2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering

Making Monero Hard-to-Trace and More Efficient

Qingyi Liu[1,3], Zhen Liu ✉[1], Yu Long ✉[1], Zhiqiang Liu ✉[1], Zhimei Sui[2], Shifeng Sun ✉[1], Shuyang Tang[1], and Dawu Gu ✉

[1]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China
Email: {3773green, liuzhen, longyu, ilu_zq, shifeng.sun, htftsy, dwgu}@sjtu.edu.cn
[2]School of Computer Science and Software Engineering, East China Normal University, Shanghai, China
Email: zhimeisui@gmail.com
[3]Shanghai Viewsource Information Science& Technology Co.Ltd

***Abstract*—Most cryptocurrencies have successfully provided anonymity in a permissionless environment. However, the pattern of transfers is open to publicity. To face this issue, Monero was proposed to provide untraceability from ring signatures by introducing mixins to obfuscate addresses. By temporal analysis, however, the transfer pattern can still be partially revealed in a stochastic approach due to inappropriate selections of mixins. Thereby, each flow of coins can be traced with high probability which disobeys the untraceability principle of Monero. In this work, we propose a hard-to-trace protocol based on Monero where each transaction output is assembled into a fixed ring set. In this way, inappropriate mixins are forbidden, and thereby the temporal analysis is resisted. Apart from the traceability issue, Monero is also challenged due to its growing difficulty of block assembly. To guarantee the privacy, "key images" with a considerable size have to be stored by each miner to verify transactions and assemble blocks. As blockchain grows, the number of key images increases and a significant burden has already been caused, making the block assembly of Monero inefficient to most miners. Aimed at a more practical block assembly, our protocol allows key image truncations to facilitate transaction verifications.**

***Index Terms*—Blockchain, Privacy-Preserving, Untraceability, Temporal analysis**

I. INTRODUCTION

Bitcoin [1] opens up an era of cryptocurrency and has been the most successful and popular cryptocurrency. One reason for Bitcoin's success is its anonymity. Adopting graph analysis on the open blockchain of Bitcoin, however, its transaction pattern can be obtained easily, which compromises its anonymity greatly [2]. To solve such a privacy issue in Bitcoin, some anonymous cryptocurrencies were proposed like Monero [3], [4], Zcash [5], [6] and Dash [7]. Among them, the total market value of Monero ranks the highest (841 million dollars by February 19, 2019).

Still, Monero faces some attacks [8]–[13], which expose the inputs and outputs of transactions and thereby weaken the privacy. In this paper, we propose a protocol to solve such a privacy threat in Monero.

A. Monero

Like all cryptocurrencies, Monero is an anonymous distributed ledger with a serialized log of transactions stored in a chain of blocks (the *blockchain*) which grows by time. The extension of the log is realized by the assembly of new blocks performed by miners (new transactions are verified by miners before recorded into a block). The transaction in Monero consists of inputs and outputs. Eash transaction output has a one-time address, which can be regarded as a public key randomly generated according to the long-term key of the receiver whose corresponding private key is obtainable to the receiver [14], [15]. The transaction input has multiple input addresses, each of which links to an output of some previous transactions.

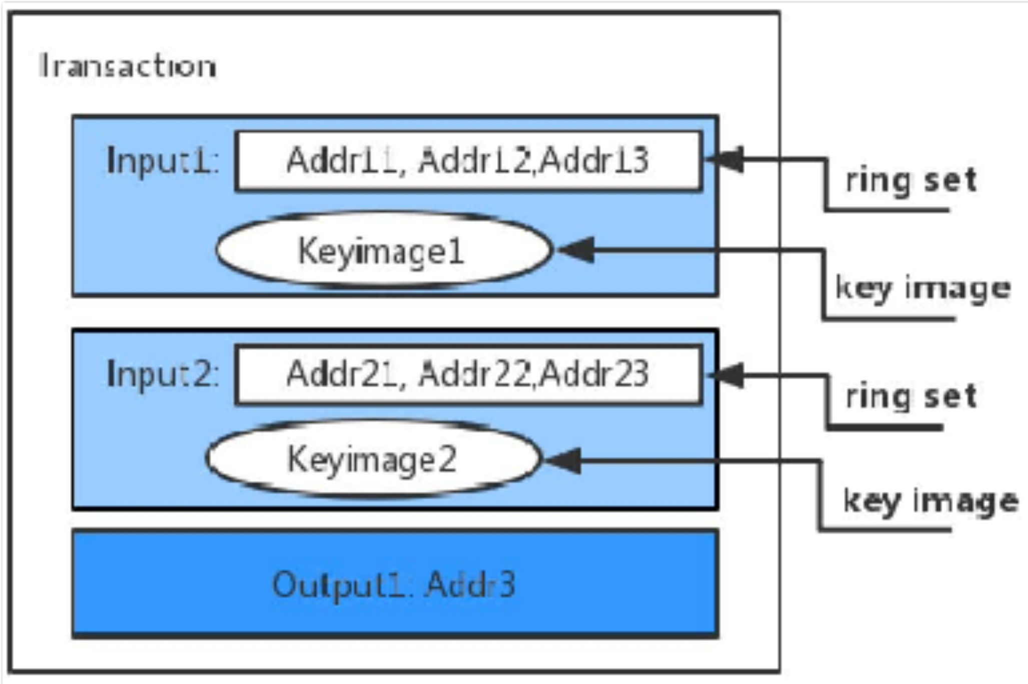

Fig. 1. A simplified diagram of the transaction in Monero

Although a transaction input has multiple input addresses, only one (the *real input*) is actually spent in the transaction and the rest, referred to as mixins of the real input, are used to hide the real input. In this paper, we call the ring set as all the addresses in an input. According to each ring set, a ring signature is generated for each transaction input [**?**], [16], [17]. In this way, Monero achieves untraceability of currency transfers since it is hard to distinguish the real input from mixins. An example is provided in Fig. 1 where the transaction has two inputs and one output. In $input_1$, the real input is hidden among three addresses–$\langle Addr_{11}, Addr_{12}, Addr_{13}\rangle$. However, such an untraceability makes double-spending hard to detect. To face this issue, for each input, a unique key image, which is generated according to the one-time private key of the real input, is provided to prove that the input has not been spent in the past [14]. Also mention about Monero uses linikable ring signature [18]–[21] as the underlying cryptographic primitives which can be used to provide linkable anonymity for various applications such as e-voting [22] or cryptocurrency such

2324-9013/19/$31.00 ©2019 IEEE
DOI 10.1109/TrustCom/BigDataSE.2019.00075

as Ring Confidential Transaction (RingCT) for blockchain protocol [17].

B. Privacy issues in Monero

In general, the larger the number of mixins, the stronger the untraceability is. However, mixins are chosen by users and thereby improperly-selected mixins can easily compromise the untraceability.

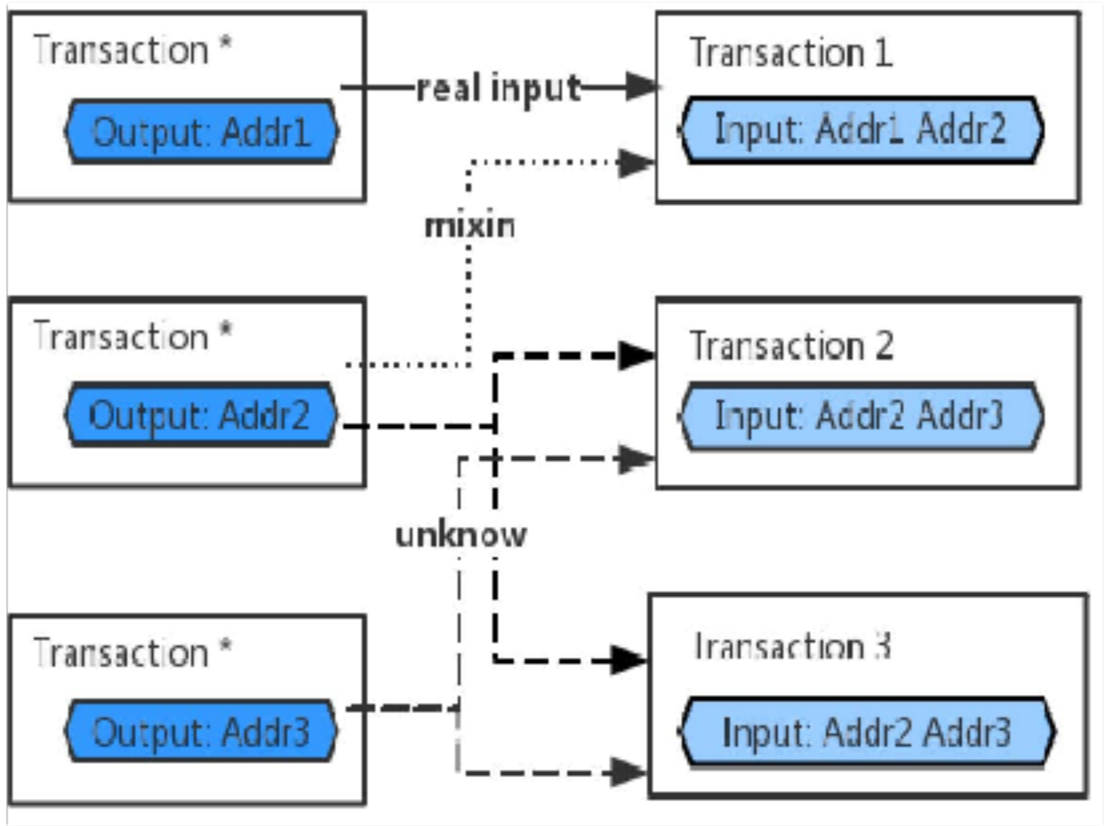

Fig. 2. An example of choosing mixins inappropriately in Monero

For example, as shown in Fig. 2, there are three transactions with one input. Transaction 1 (tx_1) has the input $\langle Addr_1, Addr_2\rangle$. Transaction 2 ($tx_2$) has the input $\langle Addr_2, Addr_3\rangle$. Transaction 3 ($tx_3$) has the input $\langle Addr_2, Addr_3\rangle$. From tx_2 and tx_3, $Addr_2$ and $Addr_3$ must be the real input of tx_2 or tx_3. Therefore, $Addr_2$ must not be the real input of tx_1. So the real input of tx_1 must be $Addr_1$. In this way, one transfer is traced and hence the pattern of currency transfer is partially revealed. This is a drawback of Monero, and there is no way for users to ensure that their transactions are untraceable, as the hard-to-trace property of any flow of currency also depends on transactions of others. This is an inevitable issue caused by the spontaneity of choosing mixins in Monero.

As a more systematic approach of currency tracing, temporal analysis [8] has been proposed in 2017, posing a greater challenge to the untraceability of Monero. As shown in [8], the real input in the input addresses of a transaction input is often the one with the smallest spend-time[1]. In fact, the real inputs of more than 90% of transactions can be deduced by temporal analysis, which seriously threatens the privacy of Monero.

To resist temporal analysis, Malte Möser et al. [9] proposed two countermeasures, namely, spend-time distribution and binned mixin sampling.

Spend-time distribution estimates the spend-time of the addresses of all transaction inputs in the blockchain to get the distribution of transaction inputs' spend-time. Users choose mixins following the above distribution to hide their real inputs. However, the sampling distribution is highly dependent on the sample and can not ensure real-time updates. In addition, since it is not guaranteed that all users use the same sampling distribution, some groups of users with their unique distribution can be exposed to the adversary.

Binned mixin sampling groups the transaction outputs in the same block or the next block into a fixed-size set, called *bins*. In this way, the temporal analysis is resisted. It takes into account the fact that miners may be malicious. In order to prevent malicious miners harming the system, transaction outputs within the same block will be shuffled.[2] In this case, miners can not control the order of the transaction outputs. However, there is a worse situation. If a malicious miner only packages the target transaction and lots of his/her own transactions, the target transaction outputs must be in a bin with the malicious miner's transaction outputs even if there is a shuffling procedure. The attacked user will immediately be exposed when the target transaction output is spent. Ordinary users cannot identify whether a miner packages his/her own transactions to surround the user's transaction. This is a weakness for the binned mixin sampling procedure.

C. Our Contribution

In this paper, we propose a practical protocol that avoids weakening the privacy of the transaction due to choosing mixins inappropriately and resists temporal analysis completely. The ring sets in our protocol are assigned by the rules instead of chosen by users and the ring sets are also fixed, which means that if $Addr_A$ is the mixin of $Addr_B$, then $Addr_B$ must be the mixin of $Addr_A$. In this way, inappropriate selection of mixins is forbidden, therefore, the anonymity of a transaction cannot be reduced. Although the ring set is fixed, the block head hash is used to shuffle the outputs, so nobody can control the members of a ring set. In addition, the addresses in a ring set come from blocks in the same time period. The adversary can not analyze which one is the real input through their spend-time.

Furthermore, our protocol can make the verification of transactions more efficient. To verify whether a transaction is valid, for each transaction input, a miner needs to check whether the addresses in the ring set are referred to existing transaction outputs, as well as, whether the key image has appeared previously. In order to verify a transaction quickly, Monero miners need to load all the outputs and all the key images on the blockchain into the memory. For miners, both key images and outputs are continuously growing, the memory growth rate is unbearable and the verification process is too time-consuming, which is very terrible for the expansion and performance of Monero. The protocol in this paper breaks the original rules of verifying transactions so that miners do not need to load all the key images and outputs. It greatly reduces the demand for miners' memory and speeds up verification of transactions.

---

[1] The spend-time is the minus of send-time (the height of the block in which it appears as an input) and receive-time (the height of the block in which an address appears as an output).

[2] If the number of transaction outputs is not divisible, the remainder transaction outputs in a bin will come from the next block.

II. OUR PROTOCOL

A. *Overview*

In order to solve the privacy issues of Monero discussed in Section. I-C, we propose a protocol that two rules should be followed. First, to avoid inappropriate and invalid selection of mixins, the mixins of an output should also be the mixin of this output. Second, to resist the temporal analysis, the members of a ring set should come from the blocks in the same time period.

Our protocol, in general, is a way to assign the fixed ring set by the rules. It breaks the original spontaneous and uncontrolled mixing law and adopts the form of fixed matching. The main idea of our protocol is to divide the transaction outputs in the consecutive N blocks into a number of fixed mixin sets as the size of k. The reason for the consecutive N blocks is that it can avoid malicious miners controlling the members in a ring set. With large enough N, nobody can control the blocks consecutively, otherwise, Monero will face 51% attack under the majority assumption.

Generally, the larger the ring set, the harder it is for the adversary to determine which is the real input. In current Monero, such ring set is constantly changing. Users choose mixins spontaneously without considering whether others also choose their outputs as the mixins. This changeable ring set can cause traceability of transactions. In this paper, the ring set is fixed, which guarantees the effectiveness of the mixins and avoids users affecting each other because of their spontaneity of choosing mixins.

In this paper, the transaction outputs in a block are divided into N parts. For the consecutive N blocks, one part is selected from each block to form a group. As shown in Fig. 3, the dotted box represents a group. The transaction outputs in the same block but different parts will be included in different groups. In order to ensure that the division fully covers every transaction output of each block, the composition of groups is like a sliding window, and any group is formed by the parts of the consecutive N blocks just before it.

It is still too many transaction outputs in a group, and they should be divided into a smaller group —- a ring set. In current Monero, the ring set is chosen by the user, while in this paper, the ring set is assigned by the rules. From a privacy perspective, even if the ring set is determined, it still should not be controlled by anyone. Due to PoW(Proof of Work [1]), block head hash is a number that cannot be easily controlled by miners. When a new block is created, the addresses in a group of previous consecutive N blocks are shuffled by the block head hash and then divided into many ring sets.

Let k be the size of the ring set and N be the number of blocks in a time period, where k and N are system parameters and can be adjusted according to the actual situation.

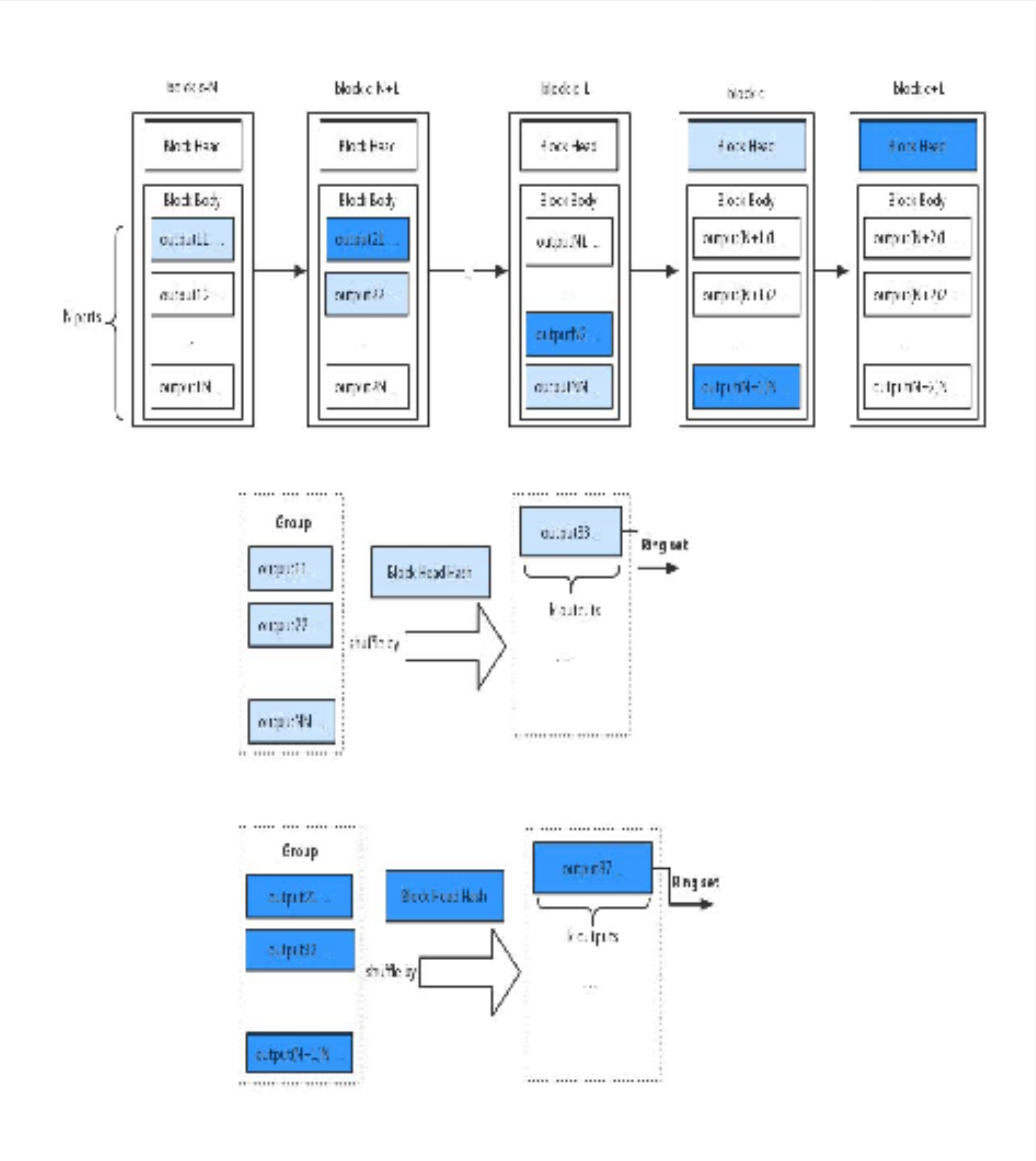

Fig. 3. Overview of our protocol

As Fig. 3 shows, the transaction outputs in a block are divided into N parts. Take $block_{c-N}$ for example, the transaction outputs are divided into N parts: $\langle part_1 : op_{11}, op...\rangle$, $\langle part_2 : op_{12}, op...\rangle$, ..., $\langle part_N : op_{1N}, op...\rangle$. The light blue parts from $block_{c-N}$ to $block_{c-1}$ are gathered together to form a group, which should be shuffled by the light blue block head hash($block_c$), and finally, be divided into lots of ring sets. Here is a point that needs attention —- if a transaction has multiple outputs, they should be regarded as separate individuals, with large probabilities being assigned to different ring sets.

B. *Protocol Run by Miners*

In the Monero network, there are two types of nodes, one is the miner, and the other is the ordinary user. Different from ordinary users, miners should verify the transactions and package the transactions into a block. Then, miners should make a block to be valid through the PoW. In general, a block consists of a block head and a block body. The transactions are contained in the block body in the form of a Merkle tree [1]. The Merkle tree's root is stored in the block head to ensure the integrity and unforgeability of the information in the block body. Miners are free to choose transactions and decide the locations of these transactions in the Merkle tree.

In our protocol, when miners are creating a block, one more thing should be done compared with the current Monero, that is, divide the transaction outputs in this block into N parts. At first, miners should calculate the sequence number for every transaction output in the block. According to the Merkle tree and outputs' order in a transaction, all transaction outputs within a block can be given a sequence number. Once the Merkle tree is constructed, the sequence numbers of the transaction outputs in this Merkle tree are determined.

Algorithm 1 Calculate sequence numbers for transaction outputs

```
Input:
    The complete Merkle Tree of a block, $MT$;
Output:
    The set of transaction outputs with a sequence number, $SN$;
from left to right, push all leaf nodes(transactions) from $MT$ into a queue $TX$;
initialize the sequence number $s = 0$;
for each $tx_i \in TX$ do
    for each $op_j \in tx_i$ do
        $SN[s] = op_j$;
        $s = s + 1$;
    end for
end for
return $SN$;
```

Through Algorithm 1, the sequence numbers for a given block are calculated, which is determined by the Merkle tree in the block body. In order to make the division of transaction outputs in a block unique, SN is of great significance and everyone can run the Algorithm 1 to get the SN of any block in Monero blockchain.

After obtaining the sequence numbers of outputs, miners divide outputs into N parts, which should follow the rules below.

a). The difference in the number of outputs per part is less than 1 (except $part_N$).

b). Adjust the size of $part_N$ to ensure that the size of a group is an integer multiple of k.

c). The size of $No.N$ part is at most k less than other parts.

d). If $i < j$, the sequence number of transaction output in $part_i$ is smaller than $part_j$.

e). If $i < j$, the size of $part_i$ is larger than $part_j$.

f). The parameters meet the following conditions: N is larger than k, $N(k-1)$ is larger than the number of transaction outputs in a block.

Following the above seven rules, the division of a given block is determined.

Through Algorithm 2, the block has been divided into the N parts that are roughly the same size. The result of Algorithm 2 —- DI is recorded in the block head in order to facilitate other miners and users to figure out which part the transaction outputs are divided into. DI is an array of $N+1$ integers, which do not take up too much space. $DI[0]$ is 1 and $DI[i] - DI[i-1]$ is the size of $part_i$.

In addition to dividing the outputs in the current block, miners should gather the parts of previous consecutive N blocks into a group. If $blockid > N$, miners do the gather procedure. If $blockid \leq N$, there are not consecutive N blocks before current block, so miners cannot do the gather procedure.

Algorithm 2 Division for transaction outputs

```
Input:
    The set of transaction outputs with a sequence number, $SN$;
    The size of ring set, $k$;
    The number of parts in a block, $N$;
    Divisions of $N-1$ blocks before the current block, $DI_{c-N+1}, DI_{c-N+2}, ..., DI_{c-1}$;
Output:
    Division for all the transaction outputs in the current block, $DI_c$;
initialize a array of size $N$: $AR = \{0\}$
initialize an integer: $j = 0$;
for each $op_i \in SN$ do
    $AR[j] = AR[j] + 1$;
    $j = (j+1) \mod N$;
end for
initialize $cnt = AR[N-1]$;
for $n = 1$ to $N-1$ do
    $cnt = cnt + (DI_{c-n}[N-n] - DI_{c-n}[N-n-1])$;
end for
$re = cnt \mod k$;
$AR[N-1] = AR[N-1] - re$;
for $t = 1$ to $re$ do
    $AR[j] = AR[j] + 1$;
    $j = (j+1) \mod N$;
end for
$DI_c[0] = 1$;
for $n = 1$ to $N$ do
    $DI_c[n] = DI_c[n-1] + AR[n-1]$;
end for
return $DI_c$;
```

Considering the initial implementation of our protocol, some transaction outputs of the initial N blocks cannot be covered. One solution is to let that the first block has only one part and the second block has two parts and so on. Therefore, there are no uncovered transaction outputs. The other solution is to gather all of these uncovered transaction outputs into a big group and shuffle them by the $N-1$th block head hash. There is a problem that the size of the big group is not a multiple of k. In this case, the miner of the $N-1$th block need to carefully package the transactions and construct the Merkle tree, ensuring that the total number of the big group is a multiple of k.

After calculating the division of a new block and record the division in the block head, the block has been assembled. Then, miners can do the PoW to obtain a legal block. After that, the new block should be broadcasted to the network and then be verified by other miners of Monero. After the verification, the block has been accepted by all. Then, the new block head hash should be used to shuffle the transaction outputs in the previous consecutive N blocks. Before the shuffling procedure, the group to be shuffled should be gathered from the N blocks.

Algorithm 3 Gather the parts of previous N blocks

```
Input:
    The sets of transaction outputs with a sequence number in previous $N$ blocks, $SN_{c-N}, SN_{c-N+1}, ..., SN_{c-1}$;
    Division of $N$ blocks before the current block, $DI_{c-N}, DI_{c-N+1}, ..., DI_{c-1}$;
Output:
    The transaction output group, $GR_c$;
initialize $GR_c = \{\}$;
for $i = 1$ to $N$ do
    for each $op_j \in SN_{c-i}[DI_{c-1}[N-i]-1, DI_{c-1}[N+1-i]-1]$ do
        add $op_j$ into $GR_c$;
    end for
end for
return $GR_c$;
```

Through Algorithm 3, miners get a newly-formed group, and then miners should do the shuffling procedure. The block head hash is introduced as a random factor that shuffles the transaction outputs in a group. The block head hash is a value that cannot be arbitrarily modified by miners so that it is suitable as a parameter for the shuffling procedure.

Algorithm 4 Shuffle the transaction outputs and divide the outputs into ring sets

```
Input:
    The transaction output group, $GR_c$;
    The current block head hash, $BHH_c$;
    The size of ring set, $k$;
Output:
    The set of ring sets, $RS_c$;
for each $op_i \in GR_c$ do
    $h_i = H\,(BHH_c \,||\, op_i\ public\ key)$, where $H$ is a hash function;
end for
sort $RS_c$ from small to large of the value $h_i$;
initialize $rs = \{\}$;
for each $op_i \in GR_c$ do
    add $op_i$ into $rs$;
    if the size of $rs$ is equal to $k$ then
        add $rs$ into $RS_C$;
        set $rs = \{\}$;
    end if
end for
return $RS_c$;
```

Through Algorithm 4, miners finally obtain many newly-formed ring sets. Next, we will discuss how miners do to make the verification of transactions more efficient.

To verify a block, miners should verify the transactions in this block. When miners verify a transaction, they should verify that each input is a legal input. The verification of a transaction input is mainly divided into three steps:

a). Check whether each address in the transaction input is valid.[3]

b). Check whether the key image has appeared in the transactions of previous blocks. If the key image has appeared, the transaction input cannot be accepted.

c). Verify the signature of the transaction input.

In our protocol, miners maintain an unspent ring set list, which is different from what miners do in current Monero. As the Fig. 4 shows, the unspent ring set has three data segments which are $counter$, $keyImages$ and $ring\ set$. When a new transaction is successfully verified and put on the blockchain, the $counter$ of its input's unspent ring set adds one and the key image of this transaction input is recorded in the $keyImages$. When the $counter$ is equal to k, this unspent ring set should be removed from the list. The role of $counter$ records the number that this ring set appears as an input, and the role of $keyImages$ is to prevent double spending.

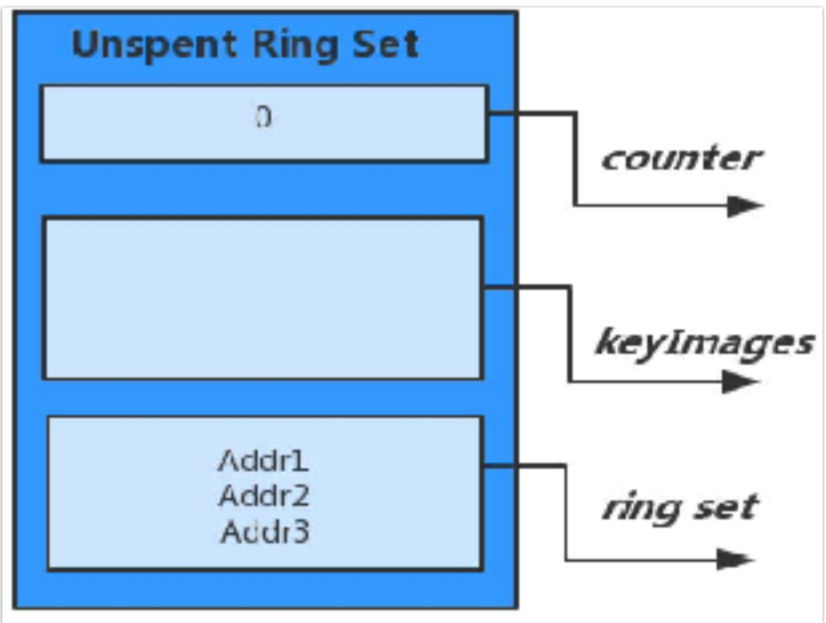

Fig. 4. The data structure of unspent ring set

For a given valid new block, miners should do the following two operations:

- *Initialize:* According to the Algorithm 4, miners obtain many newly-formed ring sets that should be inserted into the unspent ring set list. Initialize $counter = 0$, $keyImages$ is empty and $ring\ set$ is one of the newly-formed ring sets.
- *Update:* For each transaction input in the new block, miners should add one to the $counter$ of the related unspent ring set, and record the key image of the input in the $keyImages$. When a $counter$ reaches the upper limit, ie the size of the ring set, this unspent ring set should be removed from the list.

With the unspent ring set list, miners can verify a transaction easily. For a transaction input, miners do the following two steps.

a). Check whether the ring set of the input is in the list or not. Every input addresses should be the same with the addresses in $ring\ set$.

b). Check whether the key image of the input is in the $keyImages$ or not.

In this way, miners save space and time. First, miners no longer need to load all key images and transaction outputs into

---

[3] A valid input must be the transaction output in the previous blocks.

the memory. Now, miners only need to maintain an unspent ring set list, which size is almost constant by the $Initialize$ and $Update$ operations. Second, miners used to search all the transaction outputs to verify that the transaction input is valid and all the key images to make sure the key image of the transaction has never appeared in the previous blocks. Now, the miner only needs to search for the unspent ring set first, which is to verify that the transaction input is valid, and then verify that the key image does not appear in the $keyImages$, which takes less than half of the search time.

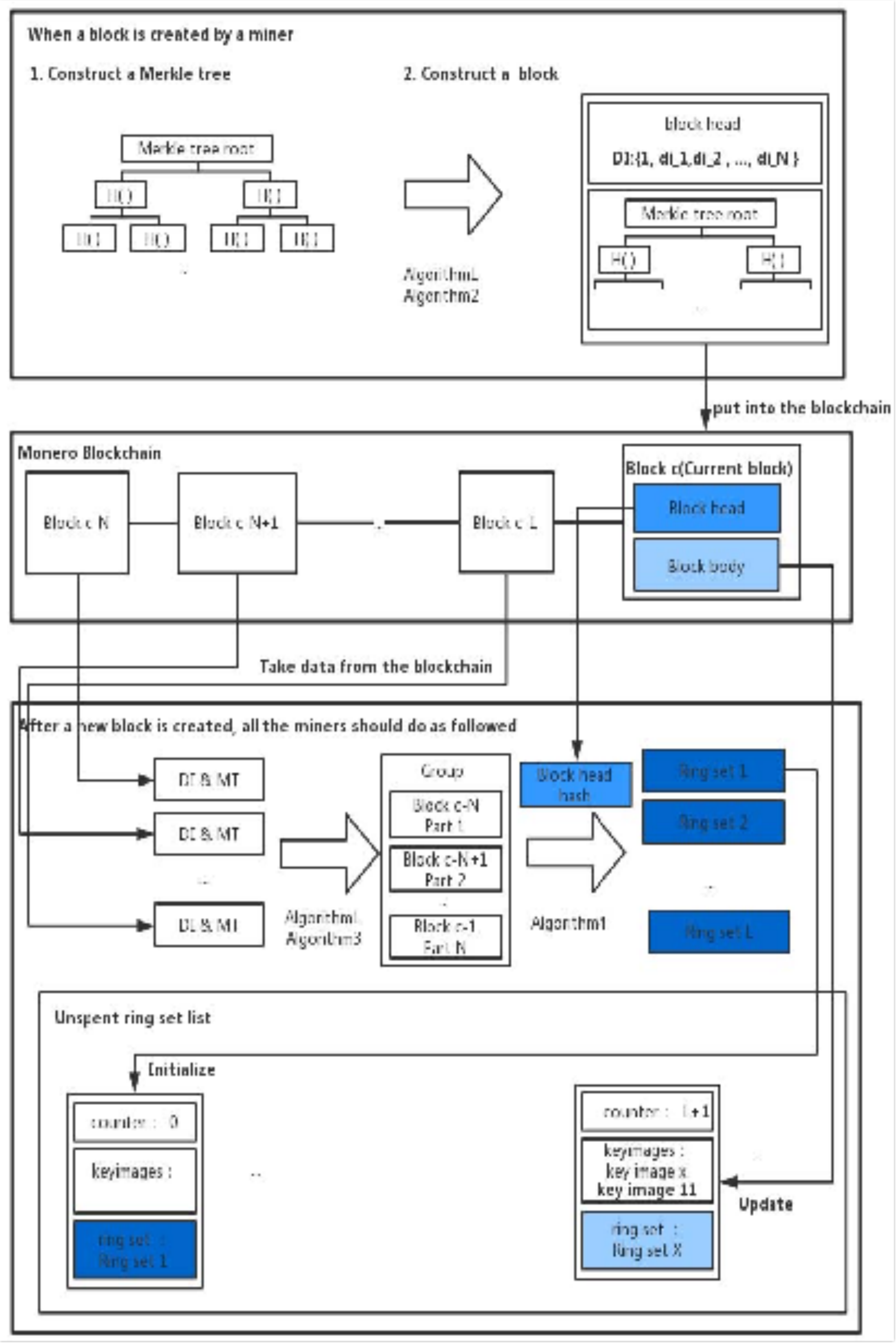

Fig. 5. The total flow diagram of miners

In conclusion, as Fig. 5 shows, before a block is created, the miner should run the Algorithm 1&2 to calculate the division of the transaction outputs in this block. After the block has been put into the blockchain, all the miners should run the Algorithm 1&3&4 to get the ring sets of previous consecutive N blocks and then initialize the unspent ring sets into the list. Moreover, miners should update the unspent ring sets according to the transaction inputs that are spent in the current block.

C. Protocol Run by Ordinary Users

Ordinary users only care about their own transactions. When a user receives a transaction output, if the user wants to create a transaction to spend the output, he/she should run the protocol to figure out which ring set its output belongs to. Suppose the user's transaction output is op_{user}, and the block that the user's transaction is in is $block_u$. $Part_i^j$ is the $part_i$ of $block_j$.

Algorithm 5 Calculate the ring set of a transaction output

```
Input:
    The Monero blockchain, $B_{chain}$;
    The user's transaction, $op_{user}$;
    The block id that user's transaction is in, $u$;
Output:
    The ring set of a transaction output, $RS_{user}$;
$MT_u \leftarrow (B_{chain}, u)$;
from left to right, push all leaf nodes(transactions) from $MT_u$ into a queue $TX_u$;
initialize the sequence number $s = 0$;
for each $tx_i \in TX_u$ do
    for each $op_j \in tX_i$ do
        if $op_j = op_{user}$ then
            break;
        else
            $s = s + 1$;
        end if
    end for
end for
$DI_u \leftarrow (B_{chain}, u)$;
initialize $p = 0$; ($p$ is the partid of $op_{user}$)
for each $j \in DI_u$ do
    if $s \geq j$ then
        break;
    else
        $p = p + 1$;
    end if
end for
calculate $\left(Part_1^{u-p+1}, ..., Part_p^u, ..., Part_N^{u-p+N}\right)$ by Algorithm 1&2;
$group_{user} = \left(Part_1^{u-p+1}, ..., Part_p^u, ..., Part_N^{u-p+N}\right)$;
calculate $RS$ with $group_{user}$ and $block_{u-p+N+1}$ $head$ $hash$ by Algorithm 4;
for each $re \in RS$ do
    for each $i \in re$ do
        if $i = op_{user}$ then
            $RS_{user} = re$;
        end if
    end for
end for
return $RS_{user}$;
```

As Fig. 6 shows, users need to wait for $N-p+1$ blocks to be able to spend the coins they have already received. However, the delay is insignificant, because users can still confirm their transactions timely. It doesn't matter if you can be sure to receive the coins and wait for a while to spend it. Moreover, assuming N is 50, the maximum waiting time for users is 100 minutes, which is a short delay.

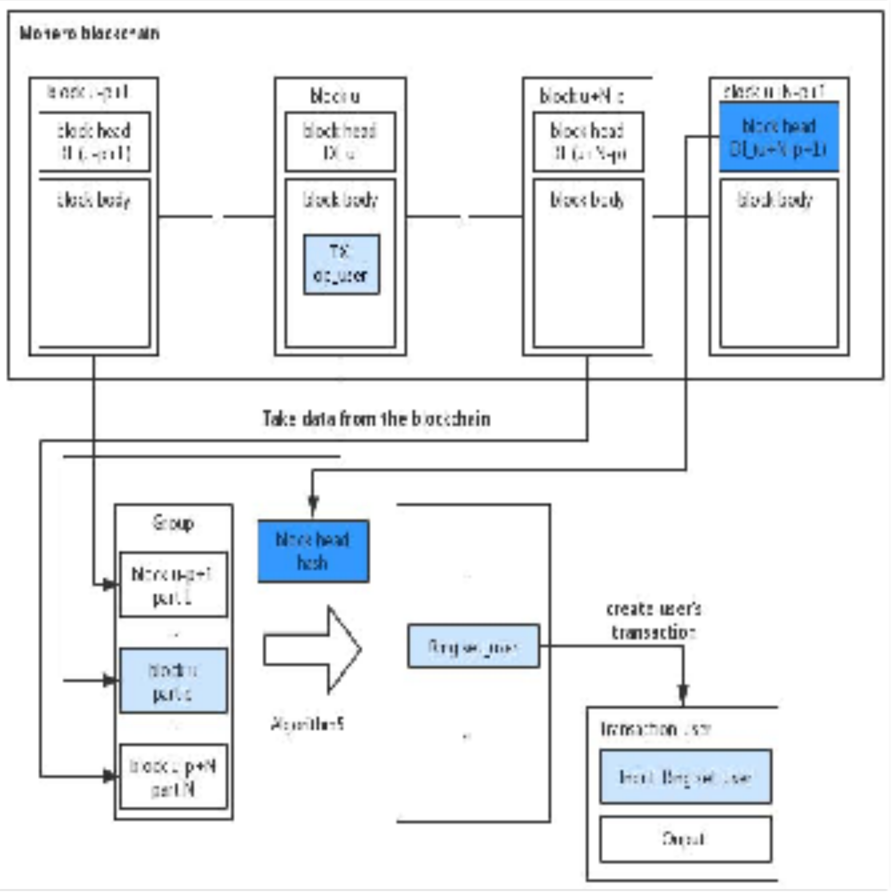

Fig. 6. The total flow diagram of ordinary users

III. SECURITY ANALYSIS

A. Analysis of the Untraceability

Untraceability is one of the most important characteristics in Monero. Ring signature is used to hide the real input and thereby Monero achieves the untraceability of transactions. However, since choosing mixins is a completely spontaneous procedure for users, it can cause many privacy risks. The most severe risks are the improper selection of mixins and temporal analysis mentioned in section one.

Our protocol enhances the untraceability in Monero. First, there is no improper selection of mixins. Our protocol makes the ring set fixed that the addresses in a ring set always appear in the same transaction input at the same time. In this way, users never have to worry about the improper selection of others. Second, our protocol can completely resist temporal analysis, because the addresses in a ring set must come from the adjacent N blocks. The spend-time of the addresses in an input have almost no difference. The adversary can not deduce which is the real input from the spent-time .

In conclusion, the protocol in this paper can deal with the current two types of traceable attacks against Monero. Compared with the current Monero, our protocol has better privacy.

B. Attack and security analysis

Miners play a vital role in the protocol of this paper. Transactions in the block are packaged by the miners, which means that the miners are free to package transactions into the block. If there are malicious miners, they do not package transactions in the normal order, which may cause certain damage to the privacy of our protocol. However, if malicious miners do not pay a considerable price on computing power, it is difficult to attack our protocol.

Due to the shuffling procedure of the transaction outputs, the only way for an adversary to attack the protocol is to control more transactions in the consequent N blocks, which means that the adversary should make its own transactions outputs in the consequent N blocks as much as possible. The more transactions the adversary controls, the worse the privacy of the transactions.

In this paper, the method of choosing mixins is fixed. When the output of a certain transaction is spent, if the adversary can control all the outputs except this output in the fixed ring set, the adversary can trace this output. When this output is spent, although it has $k-1$ mixins, the adversary can still know this output is the real input of the transaction, since the rest are controlled by the adversary. However, through probability calculations, the probability that an adversary can control all remaining outputs in a ring set is small. Besides, the adversary needs to pay a lot of computing power and obtain no transaction fees.

The range in which an adversary can attack is small, only the transaction outputs in the latest $N-1$ blocks. Because the ring sets of the transaction outputs in the block before the latest $N-1$ block have been determined and no one can change it unless someone can implement a 51% attack. The adversary has no way to attack the transaction output with the determined ring set. As more and more blocks are created and follow a target block, the probability of attacking transactions in this target block is getting smaller and smaller.

We define the model of the attack, assuming that the purpose of the adversary is to determine the real input of a transaction, in other words, to achieve traceability of the transaction. Assume that the adversary's computing power is q, and the honest miners' computing power is p, which satisfies $q+p=1$. Set the maximum capacity of a block to M and the average size of a transaction to m. The average number of transactions that a block has on average is t, which satisfies $M = tm$. Suppose each transaction has an average of two outputs, one for payment and one for change. There is an average of W transaction outputs in each block and $W = 2t$. Without considering economic incentives, we assume that the block created by the adversary contains only its own transactions, which allows the adversary to gain maximum attack power. We use the size of the anonymous set to measure the privacy of transactions. The initial size of the anonymous set is k. If the adversary controls one address in the ring set, the size of the anonymous set reduces 1. If the adversary controls $k-1$ addresses in the ring set, the adversary attacks the target output successfully.

According to the above model, the probability that the adversary can attack one target output successfully can be calculated.

The probability that the anonymous set reduces i ($i < k-1$) is:

$$\frac{\binom{qNW}{i}\binom{pNW-1}{k-1-i}}{\binom{NW-1}{k-1}}$$

The probability that the anonymous set reduces to 1 is:

$$\frac{\binom{qNW}{k-1}}{\binom{NW-1}{k-1}}$$

When $N = 50$, $M = 2MB$, $m = 1kB$, $t = 2000$, $W =$

4000 , the probability that the adversary can attack one output successfully is as followed:

TABLE I
ATTACK THRESHOLD

	q=0.05	**q=0.1**	**q=0.15**	**q=0.2**
k=5	6.3×10^{-6}	1.0×10^{-4}	5.1×10^{-4}	1.6×10^{-3}
k=10	2.0×10^{-12}	1.0×10^{-9}	3.9×10^{-8}	5.1×10^{-7}
k=15	6.1×10^{-19}	1.0×10^{-14}	3.0×10^{-12}	1.6×10^{-10}
k=20	1.9×10^{-25}	1.0×10^{-19}	2.5×10^{-16}	5.2×10^{-14}

As can be seen from table. I, the probability that the anonymous set reduces to 1 is very small. As k increases, the probability that an adversary can attack successfully is almost negligible. But considering that the larger k is, the larger the size of the transaction, the more transaction fees the users need to pay, we can choose a suitable value of k.

From the above probability formula, if the computing power ratio of the adversary is constant when N is small, the larger N is, the smaller the probability of success is, but when N increases to a certain value, the increase of N hardly affects the probability. A value of 50 is a good choice for N.

In conclusion, the protocol with fixed ring sets in this paper is more untraceable and has better privacy. Its security is guaranteed by the limited computing power of the adversary. Moreover, if the user thinks that k is too small to protect his/her privacy, the user can use some intermediate addresses to generate a series of transactions, so that the privacy is exponentially rising.

IV. CONCLUSION

This paper introduces a hard-to-trace and efficient Monero-based distributed consensus from fixed ring sets. In all, apart from privacy properties inherited from Monero, our scheme has, in addition, solved two issues. Firstly, by fixed ring sets, it resists transfer pattern leakage caused by improperly mixin selections or temporal analysis. secondly, it speeds up transaction verification, and hence block assembly, via key image truncations. To discuss the security of our scheme, we show that the probability of malicious miner determining the members of a ring set is negligible. The methodology behind this scheme is applicable not only in Monero but also in related cryptocurrencies with currency flow obfuscation from ring signatures.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their helpful feedback. The authors are supported by the National Natural Science Foundation of China(Grant No.61672339, 61672347, 61572318, 61872142), the National Crytography Development Fund(No.MMJJ20170111), Minhang Technology Innovation Program for SMEs, finance business platform based on blockchain technology(2018MH110).

REFERENCES

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2009. https://bitcoin.org/bitcoin.pdf.

[2] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 104–121, 2015.

[3] Monero. https://getmonero.org.

[4] Monero research lab. https://ww.getmonero.org/resources/research-lab.

[5] Zcash. https://z.cash.

[6] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014.

[7] Dash. https://www.dash.org.

[8] Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. A traceability analysis of monero's blockchain. In *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II*, pages 153–173, 2017.

[9] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the monero blockchain. *PoPETs*, 2018(3):143–163, 2018.

[10] Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, and Dongxi Liu. Monero ring attack: Recreating zero mixin transaction effect. In *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications / 12th IEEE International Conference On Big Data Science And Engineering, TrustCom/BigDataSE 2018, New York, NY, USA, August 1-3, 2018*, pages 1196–1201, 2018.

[11] Kevin Lee and Andrew Miller. Authenticated data structures for privacy-preserving monero light clients. In *2018 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2018, London, United Kingdom, April 23-27, 2018*, pages 20–28, 2018.

[12] Abraham Hinteregger and Bernhard Haslhofer. An empirical analysis of monero cross-chain traceability. *CoRR*, abs/1812.02808, 2018.

[13] Andrew Miller, Malte Möser, Kevin Lee, and Arvind Narayanan. An empirical analysis of linkability in the monero blockchain. *CoRR*, abs/1704.04299, 2017.

[14] Cryptonote technology. https://cryptonote.org/inside.

[15] Christopher D. Clack and Nicolas T. Courtois. Distributed ledger privacy: Ring signatures, möbius and cryptonote. *CoRR*, abs/1902.02609, 2019.

[16] Shen Noether. Ring signature confidential transactions for monero. *IACR Cryptology ePrint Archive*, 2015:1098, 2015.

[17] Shifeng Sun, Man Ho Au, Joseph K. Liu, and Tsz Hon Yuen. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II*, pages 456–474, 2017.

[18] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*, pages 325–335, 2004.

[19] Joseph K. Liu and Duncan S. Wong. Linkable ring signatures: Security models and new schemes. In *Computational Science and Its Applications - ICCSA 2005, International Conference, Singapore, May 9-12, 2005, Proceedings, Part II*, pages 614–623, 2005.

[20] Tsz Hon Yuen, Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Efficient linkable and/or threshold ring signature without random oracles. *Comput. J.*, 56(4):407–421, 2013.

[21] Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Linkable ring signature with unconditional anonymity. *IEEE Trans. Knowl. Data Eng.*, 26(1):157–165, 2014.

[22] Sherman S. M. Chow, Joseph K. Liu, and Duncan S. Wong. Robust receipt-free election system with ballot secrecy and verifiability. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*, 2008.