm the signature in bitcoin transaction.

6 Conclusions

In this paper, we have presented a new system protocol in bitcoin that not only protects the privacy of users but also enhances the efficiency of verification for signature. The new protocol keeps the size of signatures constant with a single signature regardless of the number of inputs and outputs, compresses the signatures of any number of users into a single signature and greatly reduces the

Table 3: Complexity comparisons of signature

Scheme	**KeyGen**	**Sign**	**Verify**
[27]	$(k+4)\,\tau_E$	$(2k-1)\,\tau_m + 5k\tau_E$	$k\,(\tau_m + \tau_E) + 3k\tau_B$
Ours	$(k+4)\,\tau_E$	$(2k-1)\,\tau_m + 5k\tau_E$	$(2k-1)\,\tau_m + k\tau_E + (2k+1)\,\tau_B$

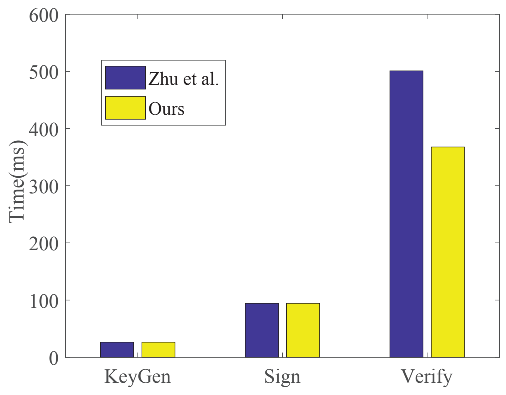


Figure 4: Comparisons of computation time in signature

storage space of signature. Furthermore, our new protocol reduces the requirements of the network bandwidth transmission, simplifies the process of verification and decreases the workload of signature verification.

Acknowledgments

This paper is supported by the National Key R&D Program of China under Grant No.2017YFB0802000, the National Natural Science Foundation of China under Grants No.61472470 and 61572390, and the Scientific Research Plan Project of Education Department of Shaanxi Province under Grant No.17JK0362.

References

[1] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *The 23rd USENIX Conference on Security Symposium (SECURITY'14)*, pp. 781–796, 2014.

[2] T. Bergan, O. Anderson, J. Devietti, L. Ceze, and D. Grossman, "Cryptonote v 2.0," in *Trend Micro 45*, pp. 1–16, 2013.

[3] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'03)*, pp. 416–432, 2003.

[4] J. Bonneau, A. Narayanan, A. Miller, J. Clark, and J.A. Kroll, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *International Conference on Financial Cryptography and Data Security (FC'14)*, pp. 486–504, 2014.

[5] T. Y. Chang, M. S. Hwang, W. P. Yang, and K. C. Tsou, "A modified ohta-okamoto digital signature for batch verification and its multi-signature version," *International Journal of Engineering and Industries*, vol. 3, no. 3, pp. 75–83, 2012.

[6] K. Croman, C. Decker, I. Eyal, A. E. Gencer, and A.Juels, "On scaling decentralized blockchains," in *Financial Cryptography and Data Security (FC'16)*, pp. 106–125, 2016.

[7] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *The 13th USENIX Conference on Networked Systems Design and Implementation (NSDI'15)*, pp. 45–59, 2015.

[8] M. S. Hwang and C. C. Lee, "Research issues and challenges for multiple digital signatures," *International Journal of Network Security*, vol. 1, no. 1, pp. 1–7, 2005.

[9] M. S. Hwang, C. C. Lee, and Y. C. Lal, "Traceability on low-computation partially blind signatures for electronic cash," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. 85, no. 5, pp. 1181–1182, 2002.

[10] M. S. Hwang, S. F. Tzeng, and C. S. Tsai, "Generalization of proxy signature based on elliptic curves," *Computer Standards and Interfaces*, vol. 26, no. 2, pp. 73–84, 2004.

[11] M. H. Ibrahim, "Securecoin: A robust secure and efficient protocol for anonymous bitcoin ecosystem," *International Journal of Network Security*, vol. 19, no. 2, pp. 295–312, 2017.

[12] L. H. Li, S. F. Tzeng, M. S. Hwang, "Generalization of proxy signature based on discrete logarithms", *Computers & Security*, vol. 22, no. 3, pp. 245–255, 2003.

[13] I. C. Lin and T. C. Liao, "A survey of blockchain security issues and challenges," *International Journal of Network Security*, vol. 19, no. 5, pp. 653–659, 2017.

[14] E. J. L. Lu, M. S. Hwang, and C. J. Huang, "A new proxy signature scheme with revocation", *Applied Mathematics and Computation*, vol. 161, no. 3, PP. 799-806, Feb. 2005.

[15] S. Micali, "Algorand: The efficient and democratic ledger," *Cryptography and Security*, May 2016. (`https://arxiv.org/abs/1607.01341`)

[16] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *IEEE Symposium on Security and Privacy (SP'13)*, pp. 397–411, 2013.

[17] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, Japan: Consulted, 2008. (`https://bitcoin.org/en/bitcoin-paper`)

[18] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *The 17th International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT'99)*, pp. 223–238, 1999.

[19] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," in *International Conference on Financial Cryptography and Data Security (FC'15)*, pp. 486–504, 2015.

[20] B. Qin, L. C. H. Chen, Q. H. Wu, Y. F. Zhang, L. Zhong, and H. B. Zheng, "Bitcoin and digital fiat currency," *Journal of Cryptologic Research*, vol. 4, no. 2, pp. 176–186, 2017.

[21] E. B. Sasson, A. Chiesa, C. Garman, M. Green, and I. Miers, "Zerocash: Decentralized anonymous payments from bitcoin," in *IEEE Symposium on Security and Privacy (SP'14)*, pp. 459–474, 2014.

[22] S. F. Tzeng, M. S. Hwang, "Digital signature with message recovery and its variants based on elliptic curve discrete logarithm problem", *Computer Standards & Interfaces*, vol. 26, no. 2, pp. 61–71, Mar. 2004.

[23] M. Vukolic, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," *International Workshop on Open Problems in Network Security*, pp. 112-125, 2015.

[24] Q. Wang, B. Qin, J. Hu, F. Xiao, and Q. Wang, "Preserving transaction privacy in bitcoin," *Future Generation Computer Systems*, 2017. (`http://dx.doi.org/10.1016/j.future.2017.08.026`)

[25] D. A. Wijaya, J. K. Liu, R. Steinfeld, S. F. Sun, and X. Huang, "Anonymizing bitcoin transaction," in *Information Security Practice and Exprience*, pp. 271–283, 2016.

[26] C. Yuan, M. X. Xu, and X. M. Si, "Research on a new signature scheme on blockchain," *Journal of Cryptologic Research*, vol. 2017, no. 2, pp. 1–10, 2017.

[27] Y. Zhu, R. Guo, G. Gan, and W. T. Tsai, "Interactive incontestable signature for transactions confirmation in bitcoin blockchain," in *IEEE 40th Annual Computer Software and Applications Conference (COMPSAC'16)*, pp. 443–448, 2016.

Biography

Zhenhua Liu received his B.S. degree from Henan Normal University, M.S., and Ph.D. degrees from Xidian University, China, in 2000, 2003 and 2009, respectively. He is currently a professor with Xidian University. His research interests include cryptography and information security.

Yuanyuan Li received her B.S. degree from Baoji University of Arts and Sciences in 2016, and she is studying for M.S. degree in Xidian University, China. Her research interests include blockchain and cryptocurrency.

Dong Yuan received her B.S. degree from Lanzhou Jiaotong University in 2016, and she is studying for M.S. degree in Xidian University, China. Her research interests include blockchain and cryptocurrency.

Yaohui Liu received his B.S. degree from Henan Institute of Science and Technology in 2016 , and he is studying for M.S. degree in Xidian University, China. His research interest include Searchable encryption.