A Mixing Scheme Using a Decentralized Signature Protocol for Privacy Protection in Bitcoin Blockchain

Ruiyang Xiao, Wei Ren, *IEEE,* Tianqing Zhu, *IEEE,* and Kim-Kwang Raymond Choo, *IEEE*

***Abstract*—Bitcoin transactions are not truly anonymous as an attacker can attempt to reveal a user's private information by tracing related transactions. Existing approaches to protect privacy (e.g. mixcoin, shuffle, and blinded token) suffer from a number of limitations. For example, some approaches assume the existence of a trusted third party, rely on exchanges among various currencies, or broadcast sensitive details before mixing. Therefore, there is a real risk of privacy breach or losing tokens. Thus in this paper, we design a mixing scheme with one decentralized signature protocol, which does not rely on a third party or require a transaction fee. Specifically, our scheme uses a negotiation process to guarantee transaction details, which is monitored by the participants. Furthermore, the scheme includes a signature protocol based on the ElGamal signature protocol and secret sharing. The proposed scheme is then proven secure.**

***Index Terms*—Blockchain, privacy protection, coin mixing, multi-party signature.**

I. Introduction

BITCOIN has been known to be exploited by criminals, for example for ransomware payment [1], [2]. This is partly fulled by the fact or belief that Bitcoin transactions are anonymous or impossible to trace, since one does not necessarily need to use his/her real identity to create an account. However, such a belief or perception is not truly accurate, as each Bitcoin transaction is linked to at least one other transaction in the previous block and all transactions in the Bitcoin blockchain can be traced back to their origin. Even if these anonymous addresses are not linked to real identities, attackers can deduce from a user's transaction network [3], [4] and infer the user's or owner's real identities using techniques such as clustering analysis [5], [6]. In addition, vulnerabilities in a Bitcoin wallet can be exploited to recover either the user's identity or gain access to the Bitcoins stored in the wallet [7], [8]. Thus, another individual (e.g. attacker or investigator) can trace existing Bitcoin transactions in the blockchain to determine a particular user's complete addresses, their transaction network, and infer their real identities (also using other available information).

Hence, several privacy-enhancing technologies have been developed and such solutions can be broadly categorized into the following:

1) Solutions that avoid having a real link between input and output addresses in a transaction but require advanced cryptographic technologies (e.g, coinjoin [9], [10], coin shuffle [11] or blinded token [12], [13]).
2) Solutions that split the relationship between several different addresses of a user but require the use of additional currencies (e.g. Altcoin [3] and Zerocash[14]).
3) Solutions that prevent the tracing of user's transactions.

In addition to these mixing approaches, a number of third-party entities provide paid mixing services (e.g. Bitcoin Fog [15], BitLaundry [16], Dark Wallet [17] and Bitmixer [18]), although there are known limitations in such services (e.g. delayed transaction, additional charges, and privacy breaches). For example, schemes that require a third-party or additional currencies require more time to complete the mixing service. Also, users who mix coins with a third-party mixing server or convert coins between pairwise currencies typically have to pay for the service. Bitcoins (or other cryptocurrencies) are also at risk of being stolen by mixing servers because the servers possess all the valid signatures. In addition, a user's initial transactions can be exposed because mixing servers are at risk of attack. Furthermore, one can predict another user's output addresses by tracing information on a bulletin board.

Therefore to mitigate these limitations, in this paper we present a mixing scheme with a decentralized signature protocol that places specific emphasis on multiple-transaction processes in Bitcoin (BTC) blockchain. Specifically, the contributions of this paper are as follows:

1) To avoid unnecessary delays, we introduce a negotiation process, where each user keeps his/her initial transaction details secret. Thus, this requires less mixing time than using a third-party.
2) To avoid incurring additional charges, we design a mixing scheme that splits the direct relationships between the initial input addresses and output addresses, in order to ensure randomness and anonymity.
3) To protect privacy, we do not rely on third parties and instead use a decentralized signature protocol based on

R.Y. Xiao is with the School of Computer Science, China University of Geosciences (Wuhan), Wuhan, China, 430074, and the School of Mathematics and Physics, China University of Geosciences (Wuhan), Wuhan, Hubei, China, 430074.

W. Ren is with the School of Computer Science, China University of Geosciences(Wuhan), Wuhan, China, 430074, the Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences (Wuhan), Wuhan, Hubei, China, 430074, and the Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guizhou, P.R. China. Corresponding Author email:weirencs@cug.edu.cn.

T.Q. Zhu is with the School of Software, University of Technology Sydney, Ultimo, NSW 2007, Australia.

K.-K.R. Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249-0631, USA.

Manuscript received November 7, 2018; revised .

ElGamal signature protocol. In the protocol, detected malicious behavior will be penalized by temporarily suspending the associated assets.

Having introduced the paper, we will briefly review related work in the next section (see Section II). In Section III, we describe the problem formulation and other relevant materials. In Section IV, we present our proposed approach, prior to presenting the evaluation findings in Section V. We conclude the paper in Section VI.

II. Related Work

We will briefly describe key mixing approaches below.

Mixcoin: The Mixcoin process [19] is completed by a single trusted third-party mixing server. Users only need to provide their input addresses and related fresh output addresses. Additionally, a number of transactions with multiple input and output addresses need to be mixed by the third-party server. However, there are several security issues. First, a malicious third-party can steal the currency it is handling (e.g. by changing the output address). Also, as this third-party has access to all original transactions, it becomes an attractive target to cybercriminals seeking to steal the cryptocurrencies. In the event that multiple users need to transfer fund to the same address (e.g. for an international conference registration), this mixing payment method means transactions are ambiguous, and the recipient may find it challenging to identify which payment is made by who. Lastly, such service is usually not free.

Coinjoin: The coin mixing process [9], [10] is conducted by a trusted third-party mixing server (i.e. a mixing server), where multiple users agree to generate a combined transaction with multiple inputs and outputs. This scheme also suffers from several limitations. For example, each output address must be a fresh new address because two users who share the same output address would otherwise not be able to recognize their own transaction details in a mixed transaction. As a result, the mixing server knows each original transaction and, if that mixing server is successfully compromised, then the transaction details will be leaked. Users also need to pay additional fees for this mixing service.

CoinShuffle: In this scheme, users broadcast their requirements, which include the amounts to be mixed. Users who are in need of the same coins can join as one group [10], [11]. They merely exchange each other's output addresses directly. However, finding a mixing group to join is difficult and waiting for another can result in time wastage. Additionally, because all users put their requirements on a bulletin board, one can potentially infer another user's real output addresses from his/her mixed amount. Moreover, malicious nodes may promise signatures and then refuse to sign after gaining other valid signatures. Such malicious nodes may also steal valid signatures for illegal uses, and CoinShuffle's punishment mechanism has little to do with financial punishments. Moreover, this scheme is completed without monitoring by a trusted third party; therefore, it is hard to prove that a node is malicious or dishonest.

Altcoin: In Altcoin [3], users convert their mixing coins into other virtual currencies, such as Zerocoin [14]. After the conversion, the users' coins are aggregated. However, an additional underlying protocol is needed for the conversion between different currencies. In addition, the values of these currencies fluctuate daily, and market fluctuations may occur between pairwise currencies. This means users may not receive the same amount of coins when the currency is returned. Moreover, this conversion process incurs additional time.

Blinded token: In this scheme, a user randomly selects other users to join and create mixing groups. A "so-called" trusted mixing server is selected by users from several third-party mixing servers [13]. Without knowing the links between the input addresses and output addresses [12], the selected server verifies the validity of the signatures. However, the mixing servers may become so busy that users experience significant delays, and additional fees for the third-party still apply.

Our proposed scheme seeks to split the initial links among input addresses and output addresses. Therefore, similar to mixing technologies like Mixcoin and Coinjoin, our scheme can form multiple input addresses and multiple output addresses in one transaction.

Although our scheme is on the basis of mixing technologies, it uses each user rather than a third-party to form the final transaction. In terms of replacing the mixing third party, we introduce a multi-signature protocol dynamically formed by a group of signers [20], which was first proposed by K. Itakura and K. Nakamura [21] and was formally defined by Silvio Micali, Kazuo Ohta, Leonid Reyzin [22].

III. Problem Formulation

A. Adversary Model

Definitions of the problems are listed below:

Dishonest/cheating behaviour This includes both a user$_i's$ and a third-party$_i's$ dishonest behavior. The former relates to a user providing fake addresses, useless signatures, and doublge spending transactions. The latter relates to a third party$_i's$ deliberate behavior, such as counterfeiting output addresses, signatures or transactions, leaking initial transaction details, and refusing to provide mixing services.

Dishonest/malicious nodes Any individuals or organizations that behave dishonestly can be thought of as a dishonest/malicious node.

Third parties Third parties are divided into two categories: negotiating third parties, which provide alternative chatting services (e.g., Wechat, a BBS, WhatsApp), and mixing third parties, which provide specific mixing services (e.g., Bitcoin Fog, BitLaundry, Dark Wallet, Bitmixer).

B. Design Goals

We outlined three main disadvantages of existing schemes in Section I and Section III-A: delay, extra charges, and privacy breaches. Now, we provide more specific details of our three design goals: time, coins, and privacy. Definitions of the specific subgoals are listed below.

Time - Shorter waiting intervals Except for essential transaction confirmation from the blockchain, users do not need to wait for mixing services.

Coins - No additional mixing fees The protocol should not require additional fees for specific mixing services. But transaction fees are allowed.

Privacy - Transaction security The security for transactions requires non-linked input and output addresses, unpredictable user identities, and untraceable initial transaction details.

Privacy - Trustless third party environments Even if a trusted third party is involved in the mixing procedure, it will not have enough information to counterfeit transaction details or even obtain the initial transaction details.

We explore a way in which each user can monitor the whole mixing process without extra cost in time, coins, and privacy.

Secret sharing is used to prevent users from distributing signatures to malicious nodes by placing all the group members into an agreement using a secret. Secret sharing can split a secret into different secret shares, or pieces, and distribute them to each group member. If the available shares do not equal the original amount, none of the group members can recover the secret. Because transactions might be eliminated if someone cheats the mixing process, we have added a negotiation process that requires the participation of all users. The validity of one transaction in the blockchain can be verified through the signature, which is solely generated by all related private keys.

Additionally, the ElGamal signature protocol [23] has been proven to be secure because discrete logarithmic problems are believed to be hard. The ElGamal signature protocol is suitable for designing a shared secret, as ElGamal$'_i s$ signature includes a commitment, which makes it possible to prove every user$'_i s$ ownership of a private key.

IV. Proposed Mix Scheme Using a Decentralized Signature Protocol

We propose a decentralized signature protocol that builds on the ElGamal signature protocol, which defends against malicious behavior by temporarily suspending assets.

However, before describing our approach, we first introduce the basic model with a negotiation process. The basic model somewhat mitigates the risk of a mixing server being compromised because negotiation before mixing ensures that no one else has direct access to anyone's real transaction details. But users still suffer from time waste in case that malicious nodes refuse to sign their final transaction. Table I lists the notations used in this protocol.

A. Basic Scheme

As mentioned a negotiation process reduces risks of a privacy leak, but this process is only required in a random-named negotiation group to ensure users do not know each other. Our scheme aims to create a new transaction, where links among the original input and output addresses are randomly split. The four steps are shown in Fig. 1.

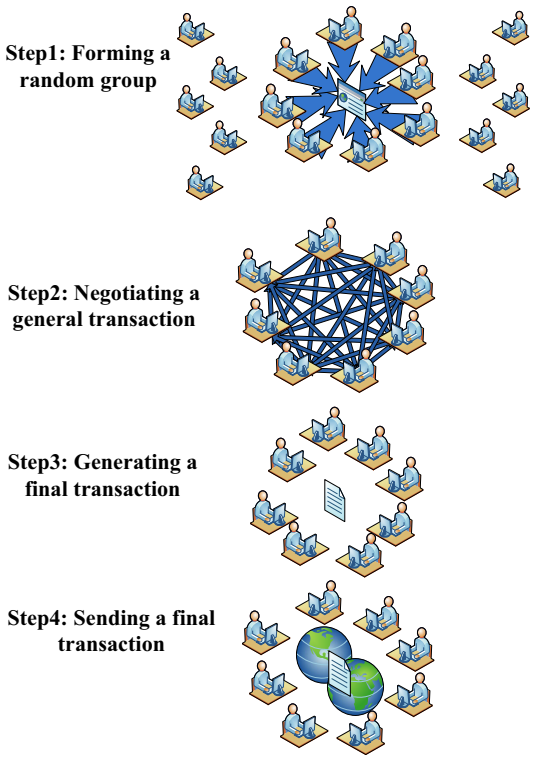

Fig. 1. Procedures of basic scheme

1) **Forming a random group**

A node that needs to mix coins broadcasts a mixing request, i.e., a $MixRes$ on the bulletin board. No sensitive details are included, such as output addresses, BTC distributions, or private keys. Every node can choose to create a new mixing group or join other node's mixing group.

For those who want to create a new mixing group (Creator), Creator's $MixRes$ only includes creator's former transaction addresses, a mixing demand, a nonce, a minimum number of group member, a end time, a timestamp and a signature. Moreover, Creator's signature signs all other information except itself in the $MixRes$.

For those who want to join other node's mixing group (Participant), the $MixRes$ includes participant's former transaction addresses, a mixing demand, the nonce of creator's $MixRes$, a timestamp and a signature, where participant's signature also signs all other information except itself in the $MixRes$.

Broadcasts with the same nonce before the creator's end time allow users to form a temporary mixing group. Only if there exist no less than minimum number of group members in creator's $MixRes$ will a mixing group be created. Otherwise, the group shall be dissolved.

Suppose that there are m nodes in this group in total($m \geq 2$), each node is denoted as $Node_i$ ($1 \leq i \leq m$) in random order. Fig. 2 shows an example of how the group is formed, which could be completed on any social media platform allowing users to share information (e.g., Wechat, a BBS, WhatsApp).

2) **Negotiating a general transaction**

As shown in Fig. 3 and Fig. 4, $Node_i$ creates fresh accounts as output addresses and generates

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2938953, IEEE Transactions on Dependable and Secure Computing

TABLE I
NOTATIONS

Notation	Description
$MixRes$	A mixing request for mixing group members
m	A sum of the nodes in the mixing group
$Node_i$	A random temporary name of a user in a mixing group
$NegRes_{ik}$	A resolution request of $Node_i$'s initial mixing request
$OutAddress_{ik}$	The output addresses in $NegRes_{ik}$
Sum_{ik}	The amount of BTC needed in $NegRes_{ik}$
$SatMess_j$	A satisfaction message that $Node_j$ can completely satisfy
$PartSatMess_j$	A satisfaction message that $Node_j$ can partly satisfy
$NegRes_{ik'}$	A new mixing transaction that $Node_j$ can not satisfy
$Sum_{ik'}$	The amount of BTC needed in $NegRes_{ik'}$
$Assessment_i$	$Node_i's$ assessment of all broadcast messages
$MixTran$	The final mixed transaction of this mixing group
p	A random chosen prime number
g	A primitive element in $Z_p^* =< Z/pZ\backslash\{0\}, * >$
x_i	A random integer number of $Node_i$
k_i	An integer number with $(k_i, p-1) = 1 (1 \leq i \leq m)$ of $Node_i$
r_i, h_i, t_i, s_i	Transmitted data from $Node_i$
$SIGN$	Signatures of $MixTran$
$Sign_1, Sign_2, Sign_3$	An element of $SIGN$

Fig. 2. Formation of a random group

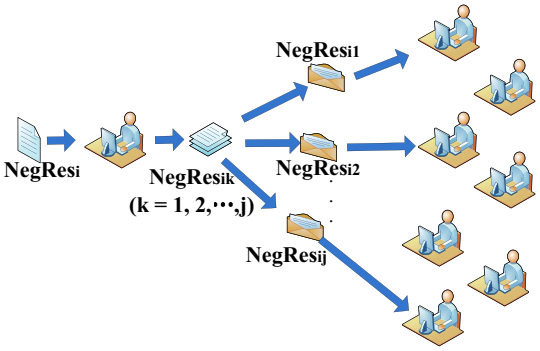

Fig. 3. Generating several disparate negotiating requests

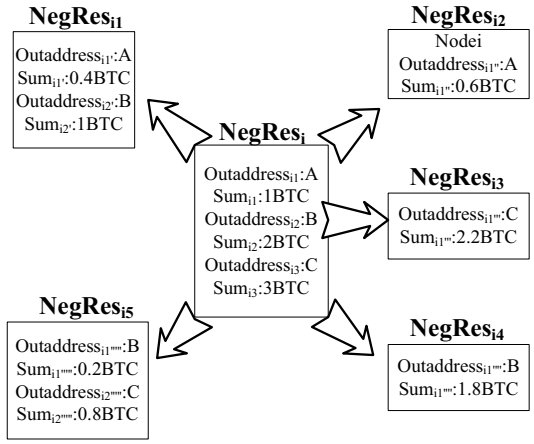

Fig. 4. Sending several specific disparate negotiating requests

several disparate negotiating requests $NegRes_{ik} (k \geq 1, 1 \leq i \leq m)$. $NegRes_{ik}$ includes all output address $OutAddress_{ik}$ and the number of coins Sum_{ik}. $Node_i$ randomly chooses different nodes to send each negotiating request to. At the same time, it receives requests from other nodes.

Once a negotiating request $NegRes_{ik}$ is received, $Node_j$ $(1 \leq j \leq m)$ in Fig. 5 judges whether or not his/her former transaction address has enough BTC to satisfy Sum_{ik}. If the answer is yes, $Node_j$ will generate a satisfaction message $SatMess_j$, and broadcast it among the mixing group. $SatMess_j$ includes $Node_j's$ former transaction address, $OutAddress_{ik}$ and Sum_{ik} in $NegRes_{ik}'s$. Otherwise, $Node_j$ will firstly generate a part-satisfaction message $PartSatMess_j$ and broadcast it among the mixing group. Then, $Node_j$ will generate a new negotiating request $NegRes_{ik'}$ and randomly choose another node to send it to. $PartSatMess_j$ includes $Node_j's$ former

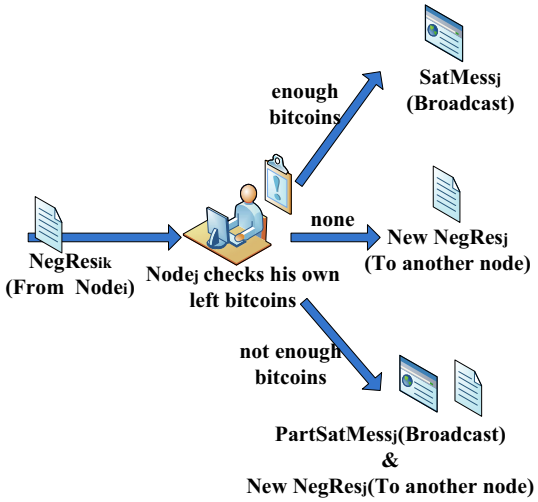

Fig. 5. Negotiating details separately

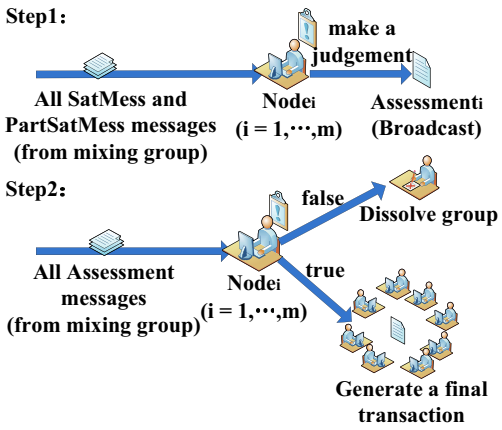

Fig. 6. Verifications made before generating a final transaction

transaction address, the $OutAddress_{ik}$ in $NegRes'_{ik}s$ and $Node'_js$ current amount of BTC. $NegRes_{ik'}$ includes the $OutAddress_{ik}$ in $NegRes_{ik}$ and the BTC $Sum_{ik'}$. Once a satisfaction message $SatMess_j$ or a part-satisfaction message $PartSatMess_j$ is broadcast, $Node'_js$ former transaction address will reduce its BTC by an equal amount, to some extent.

Example (Here we take $Node_1$ as an example to illustrate the negotiating process):

(1)Suppose that $Node_1$ generates three fresh address A, B and C as his/her mixing output addresses. $Node_1$'s real mixing request are to send A 1 BTC, to send B 2BTC and to send C 3 BTC. Then $Node_1$ chooses a random number h between 1 and the size of group member to divide his/her initial request $NegRes_1$ into several sub-request $NegRes_{1i}$(where $i = 1, 2, \cdots, h$). Here we set h as 5. Therefore, $Node_1$ can get that $NegRes_{11}$ is to send A 0.4 BTC and to send B 1 BTC, $NegRes_{12}$ is to send A 0.6 BTC, $NegRes_{13}$ is to send C 2.2 BTC, $NegRes_{14}$ is to send B 1.8 BTC, $NegRes_{15}$ is to send B 0.2 BTC and to send C 0.8 BTC.

(2)$Node_1$ chooses 5 different nodes in mixing group to send a sub-request secretly. At the same time, $Node_1$ shall receive sub-requests from other nodes.

(3)Suppose that $Node_1$ firstly receives a sub-request to send H 3 BTC. Since $Node_1$ has 6 BTC in total, he/she can satisfy this sub-request and can have 3 BTC left. Thus, he/she broadcasts a $SatMess_1$ which includes $Node_1$'s former transaction hash, $OutAddress_1 = H$ and $Sum_1 = 3$.

(4)Suppose that $Node_1$ then receives a sub-request to send W 8 BTC. Since $Node_1$ only has 3 BTC left, he/she broadcast a $PartSatMess_1$ which includes $Node_1$?s former transaction hash, $OutAddress_1 = W$ and $Sum_1 = 3$. Noted that there exist 5 BTC in this received sub-request, $Node_1$ regenerates a new sub-request which aims at sending W 5 BTC and sends it to another node.

(5) Suppose that $Node_1$ still receives sub-requests from other nodes. But considering that he/she has no coins left, $Node_1$ directly sends his/her received sub-requests to another node.

3) **Generating the final transaction**

Having finished step 2, each node verifies the validity of these messages in two aspects. First, $Node_i$ $(1 \leq i \leq m)$ judges whether or not its output addresses $OutAddress_{ik}(k \geq 1)$ have received an equal amount of BTC. This judgement aims to prevent unsatisfactory requests. Then, by making comparisons between the amount of coins in all satisfaction and part-satisfaction messages and that in all mixing requests, $Node_i$ judges whether or not there any requests have been omitted. As shown in Fig. 6, $Node_i$ makes an assessment $Assessment_i$ based on both judgments and then broadcasts it to the group. If, and only if, over two-thirds of the group$'_is$ assessments have indicated a verification will this group generate a final mixed transaction $MixTran$. $MixTran$ includes all input addresses, all output addresses, new divisions of BTC from step 2 and the hash values of $MixTran$ (see Fig. 7). Otherwise, this mixing group will be dissolved.

4) **Sending the final transaction**

Having completed Step 3, $Node_i$ $(1 \leq i \leq m)$ will broadcast his/her $Sign^i_{MixTran}$ among the mixing group. Each node in this group is able to collect everyone's signature and broadcast $(MixTran, Sign^1_{MixTran}, ..., Sign^m_{MixTran})$ to the internet. Therefore, even if malicious nodes refuse to sign signatures to final transaction, other nodes in this group will not suffer from BTC theft because the final transaction is invalid.

This scheme uses a negotiation process to replace a third party. Since messages, such as mixing requests and negotiating requests, do not include the amount of BTC and the addresses in the initial transaction, real identities cannot be predicted using the final transactions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2938953, IEEE Transactions on Dependable and Secure Computing

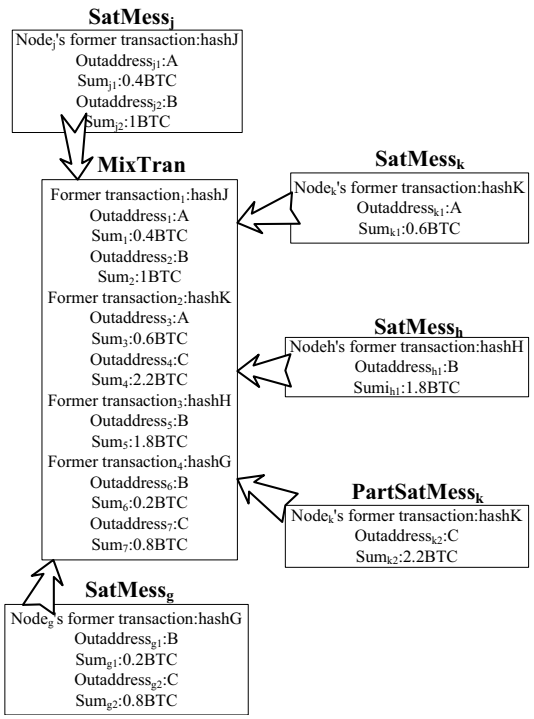

Fig. 7. Generating a MixTran

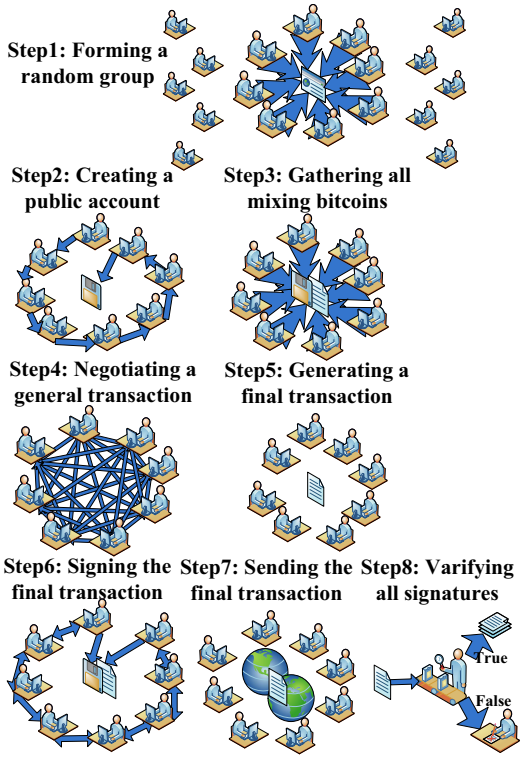

Fig. 8. Advanced scheme procedures

B. Advanced Scheme

The basic scheme introduces a negotiation process that prevents nodes from knowing others' initial transactions. And it can be further extended to tackle shortcomings, such as time wastes caused by malicious nodes not agreeing to sign transactions, no substantial- finance punishments directing at dishonest nodes even if their behaviors do not cause monetary loss and so on.

The advanced scheme includes a decentralized signature protocol based on basic scheme 8. Using this protocol, group members combine to generate a new public address without the participation of a mixing server. This group cannot recover a signature if anyone is absent because it is solely generated by all private keys. Moreover, unless a user accepts that his/her BTC will be lost, no one is able to cheat because all BTC have to be sent to their public address before negotiation.

1) **Forming a random group**

Considering that 1) Forming a random group in the advanced scheme is the same as that of basic scheme, we will simplify descriptions here.

2) **Creating a public account**

As shown in Fig. 9, the group chooses a prime number p and a primitive element $g \in Z_p^* =< Z/pZ\backslash\{0\}, * >$. Each $Node_i (1 \leq i \leq m)$ chooses a random integer number x_i as its private key.

After all the group members of $Node_i$ have chosen x_i, $Node_1$ calculates the following equation:

$$y_1 \equiv g^{x_1} \bmod p \tag{1}$$

and sends y_1 to $Node_2$. Having received y_{j-1} from $Node_{j-1}$ $(2 \leq j \leq m-1)$, $Node_j$ calculates

$$y_j \equiv y_{j-1}^{x_j} \bmod p \tag{2}$$

and sends y_j to $Node_{j+1}$. Finally, $Node_m$ calculates

$$y_m \equiv y_{m-1}^{x_m} \bmod p \tag{3}$$

and broadcasts y_m among the group as a public key. The mixing group computes the hash value $hash(y_m)$ and sets it as the public mixing address.

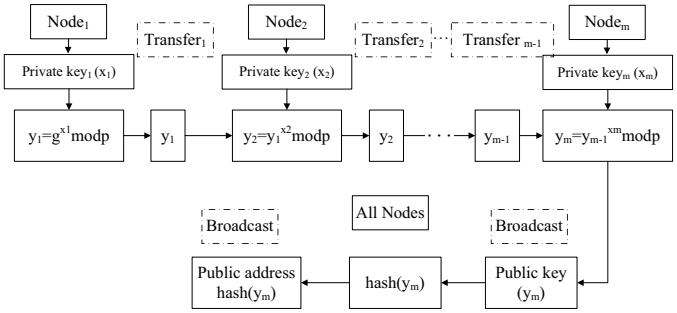

Fig. 9. Creating a public account

3) **Aggregating all mixing bitcoins**

All group members gather to generate a transaction $GaTrans$. $GaTrans$ includes all nodes' former transaction addresses of the nodes, the amount of BTC and a public mixing address. Only when all of them have attached their signatures $Sign^i_{GaTrans}$, will the transaction $GaTrans$ become valid. Every node is able to broadcast $GaTrans$ to the internet. If any malicious node refuses to sign, group can be dissolved.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2938953, IEEE Transactions on Dependable and Secure Computing

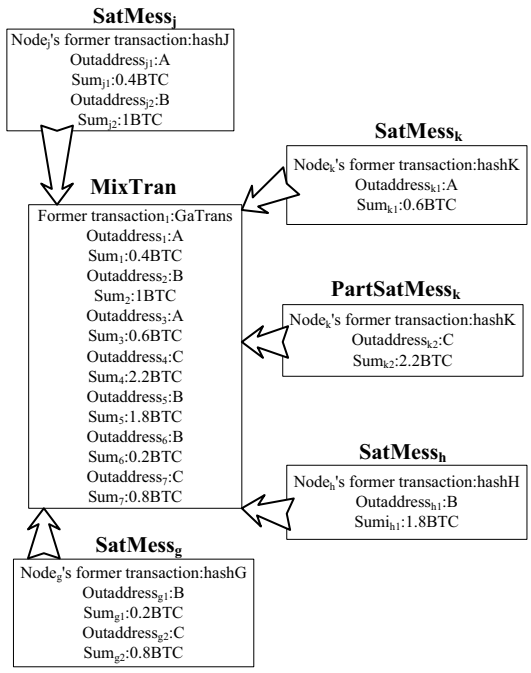

Fig. 10. Generating a final transaction

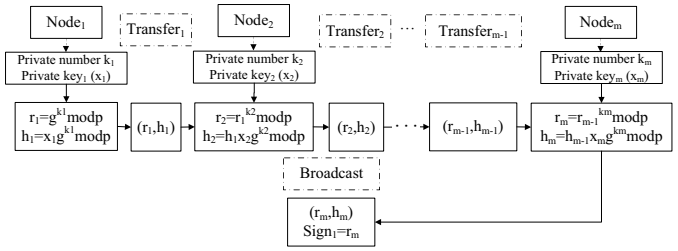

Fig. 11. The first step in signing a final transaction

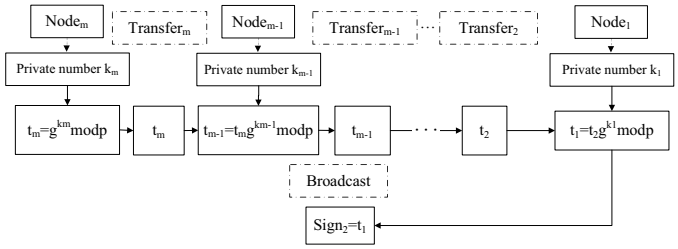

Fig. 12. The second step in signing a final transaction

4) **Negotiating a general transaction**

Considering that 4) Negotiating a general transaction in the advanced scheme is the same as that of basic scheme, we will simplify descriptions here.

5) **Generating a final transaction**

Having completed Step 4, each node verifies the validity of these messages in two aspects. To prevent unsatisfactory requests, $Node_i$ $(1 \leq i \leq m)$ judges whether or not all its output addresses will receive an equal amount of BTC. By comparing the amount of BTC in all satisfaction and part-satisfaction messages and the amount in all mixing requests, $Node_i$ determines whether or not any requests have been omitted. As shown in Fig. 6, $Node_i$ makes an assessment $Assessment_i$ based on both judgements and broadcasts it to the group. If, and only if, over two-thirds of the group's assessments have indicated a verification will this group generate a final mixed transaction $MixTran$. $MixTran$ in Fig. 10 includes all input addresses, all output addresses, new divisions of BTC from Step 4 and hash values of $GaTrans$. Otherwise, this group will be dissolved.

6) **Signing the final transaction**

Each $Node_i$ chooses an integer number k_i with $(k_i, p-1) = 1 (1 \leq i \leq m)$.

As is illustrated in Fig. 11, $Node_1$ calculates

$$r_1 \equiv g^{k_1} \bmod p \tag{4}$$

$$h_1 \equiv x_1 * g^{k_1} \bmod p \tag{5}$$

and sends (r_1, h_1) to $Node_2$. After receiving (r_{j-1}, h_{j-1}) from $Node_{j-1} (2 \leq j \leq m-1)$, $Node_j$ calculates

$$r_j \equiv r_{j-1}^{k_j} \bmod p \tag{6}$$

$$h_j \equiv h_{j-1} * x_j * g^{k_j} \bmod p \tag{7}$$

and sends (r_j, h_j) to $Node_{j+1}$. Then $Node_m$ calculates

$$r_m \equiv r_{m-1}^{k_m} \bmod p \tag{8}$$

$$h_m \equiv h_{m-1} * x_m * g^{k_m} \bmod p \tag{9}$$

and sets r_m as part of a signature $Sign_1$.

Additionally, $Node_m$ in Fig. 12 calculates

$$t_m \equiv g^{k_m} \bmod p \tag{10}$$

and sends t_m to $Node_{m-1}$. After receiving t_{j+1} from $Node_{j+1} (2 \leq j \leq m-1)$, $Node_j$ calculates

$$t_j \equiv t_{j+1} * g^{k_j} \bmod p \tag{11}$$

and sends it to $Node_{j-1}$. Then $Node_1$ calculates

$$t_1 \equiv t_2 * g^{k_1} \bmod p \tag{12}$$

and sets t_1 as part of a signature $Sign_2$.

Finally, $Node_1$ in Fig. 13 calculates

$$s_1 \equiv (MixTrans - h_m) * k_1^{-1} \bmod (p-1) \tag{13}$$

and sends s_1 to $Node_2$. After receiving s_{i-1} from $Node_{i-1} (2 \leq i \leq m-1)$, $Node_i$ calculates

$$s_i \equiv s_{i-1} * k_i^{-1} \bmod (p-1) \tag{14}$$

and sends s_i to $Node_{i+1}$. Then $Node_m$ figures out

$$s_m \equiv s_{m-1} * k_{m-1}^{-1} \bmod (p-1) \tag{15}$$

and sets s_m as part of a signature $Sign_3$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2938953, IEEE Transactions on Dependable and Secure Computing

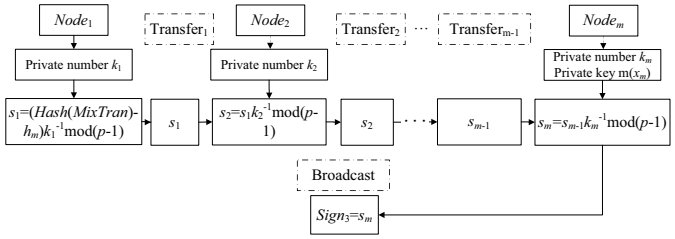

Fig. 13. The third step in signing a final transaction

Therefore, the final signature $SIGN$ should be $(Sign_1, Sign_2, Sign_3)$.

7) **Broadcasting the final transaction**
Each node in the mixing group can broadcast $(MixTran, SIGN)$ across the internet.

8) **Verifying all signatures**
To verify the validity of $(MixTran, SIGN)$, each node calculates

$$result = y_m^{Sign_2} * Sign_1^{Sign_3} \tag{16}$$

and judges whether $result$ is equal to $g^{Hash(MixTrans)}$ or not. The miners will only put the final transaction $MixTran$ and signatures $SIGN$ on the blockchain if this judgement is true.

In our scheme, the decentralized signature protocol requires aggregations of all mixed BTC to be aggregated and a new public address. However, because every mixing group member has to send his/her BTC to this public address, malicious nodes are highly unlikely to fallaciously commit to sending their BTC, which improves the security of the coin. Further, since there is no one-to-one correspondence between input addresses and output addresses, attackers can not analyze transaction details to find real links through permutations and combinations. This enhances the security of the information.

V. Analyses on Security and Performance

Our security analyses were conducted on the assumption of three types of attacks: a key-only attack (KOA) where attackers know public keys and signature verification functions, a known message attack (KMA) where attackers possess messages and related signatures, and a chosen message attack (CMA) where attackers require users to sign some specific messages. We conducted these security analyses to evaluate whether our signature protocol is safe or not. Before conducting these security analyses, we first introduce a correctness proof to verify the validity of proposed signature protocol. We also compared our scheme's mixing performance with other schemes to assess its stability and the reliability of the processing environment.

A. Correctness Proof of Proposed Signature Protocol

In Sec IV, we proposed a decentralized signature protocol to sign a final transaction. Suppose that the final transaction shall be $MixTran$ and the signature $SIGN$ shall be $(Sign_1, Sign_2, Sign_3)$, we can get

$$Sign_1 = r_m = g^{k_1 * k_2 * \cdots * k_m} \bmod p \tag{17}$$

$$Sign_2 = t_1 = g^{k_1 + k_2 + \cdots + k_m} \bmod p \tag{18}$$

$$\begin{aligned} Sign_3 = s_m =& (Hash(MixTran) - h_m) * {k_1}^{-1} * \cdots * {k_m}^{-1} \\ & \bmod (p-1) \\ =& (Hash(MixTran) - x_1 * x_2 * \cdots * x_m \\ & * g^{k_1 + k_2 + \cdots + k_m}) * {k_1}^{-1} * \cdots * {k_m}^{-1} \\ & \bmod (p-1) \end{aligned} \tag{19}$$

Considering that

$$\begin{aligned} y_m^{Sign_2} * Sign_1^{Sign_3} =& g^{x_1 * x_2 * \cdots * x_m * Sign_2} * Sign_1^{Sign_3} \\ & \bmod (p-1) \\ =& g^{x_1 * x_2 * \cdots * x_m * Sign_2 + k_1 * k_2 * \cdots * k_m * Sign_3} \\ & \bmod (p-1) \\ =& g^{x_1 * x_2 * \cdots * x_m * g^{k_1 + k_2 + \cdots + k_m}} * \\ & g^{(Hash(MixTran) \div} \\ & g^{x_1 * x_2 * \cdots * x_m * g^{k_1 + k_2 + \cdots + k_m})} \\ & \bmod (p-1) \\ =& g^{(Hash(MixTran)} \bmod (p-1) \end{aligned} \tag{20}$$

Our proposed signature protocol is thus proven secure.

B. Security Analyses for KMA

Various hypotheses associated with KMAs are outlined below along with a corresponding security analysis given a theoretical environment.

To start, we assume that attackers only possess the final signature $(MixTran, SIGN)$, the prime number p, the primitive element g and the public key y_m. These goal of the attackers is to forge a valid signature for a virtual transaction $VirTrans$. Hence, all parameters in a valid signature should follow the equation rule:

$$y_m^{Sign_2} * Sign_1^{Sign_3} = g^{Hash(VirTrans)} \tag{21}$$

1) If an attacker randomly chooses a new $Sign_1$ and wants to find its corresponding $Sign_2$ and $Sign_3$, they will not be able to calculate $Sign_2$ and $Sign_3$ using the current approaches.
2) If an attacker randomly chooses a new $Sign_2$ and wants to find its corresponding $Sign_1$ and $Sign_3$, thet will not be able to calculate $Sign_1^{Sign_3}$. However, it is difficult for this number to be spilt into a base number $Sign_1$ and a exponent number $Sign_3$. Even large quantities of such numbers could not be broken down into two pieces.
3) If an attacker randomly chooses a new $Sign_3$ and wants to find its corresponding $Sign_1$ and $Sign_2$, they will not be able to calculate $Sign_1$ and $Sign_2$ using current approaches.

The examples above illustrate that this model has the ability to defend against selective forgery.

Considering the equation

$$y_m^{Sign_2} * Sign_1^{Sign_3} = g^{Hash(MixTrans)} \tag{22}$$

The form of this equation can be changed to draw a new conclusion

$$Hash(MixTrans) = (x_1 * x_2 * ... * x_m) * Sign_2 + (k_1 * k_2 * ... * k_m) * Sign_3 \bmod (p-1) \quad (23)$$

Users can not change x_i unless they break it down because the public keys and a public address are solely generated through x_i. Therefore, we can assume that each node $Node_i$ does not choose a different nonce k_i in both transactions and that the attacker's aim must be to falsify a valid signature. Thus, in the advanced scheme, users will not change

$$Sign_1 = r_m = g^{k_1 * k_2 * \cdots * k_m} \bmod (p-1) \quad (24)$$

$$Sign_2 = t_1 = g^{k_1 + k_2 + ... + k_m} \bmod (p-1) \quad (25)$$

$$h_m = x_1 * x_2 * ... * x_m * g^{k_1 + k_2 + ... + k_m} \bmod (p-1) \quad (26)$$

Hence, every two signatures will have the following relationships [24]:

In the first transaction $(MixTrans_1, SIGN = (Sign_1, Sign_2, Sign1_3))$:

$$Hash(MixTrans_1) = (x_1 * x_2 * ... * x_m) * Sign_2 + (k_1 * k_2 * ... * k_m) * Sign1_3 \bmod (p-1) \quad (27)$$

In the second transaction $(MixTrans_2, SIGN = (Sign_1, Sign_2, Sign2_3))$:

$$Hash(MixTrans_2) = (x_1 * x_2 * ... * x_m) * Sign_2 + (k_1 * k_2 * ... * k_m) * Sign2_3 \bmod (p-1) \quad (28)$$

Then we have

$$Hash(MixTrans_2) - Hash(MixTrans_1) = (k_1 * k_2 * ... * k_m) * (Sign2_3 - Sign1_3) \bmod (p-1) \quad (29)$$

In this case, $MixTrans_1, MixTrans_2, Sign1_3$ and $Sign2_3$ are available to everyone. Given that

$$d = gcd(Sign2_3 - Sign1_3, p-1) \quad (30)$$

we have $d|(Sign2_3 - Sign1_3)$ along with $d|p-1$.

Using the definition

$$Hash(MixTrans)_{min} = \frac{MixTrans_2 - MixTrans_1}{d} \quad (31)$$

$$Sign_{min} = \frac{Sign2_3 - Sign1_3}{d} \quad (32)$$

$$p_{min} = \frac{p-1}{d} \quad (33)$$

we have

$$Hash(MixTrans)_{min} = (k_1 * k_2 * ... * k_m) * Sign_{min} \bmod p_{min} \quad (34)$$

$$gcd(Sign_{min}, p_{min}) = 1 \quad (35)$$

Note that

$$(k_1 * k_2 * ... * k_m) = Hash(MixTrans)_{min} * (Sign_{min}^{-1}) \bmod p_{min} \quad (36)$$

Moreover, $Sign_1$ is also public to everyone, which means that attackers can derive $k_1 * k_2 * ... * k_m$ by testing

$$Sign_1 = g^{k_1 * k_2 * ... * k_m} \bmod p \quad (37)$$

This example implies that the security of our scheme relies on random nonce k_i and the number of transactions. k_i can not be used for a second time. That is to say, once users need to sign a different transaction, their k_i must be changed.

C. Security Analysis for KOA

An attacker, who only has the prime number p, the primitive element g, and the public key y_m, may randomly choose a pair of numbers (u, v), where $1 \leq u, v \leq p-1$ and $gcd(v, p-1) = 1$. They will then be able to calculate

$$Sign_1 = Sign_2 = g^{-u} * y_m^v \bmod p \quad (38)$$

$$Sign_3 = -Sign_1 * v^{-1} \bmod (p-1) \quad (39)$$

$$Hash(MixTrans) = -u * Sign_3 \bmod (p-1) \quad (40)$$

Subsequently, they can claim $SIGN = (Sign_1, Sign_2, Sign_3)$ and $Hash(MixTran) = Hash(MixTrans)$ as the real final transaction parameters. However, in reality even if $Hash(MixTran)$ is available, attackers can not recover valid $MixTran$ due to residence of Hash function. Moreover, this may require us to make an extra value check on $Sign_1$ and $Sign_2$ in the verification process, where there should be such a little possibility that $Sign_1 = Sign_2$.

D. Security Analysis for CMA

Assume that one specific transaction $MixTran_1$ and its corresponding valid signature $SIGN1 = (Sign1_1, Sign1_2, Sign1_3)$ is available to all attackers. In this case, the attacker's aim is to forge a different transaction.

First, attackers can analyze the equation

$$y^{Sign1_2} * Sign1_1^{Sign1_3} = g^{Hash(MixTrans_1)} \quad (41)$$

Then they can square both sides to produce a new equation

$$(y_m^{Sign1_2} * Sign1_1^{Sign1_3})^2 = y_m^{2*Sign1_2} * Sign1_1^{2*Sign1_3} = g^{2Hash(MixTrans_1)} \quad (42)$$

In this equation, attackers can forge an artificial but valid transaction $2Hash(MixTrans_1)$ and signature $SIGN1^{'} = (Sign1_1, 2 * Sign1_2, 2 * Sign1_3)$.

Though these artificial messages may pass above signature validation, everyone will recognize them as spurious because it is impossible for one transaction hash $2hash(MixTran_1)$ to find its corresponding transaction $MixTrans_1'$. So attackers may hardly find such a specific transaction. This example implies that our scheme can defend against a CMA.

E. Performance Analyses

In our scheme, users have to send their money to the public address, thus reducing a malicious node$'s$ dishonest behavior. They cannot play tricks during the mixing process because honest nodes will not agree to sign a fake transaction. This punishment protocol is reflective of human nature, and typifies our signature protocol$'s$ security.

TABLE II
COMPARISONS BETWEEN SEVERAL MIXING SCHEMES

Protocol	Transaction Security	Trustless third party environments	No additional mixing fees	Shorter waiting hours
Mixcoin	√	×	×	×
Coinjoin	√	×	×	×
CoinShuffle	√	√	√	×
Altcoin	√	×	×	×
Blinded Token	√	×	×	√
Our scheme	√	√	√	√

Turning to the performance of existing mixing protocols, Table. II provides a comparison of the basic characteristics of each along with our proposed scheme.

The major difference between our scheme and others lies in the use of a third party. In schemes like mixcoin, coinjoin, coinshuffle, and blinded token, a third-party mixing server is required to provide mixing services. However, our proposed scheme merely requires a social media platform that enables users to communicate with each other. No additional underlying design is needed. Therefore, our background compatibility is improved because users do not need to install a specific application or software to build the environment.

Without extra mixing services, users in our scheme have more choices for information transfer than in other solutions. Neither a medium or an attacker will have an explicit object to monitor or use to breach privacy because users have a wide range of communication approaches to select from. This differs from other schemes where attackers may have specific targets like BitLaundry, Bitmixer, or Dark Wallet, which increases security risks. In the real world, safe communication applications are much more available to users, compared with a trusted mixing server.

Moreover, a busy server is much more likely to break down since it needs to handle huge amounts of transactions at a time. Our scheme will always be reliable because separate mixing groups can choose distinct mixing environments independently, which makes social media platform more stable. Furthermore, users in our scheme do not need to pay additional charges for mixing services because most social media platforms provide free chatting service.

Other schemes also have fixed long waiting periods to confirm transactions because their mixing servers require users to wait several hours for a safe transaction. For example, a user who only needs to mix 1 BTC may have to spend hours in waiting, not to mention that many users trade in less than 1 BTC. Our scheme takes less time than current schemes because even though the communication procedure is somewhat lengthy, users do not have to wait several hours for it to begin. Lastly, the scheme is based on smaller groups, which increases flexibility.

In view of the P2P size involved in transactions, there exist three negotiation process which includes forming a random group, negotiating a general transaction and signing the final transaction. First, since the random group is formed in a billboard, every node merely require send one request. Thus, the time complexity of this process is $O(m)$. Second, in process of negotiating a general transaction, we can suppose that there exist a complete graph, where each node in graph represents a user and each edge in graph represents the interaction between two nodes. Therefore, the time complexity of negotiation process is liner related to number of edges. Noted that there are m nodes in one graph, its time complexity here shall be $O(m^2)$ because there are $\frac{m(m-1)}{2}$ edges in the directed graph and $m(m-1)$ edges in the undirected graph. Finally, since each node only transfer once in each round where there are four rounds in total, time complexity in signing process turns out to be $O(m)$. To sum up, time complexity in the whole transaction is $O(m^2)$.

F. Further Discussion on Extensive Designs

As with all proposals, this scheme has some limitations and some areas for improvement.

First, significant loading times are tolerable because our mixing scheme focuses more on security. However, users who need mixing services urgently should be able to choose a very small group over a large group to further improve speed. Additionally, to reduce wasted time, each user must have a stable network environment. Unstable network environments create interruptions to the mixing processes. Suspensions may waste even more time by forcing the mixing group to reform.

Further, the processes in our scheme require a discussion platform for negotiation. For efficiency, a convenient social media platform is needed to ensure that all users can communicate with each other. But this ignores various language barriers. Additionally, users are required to retain their chat records to avoid denials in the negotiation process until the final transactions have been signed.

To improve our proposed signature protocol, we can extend a verification process during each P2P interaction. One possible solution is to introduce the concept of zero knowledge proof (which is short for ZK-Proof) in aims of keeping any transaction detail as a secret. Only ciphertexts are available to any other nodes. In this way, no one else except user itself will have access to this user's initial transaction details.

Moreover, for a better performance without a mixing third party, nodes in a mixing group can conduct broadcast processes in a lightweight blockchain among themselves. Apart from that, not only assessments but also verifications can be easily completed through smart contracts.

VI. CONCLUSION

In this paper, we proposed a mixing scheme with a decentralized signature protocol for privacy protection in a BTC blockchain. We not only designed a specific negotiation process among mixing users to circumvent the need for a third party but also introduced a distribution method to collect private keys. Our scheme can reduce the risks of privacy breaches in cryptocurrency mixing processes and, since it does not require a mixing server, additional charges for mixing

services can also be avoided. Neither a fresh address nor two users asking for the same amount of money is required.

However, there are still challenges to decrease mixing wait times in blockchain technology (e.g., transaction confirmations loads and proof of work). Therefore, our next work will address higher efficiency environments and place more emphasis on less negotiation time.

ACKNOWLEDGMENT

This research was financially supported by the Major Scientific and Technological Special Project of Guizhou Province (No. 20183001), the Open Funding of Guizhou Provincial Key Laboratory of Public Big Data (No. 2018BDKFJJ009, No. 2017BDKFJJ006), the National Science Foundation China (No. 61502362), and the Open Funding of Hubei Provincial Key Laboratory of Intelligent Geo-Information Processing (No. KLIGIP2016A05).

REFERENCES

[1] D. Y. Huang, M. M. Aliapoulios, V. G. Li, L. Invernizzi, E. Bursztein, K. McRoberts, J. Levin, K. Levchenko, A. C. Snoeren, and D. McCoy, "Tracking ransomware end-to-end," in *IEEE Symposium on Security and Privacy*. IEEE, 2018, pp. 618–631.

[2] K. Liao, Z. Zhao, A. Doupe, and G.-J. Ahn, "Behind closed doors: measurement and analysis of cryptolocker ransoms in bitcoin," in *APWG Symposium on Electronic Crime Research*. IEEE, 2016, pp. 1–13.

[3] J. Herrera-Joancomartí, "Research and challenges on bitcoin anonymity," in *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*, J. Garcia-Alfaro, J. Herrera-Joancomartí, E. Lupu, J. Posegga, A. Aldini, F. Martinelli, and N. Suri, Eds. Cham: Springer International Publishing, 2015, pp. 3–16.

[4] J. Herrera-Joancomartí and C. Pérez-Solà, "Privacy in bitcoin transactions: New challenges from blockchain scalability solutions," in *Modeling Decisions for Artificial Intelligence*, V. Torra, Y. Narukawa, G. Navarro-Arribas, and C. Yañez, Eds. Cham: Springer International Publishing, 2016, pp. 26–44.

[5] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Financial Cryptography and Data Security*, A.-R. Sadeghi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 34–51.

[6] F. Reid and M. Harrigan, *An Analysis of Anonymity in the Bitcoin System*. New York, NY: Springer New York, 2013, pp. 197–223.

[7] L. Van Der Horst, K.-K. R. Choo, and N.-A. Le-Khac, "Process memory investigation of the bitcoin clients electrum and bitcoin core," *IEEE Access*, vol. 5, pp. 22 385–22 398, 2017.

[8] T. Volety, S. Saini, T. McGhin, C. Z. Liu, and K.-K. R. Choo, "Cracking bitcoin wallets: I want what you have in the wallets," *Future Generation Computer Systems*, vol. 91, pp. 136–143, 2019.

[9] S. Meiklejohn and C. Orlandi, "Privacy-enhancing overlays in bitcoin," in *Financial Cryptography and Data Security*, M. Brenner, N. Christin, B. Johnson, and K. Rohloff, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 127–141.

[10] T. Ruffing and P. Moreno-Sanchez, "Valueshuffle: Mixing confidential transactions for comprehensive transaction privacy in?bitcoin," in *Financial Cryptography and Data Security*, M. Brenner, K. Rohloff, J. Bonneau, A. Miller, P. Y. Ryan, V. Teague, A. Bracciali, M. Sala, F. Pintore, and M. Jakobsson, Eds. Cham: Springer International Publishing, 2017, pp. 133–154.

[11] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *Computer Security - ESORICS 2014*, M. Kutyłowski and J. Vaidya, Eds. Cham: Springer International Publishing, 2014, pp. 345–364.

[12] L. Valenta and B. Rowan, "Blindcoin: Blinded, accountable mixes for bitcoin," in *Financial Cryptography and Data Security*, M. Brenner, N. Christin, B. Johnson, and K. Rohloff, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 112–126.

[13] E. Heilman, F. Baldimtsi, and S. Goldberg, "Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions," in *Financial Cryptography and Data Security*, J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 43–60.

[14] C. Garman, M. Green, I. Miers, and A. D. Rubin, "Rational zero: Economic security for zerocoin with everlasting anonymity," in *Financial Cryptography and Data Security*, R. Böhme, M. Brenner, T. Moore, and M. Smith, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 140–155.

[15] "Bitcoin fog," http://bitcoinfog.org.

[16] "Bitlaundry," http://app.bitlaundry.com.

[17] "Dark wallet," http://www.darkwallet.is.

[18] "Bitmixer," https://bitmixer.io/.

[19] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *Financial Cryptography and Data Security*, N. Christin and R. Safavi-Naini, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 486–504.

[20] L. Harn and J. Ren, "Efficient identity-based rsa multisignatures," *Computers and Security*, vol. 27, no. 1, pp. 12 – 15, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404808000059

[21] K. Itakura and K. Nakamura, "A public key cryptosystem suitable for digital multisignatures," *NEC Research and Development*, vol. 71, pp. 1 – 8, 1983.

[22] S. Micali, K. Ohta, and L. Reyzin, "Accountable-subgroup multisignatures: Extended abstract," in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, ser. CCS '01. New York, NY, USA: ACM, 2001, pp. 245–254. [Online]. Available: http://doi.acm.org/10.1145/501983.502017

[23] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, Jul 1985.

[24] W. Ren, *Digital signature and security protocol*. Tsinghua university press, 2015, pp. 15–18.

Ruiyang Xiao is a student at the School of Computer Science and the School of Mathematics and Physics, China University of Geosciences (Wuhan), China. She has been pre-admitted by University of Science and Technology of China (USTC). Her research interests include blockchain and privacy protection.

Wei Ren currently is a Professor at the School of Computer Science, China University of Geosciences (Wuhan), China. He was with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, USA in 2007 and 2008, the School of Computer Science, University of Nevada Las Vegas, USA in 2006 and 2007, and the Department of Computer Science, The Hong Kong University of Science and Technology, in 2004 and 2005. He obtained his Ph.D. degree in Computer Science from Huazhong University of Science and Technology, China. He has published more than 70 refereed papers, 1 monograph, and 4 textbooks. He has obtained 10 patents and 5 innovation awards. He is a senior member of the China Computer Federation and a member of IEEE.

Tianqing Zhu received her BEng and MEng degrees from Wuhan University, China, in 2000 and 2004, respectively, and a Ph.D degree from Deakin University in Computer Science, Australia, in 2014. Dr Tianqing Zhu is currently a senior lecturer at the School of Software in University of Technology Sydney, Australia. Before that, she was a lecturer at the School of Information Technology, Deakin University, Australia, from 2014 to 2018. Her research interests include privacy preservation, data mining, and network security.

Kim-Kwang Raymond Choo (SM'15) received his Ph.D. in Information Security in 2006 from the Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). In 2016, he was named the Cybersecurity Educator of the Year - APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is the recipient of the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, IEEE TrustCom 2018 Best Paper Award, ESORICS 2015 Best Research Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008. He is also a Fellow of the Australian Computer Society, and an IEEE Senior Member.