Polymerized RingCT: An Efficient Linkable Ring Signature for Ring Confidential Transactions in Blockchain

Junke Duan1,*, Lize Gu1 and Shihui Zheng1

1School of Cyberspace Security, Beijing University of Posts and Telecommunications, City, Beijing 100876, China
*Corresponding author’s e-mail: duanjunke@bupt.edu.cn

Abstract. In this work, we study and refer the privacy protection protocol used in the blockchain, in particular, the Ring Confidential Transaction protocol in current popular anonymous cryptocurrency Monero. Based on our observations on the underlying linkable ring signature and homomorphic commitment schemes, we design a new efficient RingCT protocol for protecting user’s transaction information. Our scheme is built on the Schnorr ring signature, the well-known Pedersen commitment and signature of knowledge. We show that the signature algorithm used in our scheme satisfies the security requirements under random oracle model and adaptive chosen message attack. Moreover, compared with other ring confidential transactions protocols, our scheme can significantly save the cost of operation and interaction linearly under multi-account operation, that is, unlike other schemes, the cost of its underlying signature algorithm doesn't grow linearly as the spender’s accounts grow.

1. Introduction
Blockchain [1] was originally used as the underlying technology of Bitcoin by Satoshi Nakamoto in 2008. Blockchain used to stored transaction ledger. Its security depends on cryptographic tools, i.e., hash function, public key encryption and digital signature. A blockchain can be thought of as a distributed ledger, which is maintained by nodes that are distributed, each with the same complete ledger. Blockchain use consensus algorithm to ensures data consistency between nodes without relying on a trusted third party, which greatly reduces the cost of validation and audit transactions. Since each block in the chain is connected to the previous block through a hash pointer, no one can tamper with the contents of the historical block, ensuring the authenticity of the data on the chain.

Recently, blockchain-based application has a broad prospect in communication, finance, medical treatment, education and other fields. However, most applications based on blockchain technology still have privacy problems. Since the user’s information recorded by record nodes is public, each participant can access all the transaction data. It leads to a series of privacy protection issues of the block chain. Although users can hide their identities by changing their account addresses, since every transaction’s input are all linked to previous transaction’s output, it is still possible to trace the identities of users by analyzing the records of funds transferred. A study by Princeton university in the United States [18] shows that in the Bitcoin system, if users use Bitcoin to conduct transactions with online merchants that accept digital currency, the merchants can easily associate users’ account address with their cookies, so as to find the real identity of users. [17] indicates that account addresses can be linked to IP addresses, then completely de-anonymizing their owners. Therefore, some means are needed to break the link relationship between the spender and receiver of one transaction, while ensuring that the verifier verify the transaction’s legitimacy properly.
1.1. Related Work
In recent years, there have been some privacy protection protocols for blockchain privacy problems. One of them is mixing protocol [2-6] which mixes the tokens of different spenders and redistribute them to the receiving address, so as to break the association between the input address and the output addresses. Mixing protocol can be divided into centralized mixing and decentralized mixing. The former requires a trusted third party to act an online mixer and swap different users’ inputs and outputs. The advantage is that the third party will charge extra service fees, so the centralized mixed coin protocol can resist the witch attack. The disadvantage is that users need to pay extra mixed fees and there is a risk that third parties will steal users' tokens. The latter mixes coins between participants, saves the extra mixing costs but it is susceptible to sybil attacks.

Another attempt to protect privacy in blockchain is Zerocoin [7], which uses the method of zero-knowledge proof [9] to converts the general tokens into commitments corresponding to their information. This allows verifiers verify the transactions without any useful information, which to some extent protects the private information of the transaction party. However, the converted amount cannot be split again, which limits the application of Zerocoin. Based on [7], Zerocash [8] use zk-SNARKs (zero knowledge succinct non-interactive argument of knowledge) to provide anonymity with a formal security proof. For achieving better privacy protection, the transaction amount and input address in [8] were kept secret. However, the proof generation process is rather expensive so that the efficiency is much worse than the normal transaction.

CryptoNote [10] uses one-time address to hide the address of the transaction receiver and a linkable ring signature [15] to hide the address of the transaction spender. Based on it, Shen Noether built ring confidential transactions (RingCT) [11] for the currently popular anonymous cryptocurrency Monero, which is one of the largest cryptocurrencies. Upon the enhancement of privacy, a major trade-off is the increase of size for the transaction. Although ring confidential transactions provides good anonymity, it has the problem of excessive volume. The size of the transaction increases linearly with the number of ring members and the spender's account addresses, which greatly increases the storage overhead of the system. Aiming at this problem, [19] put forward RingCT 2.0 which is built upon the well-known Pedersen commitment, accumulator with one-way domain and signature of knowledge. By accumulating the elements in the ring into a single piece of evidence, it reduced the size of transaction.

1.2. Our contributions
In this paper, observing the above gaps, we propose an efficient ring confidential transaction scheme. Our scheme is constructed based on Schnorr ring signature [13] and Pedersen commitment. Under the requirements of privacy protection of spender’s address and transaction amount, our scheme can hide multiple addresses of one spender’s in a ring signature(In blockchain system model, there are multiple accounts for one user[1]), and the size of the constructed transaction only increases linearly with the number of ring members, which can significantly reduce the operation and storage costs of the algorithm. Our work can be summarized as follows:

First, we present an implementation of linkable ring signature scheme based on the Schnorr ring signature. The original scheme uses signer’s private key and a group of public keys which includes signer’s public key to construct the signature. The verifier can verify that the public key of the signer is a member of the public group, but cannot determine which one belongs to the signer. Our modified scheme allows the signer to construct the signature by using her multiple private keys. In other word, our scheme allows multiple public keys of the signer to be hidden in a ring signature and the verifier cannot determine which public keys belong to the signer. In addition, our scheme has linkability, it means the signature generated by using the same private key can be identified, which can prevent the double payment attack of cryptocurrency in blockchain system.

Second, on the basis of the above and the Pedersen commitment, we present an implementation of Signature of Knowledge scheme. Our scheme allows the verifier to verify the validity of the transaction by hiding the spender's transaction amount. Specifically, the verifier of the transaction can verify that the spender has access to the account of accounts and determine whether the input amount of the
transaction is equal to the output.

Third, we put forward a new efficient RingCT protocol (Polymerized RingCT), which based on our linkable ring signature and signature of knowledge algorithm. Under the requirements of privacy protection of spender’s address and transaction amount, our scheme can hide multiple accounts of spender in one ring signature. In other RingCT schemes, the size of signature and the operation time increases linearly both with number of signer’s accounts and number of group’s accounts. In our scheme, it only increases linearly with the number of group’s accounts. The analysis shows that the proposed scheme is more efficient in the multi-account payment scenario.

1.3. Organization
The remainder of this paper is organized as follows. Section 2 reviews some preliminaries. Section 3 gives some definitions of basic primitives. Section 4 gives a generic construction and an efficient concrete implementation. Section 5 analyzes the security of our scheme. Section 6 compares the efficiency of our scheme with other schemes. Finally, Section 7 take a short summary.

2. Preliminaries

2.1. Mathematical Assumptions

**Discrete Logarithm Problem.** Let $\mathbb{G}$ be a group where the order of $\mathbb{G}$ is $q$. There exists no probabilistic polynomial time (PPT) algorithm that can find an integer $x$ such that $h = g^x$, where $g, h$ are uniformly chosen at random from $\mathbb{Z}_q$.

**Decisional Diffie-Hellman (DDH) Assumption.** Let $\mathbb{G}$ be a group where the order of $\mathbb{G}$ is $q$. $g \in \mathbb{G}$ is the generator of $\mathbb{G}$. There exists no PPT algorithm that can distinguish the distributions $(g, g^a, g^b, g^{ab})$ and $(g, g^a, g^b, g^c)$ with non-negligible probability over 1/2, where $a, b, c$ are chosen uniformly at random from $\mathbb{Z}_q$.

2.2. Linkable Ring Signature

The linkable ring signature scheme was defined by Joseph k. Liu et al. [15]. In this scheme, the private key used by the signer to generate the signature can only be used once, that means the ring signature generated with the same private key cannot pass the verification. Formally, assume there is a set of $n$ users in the system $P = \{P_1, P_2, ..., P_n\}$, each user $P_i$ corresponds to her own private-public key pair $(x_i, Y_i)$. A linkable ring signature algorithm consists of three parts ($KeyGen, Sign, Verify$) with the following syntax:

- $(x, Y) \leftarrow KeyGen(1^\lambda)$: In a polynomial time, input a security parameter $\lambda$, outputs a private-public key pair $(x, Y)$.
- $(\text{sig}, I) \leftarrow Sign(Y, m, x_\pi)$: Assume the signer is $P_\pi$, $P_\pi \in P$. Input message $m$, a set of public key $Y = \{Y_1, Y_2, ..., Y_n\}$, signer’s private key $x_\pi$ and some other parameters, outputs an signature $\text{sig}$ and an image of the private key $x_\pi$.
- $1/0 \leftarrow Verify(Y, m, \text{sig}, I)$: on input a set of public key $Y$, message $m$, signature $\text{sig}$ and image $I$, the algorithm outputs 0/1 if the signature is in/validity.

Before introducing the security model of the linkable ring signature scheme, the following definitions need to be reviewed:

**Definition 1 (Random Oracle Model [15]).** the hash function is used in many signature schemes to achieve non-repudiation. The collisional resistance of the hash function makes it almost impossible for the signer to find two different messages corresponding to the same hash value. Since hash function is an important part of the security of signature algorithm, in order to facilitate the analysis of the security of signature algorithm, many authors (e.g. [20]) put the hash function as a real random function, called "random oracle model". In this model, the hash function can generate a truly random result for each query. If the same question is queried twice, it will output the same answer. This model guarantees the security of the signature scheme If the hash function it uses is safe.

**Definition 2 (Adaptive Chosen-Message Attack [13]).** In a chosen-message attack, an adversary
\(\mathcal{A}\) is given the public data of the scheme. The PPT adversary \(\mathcal{A}\) can ask not only the oracle model \(f\), but also the signature algorithm \(so\) to sign any message, and the choice of the messages is adaptive. The adversary can adapt his queries according to previous message-signature pairs.

**Definition 3 (Security of Linkable Ring Signature [15]).** A linkable ring signature scheme should at least satisfy the properties formalized below.

**Correctness.** This property requires that the verifier can correctly verify the legal signature generated by the signer. Specifically, it holds that:

\[
\Pr[\text{Verify}(Y, m, \sigma) = 1: \sigma \leftarrow \text{Sign}(Y, m, x_g)] = 1
\]

**Unforgeability.** No PPT adversary \(\mathcal{A}\) can generate signature \(\sigma\) with a non-negligible probability under adaptive chosen-message attack. Specifically, it holds that:

\[
\Pr[\Sigma(\sigma \leftarrow \text{Sign}(Y, m, \sigma); \sigma' \leftarrow \text{Sign}(Y', m', x') : x = x') = 1] \geq 1 - \text{negl}(\lambda)
\]

**Setup.** The challenger \(\mathcal{C}\) runs the initialization algorithm, generates public parameters, and sends them to \(\mathcal{A}\).

**Hash query.** No more than \(q_H\) times, \(\mathcal{A}\) can adaptively choose message and sends to \(\mathcal{C}\), then \(\mathcal{C}\) randomly generates the answer and sends it to \(\mathcal{A}\). Without loss of generality, the adversary \(\mathcal{A}\) would not query the same message twice.

**Signature query.** No more than \(q_s\) times, \(\mathcal{A}\) can adaptively choose message and sends to \(\mathcal{C}\), then \(\mathcal{C}\) runs the signature algorithm to generate the signature of the message and sends it to \(\mathcal{A}\).

**Output.** \(\mathcal{A}\) outputs a signature \(\sigma\), if the signature is valid and the signed message did not appear in the previous query, then \(\mathcal{A}\) wins the above game.

**Anonymity.** This property requires that no PPT adversaries \(\mathcal{A}\) be able to identify which private key generated the ring signature. Specifically, it holds that:

\[
\Pr[\pi \leftarrow \mathcal{A}(m, Y, \sigma)] \leq \frac{1}{n} + \text{negl}(\lambda)
\]

where \(\pi\) is the position of signer \(P_\pi\) in the group \(P\).

**Hash query.** No more than \(q_H\) times, \(\mathcal{A}\) can adaptively choose message and sends to \(\mathcal{C}\), then \(\mathcal{C}\) randomly generates the answer and sends it to \(\mathcal{A}\). Without loss of generality, the adversary \(\mathcal{A}\) would not query the same message twice.

**Signature query.** No more than \(q_s\) times, \(\mathcal{A}\) can adaptively choose message and sends to \(\mathcal{C}\), then \(\mathcal{C}\) runs the signature algorithm to generate the signature of the message and sends it to \(\mathcal{A}\).

**Output.** \(\mathcal{A}\) outputs a number \(\pi'\), if \(\pi' = \pi\), then \(\mathcal{A}\) wins the above game.

**Linkability.** If the signer uses the same private key to construct two ring signatures, we say the two signatures are linkable. Specifically, for all PPT adversaries \(\mathcal{A}\), it holds that:

\[
\Pr[\Sigma(\sigma \leftarrow \text{Sign}(Y, m, x); \sigma' \leftarrow \text{Sign}(Y', m', x') : x = x') = 1] \geq 1 - \text{negl}(\lambda)
\]

\[
\Pr[\Sigma(\sigma \leftarrow \text{Sign}(Y, m, x); \sigma' \leftarrow \text{Sign}(Y', m', x') : x \neq x') = 0] \geq 1 - \text{negl}(\lambda)
\]

where \(\{0,1\} \leftarrow \Sigma\) is a PPT algorithm.

### 2.3. Homomorphic Commitment

Formally, a homomorphic commitment scheme consists of a pair of polynomial algorithms \((CKGen, Com)\).

- \(ctk \leftarrow CKGen(1^\lambda)\): on input a security parameter \(\lambda\), outputs a public commitment key \(ctk\), which specifies a message space \(M_{ctk}\) and a commitment space \(C_{ctk}\).

- \(c \leftarrow Com(ctk, m)\): on input a message \(m \in M_{ctk}\), generates a commitment \(c\) to \(m\).

Informally, a homomorphic commitment scheme consists of two phases: commitment and validation. In the commitment step, the spender selects a value and constructs a commitment about it, the spender...
can choose to reveal the value of the commitment in the reveal phase. After that, the verifier can verify that it is indeed the originally committed value. A homomorphic commitment scheme should satisfy the following definitions:

**Definition 4 (Security of Homomorphic Commitment [21])**. A non-interactive scheme \(HCom = (CKGen, Com)\) is called homomorphic commitment scheme and if it satisfies the following definition:

**Hiding**. The value of the original message cannot be found by commitment. More precisely, for all PPT adversaries \(\mathcal{A}\), it holds that:

\[
\Pr \left[ \mathcal{A}(c) = b : ctk \leftarrow CKGen(1^\lambda); b \leftarrow \{0,1\}; c \leftarrow Com(ctk, m_b) \right] \leq \frac{1}{2} \leq \text{negl}(\lambda)
\]

where challenger \(\mathcal{C}\) is a PPT algorithm, \(m_0, m_1 \in M\), if \(\mathcal{A}\) has exactly 1/2 chance of guessing \(b\), then the \(HCom\) has perfect hiding.

**Binding**. The same commitment cannot correspond to two different values. More precisely, for all PPT adversaries \(\mathcal{A}\), it holds that:

\[
\Pr \left[ m_0 \neq m_1; \right. \\
\left. ctk \leftarrow CKGen(1^\lambda); \right. \\
\left. Com(r_0, m_0) = Com(r_1, m_1); \right. \\
\left. (r_0, m_0, r_1, m_1) \leftarrow \mathcal{A}(ctk) \right] \leq \text{negl}(\lambda)
\]

where \(m_0, m_1 \in M\), \(r_0, r_1\) are random chosen. If the probability is exactly 0, then the \(HCom\) has perfectly binding.

**Homomorphism**. We assume that the commitment space \(\mathcal{C}_{ctk}\) is an additive group of order \(q\) and both the messages and random values are from \(\mathbb{Z}_q\), \(m_0, m_1 \in \mathbb{Z}_q; r_0, r_1 \in \mathbb{Z}_q\), it holds that:

\[
Com(r_0, m_0) + Com(r_1, m_1) = Com(r_0 + r_1, m_0 + m_1)
\]

2.4. *signature of knowledge*

The Proof of Knowledge protocols can be converted into a signature scheme by setting the challenge to the hash value related to the information to be signed [22]. Such signatures are referred to as Signatures of Knowledge (SoK) [23]. For example, we denote by \(SoK(x) : y = g^x(m)\), where \(m\) is the message, the signature contains a zero knowledge proof of discrete logarithms which is \(y = g^x\). SoK is composed of three polynomial time algorithms \((Gen, Sign, Verify)\), similar to the normal signature scheme, which is not covered in this section. An SoK scheme should satisfy the following definitions:

3. **Building Blocks**

In this section, we describe the basic primitives used to build our ring confidential transactions scheme, which include Blockchain system model, Schnorr ring signature, multilayered linkable ring signature, Pedersen commitment and the RingCT protocol for Monero.

3.1. **Blockchain Foundation**

As an example, we introduce how the Bitcoin system works on behalf of the blockchain cryptocurrency. Many anonymous cryptocurrencies, including Monero, are derived from it.

Bitcoin is a decentralized payment scheme. In Bitcoin system, there are three types of participants: nodes, spenders and receivers. When the spender wants to transfer money to the receiver, she constructs a transaction and publishes the transaction to the network by broadcasting. Then, the nodes listen, collect and package of the transactions that are broadcast to the network. In the network of Blockchain, there are multiple nodes that competitively process the transactions through consensus algorithms, e.g., the Proof of Work [1] is a consensus algorithm in Bitcoin to ensure ledger consistency between nodes. Specifically, if a node validates a batch of transactions, before it packages these transactions into blocks, it has to solve a computational puzzle which is to compute a hash value of the block header’s information that must satisfy certain conditions (e.g., less than a certain value). Nodes need to try different random numbers until the conditions are satisfied. When a node calculates a qualified hash value, then its block becomes the new block of the blockchain. This process is very hard as well as the winner of each block competition may obtain a large reward. The record nodes of Bitcoin are also called miners, and the competition process is called mining. When a node finds a solution for a new block first, it will broadcast
its block including the solution to other nodes.

After receiving the block, other miners need to verify the validity of the transaction in the block, including verifying that the input amount of each transaction is equal to its output amount; whether the spender's address has been used in past transactions (It’s often called double spending); whether the spender's signature is correct; and finally verify that the hash of the block's header satisfies the condition. If they all pass, other miners will add new blocks to their local chains, and those who produce the block will be rewarded. In order to ensure the security of the network, the system will adjust the difficulty of the puzzle according to the time of each block to make the time to generate a new block approximately constant (It's about two minutes in Bitcoin). Assuming that most participants were honest, by using consensus algorithm, the nodes could jointly maintain a unified ledger. It’s ensuring the security of the system through the calculation of the nodes.

3.2. Schnorr Ring Signature

Herranz J et al. constructed a ring signature scheme [13] based on the well-known Schnorr signature, and proves its security under the assumption of the random prediction model. [14] gives the implementation of the algorithm on the elliptic curve. In section 4.1 we will construct a new ring secret transaction based on it. In this section, we will introduce the specific steps of the algorithm:

- **KeyGen**(1^λ): on input a security parameter λ, generate a group G where the order of G is q, the algorithm generates the private key 𝑥 ∈ ℤ∗ and the corresponding public key 𝑌 = 𝑥G.

- **S 𝑖 𝑔 𝑛 ḋ (𝑌, 𝑚, 𝑥𝑔)**: on input a set of public keys 𝑌 = {𝑌₁, ..., 𝑌𝑛} including signer’s public key 𝑌𝑔, 𝜋 ∈ 1, ..., 𝑛, signer’s private key 𝑥𝑔 and the message 𝑚, the algorithm performs the following steps:

  1) Choose 𝑡 ∈ ℤ∗ uniformly at random, and for each 𝑖 ∈ 1, ..., 𝑛, 𝑖 ≠ 𝜋, choose 𝜎𝑖 ∈ ℤ∗ uniformly at random.

  2) Compute:

     \[ h = Hash(𝑚||𝑌, 𝑡G + \sum_{𝑖≠𝜋} 𝜎𝑖𝑌𝑖) \]

     \[ 𝜎_π = h - \sum_{𝑖≠𝜋} 𝜎_𝑖 \]

     \[ z = t - 𝜎_π𝑥_π \]

  3) Output the signature **sig** = ( 𝜎_1, ..., 𝜎_𝑛, 𝑧 )

- **Verify**(𝑌, 𝑚, sig): on input the set of public keys 𝑌, the message 𝑚, the signature sig, the algorithm outputs 0/1 if \( \sum_𝑖 𝜎_𝑖 \) is un/equal to \( Hash(𝑚||𝑌, 𝑧G + \sum_{𝑖=1}^{𝑛} 𝜎_𝑖 𝑌_𝑖) \).

3.3. Pedersen Commitment

Pedersen commitment was proposed by Maxwell in the confidential transaction scheme [12], which is a homomorphic commitment scheme to hide the transaction. We will construct our signature of knowledge scheme based on it in section 4. Its steps are as follow:

In the initialization step, suppose G is the generator of the additive group G, and h is another point on the same group. The discrete logarithm between h and G is unknown. Define commitment C = aG + vH, where v is the value corresponding to the commitment, and a is called the blinding factor to hide the value of v. Confidential transactions are often used in cryptocurrency, where v is used to represent the transaction’s amount. If don’t set blinding factor, then the commitment C = vH. Since the value range of v in the actual transfer operation is limited, the attacker can guess the v by exhaustive method. If an attacker gets the value of a commitment, then all commitments which hide the same secret value will be revealed.

In the commitment step, suppose that the inputs C_in and outputs C_out in a transaction are represented by commitments. The spender determines the private key ck_in = (a_in, v_in) corresponding to her commitment C_in. The receiver privately sends ck_in to spender. Then the spender computes C_out = C_in - C_out. When the sum of inputs equals to the sum of outputs, i.e., \( \sum 𝑣_𝑖 𝐻 = 0 \), it means that the spender is able to get \( zG = \sum 𝑐_𝑖 G = \sum 𝑎_𝑖 𝐺 - \sum 𝑎_𝑜𝑢𝑡 𝐺 \). Then she sends c = (C_in, C_out, z) to the verifier.

In the verification step, the verifier receives c and verifies \( \sum 𝑐_𝑖 = \sum 𝑎_𝑜𝑢𝑡 = zG \). If it satisfies the
condition, it means the transaction is valid. After validation, the verifier adds the transaction’s outputs to the local list. When the receiver wants to build a new transaction, he will use the outputs of this transaction as inputs to the new transaction.

Pedersen commitment is also applied to RingCT [11], in order to hide the spender’s amount, it subtracts the output amount from the amount of each account respectively to construct the commitments’ ring signature. Suppose the spender’s commitments are \( C_{in}, g \), it holds that:

\[
\sum C_{in,1} - \sum C_{out}, ... , \sum C_{in,n} - \sum C_{out}\]

According to the above, the spender knows \( z \) which satisfies \( zG = \sum C_{in} - \sum C_{out} \), so she can construct a ring signature to hide her commitment of input amount. However, this ring signature is not linkable, because the commitment corresponding to the amount \( v \) is not unique, and the proof \( z \) can be changed by changing \( C_{in} \), so this ring signature is used as a row in [11], and not be used alone.

3.4. RingCT protocol in Monero

Ring confidential transaction is a privacy protection scheme used in the currently popular anonymous cryptocurrency Monero, which is based on the CryptoNote protocol [10] to hide the spender’s address and transaction amount. In the Monero system, the user’s information is stored locally on the node in the form of an "address-commitment", where the "address" is the user’s public key, and the "commitment" is the amount in the form of a Pedersen commitment whose value is unknown to others. Informally, a ring confidential transaction scheme consists of two phases: generation and validation. In the transaction generation phase, the spender accesses the accounts’ list of the node to obtain the set of accounts, including other users accounts, together with her own accounts to construct the ring confidential transaction. Then she broadcast the constructed transactions to the network, and the nodes collect the transactions and then verify them through the verification algorithm. After verification, the transactions are packaged into blocks and published to the network. [19] gave a formal syntax of RingCT protocol. We will show our implementation with this syntax in section 4.3. The formalized algorithm is as below. In general, a RingCT protocol consists of a tuple of polynomial time algorithms \( (KeyGen, Mint, Spend, Verify) \), the syntax of which are described as follows:

- \((x, Y) \leftarrow KeyGen(1^\lambda)\): on input a security parameter \( \lambda \), the algorithm outputs corresponding private-public key pair \((x, Y)\). In Monero, the public key \( Y \) is always taken as the user’s address, which together constitutes an account with a commitment of amount.

- \((C, ck) \leftarrow Mint(Y, v)\): on input an amount \( v \) and a valid address \( Y \), the algorithm outputs a commitment \( C \) for \( Y \) as well as the associated commitment key \( ck \). The commitment \( C \) and \( Y \) together forms an account \( act = (Y, C) \), the corresponding secret key of which is \( ask = (x, ck) \). The user must know the secret key \( ask \) before using the account.

- \((tx, sig, S) \leftarrow Spend(m, A_{\pi}, K_{\pi}, A_{out})\): on input a set \( A_{\pi} \) of accounts with the corresponding set of account secret keys \( K_{\pi} \), a set \( A \) of input accounts containing \( A_{\pi} \), a set \( A_{out} \) of output accounts and some transaction string \( m \in \{0,1\}^* \), the algorithm outputs a transaction \( tx \) (containing \( m, A, A_{out} \)), a signature \( sig \) and a set \( S \) of serial numbers.

- \(1/0 \leftarrow Verify(tx, sig)\): on input transaction \( tx \), signature \( sig \) and the serial numbers \( S \), the algorithm verifies whether a set of accounts with serial numbers \( S \) is spent properly for the transaction \( tx \) towards addresses \( A_{out} \), and outputs 0/1 when the spending is in/valid.

4. Our RingCT Scheme

In this section, we introduce our polymerized ring confidential transaction scheme (polymerized RingCT for short). In the scheme [11], when the spender has \( k \) accounts to spend, the transferor needs \( n \times k \) accounts in total to construct the signature (where \( (n-1) \times k \) accounts are used to hide the spender’s \( k \) accounts). In other words, each spender’s accounts need another \( n-1 \) address to hide itself. In our scheme, we will just use \( n \) addresses to hide all accounts of the sender. We will first give an implementation of linkable ring signature based on [14], which allows multiple public keys of the signer to be hidden in a ring signature and the verifier cannot determine which public keys belong to the signer. Then, we present
an implementation of Signature of Knowledge scheme, which allows the verifier to verify the validity of the transaction as well as hides the spender's transaction amount. Finally, we follow the syntax of [19] to give the complete steps of our scheme.

4.1. Multiple Linkable Ring Signature
In this section, we introduce our signature scheme according to the syntax of Section 2.2, the steps of which are described as follows:

- **KeyGen(1^λ):** on input a security parameter λ, the algorithm prepares an additive group \( \mathbb{G}_q \) whose generator is \( G \) and order is \( q \), and generates a private-public key pair \( (x, Y = xG) \in \mathbb{Z}_q \times \mathbb{G}_q \).
- **Sign(m, Y, Y_{π}, X_{π})**: on input transaction message \( m \in \{0,1\}^* \), a set of public keys by \( Y = \{Y_1, Y_2, ..., Y_{n}\} \), a set of signer’s public keys \( Y_π = \{Y_{π1}, ..., Y_{πk}\} \) where \( Y_π \subset Y \), the corresponding secret keys of which are \( X_π = \{x_1, ..., x_k\} \), and denote a set \( J \) which satisfies \( \forall j \in [1, ..., n], Y_j \in Y_π \).
  1) For each \( j \in [1, ..., k] \), compute:
     \[ I_j = x_j^{-1}H_p(Y_j) \]  (4)
  2) Choose \( t \in \mathbb{Z}_q^* \) randomly, and for each \( i \in [1, ..., n], i \notin J \), choose \( a_i \in \mathbb{Z}_q^* \) randomly.
  3) Compute:
     \[ h_Y = Hash(m||Y, \sum_{j=1}^{k} z_j(G + I_j) + \sum_{i \notin J} (a_i Y_i + a_i H_p(Y_i))) \]  (5)
     \[ \sigma_π = h_Y - \sum_{i \notin J} a_i \]  (6)
  4) For each \( j \in [1, ..., k-1] \), choose \( \sigma_j^{k-1} \in \mathbb{Z}_q^* \).
  5) Compute:
     \[ \sigma_π = \sigma_π - \sum_{i \notin J} a_i \]  (7)
     \[ z_j = t - \sum_{i \notin J} (a_i + \sigma_j) \]  (8)
  6) Output the signature \( sig_π = (σ_1, ..., σ_n, z_1, ..., z_k) \) and the set of images by \( I = (I_1, ..., I_k) \).

- **Verify(Y, m, sig_π, I)**: on input a set of public key \( Y \), message \( m \), signature \( sig_π \) and image \( I \), the algorithm verifies that:
     \[ \sum_{i=1}^{n} a_i = Hash(m||Y, \sum_{j=1}^{k} z_j(G + I_j) + \sum_{i=1}^{n} (a_i Y_i + a_i H_p(Y_i))) \]  (9)
  If true, it outputs 1, otherwise outputs 0.

4.2. SoK of Transaction’s Amount
In this section, we present a signature of knowledge scheme for proving that the spender has the right to spend her accounts and the transaction is legal. The steps of which are described as follows:

- **Gen(1^λ)**: on input a security parameter \( \lambda \), the algorithm prepares an additive group \( \mathbb{G}_q \) with the generator \( G \), the order \( q \) and another element \( H \in \mathbb{G} \). The discrete logarithm between \( G \) and \( H \) is unknown. Output the commitment \( C ≜ aG + vH \), where \( a \in \mathbb{Z}_q \) is the blinding factor and \( v \in \mathbb{Z}_q \) is the value of commitment, and commitment key \( ck ≜ (a, v) \).
- **Sign(m, Y_π, X_π)**: on input transaction message \( m \in \{0,1\}^* \), a set of commitments by \( Y_π = \{C_1, ..., C_n\} \), a set of signer’s commitments \( C_π = \{C_{π1}, ..., C_{πk}\} \) where \( C_π \subset C \), the corresponding secret keys of which are \( c_π = \{c_{π1}, ..., c_{πk}\} \), and denote a set \( J \) as before.
  1) Choose \( t \in \mathbb{Z}_q^* \) randomly, and for each \( i \in [1, ..., n], i \notin J \), choose \( \varphi_i \in \mathbb{Z}_q^* \) randomly.
  2) Compute:
     \[ h_C = Hash(m||C, t \cdot G + \sum_{i \notin J} (\varphi_i G + C_i - \frac{1}{k} C_{\text{out}})) \]  (10)
     \[ \varphi_π = h_C - \sum_{i \notin J} \varphi_i \]  (11)
  3) For each \( j \in [1, ..., k-1] \), choose \( \varphi_j^{k-1} \in \mathbb{Z}_q^* \).
  4) Compute:
     \[ \varphi_π = \varphi_π - \sum_{j=1}^{k-1} \varphi_j^j \]  (12)
     \[ z_π = t - \sum_{i \notin J} (\varphi_i + a_i) + a_{\text{out}} \]  (13)
5) Output the signature $\mathit{sig}_C = (\varphi_1, \ldots, \varphi_n, z_C)$. 

- \text{Verify}(C, m, \mathit{sig})$: on input a set of commitments $C$, message $m$ and signature $\mathit{sig}_C$, the algorithm verifies

$$
\sum_{i=1}^n \varphi_i z_C = \text{Hash}(m || C, z_C) \cdot G + \sum_{i=1}^n (\varphi_i G + C_i - \frac{1}{k} C_{\text{out}})).
$$

If true, it outputs 1, otherwise outputs 0.

4.3. Description of Complete Scheme

In this section, we present our new RingCT scheme under the formalized syntax mentioned in Section 3.4. Based on the algorithm in Section 4.1 and Section 4.2, our scheme is designed as follows:

\text{KeyGen}(1^\lambda): on input a security parameter $\lambda$, the algorithm prepares an additive group $\mathbb{G}_q$ whose generator is $G$, order is $q$, another element $H \in \mathbb{G}$ whose discrete logarithm between $G$ is unknown, and generates a private-public key pair $(x, Y = xG) \in \mathbb{Z}_q \times \mathbb{G}_q$.

\text{Mint}(Y, v)$: on input value $v \in \mathbb{Z}_q$ and a valid address $Y$, the algorithm mints a coin for $Y$: chooses $r \in \mathbb{Z}_q$ randomly, compute commitment $C = r \cdot G + v \cdot H$, where $a \in \mathbb{Z}_q$ is the blinding factor and $v \in \mathbb{Z}_q$ is the value of commitment, and corresponding commitment key $ck = (a, v)$, and then returns an account $act = (Y, C)$, which is corresponding to secret key $ask = (x, (r, v))$.

\text{Spend}(m, A_{\text{in}}, K_{\text{in}}, A, A_{\text{out}})$: on input a set $A_{\text{in}} = \{ (Y_1, C_1), \ldots, (Y_k, C_k) \}$ of accounts with the corresponding set of account secret keys $K_{\text{in}} = \{ ask_1, \ldots, ask_k \}$, a set $A = \{ (Y_1, C_1), \ldots, (Y_n, C_n) \}$ of input accounts containing $A_{\text{in}}$ a set of output accounts (For simplicity, we denote there is only one output account $Y_{\text{out}}$ in next steps, It is still used when there are multiple output addresses) and some transaction string $m \in \{0,1\}^*$. 

1) Decide amount $v_{\text{out}}$ for output address $Y_{\text{out}}$ which satisfies $\sum_{j=1}^k v_{\text{in}_j} = v_{\text{out}}$. Choose $r_{\text{out}} \in \mathbb{Z}_q$ randomly. Compute commitment $C_{\text{out}} = r_{\text{out}} \cdot G + v_{\text{out}} \cdot H$. Generate output account $act_{\text{out}} = (Y_{\text{out}}, C_{\text{out}})$ and it’s corresponding secret key $ck_{\text{out}} = (r_{\text{out}}, v_{\text{out}})$, then privately send $ck_{\text{out}}$ to the receiver.

2) Input the set of addresses $Y = (Y_1, \ldots, Y_n)$ and corresponding private keys $X = (x_1, \ldots, x_k)$ and other transaction message $m \in \{0,1\}^*$, perform the signature algorithm in Section 4.1, then get signature $\mathit{sig}_y = (\sigma_1, \ldots, \sigma_n, z_1, \ldots, z_k)$ and serial numbers $S = (I_1, \ldots, I_k)$.

3) Input the set of commitments $C = (C_1, \ldots, C_n)$ and corresponding secret keys $CK = (ck_1, \ldots, ck_k)$, the output commitment $C_{\text{out}}$ and other transaction message $m$, perform the SoK algorithm in Section 4.2, then get $\mathit{sig}_C = (\varphi_1, \ldots, \varphi_n, z_C)$.

4) Output a signature $\mathit{sig} = (\mathit{sig}_y || \mathit{sig}_C) = (\sigma_1, \ldots, \sigma_n, \varphi_1, \ldots, \varphi_n, z_1, \ldots, z_k)$, a transaction $tx = (m, A, A_{\text{out}})$ and a set of serial numbers $S = (I_1, \ldots, I_k)$.

\text{Verify}(tx, \mathit{sig})$: Input transaction $tx$, signature $\mathit{sig}$ and the serial numbers $S$, execute the validation algorithm in Section 4.1 and Section 4.2 respectively. If they both true, outputs 1, otherwise outputs 0.

5. Security Analysis

In this part, we will analyze the underlying signature algorithm’s security of our scheme under the formalized security models which are given in Definition 3.

\textbf{Theorem 1.} In this section, we will prove the correctness of the equations which the verifier needs to verify in section 4.1 and 4.2 respectively. As follows:

\textbf{Proof.} For the equations that the verifier needs to verify in section 4.1, we have:

$$
\text{Hash}(m || Y, \sum_{j=1}^k z_j (G + I_j) + \sum_{i=1}^n (\sigma_i Y_i + \sigma_i H_p(Y_i)))
$$

$$
= \text{Hash}(m || Y, \sum_{j=1}^k (t - \sigma_j x_j) (G + I_j) + \sum_{i \in j}(\sigma_i Y_i + \sigma_i H_p(Y_i)) + \sum_{i \in j}(\sigma_i Y_i + \sigma_i H_p(Y_i)))
$$

If true, it outputs 1, otherwise outputs 0.
\[ \text{Hash}(m|Y, \sum_{j=1}^{k} t(G + l_j) - \sum_{i \in E_j} \sigma_i x_i G - \sum_{i \in E_j} \sigma_i x_i x_i^{-1} H_p(Y_i^j) + \sum_{i \in E_j} (\sigma_i Y_i + \sigma_i H_p(Y_i)) + \sum_{i \in E_j} (\sigma_i Y_i + \sigma_i H_p(Y_i))) \]

\[ = \text{hash}(m|Y, \sum_{j=1}^{k} t(G + l_j) + \sum_{i \in E_j} (\sigma_i Y_i + \sigma_i H_p(Y_i))) \]

\[ = h_y \]

\[ = \sigma_g + \sum_{i \in E_j} \sigma_i \]

\[ = \sum_{i \in E_j} \sigma_i + \sum_{i \in E_j} \sigma_i \]

\[ = \sum_{i=1}^{n} \sigma_i \]

For the equations that the verifier needs to verify in section 4.2, we have:

\[ \text{hash}(m|C, z_g G + \sum_{i=1}^{n} (\varphi_i G + C_i - \frac{1}{k} C_{out})) \]

\[ = \text{hash}(m|C, t \cdot G - \sum_{i \in E_j} (\varphi_i + a_i) + a_{out} G + \sum_{i \in E_j} (\varphi_i G + C_i - \frac{1}{k} C_{out})) + \]

\[ \sum_{i \in E_j} (\varphi_i G + C_i - \frac{1}{k} C_{out})) \]

\[ = \text{hash}(m|C, t \cdot G - \sum_{i \in E_j} (\varphi_i G + a_i G) + a_{out} G + \sum_{i \in E_j} (\varphi_i G + C_i - \frac{1}{k} C_{out})) + \]

\[ \sum_{i \in E_j} (\varphi_i G + a_i G + v_i H) - a_{out} G - v_{out} H) \]

\[ = \text{hash}(m|C, t \cdot G + \sum_{i \in E_j} (\varphi_i G + C_i - \frac{1}{k} C_{out})) \]

\[ = h_c \]

\[ = \varphi_g + \sum_{i \in E_j} \varphi_i \]

\[ = \sum_{i \in E_j} \varphi_i + \sum_{i \in E_j} \varphi_i \]

\[ = \sum_{i=1}^{n} \varphi_i \]

**Theorem 2.** In terms of the discrete logarithm problem and random oracle model, our scheme is unforgeable for any PPT adversary \( \mathcal{A} \).

**Proof.** Suppose there is a PPT adversary \( \mathcal{A} \) and a challenger \( \mathcal{C} \) which is responsible for answering the queries of \( \mathcal{A} \). If \( \mathcal{A} \) can forge a valid signature with non-negligible probability, then there is an PPT algorithm \( \mathcal{M} \) can solve the discrete logarithm problem by using \( \mathcal{A} \).

Setup. The challenger \( \mathcal{C} \) runs the initialization algorithm, generates the security parameter \( \lambda \), an additive group \( \mathbb{G}_q \) with generator \( G \) and order \( q \). For each \( i \in \{1, ..., n-1, n+1, q_H\} \), \( \mathcal{C} \) picks \( x_i \in \mathbb{Z}_q \) randomly, and generates a set of public keys \( Y = (Y_1, ..., Y_{n}, ..., Y_{q_H}) \) where \( Y_i = x_i G, i \neq \pi \) and \( Y_{\pi} \) is chosen randomly from \( \mathbb{G}_q \). \( \mathcal{C} \) sends \( \lambda, \mathbb{G}_q, G, Y \) to \( \mathcal{A} \).

Hash query. No more than \( q_H \) times, \( \mathcal{A} \) adaptively chooses message and sends to \( \mathcal{C} \), then \( \mathcal{C} \) randomly generates the answer and sends to \( \mathcal{A} \). The adversary \( \mathcal{A} \) would not query the same message twice.

Signature query. No more than \( q_s \) times, \( \mathcal{A} \) adaptively chooses message \( m \), a set of public keys \( Y' \subseteq Y \), the signer’s public key \( Y_{\pi} \in Y' \) and sends them to \( \mathcal{C} \). If \( Y_{\pi} \neq Y_{\pi} \), \( \mathcal{C} \) runs the signature algorithm in Section 4.1 to generate the signature of the message and sends it to \( \mathcal{A} \). If \( Y_{\pi} = Y_{\pi} \), \( \mathcal{C} \) performs the following steps:

1) Choose \( I_{\pi} \in \mathbb{G}_q \) randomly, \( I_{\pi} \neq G \).
2) Choose \( z_{\pi} \in \mathbb{Z}_q^* \) randomly, and for each \( i \in \{1, ..., n\} \), choose \( \sigma_i \in \mathbb{Z}_q^* \) randomly.
3) Store the relationship \( \sum_{i=1}^{n} \sigma_i = \text{Hash}(m|W, z_{\pi} I_{\pi} + \sum_{i=1}^{n} \sigma_i H_p(Y_i)) \).
4) Output the signature \( \text{sig} = (\sigma_1, ..., \sigma_n, z_{\pi}, ..., z_{\pi}, I_{\pi}) \).

Output. \( \mathcal{A} \) outputs a signature \( \text{sig} \), if the signature is valid and the signed message did not appear in the previous query, then according to the forking lemma in [18], there is a PPT algorithm \( \mathcal{M} \) which can generate two valid signatures \( \text{sig} \) and \( \text{sig}' \) by using \( \mathcal{A} \) where

\[ \text{sig} = (\sigma_1, ..., \sigma_n, z_{\pi}, I_{\pi}) \]

\[ \text{sig}' = (\sigma_1, ..., \sigma_n, z_{\pi}, I_{\pi}) \]

where \( \sigma_{\pi} \neq \sigma_{\pi}', z_{\pi} \neq z_{\pi}' \).
By \( \frac{z_\pi - z_\pi'}{\sigma_\pi - \sigma_\pi} = \frac{t - \sigma_\pi x_\pi + \sigma_\pi x_\pi'}{\sigma_\pi - \sigma_\pi} = x_\pi \), \( \mathcal{M} \) gets an instantiation to solve discrete logarithm, which contradicts the discrete logarithm assumption.

**Theorem 3.** In terms of DDH assumption and random oracle model, our scheme is anonymous for any PPT adversary \( \mathcal{A} \).

**Proof.** Suppose there is a PPT adversary \( \mathcal{A} \) and a challenger \( \mathcal{C} \) which is responsible for answering the queries of \( \mathcal{A} \). If \( \mathcal{A} \) can identify signer’s public key from the set of public keys with

\[
Pr > \frac{1}{n} + \frac{1}{\text{negl}(\lambda)}
\]

then there is a PPT algorithm \( \mathcal{M} \) can solve DDH assumption with

\[
Pr > \frac{1}{2} + \frac{1}{\text{negl}(\lambda)}
\]

by using \( \mathcal{A} \).

**Setup.** The challenger \( \mathcal{C} \) runs the initialization algorithm, generates the security parameter \( \lambda \), an additive group \( \mathbb{G}_q \) with generator \( G \) and order \( q \). For each \( i \in \{1, \ldots, n\} \), \( \mathcal{C} \) picks \( x_i \in \mathbb{Z}_q \) randomly, and generates a set of public keys \( Y=(Y_1, ..., Y_n) \) where \( Y_i = x_i G \), then \( \mathcal{C} \) picks \( l_j \) where \( j \in \{0,1\} \) is chosen randomly and \( I_1 = I_{0_1}, I_0 \neq I_\pi \). \( \mathcal{C} \) performs the sign algorithm in Section 4.1 with \( Y, x_\pi \), a string of message \( m \in \{0,1\}^* \) and sends \( I, \) the signature and other public parameters to \( \mathcal{A} \).

**Hash query.** No more than \( q_H \) times, \( \mathcal{A} \) adaptively chooses message and sends to \( \mathcal{C} \), then \( \mathcal{C} \) randomly generates the answer and sends to \( \mathcal{A} \). The adversary \( \mathcal{A} \) would not query the same message twice.

**Signature query.** No more than \( q_s \) times, \( \mathcal{A} \) adaptively chooses message \( m \), a set of public keys \( Y \subseteq Y \), the signer’s public key \( Y_\pi \in Y' \) and sends them to \( \mathcal{C} \). \( \mathcal{C} \) runs the signature algorithm in Section 4.1 to generate the signature of the message and sends it to \( \mathcal{A} \).

**Output.** If \( \mathcal{A} \) can outputs an index \( \pi' \) with \( Pr > \frac{1}{n} + \frac{1}{\text{negl}(\lambda)} \) where \( \pi' = \pi \), for an instantiation of DDH assumption \( Q = \{(G, x_\pi G, l_1, H_p(Y_\pi)) \rightarrow j, j \in \{0,1\}, I_1 = l_1, I_0 \neq I_\pi \} \), there is a PPT algorithm \( \mathcal{M}(Q) \) which outputs 1 if \( \pi' = \pi \) and outputs 0/1 with the same probability if \( \pi' \neq \pi \), then we have:

\[
Pr[\mathcal{M}(Q) = j|j = 1] = Pr[\mathcal{M}(Q) = j|j = 1, \mathcal{A} = \pi] + Pr[\mathcal{M}(Q) = j|j = 1, \mathcal{A} \neq \pi] \\
\geq 1 \cdot \left( \frac{1}{2} + \frac{1}{Q(K)} \right) + \frac{1}{2} \left( 1 - \frac{1}{n} - \frac{1}{Q(K)} \right) \\
\geq \frac{1}{2} + \frac{1}{2n} + \frac{1}{\text{negl}(\lambda)}
\]

If \( j = 0 \), then all signers are symmetric from \( \mathcal{A}'s \) perspectives, and \( \mathcal{A} \) can do no better than random guessing. Averaging over \( \mathcal{M}'s \) random choice of \( \pi, 1 \leq \pi \leq n \), we obtain:

\[
Pr[\mathcal{M}(Q) = j|j = 0] = Pr[\mathcal{M}(Q) = j|j = 0, \mathcal{A} = \pi] + Pr[\mathcal{M}(Q) = j|j = 0, \mathcal{A} \neq \pi] \\
\geq 0 \cdot \frac{1}{n} + \frac{1}{2} \cdot (1 - \frac{1}{n}) \\
Pr[\mathcal{M}(Q) = j] = \frac{1}{2} \cdot Pr[F = j|j = 1] + \frac{1}{2} \cdot Pr[F = j|j = 0] \\
\geq \frac{1}{2} + \frac{1}{4\text{negl}(\lambda)}
\]

Then \( \mathcal{M} \) gets an instantiation to solve this problem, which contradicts the DDH assumption.

**Theorem 4.** Under the formalized syntax mentioned in Section 2.2, our scheme is linkable for any PPT adversary \( \mathcal{A} \).

**Proof.** Assume there is an algorithm \( \Sigma(sig, sig') \rightarrow \{1,0\} \): \( I_\pi \in sig, I'_\pi \in sig' \), if \( I_\pi = I'_\pi \), \( \Sigma(sig, sig') = 1 \); else if \( I_\pi \neq I'_\pi \), \( \Sigma(sig, sig') \rightarrow 0 \). \( \mathcal{A} \) uses the same private key \( x \) to construct two signatures \( \sigma \) and \( \sigma' \), the corresponding public keys' index are \( \pi \) and \( \pi' \). According to Section 3.4, we know that the valid signature must satisfy the following condition:

\[
\text{Hash}(m||Y, z(G + I_\pi) + \sum_{i=1}^{m}(\alpha_i Y_i + \sigma_i H_p(Y_i))) = \text{Hash}(m||Y, t(G + I_\pi) + \sum_{i=\pi}(\alpha_i H_p(Y_i)))
\]
Since the probability of hash collision is negligible, we have:
\[
\begin{align*}
&z(G + l_n) + \sum_{i=1}^{n} (\sigma_i Y_i + \sigma_i H_p(Y_i)) = t(G + l_n) + \sum_{i=\pi} (\sigma_i H_p(Y_i)) \\
&z(G + l_n) + \sigma_\pi Y_\pi + \sigma_\pi H_p(Y_\pi) = t(G + l_\pi) \\
&(t - \sigma_\pi x_\pi)(G + l_n) + \sigma_\pi Y_\pi + \sigma_\pi H_p(Y_\pi) = t(G + l_\pi) \\
&x_\pi l_\pi = H_p(Y_\pi) \\
&l_\pi = x_\pi^{-1} H_p(Y_\pi)
\end{align*}
\]
As above, for $\text{sig}'$ we have:
\[
l_{\pi}^{x_{\pi}x_{\pi}'^{-1}H_p(Y_\pi)} \rightarrow l_{\pi}' = l_{\pi}', \Sigma(\text{sig}, \text{sig}') = 1.
\]

Since there exists an algorithm which satisfies the condition in Section 2.2, we say our scheme is linkable for any PPT adversary $\mathcal{A}$.

6. Efficiency Analysis
In this section, we briefly compare the efficiency of our scheme with that in [11] and [19]. The linkable ring signature used in scheme [11] needs to construct a ring signature for each account of the sender, whose signature size increases linearly with the number $m$ of accounts of the sender and the number $n$ of accounts required to construct one signature. More concretely, it is almost $O(n(m + 1))$. Scheme [19] uses one-way accumulator to compress the size of each group of signatures, whose communication complexity is about $O(m)$. However, the efficiency of constructing and validating the signature is still $O(n(m + 1))$. Also, because the knowledge signature it constructs is associated with the length $\lambda$ of the group element. When $n < \lambda$, the communication complexity is still higher than that of the scheme [11]. In fact, currently the ring membership $n$ of the Monero is 11, and it uses the Ed25519 standard of elliptic curve, which has 256 bits of group elements. It’s much larger than $n$. In our scheme, the above complexity only increases linearly with $n$, in other words, it is almost $O(n)$. When the sender uses more accounts, the advantage of our scheme becomes more obvious. The detailed comparison is shown in table 1. For intuition, we just list the highest order terms and hide the terms that have negligible impact on performance.

7. Conclusion
In this work, we design an efficient ring confidential transaction scheme based on the CryptoNote protocol and Schnorr ring signature. We first present an implementation of linkable ring signature scheme based on the Schnorr ring signature. Then we propose a new efficient RingCT protocol based on our linkable ring signature and signature of knowledge algorithm. We also prove that our protocol is safe under the random oracle model. We briefly compare the efficiency of our scheme and other RingCT schemes. The result is that our scheme is more efficient when using multiple accounts to transfer funds. The downside is that, like other schemes, our scheme will reveal the number of accounts used by the originator of the transaction, and we hope to improve this in future work.

Acknowledgements
This work would never have existed without the support of the National Key R&D Program of China (Grant Number 2017YFB080301). Thank to all the cooperate author and reviewers for their helpful advices. Thanks to professor Lize Gu and professor Shihui Zheng for providing me lots of help and many valuable suggestions to my research.
Table 1. Comparison of RingCT Schemes

| Ref. | Spender Communication | Verifier Communication |
|------|-----------------------|------------------------|
| [13] | \(2.1(m + 1)n \cdot exp\) | \(2.2(m + 1)n \cdot exp\) | \(mn|\mathbb{Z}_p|\) |
| [21] | \(0.4(m + 1)\cdot(n + 1)\cdot exp\) | \(3.2\lambda(m + 1)\cdot exp\) | \((\lambda + 5)(m + 1)|\mathbb{G}|\) |
|       |                       | \(+ (1.1\lambda + 0.4n + 4)\cdot(m + 1)\cdot exp\) | \(+ (3\lambda + 6)\cdot(m + 1)|\mathbb{Z}_p|\) |
| Ours | \((m + 2n + 1)\cdot exp\) | \((m + 2n + 1)\cdot exp\) | \(m|\mathbb{G}| + (m + 2n + 1)|\mathbb{Z}_p|\) |

\(n\): The number of accounts required to construct one signature.
\(m\): The number of accounts of the sender.
\(\lambda\): The number of bits of the group element
\(|\mathbb{Z}_p|\): The length of element in \(\mathbb{Z}_p\), similarly in \(|\mathbb{G}|\)
\(\exp\): An exponentiation operation

References
[1] Nakamoto, S., & Bitcoin, A. (2008). A peer-to-peer electronic cash system. URL: https://bitcoin.org/bitcoin.pdf.
[2] Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J. A., & Felten, E. W. (2014). Mixcoin: Anonymity for bitcoin with accountable mixes. In: International Conference on Financial Cryptography and Data Security. Barbados. pp. 486-504.
[3] Maxwell, G. (2013). Coinjoin: Bitcoin privacy for the real world, 2013. URL: https://bitcointalk.org/index.php.
[4] Barber, S., Boyen, X., Shi, E., & Uzun, E. (2012). Bitter to better—how to make bitcoin a better currency. In: International Conference on Financial Cryptography and Data Security. Kralendijk, Bonaire, Sint Eustatius and Saba. pp. 399-414.
[5] Bissias, G., Ozisik, A. P., Levine, B. N., & Liberatore, M. (2014). Sybil-resistant mixing for bitcoin. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society. Scottsdale Arizona. pp. 149-158.
[6] Ruffing, T., Moreno-Sanchez, P., & Kate, A. (2014). Coinshuffle: Practical decentralized coin mixing for bitcoin. In: European Symposium on Research in Computer Security. Wroclaw. pp. 345-364.
[7] Miers, I., Garman, C., Green, M., & Rubin, A. D. (2013). Zerocoin: Anonymous distributed e-cash from bitcoin. In: 2013 IEEE Symposium on Security and Privacy. California. pp. 397-411.
[8] Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., & Virza, M. (2014). Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. California. pp. 459-474.
[9] Goldwasser, S., Micali, S., & Rackoff, C. (1989). The knowledge complexity of interactive proof systems. SIAM Journal on computing., 18(1): 186-208.
[10] Van Saberhagen, N. (2013). Cryptonote v 2.0. URL: https://cryptonote.org/whitepaper.pdf.
[11] Noether, S., & Mackenzie, A. (2016). Ring confidential transactions. Ledger., 1: 1-18.
[12] Maxwell, G. (2015). Confidential transactions. URL: https://people.xiph.org/greg/confidential values.txt
[13] Herranz, J., & Sáez, G. (2003). Forking lemmas for ring signature schemes. In: International Conference on Cryptology in India. Mumbai. pp. 266-279.
[14] W. Zhaoxia., Z Yingying. (2018). An Efficient Ring Signature without Bilinear Pairing. Journal of Natural Science Edition., 35(04):473-478.
[15] Liu, J. K., Wei, V. K., & Wong, D. S. (2004). Linkable spontaneous anonymous group signature for
ad hoc groups. In: Australasian Conference on Information Security and Privacy. Sydney. pp. 325-335.

[16] Bellare, M., & Neven, G. (2006). Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of the 13th ACM conference on Computer and communications security. London. pp. 390-399.

[17] Biryukov, A., & Pustogarov, I. (2015). Bitcoin over Tor isn't a good idea. In 2015 IEEE Symposium on Security and Privacy (pp. 122-134). IEEE.

[18] Goldfeder, S., Kalodner, H., Reisman, D., & Narayanan, A. (2018). When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. Proceedings on Privacy Enhancing Technologies., 2018(4), 179-199.

[19] Sun, S. F., Au, M. H., Liu, J. K., & Yuen, T. H. (2017). Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In: European Symposium on Research in Computer Security. Oslo. pp. 456-474.

[20] Fiat, A., & Shamir, A. (1986). How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques. Santa Barbara. pp. 186-194.

[21] Groth, J., & Kohlweiss, M. (2015). One-out-of-many proofs: Or how to leak a secret and spend a coin. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Sofia. pp. 253-280.

[22] Feige, U., Fiat, A., & Shamir, A. (1988). Zero-knowledge proofs of identity. Journal of cryptology., 1(2): 77-94.

[23] Camenisch, J., & Stadler, M. (1997). Efficient group signature schemes for large groups. In: Annual International Cryptology Conference. Santa Barbara. pp. 410-424.

[24] Chase, M., & Lysyanskaya, A. (2006, August). On signatures of knowledge. In: Annual International Cryptology Conference. Santa Barbara. pp. 78-96.