CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin*

Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate

MMCI, Saarland University
`{tim.ruffing,pedro,aniket}@mmci.uni-saarland.de`

Abstract. The decentralized currency network Bitcoin is emerging as a potential new way of performing financial transactions across the globe. Its use of pseudonyms towards protecting users' privacy has been an attractive feature to many of its adopters. Nevertheless, due to the inherent public nature of the Bitcoin transaction ledger, users' privacy is severely restricted to *linkable anonymity*, and a few Bitcoin transaction deanonymization attacks have been reported thus far.
In this paper we propose CoinShuffle, a completely decentralized Bitcoin mixing protocol that allows users to utilize Bitcoin in a truly anonymous manner. CoinShuffle is inspired by the accountable anonymous group communication protocol Dissent and enjoys several advantages over its predecessor Bitcoin mixing protocols. It does not require any (trusted, accountable or untrusted) third party and it is perfectly compatible with the current Bitcoin system. CoinShuffle introduces only a small communication overhead for its users, while completely avoiding additional anonymization fees and minimizing the computation and communication overhead for the rest of the Bitcoin system.

1 Introduction

Bitcoin [3] is a fully decentralized digital crypto-currency network that does not require any central bank or monetary authority. Over the last few years we have observed an unprecedented and rather surprising growth of Bitcoin and its competitor currency networks [4,5,6,7]. Despite a few major hiccups, their market capitalizations are increasing tremendously [8]. Many now believe that the concept of decentralized crypto-currencies is here to stay.

Nevertheless, these decentralized currency systems are far from perfect. Traditional payment systems rely on a trusted third party (such as a bank) to ensure that money cannot be spent twice. Decentralized currencies such as Bitcoin employ a global replicated append-only transaction log and proof-of-work (POW) instead to rule out double-spending. This requires managing a public ledger such that every transaction is considered to be verified only after it appears in the ledger. However, given that the Bitcoin transactions of a user (in particular, of

* The full version of this paper is available on the project webpage [1], and we offer a discussion thread about CoinShuffle on the Bitcoin forum [2].

her pseudonyms, called *Bitcoin addresses*) are linkable, the public transaction ledger constitutes a significant privacy concern: Bitcoin's reliance on the use of pseudonyms to provide anonymity is severely restricted.

Several recent studies analyzing the privacy implications of Bitcoin indicate that Bitcoin's built-in privacy guarantees are not satisfactory. Barber et al. [9] observe that Bitcoin exposes its users to the possible linking of their Bitcoin addresses, which subsequently leads to a weak form of anonymity. Meiklejohn et al. [10] demonstrate how to employ a few basic heuristics to classify Bitcoin addresses that are likely to belong to the same user; this is further refined by Spagnuolo, Maggi, and Zanero [11]. Koshy, Koshy, and McDaniel [12] show that it is also possible to identify ownership relationships between Bitcoin addresses and IP addresses.

Recently, some efforts have been made towards overcoming the above attacks and providing stronger privacy to the Bitcoin users by *mixing* multiple transactions to make input and output addresses of transactions unlinkable to each other. In this direction, some third-party Bitcoin mixing services [13,14,15] were first to emerge, but they have been prone to thefts [10]. Mixcoin [16] allows to hold these mixing services accountable in a reactive manner; however, the mixing services still remain single points of failure and typically require additional mixing fees. Zerocoin [17] and its successors [18,19,20] provide strong anonymity without any third party, but lack compatibility with the current Bitcoin system.

Maxwell proposes CoinJoin [21] to perform mixing in a perfectly compatible manner with Bitcoin, while ensuring that even a malicious mixing server cannot steal coins. CoinJoin is actively used in practice [22] but suffers from a substantial drawback: The mixing server still needs to be trusted to ensure anonymity, because it learns which coins belong to which user. To tackle this problem, Maxwell mentions the possibility to use secure multi-party computation (SMPC) in CoinJoin to perform the mixing in an oblivious manner. Yang [23] proposes a concrete scheme based on SMPC sorting. However, against a fully malicious attacker, generic SMPC as well as state-of-art SMPC sorting [24,25] is not yet practical for any reasonable number of parties required in mixing to ensure a good level of anonymity. Furthermore, it is not clear how to ensure robustness against DoS attacks in these approaches, because a single user can easily disrupt the whole protocol while possibly remaining unidentified. Consequently, defining a practical and secure mixing scheme is considered an open problem by the Bitcoin community [26,27,28].

Our Contribution. We present CoinShuffle, a completely decentralized protocol to allow users to mix their coins with those of other interested users. CoinShuffle is inspired by CoinJoin [21] to ensure verifiability and by the accountable anonymous group communication protocol Dissent [29] to ensure anonymity and robustness against active attacks. The key idea is similar to decryption mix networks, and the protocol requires only standard primitives such as signatures and public-key encryption. CoinShuffle is a practical solution for the Bitcoin mixing problem and its distinguishing features are as follows:

No Third Party: CoinShuffle preserves Bitcoin's decentralized trust ideology: It is executed exclusively by the Bitcoin users interested in unlinkability for their Bitcoin transactions, and it does not require any trusted, accountable or untrusted third party. The unlinkability of transactions is protected as long as at least any two participants in a run of the protocol are honest.

Compatibility: CoinShuffle is fully compatible with the existing Bitcoin network. Unlike other decentralized solutions, it works immediately on top of the Bitcoin network without requiring any change to the Bitcoin rules or scripts.

No Mixing Fee: In absence of a third party that acts as a service provider, CoinShuffle does not charge its users any additional mixing fees. It also performs well in terms of Bitcoin transaction fees, because the participants are only charged the fee for a single mixing transaction.

Small Overhead: Our performance analysis demonstrates that CoinShuffle introduces only a small communication overhead for a participant (less than a minute for an execution with 20 participants), while the computation overhead remains close to negligible. Finally, CoinShuffle introduces only minimal additional overhead for the rest of the Bitcoin network.

2 Background

In this section we present the basics of the Bitcoin protocol and explain Bitcoin mixing, the most prevalent approach to strengthening users' anonymity in Bitcoin. We explain only the aspects of Bitcoin that are relevant for Bitcoin mixing and refer the reader to the original Bitcoin paper [3] for further details.

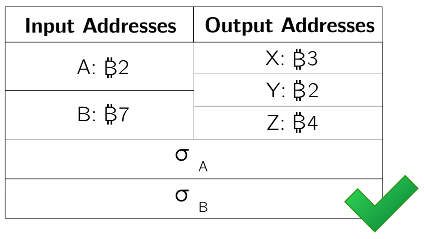
2.1 Bitcoin

Bitcoin (symbol: ₿) is a digital currency run by a decentralized network. The Bitcoin network maintains a public ledger (called the *blockchain*) whose purpose is to reach consensus on the set of transactions that have been validated so far in the network. As long as the majority of computation power in the system is honest, transactions accepted by the system cannot be changed or invalidated, thus preventing double-spending of money.

User accounts in the Bitcoin system are identified using pseudonymous addresses. Technically, an address is the hash of a public key of a digital signature scheme. To simplify presentation, we do not differentiate between the public key and its hash in the remainder of the paper. Everybody can create an arbitrary number of addresses by creating fresh key pairs.

Transactions. The owner of an address uses the corresponding private key to spend coins stored at this address by creating transactions. In the simplest form, a transaction transfers a certain amount of coins from one address (the *input address*) to another address (the *output address*). While multiple sets of coins may be stored at one address, we assume in the remainder of the paper that only one set of coins is stored at an address; these coins can only be spent together. This simplification is purely for the sake of readability.

As depicted in Fig. 1, transactions can include multiple input addresses as well as multiple output addresses. Three conditions must be fulfilled for a transaction to be valid: First, the coins spent by the transaction must not have been already spent by another transaction in the blockchain. Second, the sum of the input coins must equal the sum of the output coins.[1] Third, the transaction must be signed with the private keys corresponding to all input addresses.



Input Addresses	Output Addresses
A: ₿2	X: ₿3
	Y: ₿2
B: ₿7	Z: ₿4
σ_A	
σ_B	✓

Fig. 1. A valid Bitcoin transaction with multiple input addresses and multiple output addresses. This transaction is signed using both the private key for input address A and the private key for input address B; the corresponding signatures are denoted by σ_A and σ_B, respectively.

2.2 Bitcoin Mixing

The most prevalent approach to improve anonymity for Bitcoin users is the idea of hiding in a group by *Bitcoin mixing*: the users in the group exchange their coins with each other to hide the relationship between the user and the coin from an external observer. Assume that in a group of several users, every user owns exactly one Bitcoin (₿ 1). In the simplest form, mixing is done with the help of a trusted third-party mixing server, the *mix*: every user sends a fresh address in encrypted form to the mix and transfers her coin to the the mix. Then, the mix decrypts and randomly shuffles the fresh addresses and sends ₿ 1 back to each of them. While such public mixes are deployed in practice [30,13,14,15], they suffer from two severe drawbacks: First, the mix might just steal the money and never return it to the users. Second, the mix learns the permutation of the output addresses. Thereby, user's anonymity relies on the assumption that the mix does not log or reveal the permutation.

2.3 Bitcoin Mixing With A Single Transaction

To solve the problem that the mix can steal the money, Maxwell proposes CoinJoin [21]: Assume a group of users would like to mix their coins. In CoinJoin this group jointly generates one single *mixing transaction* containing the users' current addresses as inputs and the shuffled fresh addresses as outputs. Recall that a transaction with several input addresses is only valid if it has been signed with

[1] In practice, a small transaction fee might be required. In that case, the sum of input coins must exceed the sum of the output coins by the amount of the fee.

all keys belonging to those input addresses. Thus each user can verify whether the generated mixing transaction sends the correct amount of money to her fresh output address; if this is not true the user just refuses to sign the transaction and the protocol aborts without transferring any coins.

Several implementations of CoinJoin are already actively being used [22,31], at least one more implementation is under way [32], and the Bitcoin developers consider adding CoinJoin to the official Bitcoin client [33]. Still, the problem that the mix learns the relation between input and output addresses persists, and no fully anonymous and efficient solution has been proposed to the best of our knowledge.

3 Problem Definition

Here, we define the properties that a Bitcoin mixing protocol should satisfy, and explain the threat model under which we would like to achieve these properties.

3.1 Design Goals

A Bitcoin mixing protocol must achieve the following security and privacy goals.

Unlinkability. After a successful Bitcoin mixing transaction, honest participants' input and output addresses must be unlinkable.
Verifiability. An attacker must not be able to steal or destroy honest participants' coins.
Robustness. The protocol should eventually succeed even in the presence of malicious participants.

Besides ensuring security and privacy, a Bitcoin mixing protocol must additionally overcome the following challenges:

Compatibility. The protocol must operate on top of the Bitcoin network, and should not require any change to the existing system.
No Mixing Fees. The protocol should not introduce additional fees specifically required for mixing. As every mixing transaction necessarily requires a Bitcoin transaction fee, the protocol must ensure that this transaction fee remains as low as possible.
Efficiency. Even users with very restricted computational capacities should be able to run the mixing protocol. In addition, the users should not be required to wait for a transaction to be confirmed by the Bitcoin network during a run of the protocol, because this inherently takes several minutes.[2]
Small Impact on Bitcoin. The protocol should not put a large burden on the efficiency of the Bitcoin network. In particular, the size of the executed transactions should not be prohibitively large because all transactions have to be stored in the blockchain and verified by all nodes in the network.

[2] A confirmation takes 10 minutes on average. As mixing inherently requires at least one transaction, it is acceptable to wait for a confirmation at the end of the protocol, provided the protocol fails gracefully if the transaction is not accepted by the network.

3.2 Non-Goals

Bitcoin users who wish to participate in a mixing protocol need a bootstrapping mechanism to find each other, e.g., through a public bulletin board acting as facilitator or through a peer-to-peer protocol specifically crafted for this purpose. A malicious facilitator may try to undermine unlinkability by forcing an honest participant to run the protocol only with malicious participants. Thus, in general, the bootstrapping mechanism should resist attempts to exclude honest users from the protocol. Since the Bitcoin network does not allow nodes to send arbitrary messages, the participants must agree on a channel for further communication during bootstrapping. Similar to other decentralized mixing protocols, we consider bootstrapping to be orthogonal to our work and assume that it is available to all Bitcoin users.

The main goal of a Bitcoin mixing protocol is the unlinkability between input and output addresses in a mixing transaction. If *after* the mixing, a user would like to spend the mixed coins associated with the output address while maintaining her anonymity, she has to ensure that network metadata, e.g., her IP address, does not reveal her identity or make the spending transaction linkable to a run of the mixing protocol. This problem is not in the scope of the Bitcoin mixing protocol and can be addressed, e.g., by connecting to the Bitcoin network via an anonymous communication protocol such as Tor [34].

3.3 Threat Model

For unlinkability and verifiability, we assume an active network attacker. (Robustness cannot be ensured in the presence of an active network attacker, because such an attacker can always stop the communication between two honest participants.)

We do *not require any trust assumption on a particular party*: for verifiability, we assume that an honest participant can be faced with an arbitrary number of malicious participants. For unlinkability and robustness, we require that there are at least two honest participants in the protocol. Otherwise the attacker can trivially determine the mapping between input and output addresses and meaningful mixing is not possible.

4 Solution Overview

Our main contribution is *CoinShuffle,* a Bitcoin mixing protocol that achieves the aforementioned goals. In this section, we give an overview of our solution.

4.1 Main Idea

To ensure verifiability, our protocol follows the same paradigm as CoinJoin (Section 2.3): A group of users jointly create a single mixing transaction and each of them can individually ensure that she will not lose money by performing the transaction. In case of a fraud attempt, the user will just refuse to sign the transaction.

Unlinkability and robustness, however, are the most challenging problems: To create a mixing transaction while assuring that input addresses are not linkable to fresh output addresses, the participants shuffle their output addresses in an oblivious manner, similar to a decryption mix network [35]. This shuffling is inspired from the accountable anonymous group messaging protocol Dissent [29,36]. Mainly due to the special nature of the problem that we would like to solve, we are able to simplify and optimize ideas from Dissent significantly. We refer to the full version [1, App. A] for a high-level comparison with Dissent.

The shuffling provides robustness in the sense that attacks that aim to disrupt the protocol can be detected by honest users and at least one misbehaving participant can be identified and excluded.[3] The other participants can then run the protocol again without the misbehaving participant.

4.2 Protocol Overview

The main part of the CoinShuffle protocol can roughly be split into three phases as depicted in Fig. 2. If the protocol does not run successfully, an additional blame phase will be reached. In the following we give an overview of every phase. A detailed description of the protocol can be found in the full version [1].

Assume that every participant holds the same amount of coins at some Bitcoin address. This address will be one of the input addresses in the mixing transaction, and every protocol message is signed with the sender's private signing key that belongs to this address.

Announcement. Every participant generates a fresh ephemeral encryption-decryption key pair, and broadcasts the resulting public encryption key.

Shuffling. Every participant creates a fresh Bitcoin address, designated to be her output address in the mixing transaction. Then the participants shuffle the freshly generated output addresses in an oblivious manner, similar to a decryption mix network [35].

In more detail, every participant (say participant i in a predefined shuffling order) uses the encryption keys of every participant $j > i$ to create a layered encryption of her output address. Then, the participants perform a sequential shuffling, starting with participant 1: Each participant i expects to receive $i - 1$ ciphertexts from participant $i - 1$. Upon reception, every participant strips one layer of encryption from the ciphertexts, adds her own ciphertext and randomly shuffles the resulting set. The participant sends the shuffled set of ciphertexts to the next participant $i + 1$. If everybody acts according to the protocol, the decryption performed by the last participant results in a shuffled list of output addresses. The last participant broadcasts this list.

[3] This property is called *accountability* in Dissent. We use a different term to avoid confusion with the concept of accountable Bitcoin mixing services in Mixcoin [16].

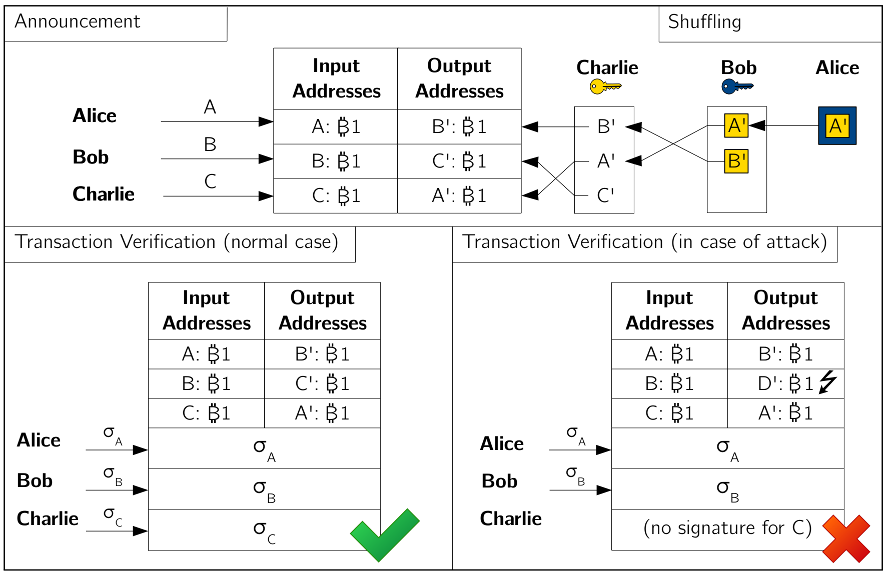


Fig. 2. Overview of CoinShuffle: First, the participants announce their input addresses. Second, they perform a shuffling of fresh output addresses. (Colored boxes represent ciphertexts encrypted with the respective encryption key.) Third, the participants check if their output address is contained in the final list of output addresses. In this case (left-hand side), the transaction is signed by the participants and submitted to the Bitcoin network. If, on the contrary, an output address is missing (e.g., C' has been replaced by D', right-hand side), the transaction does not become valid and the participants enter the blame phase to find out which participant deviated from the protocol specification.

Transaction Verification. Each participant can individually verify if her output address is indeed in the list. If this is true, every participant deterministically creates a (not yet signed) mixing transaction that spends coins from all input addresses and sends them to the shuffled list of output addresses. Every participant signs the transaction with her Bitcoin signing key and broadcasts the signature.

Upon receiving signatures from all other participants, every participant is able to create a fully-signed version of the mixing transaction. This transaction is valid and can be submitted to the Bitcoin network.

Blame. In every step of the previous phases, every participant ensures that no other participant deviates from the protocol. In such a case, an honest participant would report the deviation and the protocol enters the blame phase, which is then performed to identify the misbehaving participant. The misbehaving participant can then be excluded from a subsequent run of the protocol. There are three cases in which participants enter the blame phase. First, the blame phase is entered if some participant does not have enough bitcoins at her input address to perform the mixing transaction, or if she just spends the money at the input address before the mixing protocol is completed. In both situations, the Bitcoin network provides evidence for the misbehavior. Second, the blame phase is entered if the shuffling has not been performed correctly. In that case, the participants can broadcast their ephemeral decryption keys, along with the messages they have received. This information allows every participant to replay the computations of the rest of participants and expose the misbehaving one. Third, participants could equivocate in the broadcasts of the protocol, e.g., by sending different public keys to different participants in the announcement phase. All participants exchange messages before creating the mixing transaction to ensure that nobody has equivocated. If the equivocation check fails, the blame phase is entered. Since all protocol messages are signed, the equivocating participant can be identified; two signed messages that are different but belong to the same sender and the same broadcast step provide evidence of the misbehavior.

4.3 Practical Considerations

Transaction Fees. In practice, the Bitcoin network charges a small fee for mixing transactions[4] to prevent DoS attacks that flood the network with a high number of transactions [37]. Transaction fees can easily be dealt with in CoinShuffle. Before creating the transaction, the N participants calculate the required fee μ and reduce the size of each output by μ/N (assuming the fee should be split equally among the participants). This ensures that the transaction will be accepted by the Bitcoin network. If a participants tries to cheat by deviating from this policy, e.g., to pay a lower fee, the mixing transaction will not become valid as only the correct transaction will be signed by the honest participants.

Change Addresses. A user that would like to spend exactly ₿ x typically does not hold an input address with exactly this balance, but rather an address with a higher balance ₿ $(x + y)$. In order to perform the payment, the user will create a transaction with one input ₿ $(x + y)$ and two outputs: ₿ x go to the address of the payee and ₿ y go to a *change address* belonging to the original user.

The use of change addresses is supported in CoinShuffle: Participants can announce additional change addresses in announcement phase, if they do not have an address holding exactly the mixing amount ₿ x. In transaction verification phase, every participant adds all the change addresses as outputs of the mixing transaction before it is signed. CoinShuffle still preserves the unlinkability between the input addresses and the (regular) output addresses of the honest participants.

Communication and Liveness. In practice, broadcasts can be implemented by sending all messages to a randomly chosen *leader* that relays the messages to all participants. Furthermore, instead of active misbehaving, participants might

[4] At the time of writing, a fee of ₿ 0.0001 ($\approx$ \$ 0.06) per 1000 bytes of transaction size is mandatory for transactions of at least 1000 bytes. Due to their nature, mixing transactions contain several addresses and are typically larger than 1000 bytes. A mixing transaction with 25 participants has an approximate size of 5000 bytes.

passively disrupt a protocol round by simply going offline at any time, either maliciously or due to a network failure or asymmetric connectivity. This problem is not particular to CoinShuffle and can be handled using the same techniques as in Dissent [29], which in turn borrows ideas from PeerReview [38]. We only present the idea and refer the reader to the original papers for details. When the protocol states that a participant i must receive a properly signed message from participant j, but participant j does not send such a message within a predefined timeout period, i suspects j. In this case, i asks another participant k (or several participants) to request the desired message from j and relay it to i. If k does not receive the message either, also k suspects j and can in turn ask other members. In case nobody receives the message from j, i.e., everybody suspects j eventually, the participants can start a new run of the protocol without j.

5 System Discussion

In this section, we discuss why CoinShuffle achieves all desired system-level design goals as described in Section 3.1. We refer the reader to the full version [1] for a security analysis of the protocol.

Compatibility. CoinShuffle does not require any change to the Bitcoin protocol or to the transaction format, because a successful run of CoinShuffle results in a transaction that is valid according to the current rules of Bitcoin. Thus the protocol is immediately deployable.

No Mixing Fees. Systems in which a trusted third party performs the mixing typically charge users two fees: a transaction fee as defined in Bitcoin and a mixing fee required by the mixing server [13,14,15]. In CoinShuffle, however, no mixing fee is required. Users who jointly execute CoinShuffle are only charged the transaction fee as defined in the currently deployed Bitcoin protocol.

Efficiency. Aside from digital signatures that are already deployed in Bitcoin, CoinShuffle requires only standard public-key encryption and a hash function. This makes it possible to run the protocol even on computationally restricted hardware. Our performance evaluation (Section 6) shows the feasibility of CoinShuffle in practice.

Small Impact on Bitcoin. Upon successful protocol execution, CoinShuffle participants jointly create only a single Bitcoin transaction that is stored in the blockchain and has to be verified by all nodes in the network. Thus, the execution of CoinShuffle introduces only a minimal overhead in terms of storage and computation for the Bitcoin system.

6 Performance Evaluation

We have developed a proof-of-concept implementation [1] of CoinShuffle leveraging an existing implementation of the Dissent protocol. In particular, we have implemented the announcement, shuffling and transaction verification phase, which suffice to measure the performance of a single successful run without disruption attempts.

The implementation is written in Python and uses OpenSSL to sign and encrypt messages. As required by the Bitcoin network, signatures have been implemented using ECDSA on the 256-bit curve secp256k1 [39]. We use the Elliptic Curve Integrated Encryption Scheme (ECIES) [39] on the same curve together with 256-bit AES in CBC mode for encryption. The communication among the participants has been implemented using TCP. When a message is broadcast, it is first sent to the first participant in the shuffling order, who in turn sends a copy to every participant.

We tested our implementation in Emulab [40], a testbed for distributed systems, in which network parameters such as topology or bandwidth of links can be easily configured. In this setting, we have run several experiments under controlled network conditions. We consider two scenarios: a local network and a global network. In the former, we connected all the participants to a LAN with 100 Mbit/s bandwidth without delays. In the latter, we split the participants in two LANs of 100 Mbit/s bandwidth each. Both LANs were connected through a router with a bandwidth of 20 Mbit/s and a delay of 50 ms. In the global network scenario we considered the worst case for the shuffling phase: participants with an odd index in the shuffling ordering were placed in one LAN while participants with an even index were placed in the second LAN. Thus every message in the shuffling phase had to traverse the whole network.

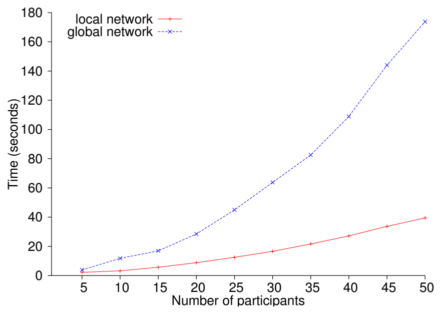


Fig. 3. Overall execution time

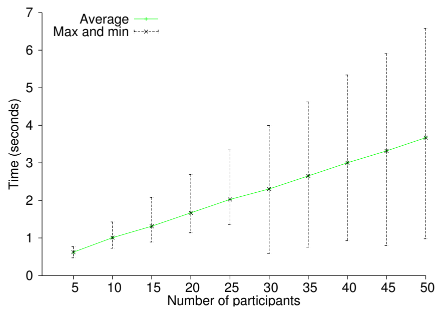


Fig. 4. Average processing time per node

We have run the protocol with different numbers of participants, ranging from 5 to 50. Figure 3 shows the overall time needed to create a Bitcoin transaction in a run without misbehaving participants. In the local network setting, 50 participants need approximately 40 seconds to run CoinShuffle, while in the global network, slightly less than 3 minutes are necessary to complete the protocol.

Figure 4 shows the overhead of the computation carried out by every participant on average. As expected, the average processing time increases linearly with the number of participants, because every participant must shuffle a vector of ciphertexts containing one ciphertext more than the previous participant. Furthermore, the computation overhead constitutes only a small fraction of the overall time. In the case of 50 participants, the average computation time is slightly larger than 3 seconds, which constitutes approximately 2% of the overall time in the local network scenario and less than 1% in the global network setting.

In summary, the experimental results demonstrate the feasibility of the CoinShuffle protocol even in scenarios where the number of participants is large.

7 Related Work

Zerocoin [17], an extension to Bitcoin, was among the first proposals to provide unlinkability between individual Bitcoin transactions without introducing a trusted party. It employs a cryptographic accumulator of minted *zerocoins* and a zero-knowledge proof of inclusion of a certain zerocoin within the accumulator. Zerocoin introduces a significant computation and communication overhead: the size of the proof that has to be stored in the blockchain for each transaction is prohibitively large (i.e., approximately 25 KB) and far exceeds the size of the Bitcoin transaction itself.

Recently, there have been some proposals to reduce the Zerocoin proof size. Garman et al. [19] propose a set of extensions to Zerocoin that reduces the proof size by modeling the cost of forging a coin and picking cryptographic parameters to make such forgery uneconomical. Both Pinocchio Coin [18] and Zerocash [20] are promising improvements of Zerocoin that significantly reduce the proof size (to less than 1 KB) and the computational costs. Nevertheless, this line of research is severely restricted in terms of adaptability. Zerocoin and all of the above extensions require substantial modifications to the Bitcoin system. Thus, Zerocoin and its variants cannot be directly deployed in Bitcoin. Instead, they would need an incremental deployment that requires acceptance by the majority of the Bitcoin nodes (measured in computational resources). So far, it looks unlikely for the Bitcoin network to employ the Zerocoin strategy [41].

In contrast, while requiring more communication, CoinShuffle is immediately adaptable and works on top of the existing Bitcoin network without any modifications.

The Mixcoin [16] protocol facilitates anonymous Bitcoin payments without making any modifications to the Bitcoin protocol. Here, Bitcoin users send their coins to a central accountable mixing server which in turn replies with a guarantee of returning the funds to the user. Afterwards, the mix sends the coins back to the user ensuring unlinkability between the user input and output addresses. Although the mixing server can be held accountable for any wrongdoing, the system still has several drawbacks. First, the use of a central mix introduces a single point of failure, where the mix becomes a suitable target for DoS attacks from competing mixing servers as well as governmental agencies. Second, the provided accountability is reactive in nature, and the mix can still steal users' coins before going out of business. Third, a payment in Mixcoin requires two Bitcoin transactions and additionally the mixing server's fee. Finally, unlinkability is only guaranteed against external observers, because the mixing server learns which address belongs to which user. In comparison to Mixcoin, CoinShuffle relies on the interaction between the users in the mixing to achieve unlinkability against malicious servers, verifiability, robustness, and cost effectiveness.

Maxwell [42] sketches a modification to the CoinJoin protocol using blind signatures to avoid the problem of a centralized mix learning the relation between input and output addresses. This protocol employs the anonymous communication network Tor [34] as a building block to provide unlinkability. In contrast, CoinShuffle provides full resistance against traffic correlation attacks by using a decentralized high-latency mix network run only by the participants.

8 Conclusion

The linkable pseudonymity provided by the Bitcoin system leads to significant privacy concerns for its users. A few solutions that aim at mixing transactions of a group of users have been proposed in the last two years to address this concern; however, none of them has been found to be satisfy all requirements of a practical and compatible solution. In this paper, we have presented the Bitcoin mixing protocol CoinShuffle, which is secure, robust, and perfectly compatible with the existing Bitcoin system. Adhering to the Bitcoin ideology, CoinShuffle is completely decentralized, and it neither requires any third party nor introduces any additional anonymization fees for the users.

We implemented CoinShuffle and tested it in a local as well as in a global network scenario in the Emulab environment. Our experiments demonstrate that CoinShuffle introduces only a small (suitable for Bitcoin) computation and communication overhead to a participant, even when the number of CoinShuffle participants is large (≈ 50). Moreover, CoinShuffle leads to only a minimal overhead for the Bitcoin blockchain and the rest of the Bitcoin users.

Finally, although we have focused on the crypto-currency Bitcoin in the paper, we stress that our protocol is compatible with all competing currencies derived from Bitcoin, e.g., Litecoin [4], Mastercoin [6], and others.

Acknowledgments. We thank Bryan Ford for his insightful comments on an earlier draft and Henry Corrigan-Gibbs for helping us with running the proof-of-concept implementation on Emulab. We further thank the anonymous reviewers for their helpful comments. This work was supported by the German Universities Excellence Initiative.

References

1. Ruffing, T., Moreno-Sanchez, P., Kate, A.: CoinShuffle: Practical decentralized coin mixing for Bitcoin. Full version of this paper and prototype implementation. `http://crypsys.mmci.uni-saarland.de/projects/CoinShuffle`

2. Ruffing, T., Moreno-Sanchez, P., Kate, A.: CoinShuffle: Practical decentralized coin mixing for Bitcoin. Post on Bitcoin Forum. `https://bitcointalk.org/index.php?topic=567625` (2014)
3. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Technical report. `https://bitcoin.org/bitcoin.pdf` (2008)
4. Litecoin. `https://litecoin.org`
5. Ripple. `https://ripple.com`
6. Mastercoin. `http://www.mastercoin.org`
7. Dodgecoin. `http://dogecoin.com`
8. BitInfoCharts (accessed on 2014-03-28). `http://bitinfocharts.com/comparison/transactions-marketcap-btc-ltc.html`
9. Barber, S., Boyen, X., Shi, E., Uzun, E.: Bitter to better — how to make Bitcoin a better currency. In: Proc. of the 15th Conference on Financial Cryptography and Data Security. FC'12, Springer (2012) 399–414
10. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: Characterizing payments among men with no names. In: Proc. of the 2013 Conference on Internet Measurement Conference. IMC'13, ACM (2013) 127–140
11. Spagnuolo, M., Maggi, F., Zanero, S.: BitIodine: Extracting intelligence from the Bitcoin network. In: Proc. of the 17th Conference on Financial Cryptography and Data Security. FC'14 (2014)
12. Koshy, P., Koshy, D., McDaniel, P.: An analysis of anonymity in Bitcoin using P2P network traffic. In: Proc. of the 17th Conference on Financial Cryptography and Data Security. FC'14 (2014)
13. Bitcoin Fog. `http://www.bitcoinfog.com`
14. BitLaundry. `http://app.bitlaundry.com`
15. BitLaunder. `https://bitlaunder.com`
16. Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J.A., Felten, E.W.: Mixcoin: Anonymity for Bitcoin with accountable mixes. In: Proc. of the 17th International Conference on Financial Cryptography and Data Security. FC'14, Springer (2014)
17. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: Anonymous distributed e-cash from Bitcoin. In: Proc. of the 34th Symposium on Security and Privacy. S&P'13, IEEE (2013) 397–411
18. Danezis, G., Fournet, C., Kohlweiss, M., Parno, B.: Pinocchio Coin: Building Zerocoin from a succinct pairing-based proof system. In: Proc. of the 1st ACM Workshop on Language Support for Privacy-enhancing Technologies. PETShop'13, ACM (2013) 27–30
19. Garman, C., Green, M., Miers, I., Rubin, A.D.: Rational Zero: Economic security for Zerocoin with everlasting anonymity. In: 1st Workshop on Bitcoin Research. (2014)
20. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from Bitcoin. In: Proc. of the 35th Symposium on Security and Privacy. S&P'14, IEEE (2014)
21. Maxwell, G.: CoinJoin: Bitcoin privacy for the real world. Post on Bitcoin Forum. `https://bitcointalk.org/index.php?topic=279249` (August 2013)
22. Qkos Services Ltd: Shared Coin. `https://sharedcoin.com`
23. Yang, E.Z.: Secure multiparty bitcoin anonymization. Technical report. `http://blog.ezyang.com/2012/07/secure-multiparty-bitcoin-anonymization/` (July 2013)

24. Jónsson, K.V., Kreitz, G., Uddin, M.: Secure multi-party sorting and applications. ePrint Cryptology Archive 2011/122. `https://eprint.iacr.org/2011/122.pdf` (2011)
25. Hamada, K., Kikuchi, R., Ikarashi, D., Chida, K., Takahashi, K.: Practically efficient multi-party sorting protocols from comparison sort algorithms. In: Proc. of the 15th International Conference on Information Security and Cryptology. ICISC'12, Springer (2013) 202–216
26. Rosenfeld, M.: Using mixing transactions to improve anonymity. Post on Bitcoin Forum. `https://bitcointalk.org/index.php?topic=54266` (December 2011)
27. Murphant (pseudonym). Post on Bitcoin Forum. `https://bitcointalk.org/index.php?topic=279249.msg3057216#msg3057216` (August 2013)
28. Maxwell, G. Post on Bitcoin Forum. `https://bitcointalk.org/index.php?topic=279249.msg3013970#msg3013970` (September 2013)
29. Corrigan-Gibbs, H., Ford, B.: Dissent: Accountable anonymous group messaging. In: Proc. of the 17th Conference on Computer and Communications Security. CCS'10, ACM (2010) 340–350
30. Möser, M., Böhme, R., Breuker, D.: An inquiry into money laundering tools in the Bitcoin ecosystem. In: Proc. of the APWG eCrime Research Summit. ECRIME'13, IEEE (2013)
31. Duffield, E., Hagan, K.: Darkcoin: Peer-to-peer crypto currency with anonymous blockchain transactions and an improved proof-of-work system. Technical report (March 2014)
32. Buterin, V., Malahov, J., Wilson, C., Hintjens, P., Taaki, A., et al.: Dark Wallet. `https://darkwallet.unsystem.net`
33. van der Laan, W.J.: Implement coinjoin in wallet. GitHub Issue #3226 of official Bitcoin repository. `https://github.com/bitcoin/bitcoin/issues/3226`
34. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proc. of the 13th USENIX Security Symposium. USENIX Security'04, USENIX (2004) 21–37
35. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **24**(2) (1981) 84–90
36. Syta, E., Johnson, A., Corrigan-Gibbs, H., Weng, S.C., Wolinsky, D., Ford, B.: Security analysis of accountable anonymous group communication in Dissent. `http://dedis.cs.yale.edu/dissent/papers/analysis.pdf`
37. Transaction fees. Bitcoin Wiki (revision as of 2014-03-28). `https://en.bitcoin.it/w/index.php?title=Transaction_fees&oldid=45501`
38. Haeberlen, A., Kouznetsov, P., Druschel, P.: PeerReview: Practical accountability for distributed systems. In: Proc. of 21st Symposium on Operating Systems Principles. SOSP'07, ACM (2007) 175–188
39. Certicom Research: Sec 1: Elliptic curve cryptography. `http://www.secg.org/download/aid-780/sec1-v2.pdf`
40. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. In: OSDI'02, USENIX (December 2002) 255–270
41. Thread on Bitcoin Forum. `https://bitcointalk.org/index.php?topic=175156.0`
42. Maxwell, G. Post on Bitcoin Forum. `https://bitcointalk.org/index.php?topic=279249.msg2984051#msg2984051` (2013)