HIGH TECHNOLOGY LETTERS | Vol. 28 No. 3 | Sep. 2022 | pp. 259-271

doi:10.3772/j.issn.1006-6748.2022.03.005

Efficient and fair coin mixing for Bitcoin①

GONG Xunwu (龚循武)②[*,**], HU Bin[*], LIU Xiaodong[*], ZHAO Xiaofang[*,***]

([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, P. R. China)

([**] University of Chinese Academy of Sciences, Beijing 100049, P. R. China)

([***] Institute of Intelligent Computing Technology, Chinese Academy of Sciences, Suzhou 215000, P. R. China)

Abstract

Bitcoin transactions are pseudo-anonymous, which can be exploited to reveal a user's private information. To eliminate this threat, this paper presents FairMixer, a highly secure and efficient Bitcoin mixing system using the trusted execution environments (TEEs). With the TEE's confidentiality and integrity guarantees for code and data, FairMixer enables a correct and privacy-preserving mixing process. However, a TEE-based implementation cannot prevent the manipulation of inputs to the mixer, such as mixing request submissions and blockchain feeds. Against this background, FairMixer captures users' mixing requests via Bitcoin transactions for deterring a malicious service provider from dropping benign participants. To constrain misbehavior during a mixing mission, a misconduct monitoring mechanism and a penalty mechanism are introduced. The proposed scheme is fully compatible with Bitcoin and forces mixes to be accountable. Finally, a prototype of FairMixer is provided using Intel Software Guard Extensions (SGX) and its performance is evaluated in the Bitcoin Testnet. FairMixer mixes 700 inputs in just 8.39 s, which outperforms most existing decentralized mixers.

Key words: coin mixing, trusted execution environment (TEE), blockchain, accountable, anonymity

0 Introduction

Bitcoin and other cryptocurrencies keep all transaction records publicly available on the blockchain. User privacy can be violated by linking related transactions. Bitcoin users can stay pseudo-anonymous by generating multiple cryptographic addresses for receiving funds. However, they have to face a threat that their real identities and financial privacy may be leaked through transaction aggregation and analysis[1-2]. Even though these anonymous addresses could not be linked with real identities[3], attackers can deduce from a user's transaction network and infer the user's real identities combining other techniques such as clustering analysis[4].

Hence, several privacy-enhancing mixers have been proposed. The first type is centralized schemes[5-8]: a user Alice sends a certain amount of bitcoins to the mixing service provider (Bob), and then Bob sends the funds received to the address specified by Alice. Advantages of these designs are obvious as follows.

(1) Indistinguishability, meaning the mixing transaction cannot be distinguished from normal transaction.

(2) Scalability, meaning these schemes can be easily extended to mix with thousands of users due to the low cost of the mixing mission.

(3) Resistance to DOS attacks from participants, meaning malicious users that can be rejected as the mixer needs to confirm the receipt of users' funds before mixing.

In centralized mixers, Bob takes possession of all funds and mixing details. Hence, Alice is vulnerable to coin theft and privacy disclosure if bob is breached or other forms of malfeasance takes place.

Risky trust hypothesis in centralized mixers has led to the rising popularity of decentralized mixers[9-14]. In these systems, the coin mixing protocol succeeds if and only if all parties' signatures of the mixing transaction have been collected. It can be seen that these schemes provide theft prevention and no service fees. However, these schemes are not under widespread use due to the following limitations.

(1) Lengthy waiting time, meaning it is uncertain when a random group of coin mixing will be formed through a time-consuming bootstraping process.

① Supported by the National Key Research & Development Program of China (No. 2018YFB0904503).

② To whom correspondence should be addressed. E-mail: gongxunwu18b@ict.ac.cn.

Received on Sep. 10, 2021

(2) Limited scalability, meaning they support a fewer number of users to mix due to the heavy communication cost.

(3) No resistance to DOS from participants, meaning some participants refuse to sign for the mixing transaction.

To mitigate these limitations, this paper presents a secure mixing protocol and system called FairMixer. FairMixer is a best-of-both-worlds design by incorporating the advantages of both existing centralized and decentralized systems. It not only offers guarantees of anonymity and theft protection, but also ensures an efficient and correct mixing operation in a strong threat model. Furthermore, a fairness issue is considered during a mixing mission. It means penalties will be paid for misconducts. This is inspired by the research effort that is dedicated to solve the fairness problem in secure multiparty computation[15-16].

FairMixer relies on a trusted execution environment (TEE)[17]. This technology allows applications to execute within a protected environment called an enclave, which ensures confidentiality and software integrity. TEE enables FairMixer to behave like a trusted third party, conducting a secure mixing mission. It is emphasized that FairMixer requires only a single TEE. The TEE is owned by an operator, who is either one of the participants or an external service provider. The security and trust assumptions of FairMixer are quite conservative, cf. subsection 2.2. In the proposed mixing scheme, one participant desiring to mix his coins sends mixing fee and collateral (e.g., 20% of mixing amount) to the enclave by submitting a deposit transaction, instead of sending the whole mixing amount of coins as existing centralized mixers do.

The reference implementation is built using Intel's Software Guard Extensions (SGX)[18-20], which is a prominent TEE instantiation built into most recent Intel processors. While side-channel attacks on SGX that may endanger security and privacy of the enclave have been demonstrated[21], e.g., prominently Foreshadow[22], mitigating side-channel leakage is considered as an orthogonal problem and out of scope for this paper. TEEs provide powerful memory isolation property that can enhance the security and efficiency of Bitcoin mixers. Whereas, malicious operators could potentially control the entire worldview of the enclaved mixer by manipulating data feeds of the enclave. This incurs nontrivial design challenges which will be discussed in subsection 2.1.

In a nutshell, contributions in this paper are as follows:

- This paper proposes FairMixer, a TEE-backed coin mixing scheme that can support privacy protection and fair mixing. The notion of fair mixing, introduced in this work, means malicious participants or operator will be punished. To this end, a transaction-based mixing request submission mechanism, a misconduct monitoring mechanism and a corresponding penalty mechanism are introduced.
- This paper considers powerful network adversaries who can manipulate the worldview of the isolated mixer in order to reduce the anonymity set size. This is done by designing a stateless enclave, augmenting extra block confirmations and securing data exchanges between an enclave and participants.
- This paper implements proof-of-concept of FairMixer based on the Intel SGX technology, and demonstrate its feasibility and effectiveness for various sizes of participants.

1 Related work

1.1 Bitcoin mixers

Research effort has been dedicated towards the study of achieving unlinkability of transactions on Bitcoin over the years. Below, a non-exhaustive summary of prior work is provided. Existing work can be classified into two main groups, namely centralized and decentralized ones (see Table 1).

Table 1 Comparison of key features

Mixers	Permutation leak prevention	Coin theft prevention	Minimum mixing set size guarantee	Guaranteed fairness	Join-then-abort resistance
Coinjoin[10]	×	√	√	×	×
CoinShuffle[11]	√	√	√	×	×
CoinParty[13]	√	√	√	×	√
Xiao[14]	√	√	√	×	×
MixCoin[6]	×	×	×	×	√
BlindCoin[7]	√	×	×	×	√
Obscuro[9]	√	√	√	×	√
FairMixer	√	√	√	√	√

In the centralized design, users only need to provide their input addresses and fresh output addresses. A number of transactions with multiple input and output addresses need to be mixed by the third-party server[5-8]. Note that there are several security threats since the mixer may steal users' funds and leak the links between the input and output addresses. These risks have led to MixCoin[5] and its successor BlindCoin [6] to provide users signed certificates for accountability. The certificate can hold the malicious mixer accountable for theft by damaging the mixer's reputation and its business model. While the accountability prospects are arguable, it is not certain that the mixing server will not leak the links between the input and output addresses.

TumbleBit introduces an untrusted intermediate payment hub between the payer and payee[7]. Although TumbleBit achieves many desired security properties, it fails to defend against Sybil-based anonymity reduction attacks as it allows a malicious tumbler to selectively drop benign users.

Ref. [8] proposed Obscuro, the first Bitcoin mixer that utilizes trusted execution environments. However, Obscuro cannot prevent eclipse attacks from powerful network adversaries and the misconduct of the service provider. Moreover, funds of honest parties will be locked meaninglessly for a long period of time if the mixing protocol fails for these reasons.

The strong trust assumption in most centralized coin-mixing schemes has led to the rising popularity of decentralized mixers. CoinJoin[9] protocol proposed group transactions. The main drawback of CoinJoin is that the participants learn about the links between the input and output addresses of each other. CoinShuffle[10] improved CoinJoin by introducing a way to mix tokens that no participant learns about the map relationships between input and output addresses. The protocol utilizes decryption mixing nets for output address shuffling.

As the successor of CoinShuffle, CoinShuffle ++[11] is considerably simpler and easier to implement and outperforms CoinShuffle in terms of efficiency. CoinParty[12] improves mixing through group transactions by employing threshold ECDSA signatures. Ref. [13] proposed a mixing scheme with a decentralized signature protocol based on ElGamal signature protocol.

These decentralized systems don't need users to trust each other and eliminate the risk of theft and privacy concerns. Unfortunately, these schemes are vulnerable to join-then-abort DoS attacks. Moreover, decentralized mixing schemes demonstrate limited scalability as they require a heavy communication overhead between the participants (specifically, quadratic in the number of participants). For instance, CoinShuffle only supports a mix of up to 50 participants. CoinParty relies on an assumption that 2/3 of the peers are honest, which could be easily violated in reality.

1.2 TEEs for Blockchains

Trusted execution environment technology has been treated as an effective tool for different kinds of cryptocurrency use cases, such as off-chain payment channels[23], smart contract execution[24-25], reputable data feed services[26]. These schemes offer better efficiency and features by placing more trust in the hardware manufacturer.

Teechain[23] is a system to perform off-chain payments on top of Bitcoin who leverages SGX to set up stateful payment channels among mistrusting parties. With the usage of SGX, Teechain relaxes the synchrony assumptions in existing payment channels and gains efficiency.

Ekiden[24] uses a TEE to support execution of multiparty computations and support contracts that handle coins. It aims at moving heavy smart contract execution off the chain to reduce the cost of executing complex contract functions. In contrast, FASTKITTEN[25] focuses on efficient off-chain execution of multi-round contracts between a set of parties using a TEE.

TownCrier[26] employs SGX to implement a trusted party for vetting external contents and importing them to the blockchain. These systems rely on a trust model which is weaker than that of a purely cryptographic system. In particular, their security is dependent upon a trusted computing base (TCB) that is running inside the trusted hardware. Smaller TCBs mean better security.

2 System settings

2.1 Problem definition

In this work, it considers the problem of designing a Bitcoin mixer that transfers coins from a set of sending addresses $S = \{s_1, s_2, \cdots, s_n\}$ to a set of receiving addresses $R = \{r_1, r_2, \cdots, r_n\}$, while achieving strong anonymity and security properties in the presence of a wide range of attacks against the mixer. Particularly, a mixer must provide unlinkability between S and R. That is, a user should be the only entity that knows the mapping from his input address to his output address. Also, the system should not be vulnerable to DoS attacks by malicious parties or operator. Furthermore, the mixer needs to be secure against a malicious operator of the mixing service, when the operator at-

tempts to undermine the mixer's service (e. g., by stealing coins). In addition, the mixer should easily scale to a large number of users and large anonymity sets. Finally, the system should be compatible with the current Bitcoin system to allow for practical integration.

2.2 System description

The system architecture of FairMixer is depicted in Fig. 1, which involves seven elements, namely operator, enclave, host process, participants, the Bitcoin blockchain, public bulletin board and Intel Attestation Service (IAS).

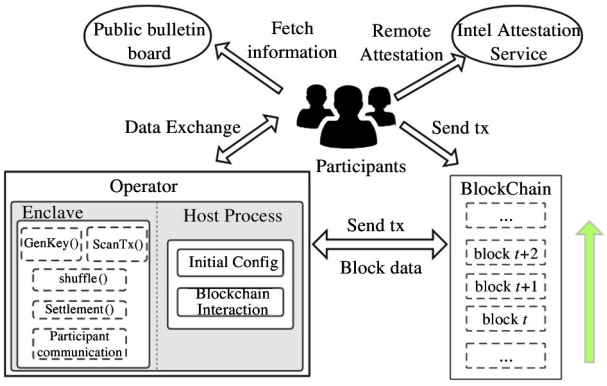


Fig. 1 Architecture of the FairMixer

- Operator is the service provider who controls the platform that hosts the FairMixer instance. It is in charge of running the FairMixer execution facility composed of a host process and an enclave. The operator also handles the participant and blockchain communication over the network. Although this means that the operator has complete control over these parts, his influence on a running enclave is limited. He can interrupt enclave execution but not tamper with it.
- Enclave is a trusted component in the SGX programming model. The key procedures of the mixing protocol are assigned in the enclave, such as the key pair generation, block scanning and generating mixing transaction. It is initialized and fed by the untrusted host process.
- Host process is an untrusted component that is outside of the enclave. It is responsible for initializing the enclave and interacting with the Bitcoin blockchain.
- Participants are Bitcoin users desiring to mix his bitcoins. Once participants confirm a valid coin mixing round, he sends a deposit transaction to the enclave. In addition, participants need to sign a unsigned mixing transaction once receiving it from the enclave.
- The Bitcoin blockchain is a distributed shared ledger, which builds trust relationship among peers that distrust each other without the involvement of a trusted authority.
- Public bulletin board is used for the availability of the mix-related data, such as a fresh public key of the enclave and some basic information about a mixing round. Participants fetch the mix-related information from the public bulletin boards. The public bulletin boards do not need to be trusted since tampering with in any form will lead to a failed verification by the IAS. There are several practical implementations of public bulletin boards, including IPFS[27], data servers with replications, and public blockchains such as Bitcoin itself.
- Intel Attestation Service allows a third party to be convinced that the correct application code has been loaded in an enclave.

2.3 Threat model

This work considers a strong adversary that could compromise the operating system (OS) of the mixer platform. A compromised OS can access any system resource that is under its control (e. g., access any physical memory). This means that the data between the enclave and the outside may be falsified or even dropped, or refuse to provide mixing services suddenly. The adversary can also leverage the compromised OS to actively read any message that should be sent or received by the mixer. In practice, the operator of the mixer platform may be malicious, and even if the operator is honest the mixer's OS may be compromised by a remote adversary. Furtherly, it is assumed that the malicious OS can collude with a non-negligible fraction of the Bitcoin miners. The fraction of computational power that is controlled by the adversary is assumed to be below half, otherwise the security of the Bitcoin system itself does not hold. The compromised OS has several attack goals. It aims to steal coins that honest users submitted to the mixer, deanonymize a targeted user transaction, or reduce the anonymity set size (i. e., the number of benign deposits that are mixed in a mixing round).

2.4 Design challenges

To build a secure TEE-based coin mixing platform, the following challenges need to be solved.

Guard against malicious operators. First, the malicious operator can block or modify the blockchain feeds from the outside world to the enclaved mixer, with fake or stale blocks that may "eclipse" the worldview of the enclave. Second, the malicious operator

can selectively prevent benign users from participating in the mixing service. This would reduce the anonymity set size of a mixing operation. Third, the malicious operator may manipulate the user's feedback data, or prevent the enclave from obtaining the user's signature for mixed transaction.

To overcome these challenges, FairMixer employs an indirect mixing request submission via the blockchain itself. Since the users interact only with the decentralized Bitcoin network, benign user participation is assured. FairMixer is also designed to be stateless in order to eliminate the adversary's advantage from rewinding the mixer's state to the past. Consider a situation where the enclave does not obtain the signature of party P_i. The enclave is not clear whether the operator misbehaved and did not forward the message to the enclave deliberately, or party P_i did not send the required signature to the operator. To resolve this conflict, this paper presents a misconduct monitoring mechanism via the blockchain and corresponding penalty mechanism is furtherly introduced. For the enforcement of the penalty, FairMixer requires that the operator pays a certain amount of collateral to the enclave. If the enclave is restarted or terminated by a malicious operator, the funds could be spendable after a time-lock.

Guard against malicious parties. After the enclave sends an unsigned mixing transaction to all parties, some malicious party may refuse to sign. This results in the abortion of the mixing mission. Even if all participants provide signatures, double spending may occur. Such attempts would amount to DOS attacks on the mixer, so FairMixer dictates that users must submit part of funds to be mixed to the enclave's address. This makes users have no incentives to refuse to sign or double spending since they would lose coins once caught.

Verification of blockchain. To ensure that a malicious operator cannot make up a fake blockchain, the enclave needs to design a reliable and efficient blockchain validation algorithm. FairMixer achieves this by simplifying the verification process typically by using a *checkpoint block* to serve as the initial starting point for verification. To further speed up the verification, FairMixer only validates correctness of block headers according to the Bitcoin consensus rules. Then the integrity of the transactions of each block will be verified by recomputing their Merkle root. Finally, when the enclave needs to verify whether a certain transaction, e. g. , the deposit transaction of participants, was integrated into a block, it sets a minimum number of blocks that must confirm a transaction. This makes it computationally infeasible to provide a valid-looking chain.

3 FairMixer description

In this section a more detailed description of the proposed protocol is given. The interaction between the operator, user_i and the blockchain is shown in Fig. 2. The interactions with the blockchain and TEE are presented firstly.

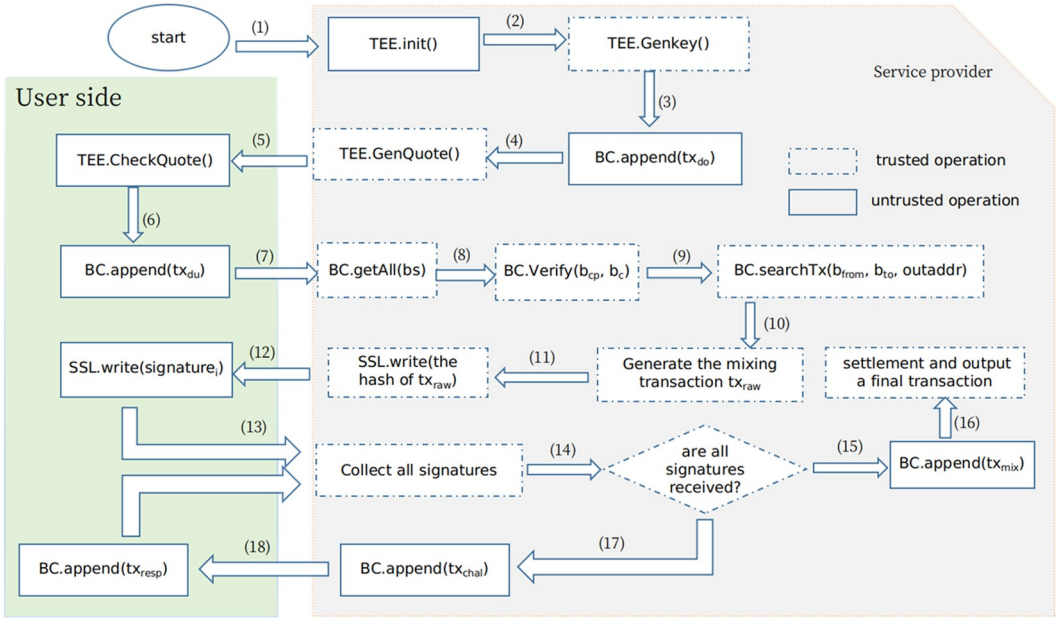


Fig. 2 The core process of the mixing protocol

3.1 Prerequisites

3.1.1 Modeling the bitcoin blockchain

Bitcoin is built on a blockchain that is maintained by special parties called miners under the Nakamoto consensus. The blockchain is composed of blocks linked in chronological order as a distributed ledger. Due to the consensus mechanism, a new valid block is appended every 10 minutes on average in Bitcoin[28-29].

Bitcoin transactions. A Bitcoin transaction may include multiple inputs and multiple outputs. A transaction input is a reference to the previous unspent transaction output. The total input amount should be greater than or equal to the total output amount while the difference between them is regarded as the reward fee for miners. Except inputs and outputs, there could be other information in a transaction, such as the time lock and extra data. Similar to FASTKITTEN[25], transactions are represented by tables as shown in Table 2.

Table 2 Transaction Tx

Transaction Tx	
Tx. Input:	coins from previous unspent transaction output
Tx. output:	send coins to the receiver
Tx. timelock:	coins can be spentable after timelock (optional)
Tx. data:	some data (optional)

Bitcoin scripts. Bitcoin utilizes a stack-based scripting language to describe the transaction details. Each transaction output contains a locking script called *scriptPubKey* for specifying who can spend the funds. Each transaction input is composed of *PreviousOutPoint* and an unlocking script called *scriptSig*. The *PreviousOutPoint* contains the hash of the previous unspent transaction (a. k. a. TxID) and a specified index of the output field. The *scriptSig* script stores the signature of the transaction sender. Only some specific signatures specified in the *scrpitPubKey* script could pass the transaction verifying. With an opcode called OP _ CHECKLOCKTIMEVERIFY[30], the transaction supports a time lock. This means a transaction output cannot be spent until the blockchain gets to the specified height. Utilizing an opcode called OP _ RETURN, up to 80 bytes of data could be stored in the transaction[31]. There are several optional transaction scripts in Bitcoin and two of them are applied in FairMixer: Pay-To-Script-Hash (P2SH) and Pay-To-Public-Key-Hash (P2PKH).

In order to facilitate the interaction with Bitcoin, BC is used to denote a simplified blockchain function. Internally it records a block counter c which starts initially with 0 and increases one every 10 minutes on average. Parties and the enclave can interact with the blockchain functionality BC using the following functions:

- *BC. append*(tx): this function submits a transaction tx that will be appended within the next δ blocks if it is a valid transaction. The parameter δ ensures the liveness of the blockchain system.
- *BC. getLatest*(): this function returns the newest block of the blockchain.
- *BC. getAll*(b_s): this function returns a list of blocks that extend $b_s = \{b_{s+1}, b_{s+2}, \cdots, b_c\}$.
- *BC. Verify*(b_{cp}, b_c): this function verifies the validity and integrity of blocks from the *checkpoint* block to the latest block.
- *BC. searchTx*(b_{from}, b_{to}, *outaddr*): this function is used to find all transactions that assigns coins to *outaddr* from the b_{from} block to the b_{to} block.

3.1.2 Modeling the SGX

As mentioned above, the implementation of a program using the SGX programming model, needs a trusted enclave component and a untrusted host process. To allow the remote third party to obtain evidence to prove that the target program is running safely in an enclave, Intel SGX provides remote attestation. The enclave application generates a remote evidence called quote signed with SGX private key through a quoting enclave[32]. The private key used here is embedded into the CPU by the CPU manufacturer and cannot be changed. Further, the quote is sent to the questioner who forwards it to Intel Attestation service for verification. FairMixer requires that the enclave application demonstrates remote evidence on the bulletin board so that users can directly obtain and verify it. There are several functions about SGX referred in FairMixer:

- *TEE. init*(): this function creates and initializes an enclave and then returns the enclave ID *eid*.
- *TEE. destroy*(*eid*): this function destroys the enclave identified as *eid*.
- *TEE. ECALL*(*funcA*): this function is called by the untrusted zone to run the function funcA in the trusted zone.
- *TEE. OCALL*(*funcB*): this function is called by the trusted zone to run the function funcB in the untrusted zone.
- *TEE. GenQuote*(): this function returns the evidence *quote _ report* which proves the target program is running safely in an enclave.
- *TEE. CheckQuote*(*quote report*): if quote report can be verified this function returns 1. Assuming that an attacker cannot forge a quote so that the function outputs 1.

3.2 Detailed protocol description

The proposed protocol proceeds in three phases: requesting phase, generation phase and final settlement phase.

(1) In the requesting phase, the host process creates an enclave and loads the coin mixing program into the enclave. The enclave executes the initialization function to generate the public key and private key of the enclave (pk_{mixer}, sk_{mixer}). In order to significantly improve the entropy of the random number used for key generation, the randomness in the scheme could aggregate the following sources: a hardware-based randomness instruction (RDRAND with SGX), the hashes of the latest block in blockchain and randomness provided by OS (via /dev/random). Then, the operator needs to make a deposit transaction tx_{do} (see Table 3). This transaction assigns q coins to the enclave as collateral by P2SH scripts. To be specific, the operator constructs a redeem script that follows a predefined format (see Table 4). This script allows the mixer to spend the deposit, and allows the operator to claim back funds after the lock-time c_{total}. The c_{total} is set to $5(n+\delta)$, where δ represents the liveness of the blockchain system (i.e., valid transactions are guaranteed to be included within the next δ blocks), n denotes the confirmation of the blockchain and 5 means that five Bitcoin transactions are needed at most.

Table 3 Transaction tx_{do}

Transaction Tx_{do}	
Tx. Input:	some unspent transaction output from the operator
Tx. output:	assign q coins to the hash of pk_{mixer}
Tx. timelock:	spendable for the enclave immediately while spendable for the operator after c_{total} blocks

Table 4 Redeem script

The Redeem Script
OP _ IF
< mixerpubkey > OP _ CHECKSIG
OP _ ELSE
< ctotal > OP _ CHECKLOCKTIMEVERIFY OP _ DROP
< operator _ pubkey > OP _ CHECKSIG
OP _ ENDIF

Once the enclave confirms that the tx_{do} is included in the blockchain ledger, it can obtain its block number and transaction ID. Afterwards, the general information of tx_{do}, the mix amount, the minimum mixing set size $N_{\min}$, public key pk_{mixer} and some metadata for remote verification are displayed on the bulletin board. After fetching the information, users desiring to mix his coins seek remote verification from Intel Attestation Service. This can ensure that the enclave program is correctly imported and executed. Furtherly, these users need to check the authenticity of Tx_{do} showed on the bulletin board. Afterwards, users output their deposit transactions Tx_{du}. Each Tx_{du} assigns p coins (e.g., 30% of mixing amount) and f (e.g., 2% of mixing amount) to the enclave as security deposit and the mixing fee, respectively. The change m of this transaction output will be fetched as the input of incoming mixing transaction. Note that the mixing fees are paid to some reputable charity organizations instead of the operator. This is because a malicious recipient can generate a large number of Sybil deposits without actually paying any mixing fees. Similar to tx_{do}, tx_{du} also constructs the P2SH script such that the funds can be spent by the enclave or refunded to users if the mixing protocol is aborted. Table 5 shows the structure of users' deposit transactions.

Table 5 Structure of the transaction Tx_{du}

Structure of the transaction Tx_{du}
Input:
scriptsig: < user signature > < user pubkey >
PreviousOutPoint: (preTxID, index)
Output:
Index: 0
Value: 0 BTC
scriptPubKey: OP _ RETURN < identifier > < encrypted out addr >
Index: 1
Value: $p+f$ BTC
scriptPubKey: OP _ HASH160 < Hash160 (redeem _ script) > OP _ EQUAL
Index: 2
Value: m BTC
scriptPubKey: OP _ DUP < < OP _ HASH160 < < user _ pubkey _ hash
< < OP _ EQUALVERIFY < < OP _ CHECKSIG

In addition, users need to delivery their specified output address and IP address via the deposit transaction. With IP address, the encalve can receive securely signatures of the mixing transaction from each participant. For privacy and security, users need to encrypt these two items with the pk_{mixer} of the enclave and store the ciphertext in OP _ RETURN field. The public key

encryption scheme for FairMixer must be compact since the ciphertext should be shorter than 80 bytes. This work uses the Elliptic Curve Integrated Encryption Scheme (ECIES) over the secp256k1 elliptic curve. ECIES is CCA secure[33] and its ciphertexts are quite compact for 128 bit security; a ciphertext is a tuple (R, c, d), where R is a 33-byte compressed elliptic curve point, c is the 24-byte encrypted output address and IP address with AES counter mode, and d is a 16-byte HMAC tag. An extra byte is set as the identifier for fast transaction scanning. A total of 74 bytes are used in the OP _ RETURN field.

(2) In the generation phase, the enclave verifies the validity and integrity of blocks fetched from the host process. After all blocks are checked, all the FairMixer-compliant deposit transactions are extracted. In the meantime, the enclave decrypts output addresses and IP addresses in the field of OP _ RETURN. When the number of transactions received caters to the minimum user size $N_{\min}$ specified on the bulletin board within a specified time, the protocol proceeds; otherwise the protocol jumps directly to the final settlement phase where deposits of users and the operator will be refunded.

Based on the mixing policy (i. e., $N_{\min}$), FairMixer determines the transaction set for a mixing round. Taking the changes of each participant's $\mathrm{tx}_{\mathrm{du}}$ as the transaction input and the output address given by each participant as the corresponding recipient, the enclave generates a giant mixing transaction $\mathrm{tx}_{\mathrm{raw}}$. Based on the shuffle( ) function that is implemented in Obscuro[8] with the Fisher-Yates shuffle algorithm[34], output addresses are shuffled using the trusted randomness generator. On the other hand, the enclave establishes SSL channels with all participants in order to collect valid signatures of input transactions. The enclave first sends the unsigned transaction $\mathrm{tx}_{\mathrm{raw}}$ to participants respectively. Then each participant checks if the mixing transaction $\mathrm{tx}_{\mathrm{raw}}$ contains his input transaction and output address with appropriate bitcoins. If all the information is correct, he signs for the giant mixing transaction using his private key and sends the signature back. If each participant and the operator behave honestly, a valid mixing transaction $\mathrm{tx}_{\mathrm{mix}}$ would be formed by recombining all valid signatures and submitted to the blockchain. Otherwise, the mixing mission fails and the protocol enters next phase.

Until the mixing transaction $\mathrm{tx}_{\mathrm{mix}}$ is appended to the blockchain successfully, current phase is completed. However, signatures may not be fetched smoothly. Note that the operator may be malicious and pretend that it actually did not receive a signature from some user Alice. On the other hand, Alice may be dishonest and did not send the signature. The enclave cannot distinguish between these two cases without additional information.

How to generate a proof to attribute the misbehavior to the dishonest party? This work introduces a misconduct monitoring mechanism. First, the operator issues a challenge transaction $\mathrm{Tx}_{\mathrm{chal}}$. It assigns t coins (t can be enough small) to a participant user_i and stores the hash of the unsigned mixing transaction $\mathrm{tx}_{\mathrm{raw}}$ and the signature of the message from the enclave. Note that the enclave's signature can guarantee the validity of the message. If the operator fails to provide a valid block containing $\mathrm{tx}_{\mathrm{chal}}$, the operator is supposed to be malicious since it does not issue the challenge transaction.

Challenge transaction $\mathrm{tx}_{\mathrm{chal}}(i)$

Tx. input: some unspent transaction output from the operator

Tx. output: spend t coins to user_i

Tx. data: hash of the unsigned mixing transaction

Once $\mathrm{tx}_{\mathrm{chal}}$ is appended in the blockchain, user_i can fetch the hash value of the unsigned mixing transaction. The party should reply with a transaction $\mathrm{tx}_{\mathrm{resp}}$ that includes his signature of the transaction and equal coins. If the enclave confirms the transaction $\mathrm{tx}_{\mathrm{resp}}$ is included in the blockchain, It is considered that the instability network is responsible for the event. Otherwise, it is concluded that user_i has malicious behavior.

Challenge transaction $\mathrm{tx}_{\mathrm{resp}}(i)$

Tx. input: output of challenge Transaction $\mathrm{tx}_{\mathrm{chal}}$

Tx. output: spend t coins to the operator

Tx. data: signature data

Once it is proved that Alice or the operator misbehaves, the protocol jumps to the final settlement phase, where related party gets punishment.

(3) In the final settlement phase, the enclave outputs a final transaction $\mathrm{tx}_{\mathrm{final}}$ that assign all the coins received by the enclave to all honest parties. The details of assignment are determined by the system status that has four different implications: the failure of setup where the number of received transactions does not satisfy the minimum user size, successful completion of the mixing mission, an aborted mixing protocol caused by the malicious operator or parties. In the sequel, out_{repu}, out_{op}, out_i, out_k denote the distributed amount of bitcoins to the reputable charity organization, the operator and each benign party, respectively.

- if the setup of the mixing protocol fails, all mixing fees and collaterals are claimed back after the lock-time. To be specific:

$out_{repu} = 0$, $out_{op} = q$, $out_i = p + f$.

- For the second case, all mixing fees are paid to charity organizations. In addition, the operator's deposit are refunded to the operator while all collaterals from users are claimed back by users respectively. To be specific: $outr_{epu} = nf$, $out_{op} = q$, $out_i = p$.
- If the operator aborts the protocol, it will lose its deposit that will be allocated evenly to all participants. Furthermore, all users will get their initial deposits and fees back. To be specific: $out_{repu} = 0$, $out_{op} = 0$, $out_i = p + f + q/n$.
- In the case that a party $user_i$ is caught cheating by the enclave, its deposit and mixing fee will be assigned evenly to all honest parties and the operator. Moreover, all honest parties and the operator will get their initial deposits or fees back. To be specific: $out_{repu} = 0$, $out_{op} = q + (p + f)/n$, $out_i = p + f + (p + f)/n$.

4 Security

This section gives the underlying security considerations from five aspects: guaranteed minimum mixing size, resistant to DoS attacks, preventing selectively benign participants, eclipse attacks and Sybil attacks.

To clarify the prospects of a de-anonymization attack, a simple lemma is provided firstly. Let $u(x)$ denote a negligible function, i. e., for every positive polynomial $poly(\ )$ there exists a sufficiently large n so that $\forall m > n$: $u(m) < 1/p(m)$.

Lemma 1 Suppose that H is the number of honest participants in a mixing round of *FairMixer*. Given an input of an honest user U, the adversary can guess the corresponding output of U in the mixing task with probability $1/H + u(\kappa)$ at most, where κ is the security parameter of a FairMixer instantiation.

Proof First, two events are defined: E_1 denotes that the adversary correctly guesses the output address of a user, and E_2 denotes that the adversary cracks the encryption of the output address. Normally, if the adversary does not know the private key of the enclave, his guess will be correct with the probability $1/H$. Otherwise, when the adversary breaks the encryption of the output addresses, the probability that the adversary can guess the output of U is more than $1/H$. Since the security parameter of the encryption scheme is κ, it is quite plausible to suppose that the adversary can crack the cryptography with only $u(\kappa)$ probability:

$$\begin{aligned} P(E_1) &= P(E_1 \cap \neg E_2) + P(E_1 \cap E_2) \\ &= P(E_1 \mid \neg_{E2}) P(\neg E_2) + P(E_1 \mid E_2) P(E_2) \\ &\leqslant P(E_1 \mid \neg E_2) + P(E_2) \leqslant 1/H + u(\kappa) \end{aligned}$$

Given Lemma 1, it is time to show that H will indeed be large.

4.1 Guaranteed minimum mixing set size

A minimum mixing set size acts as a lower bound on the quality of the mixing service. This work leverages the TEE integrity guarantee to enforce a minimum mixing set size $N_{\min}$. This policy can be verified by inspecting the attested enclave code. When the number of valid users is less than $N_{\min}$, the design idea of the FairMixer enclave is to refund the users' funds not to continue the protocol. The reason is that the honest users pay a considerable mixing fee in order to be secure against Sybil attacks. Hence, they expect a large enough mixing transaction in return.

4.2 Resistant to DoS attacks

Similar to all the centralized mixing services, FairMixer is susceptible to DoS attacks on the mixer's server. During the protocol, the operator may discard or tamper with messages from the participants. Furthmore, some participants may refuse to sign for the mixing transaction. This is a join-then-abort attack in most decentralized mixers. These events amount to DoS attacks and will cause failure of mixing protocol. In order to resist above DoS attacks, a punishment mechanism is introduced. With the mechanism, the funds of malicious operator or participants can be allocated evenly to honest users once they are caught cheating by the enclave. To this end, it requires that both the service provider and the participants send time-locks coins to the enclave at the beginning of a mixing round. If considering only incentive-driven operator or participants, the protocol protects against such DoS attacks. Preventing a malicious OS from conducting an eclipse attack that creates a fake blockchain view to the enclave is another form of DoS attacks and will be discussed in Section 4.4.

4.3 Preventing selectively benign participants

FairMixer's design prevents a compromised OS from selectively dropping participants. In FairMixer, a user first fetches FairMixer's information (i. e., pk_{mixer}). Then he uses the meta-data to verify that FairMixer is running the correct code inside an enclave, and that pk_{mixer} was generated by FairMixer's enclave code. Instead of querying the mixer's server directly, the participant fetches the information from public bulletin boards. Users will join in the mixing only if they confirm that the data was uploaded to several of these bulletin boards. Moreover, each user is advised to upload the information to additional public bulletin boards for wider dissemination. If the mali-

cious OS prevents FairMixer from publishing the metadata to all of the public bulletin boards, then this would be a straightforward DoS attack. Once the user fetches the data, he verifies FairMixer's code and the pk_{mixer} value by using the trusted remote attestation service.

Then the user joins in a mixing round by sending a deposit transaction to the enclave. FairMixer needs to scan the transaction from the blockchain directly. This makes it hard and expensive to exclude benign participants selectively. The malicious OS can still try to drop users from the mixing by conducting an eclipse attack that creates a fake blockchain view to the enclave (see subsection 4.4).

4.4 Eclipse and state-rewind attacks

Assuming that an adversary A1 controls less than 50% fraction of the computational power of the Bitcoin blockchain and also has physical access to the FairMixer server. Therefore, A1 can cut the communication between the enclave and the network and modify the blockchain feeds.

Assuming a raw enclave implementation, A1 can conduct an Eclipse attack[35] as the following example demonstrates: A1 cuts the enclave off from the Bitcoin network and feeds it with fake blocks, which makes some fake transactions involved in a bunch of honest users' transactions. To resist this attack, FairMixer depends on the fact that the rate at which A1 can feed fake blocks to the enclave is at least twice slower than in the absence of an attack. This is because A1 controls less than half of the computational power. Assuming that the enclave has a trusted clock, the enclave can introduce a rule which requires the enclave to wait for additional confirmations if the blocks arrive too slowly as the confirmation is usually 6 blocks in Bitcoin. Due to the consensus rule of proof of work (POW), the more fake blocks to be created, the less likely the attacker could generate these blocks successfully. Therefore, an optional mechanism is added to the proposed scheme: once the average interval between multiple blocks provided by the operator is detected to be greater than 15 min, the enclave will be triggered to wait for N extra block confirmation before accepting any deposits.

Even when eclipse attacks are not blocked, the attacks are futile as a de-anonymization technique due to FairMixer's stateless enclave. Such attempts would amount only to a DoS attack on the mixer. To be specific, the resulting giant mix transaction will never be accepted by the Bitcoin network. Honest users will instead retrieve their coins (i.e., mixing fee and collateral for mixing) after the timeout expires. Although the failure of coin mixing will not lead honest users to lose their deposits, the unlinkability between payers and payees would be in danger when fake transactions are more than benign ones.

In any case, FairMixer completely prevents state-rewind attacks by a stateless design for the FairMixer enclave. Instead of storing some system state for consecutive operation and recovery from reboots, FairMixer stores no states of the mixer. Each time that the enclave code is initialized, it uses secure randomness sources to generate a unique receiving address. Users who wish to deposit to the new receiving address should encrypt a fresh output address. It means that mixing requests in current mixing round will be disjoint from any other set of requests in other mixing round.

4.5 Sybil attacks

Assuming that an attacker (A2) can reduce the size of benign deposits in a mixing round while increasing the Sybil deposits, A2 thus can effectively reduce the anonymity set size without being detected by the honest users. FairMixer deters Sybil attacks by enforcing mixing fees (on top of the transaction fees), as with other cryptocurrency mixers[7,9]. However, it does not prevent cost-insensitive adversaries from launching Sybil attacks that flood the mixer with large numbers of adversarially-generated deposits. To highlight the impracticality of such attacks, it is worth analyzing the expected cost of Sybil attacks on FairMixer.

As the enclave guarantees a fair and random selection of the mixing requests, the probability of honest and Sybil deposits being selected to enter a mixing round is the same. Let H and S be the number of benign and Sybil deposits submitted to FairMixer for the current mixing round, respectively. $N_{\max}$ denotes the maximum mixing capacity of a single mixing round, e.g., 430 transactions assuming a standard Bitcoin transaction size of 100 kB. If the total number of mixing requests in this round does not exceed $N_{\max}$ (i.e., $H + S \leqslant N_{\max}$), the effective anonymity set size is guaranteed to be H. This is attributed to the indirect mixing request submission property. Otherwise, some mixing requests will be excluded from the current mixing round. That is, the expected anonymity set size is $N_{\max} \times H$, which is inversely $H + S$ proportional to the size of the Sybil deposits S. Thus, in order to obtain a small effective anonymity set size, A2 needs a prohibitively large number of Sybil deposits S.

For example, consider a conservative scenario in which there are $H = 100$ honest requests and the adversary aims to reduce the effective anonymity set size to

20. To this end, the adversary needs to create 2400 Sybil deposits (assuming N_{max} = 500). With 2% mixing fees (where typical mixing fees on the market are in the range of 1% ~3%), this attack costs 48 bitcoins, where the denomination of the mixing round is 1 bitcoin.

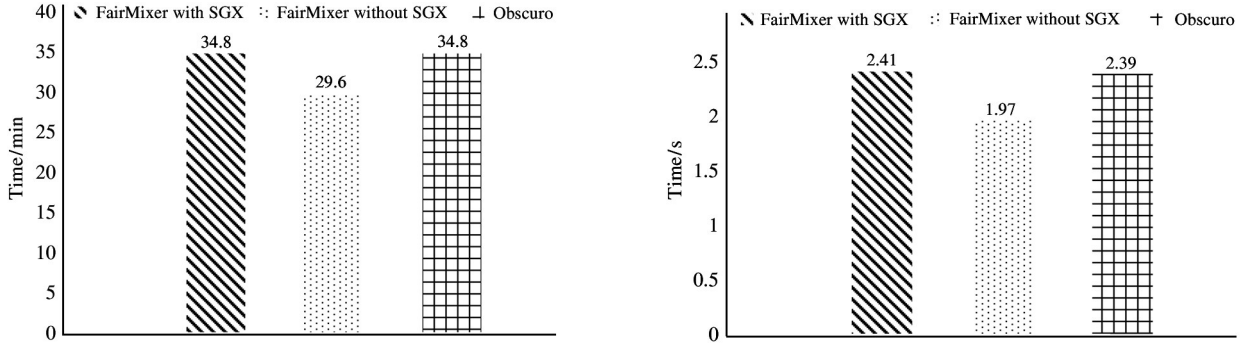
5 Performance evaluation

A prototype of the proposed scheme has been implemented by porting a Bitcoin Core's codebase version v0. 13. 1[36], an openssl library called SGX-OpenSSL[37] and the ECDSA library libsecp256k1[38] into the enclave while following the programming model of Intel SGX application. SGX-OpenSS contains modified OpenSSL codes and necessary wrapper functions to be used for SGX-enabled applications. The prototype minimizes the risk of side-channel attacks by using constant-time code[38] for transaction signing and signature verification in FairMixer. It realizes the generation, signing, signature verification and retrieval of all Bitcoin transactions in the enclave, as well as the establishment of SSL secure channel.

A number of performance measurements of FairMixer are performed on the following two machines: one machine running Ubuntu 16.04.5 LTS with an Intel(R) Xeon(R) Bronze 3104 CPU clocked at 1.7 GHz and 32 GB RAM, which does not support SGX and plays the role of the mixing service provider; the other a machine running Ubuntu 14.04.4 LTS on an Intel i5-3210M CPU clocked at 2.5 GHz with 6 GB RAM, which acts all participants involved in the protocol.

To evaluate the performance of trusted functions inside the enclave, FairMixer is tested in Bitcoin Regression Test Mode (i.e., RegTest). Each experiment is run 20 times with two versions of FairMixer, one in SGX application model and one without using SGX, and the average results are used to measure the added complexity of SGX-related operations. In addition, the proposed scheme is compared with Obscuro[8] in some aspects.

Block validation. The time taken to verify the up-to-date Bitcoin blockchain is measured. As of this writing, the latest block is approximately 698 000 blocks ahead of the most recent blockchain checkpoint. A checkpoint block is set to serve as the initial starting point for verification and measure the time to verify the block headers of all the blocks (in total 200 000 blocks) after the checkpoint. Fig. 3(a) shows that it takes approximately 34.8 min, 29.6 min and 34.9 min for the FairMixer with SGX, the FairMixer without SGX and Obscuro, respectively. That is, it takes approximately 0.01 s to verify a block, which is easily acceptable in practice.



(a) Verifying 200 000 blocks (b) Scanning transactions in 100 blocks

Fig. 3 Measured time for verifying blocks and scanning transactions

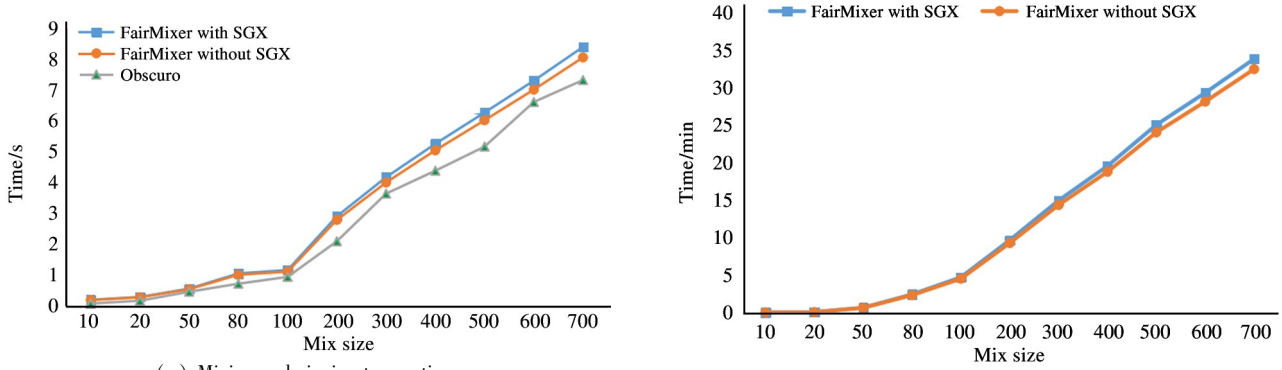
Scanning for FairMixer deposits. The time that FairMixer takes to scan Bitcoin blocks and extract FairMixer-compliant deposit transactions is measured. Specifically, assume a conservative scenario in which it should scan 100 blocks to search for 2000 FairMixer deposits transactions among a total of 4000 transactions. Fig. 3(b) shows that FairMixer scans the deposits almost as fast as Obscuro does (2.41 s with SGX, 1.97 s without SGX and 2.39 s for Obscuro).

Mixing and signing transaction. The time taken to generate a giant unsigned mixing transaction and collect signatures from all participants is measured. The shuffling and signature acquisition with different sizes of the mixing set are tested, ranging from 10 to 700 transactions. Fig. 4(a) shows that the cost time increases as the size of the mixing set increases and SGX programming model brings a very small extra execution time (approximately 4%). Furthermore, FairMixer can mix seven hundred input transactions within seconds (specifically, 700 inputs in 9.34 s), demonstrating that a practical deployment of FairMixer is scalable and efficient in mixing a large set of transactions.

FairMixer needs more time than Obscuro in terms of signature acquisition since it introduces extra communication cost.

Settlement transaction. The time taken to generate the final settlement transaction is measured. Note that Obscuro does not introduce settlement transaction since it does not consider how to resist to DoS attack from a malicious operator. Assuming all parties comply with the protocol, Fig. 4 (b) shows that the total time required by FairMixer varies from 0.02 s to 33.89 s when the size of mixing set range from 10 to 700. This is acceptable in practice since it is far less than the time of a block confirmation.

Trusted computing base. The entire TCB of FairMixer consists of approximately 1670 source lines of code (SLoC) for the functionalities that run inside the SGX enclave. This figure excludes the Bitcoin Core implementation which contributes 1293 SLoC and two widely used cryptographic libraries (i. e., libsecp256k1 and OpenSSL).



(a) Mixing and signing transactions

(b) Generating the settlement transaction

Fig. 4 Measured time for mixing and signing transactions and generating the settlement transaction

6 Conclusion

Fungibility is one of ideal feature for cryptocurrencies. However, the linkability of the mainstream cryptocurrency transactions seriously damages this feature. To break the dilemma, this paper presents a centralized coin mixing scheme with privacy and fairness by using TEE technologies. FairMixer is a best-of-both-worlds design, because it combines the advantages of the existing decentralized and centralized mixers but circumvents their respective disadvantages. FairMixer demonstrates efficient operation and strong privacy guarantees which can facilitate the development of Bitcoin and other cryptocurrencies that are subject to linkable transactions.

References

[1] MEIKLEJOHN S, POMAROLE M, JORDAN G, et al. A fistful of bitcoins: characterizing payments among men with no names[J]. *Communications of the ACM*, 2016, 59(4): 86-93

[2] RON D, SHAMIR A. Quantitative analysis of the full bitcoin transaction graph[C]// International Conference on Financial Cryptography and Data Security, Okinawa, Japan, 2013: 6-24

[3] REID F, HARRIGAN M. An analysis of anonymity in the bitcoin system[C]//2011 IEEE 3rd International Conference on Privacy, Security, Risk and Trust (PASSAT)/2011 IEEE 3rd International Conference on Social Computing (SocialCom), Boston, USA, 2011: 1318-1326

[4] ANDROULAKI E, KARAME G O, ROESCHLIN M, et al. Evaluating user privacy in bitcoin[C]// International Conference on Financial Cryptography and Data Security, Berlin, Germany, 2013: 34-51

[5] BONNEAU J, NARAYANAN A, MILLER A, et al. Mixcoin: anonymity for bitcoin with accountable mixes[C]// International Conference on Financial Cryptography and Data Security, Berlin, Germany, 2014: 486-504

[6] VALENTA L, ROWAN B. Blindcoin: blinded, accountable mixes for bitcoin[C]// International Conference on Financial Cryptography and Data Security, Berlin, Germany, 2015: 112-126

[7] HEILMAN E, ALSHENIBR L, BALDIMTSI F, et al. Tumblebit: an untrusted bitcoin-compatible anonymous payment hub[C]//The 24th Annual Network and Distributed System Security Symposium, San Diego, USA, 2017

[8] TRAN M, LUU L, KANG M S, et al. Obscuro: a bitcoin mixer using trusted execution environments[C]// Proceedings of the 34th Annual Computer Security Applications Conference, San Juan, USA, 2018: 692-701

[9] MAXWELL G. CoinJoin: Bitcoin privacy for the real world [EB/OL]. https://bitcointalk.org/index.php?topic=279249: Bitcoin Forum, [2021-08-05]

[10] RUFFING T, MORENO-SANCHEZ P, KATE A. CoinShuffle: practical decentralized coin mixing for bitcoin[C]//European Symposium on Research in Computer Security, Wroclaw, Poland, 2014: 345-364

[11] RUFFING T, MORENO-SANCHEZ P, KATE A. P2P mixing and unlinkable bitcoin transactions[C]//The 24th

Annual Network and Distributed System Security Symposium, San Diego, USA, 2017: 1-15
[12] JAN, HENRIK, ZIEGELDORF, et al. Secure and anonymous decentralized bitcoin mixing[J]. *Future Generations Computer Systems*, 2018, 80: 448-466
[13] XIAO R, REN W, ZHU T, et al. A mixing scheme using a decentralized signature protocol for privacy protection in bitcoin blockchain[J]. *IEEE Transactions on Dependable and Secure Computing*, 2019, 18(4): 1793-1803
[14] LIU Y, LIU X, TANG C, et al. Unlinkable coin mixing scheme for transaction privacy enhancement of bitcoin [J]. *IEEE Access*, 2018(6): 23261-23270
[15] BAUM C, DAVID B, DOWSLEY R. Insured MPC: efficient secure computation with financial penalties[C]// International Conference on Financial Cryptography and Data Security, Kota Kinabalu, Malaysia, 2020: 404-420
[16] COHEN R, LINDELL Y. Fairness versus guaranteed output delivery in secure multiparty computation[J]. *Journal of Cryptology*, 2017, 30(4): 1157-1186
[17] ZHANG F, ZHANG H. SoK: a study of using hardware-assisted isolated execution environments for security [C]// Proceedings of the Hardware and Architectural Support for Security and Privacy, Seoul, Korea, 2016: 1-8
[18] MCKEEN F, ALEXANDROVICH I, BERENZON A, et al. Innovative instructions and software model for isolated execution[C]// Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, Tel-Aviv, Israel, 2013: 1-8
[19] ANATI I, GUERON S, JOHNSON S, et al. Innovative technology for CPU based attestation and sealing[C]// Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, Tel-Aviv, Israel, 2013: 1-7
[20] HOEKSTRA M, LAL R, PAPPACHAN P, et al. Using innovative instructions to create trustworthy software solutions[C]// Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, Tel-Aviv, Israel, 2013: 2487726-2488370
[21] XU Y, CUI W, PEINADO M. Controlled-channel attacks: deterministic side channels for untrusted operating systems[C]//2015 IEEE Symposium on Security and Privacy, San Jose, USA, 2015: 640-656
[22] VANBULCK J, MINKIN M, WEISSE O, et al. Foreshadow: extracting the keys to the intel SGX kingdom with transient out-of-order execution[C]// The 27th USENIX Security Symposium, Baltimore, USA, 2018: 991-1008
[23] LIND J, NAOR O, EYAL I, et al. Teechain: a secure payment network with asynchronous blockchain access [C]// Proceedings of the 27th ACM Symposium on Operating Systems Principles, Huntsvile, USA, 2019: 63-79
[24] ZHANG F, HE W, CHENG R, et al. The Ekiden platform for confidentiality-preserving, trustworthy, and performant smart contracts[J]. *IEEE Security & Privacy*, 2020, 18(3): 17-27
[25] DAS P, ECKEY L, FRASSETTO T, et al. Fastkitten: practical smart contracts on bitcoin[C]// The 28th USENIX Security Symposium, Santa Clara, USA, 2019: 801-818
[26] ZHANG F, CECCHETTI E, CROMAN K, et al. Town crier: an authenticated data feed for smart contracts[C]// Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 2016: 270-282
[27] BENET J. Ipfs-content addressed, versioned, p2p file system[EB/OL]. https: // arxiv. org/pdf/1407. 3561. pdf: arXiv, (2014-07-14), [2021-08-05]
[28] GARAY J, KIAYIAS A, LEONARDOS N. The bitcoin backbone protocol: analysis and applications[C]// Annual International Conference on the Theory and Applications of Cryptographic Technques, Sofia, Bulgaria, 2015: 281-310
[29] NAKAMOTO S. Bitcoin: a peer-to-peer electronic cash system[J]. *Social Science Electronic Publishing*, DOI: 10.2139/ssrn. 3440802
[30] TODD P. Bip 65: Op checklocktimeverify[EB/OL]. https: // github. com/bitcoin/bips/blob/master/bip-0065. mediawiki: github, [2021-08-05]
[31] BARTOLETTI M, POMPIANU L. An analysis of bitcoin op return metadata[C]// International Conference on Financial Cryptography and Data Security, Sliema, Malta, 2017: 218-230
[32] PASS R, SHI E, TRAMER F. Formal abstractions for attested execution secure processors[C]// Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, 2017: 260-289
[33] BLACK P E. Fisher-yates shuffle, dictionary of algorithms and data structures [J]. *National Institute of Standards and Technology*, *retrieved*, 2007, doi: 10. 18434/T4/1422485
[34] HEILMAN E, KENDLER A, ZOHAR A, et al. Eclipse attacks on bitcoin's peer-to-peer network[C]// The 24th USENIX Security Symposium, Washington DC, USA, 2015: 129-144
[35] GARAY J, KIAYIAS A, LEONARDOS N. The bitcoin backbone protocol with chains of variable difficulty[C]// Annual International Cryptology Conference, Santa, Barbara, USA, 2017: 291-323
[36] MUGWARA T, YU L, HARARE Z. Bitcoin core version 0.13.0 released[EB/OL]. https: // bitcoin. org/en/release/v0. 13. 1, [2021-08-05]
[37] JUHYENG H. SGX-OpenSSL[EB/OL]. https: // github. com/sparkly9399/SGX-OpenSSL: github, [2021-08-05]
[38] WUILLE P. libsecp256k1[EB/OL]. https: // github. com/bitcoin-core/secp256k1: github, [2021-08-05]

GONG Xunwu, born in 1990. He is working toward the Ph. D degree in computer science at University of Chinese Academy of Sciences. He received the B. S. degree in electronic science and technology from Nanchang Hangkong University in 2012 and M. S. degree in electronics and communication engineering from University of Chinese Academy of Sciences in 2016. His research focuses on access control, applied cryptography, data security and privacy.