Distributed and Hash-Based Mixers for User Anonymity on Blockchain

P. Guna Shekar, Raghwendra Singh, Debanjan Sadhya, and Bodhi Chakraborty

1 Introduction

In its essence, a Blockchain is a distributed ledger technology enabling data immutability. It helps facilitate financial transactions in a decentralized and peer-to-peer manner without the involvement of a third party. While third parties like banks maintain the ledger of financial transactions in the traditional scenario, Blockchain acts as a public ledger, maintaining all the transactions on the network in a decentralized and distributed fashion. All the transactions on the network are collected and organized into *blocks*, with each block containing a limited number of transactions. As and when more transactions come into the network, new blocks get appended to the existing set of blocks, thus forming a chain of blocks. The blocks are connected using cryptographic methodologies like hashing, as shown in Fig. 1. Since Blockchain is a decentralized system, it is crucial to maintain consensus about the content of each block in the Blockchain among all the network participant nodes.

Anonymity in Blockchain refers to its property of not revealing the user's identity to the network. This property enables users to stay unidentified on the Blockchain network, thus enhancing user privacy. Though the user's identity needs not to be revealed to make transactions, de-anonymization inference attacks can be used to link a set of transactions to a user. Unlinkability refers to the property where a particular set of transactions cannot be related to a single entity with a high confidence

P. G. Shekar · R. Singh (✉) · D. Sadhya
ABV-Indian Institute of Information Technology and Management Gwalior, Madhya Pradesh, Gwalior, India
e-mail: 202109@iiitm.ac.in

D. Sadhya
e-mail: debanjan@iiitm.ac.in

B. Chakraborty
ITM University, Madhya Pradesh, Gwalior, India
e-mail: bodhi.cse@itmuniversity.ac.in

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2024

B. K. Roy et al. (eds.), *Cryptology and Network Security with Machine Learning*, Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-99-2229-1_17

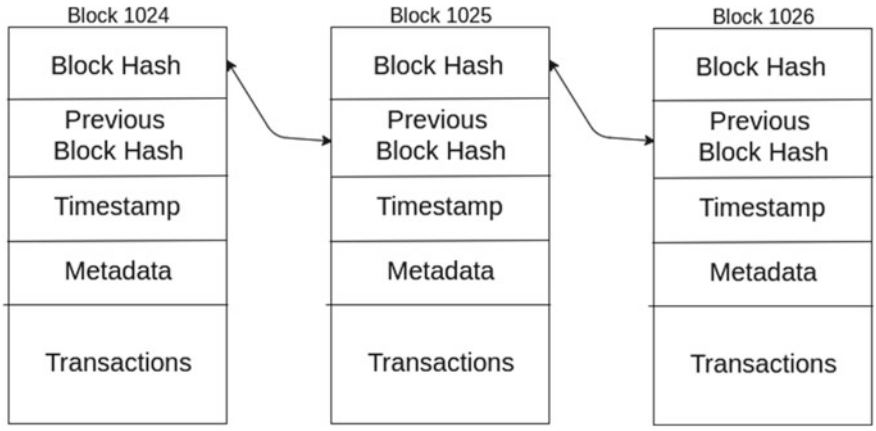

Fig. 1 Basic structure of a Blockchain

level. This property is essential despite anonymity because once a set of transactions can be linked to a particular entity, other details such as account balance, i.e., type of merchants/transactions and the frequency can be easily inferred. In turn, malicious parties can use these statistics to reveal the user's true identity. For instance, a statistical analysis based on the origin IP Address and the physical location of transactions can be used to trace back the transactions and link a certain set of transactions to a particular user [1].

Mixing is a service that allows users to regain their privacy on Blockchain that was challenged by recent works on de-anonymization. Mixing refers to transferring coins or tokens from one account to another unrelated account before finally transferring them to the desired destination to prevent attackers from relating transactions to each other. In this work, we devise a solution for maintaining anonymity over a Blockchain network while using mixers over a Merkle Patricia Trie. The proposed solution has an interface for the users to deposit their coins to the mixer, which would then mix them to multiple addresses. The mixer would take note of the amount it owes to the user and return it to the desired destination as requested by the user.

2 Related Work

In this section, we discuss and analyze the existing studies on unlinkability and anonymity on the Blockchain network. Dupont and Squicciarini [2] provided a demonstration on obtaining a user's real-life information based on their bitcoin transactions. To achieve this, they used publicly available Bitcoin transaction data. They performed a statistical analysis of the users' spending habits to determine their physical location worldwide. The authors considered the timings during which the transactions were made from a particular user and were able to decode a user's timezone

and, in some cases, their country with 75% accuracy. Jawaheri et al. [3] examined the possibility of de-anonymizing Tor hidden service users who pay with Bitcoin using public information obtained from online social networks, the Blockchain, and onion websites. This experiment successfully enabled to link of a user's Twitter handle to their Tor hidden service handle based on the transactions they made using Bitcoin. The authors conducted a real-world experiment simulating a passive, limited adversary to demonstrate the feasibility of this de-anonymization attack. Specifically, 1500 hidden services were crawled, and 88 different Bitcoin addresses were gathered. The authors subsequently crawled 5 billion tweets and 1 million BitcoinTalk forum pages, thus collecting 4.2 million and 41 thousand unique Bitcoin addresses, respectively. Each user's address was linked to the online identity and public profile information. A total of 125 individual users were linked to 20 Tor hidden services, including sensitive ones like The Pirate Bay and Silk Road.

Mixcoin, a protocol to enable users to have anonymity and unlinkability on the Blockchain, was proposed by Bonneau et al. [4]. The authors expanded on the emerging area of currency mixes by introducing an accountability mechanism to disclose thievery. It was demonstrated how the incentives of the mixes and the clients could be aligned to ensure that rational mixes do not steal. Gregory Maxwel developed a method to anonymize user transactions on the Blockchain using a joint transaction approach [1]. In this approach, if a user wishes to send a transaction, they would find another user who also wants to make a transaction and send joint transactions to their respective destinations. This process improves anonymity as patterns of a user would be very difficult to identify when their transactions are combined with that of other users. CoinJoin also has a requirement that the users would have to find a pair for themselves to make a joint transaction.

Some older mixing protocols, such as SharedCoin, proposed by Henrique et al. [5] used a centralized server to facilitate user matching. This process lessened the load on users to find a pair for themselves to make a joint transaction. However, they acted as a single point of failure for the entire system. The model also maintained the original addresses of users and transaction logs, leakage of which would cause de-anonymization of the users.

3 Merkle Patricia Trie

A Merkle Patricia Trie is an improved combination of a Merkle Tree and a Trie. An example of a Merkle Patricia Trie can be seen in Fig. 2. Each node has a hash, which is used as the identifier for the node. Starting from the leaf node, all children of a particular node are hashed to get the parent's hash; hence, the root of the tree can act as a cryptographic hash for the entire tree. A Merkle Patricia Trie has four different types of nodes.

1. **Empty Nodes**: These blank nodes do not contain any data.

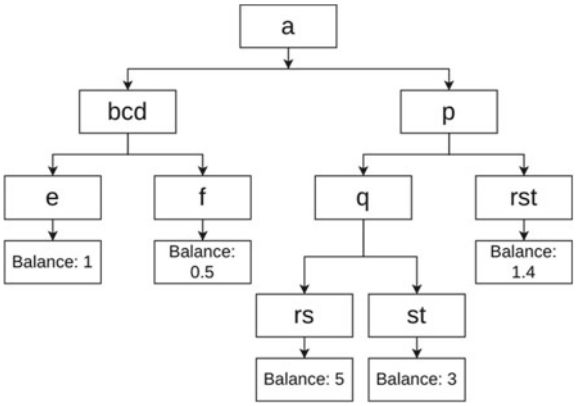

Fig. 2 Example of a Merkle Patricia Trie with five hashes of user addresses–abcde, abcdf, apqrs, apqst, aprst; and their balances

2. **Leaf Nodes**: Lowermost nodes in the tree containing the final key-value pair data (in our case, the key is the hash of the user's address, and the value is the amount the user deposited in our mixer).
3. **Branch Nodes**: These are internal nodes, a list of characters that link to either other branch nodes or a leaf node. The list is the size of the alphabet used in the data structure.
4. **Extension Nodes**: These nodes contain the hash of another node as its value.

The Merkle Patricia Trie is much more efficient in terms of space than a Merkle Tree or a Trie because of the extension nodes that make it compact. Searching for a particular element in a Merkle Patricia Trie takes (1) time. While indexed databases can search for elements in (1) time, the indices take up extra storage space. At the same time, Merkle Patricia Trie does not utilize extra storage space for any form of indices. The Merkle Patricia Trie also provides a Merkle Proof via the cryptographic hash of the root node, using which other servers in the network can verify balances using (1) time without having to fetch all balances. In case of balance differences, comparison and update among a distributed set of Merkle Patricia Trie happen in (log n) time, where the maximum value of n is the number of branches from an element in the Merkle Patricia Trie. A Merkle Patricia Trie is beneficial when hashes are stored in it because hashes have a fixed domain; hence, the Trie width will always remain less than or equal to the size of the domain.

4 Proposed Methodology

With user anonymity being the primary objective of the implementation, we now discuss the design details of our model.

4.1 Model Overview

Our solution adopts a distributed approach to avoid such a single point of failure. Instead of having a single mixer handling all users' requests, the proposed solution will have multiple servers interacting with each other, as shown in Fig.3. The database too will be distributed across a cluster to ensure data replication. In this case, the data will be stored in a data structure called the Merkle Patricia Trie. In order to prevent location-based de-anonymization, the mixer servers are deployed on a cloud cluster with each node located in different time zones. Hence, a particular user's transactions will also occur randomly from different locations worldwide, thus preventing location-based de-anonymization.

The second aspect of our design is to provide a better user experience. Older mixers put the burden of finding a group/partner for mixing on the end user. Moreover, it would not be a viable solution if the problem had to be solved for the masses and scale to a massive number of users. A much more scalable approach would be to have the mixers take on the responsibility of creating one-time-use addresses for mixing, as shown in Fig. 4.

The final aspect of our solution is the protection of user anonymity. To efficiently mix the coins to ensure unlinkability, it is not necessary to store the user addresses

Fig. 3 Illustration of distributed servers to avoid a single point of failure

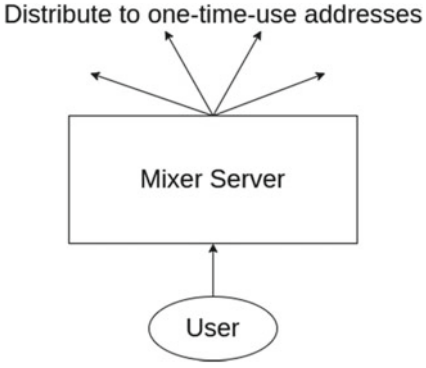

Fig. 4 Mixer server distributing the coins to one-time-use addresses

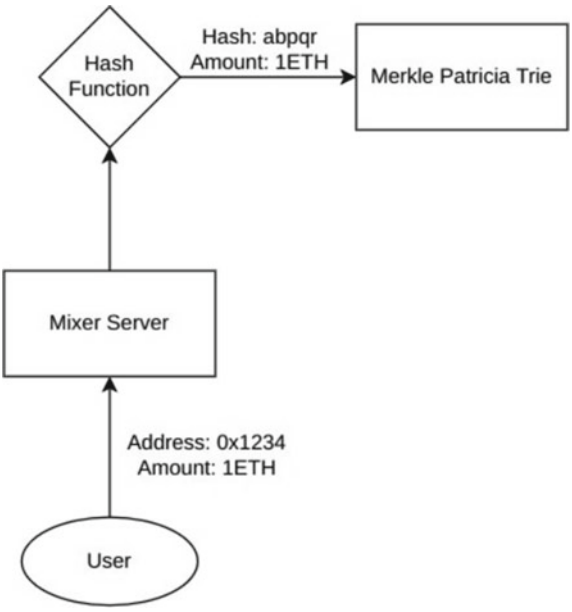

Fig. 5 Storing only the hash to protect user anonymity

directly. Instead, any other identifier that only the user would have access to would be sufficient to prove that the mixer owes a certain amount of coins to the user. For this purpose, instead of the user's original address, the address's hash will be stored in the database of the proposed solution, as shown in Fig. 5. Users who wish to withdraw their amount or send it to another account will send a signed transaction to the mixer. The signed transaction would come from the user's original address, which is enough proof that it is the original user requesting their coins back. The mixer would then hash the user's addresses, check if the mixer owes any coins to that user, and finally send the coins to the destination accordingly.

4.2 Detailed Architecture

There are three main components to consider for architecture design: (1) user interaction with the mixer while depositing coins, (2) the mixing protocol, and (3) user interaction with the mixer while withdrawing coins. Now we individually discuss these mechanisms.

4.2.1 Depositing Coins

For depositing the coins, the client chooses a random mixer from the available n mixers in the distributed system. The user would be presented with a user interface

using which they can enter the amount they want to deposit. They would then sign a transaction message to verify that they are the valid owner of the coins they are depositing. This message would then be sent to the mixer server. The mixer would then validate the request and hash the user's address for storage. The hashed address and the amount the mixer now owes to the user are stored in the Merkle Patricia Trie and eventually replicated over other mixer servers.

4.2.2 Mixing Protocol

Once the user sends the coins to the mixer, the mixer proceeds to carry out a protocol for mixing. The total number of coins is divided into three to five chunks with random amounts in each, totaling up to the original amount. The chunks are random and not uniform to ensure that an adversary cannot multiply the amount in a chunk to derive insights into what the original amount could be. The mixer then creates n one-time-use addresses, n being the number of chunks the original amount was divided into and transfers the amount in each chunk to each of the newly created addresses. At this point, the user's coins have reached new addresses, and future transactions cannot be linked to the user's original account. To increase the number of jumps before reaching the final destination and improve unlinkability, the coins will be shuffled between the newly created addresses once for a certain period (viz., one month).

4.2.3 Withdrawing Coins

When users wish to withdraw their coins, they first sign a message with the private key of the address whose coins are stored with the mixer. Along with this message, they also enter a new address to which they wish to transfer their coins and the number of coins they wish to transfer. This message and the details are then sent to the mixer. The mixer validates the message, verifies whether the mixer owes the amount to the given address, and proceeds to transfer the funds. For transferring funds, the mixer chooses three to five random addresses it previously created, whose total balance is at least as much as the amount the user wishes to withdraw. These n addresses are then made to transfer chunks of the amount, totaling up to the desired amount the user wants to withdraw, to the final destination where the user wishes to withdraw their coins. Finally, the destination address has now received the total sum requested by the user, but from n different and unrelated addresses, which cannot be linked back to the original address of the user. The process of withdrawing coins is demonstrated in Fig. 6.

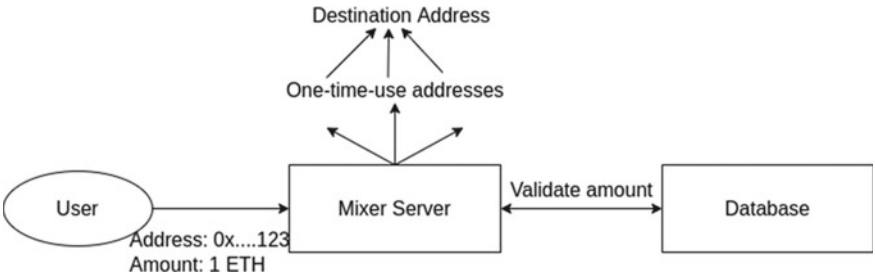

Fig. 6 The process of withdrawing coins by a user

5 Experimental Results

The developed model is a distributed and hash-based mixer server that provides user anonymity and transaction unlinkability on Blockchain. For simulation and testing purposes, the servers have been deployed on Microsoft Azure's Virtual Machines, as shown in Fig. 7.

5.1 Usage of Merkle Patricia Trie

While providing a data storage solution for hashes, the Merkle Patricia Trie proved more efficient than a relational database. The Merkle Patricia Trie's insertion time is significantly lesser than a relational or a document database. The insertion of a hash of a user address along with their balance took 21.755 ms in the case of a Merkle

Fig. 7 Mixer servers are deployed at different data centers to prevent a single point of failure and location-based de-anonymization

Patricia Trie, while it took 49.262 and 45.278 ms in the case of a Document and Relational Database, respectively. Hence, Merkle Patricia Trie proves to be better in terms of computational memory and time.

5.2 Mixer Server

The Mixer Server is a Node.js-based server that contains the core logic of the mixer. This server is deployed on a cloud cluster on Microsoft Azure. The mixer has the following APIs.

1. **Deposit Funds**: The Deposit Funds API accepts a signed transaction from a user that contains a certain amount of cryptocurrencies. The signature on the transaction is done using the Metamask wallet on the client side. This signature is performed by the user's private key on the Blockchain network that can be used to ensure that it is the legitimate user who is sending the transaction. The API then hashes the user's address to provide anonymity on our server and then updates the user's balance in the Merkle Patricia Trie. Once the balance has been updated, the API distributes the funds to one-time-use addresses according to the mixing protocol described previously.
2. **Fetch Balance**: The Fetch Balance API accepts the user's address in the form of a signed transaction with zero funds to ensure the legitimacy of the user. The user's address is then hashed, and the record is fetched from the Merkle Patricia Trie. The balance fetched from the Merkle Patricia Trie is sent to the client, which then displays it to the user.
3. **Withdraw Funds**: The Withdraw Funds API accepts a destination address, the amount to be withdrawn, and a signed transaction with zero funds to ensure the legitimacy of the user. The first step is to hash the user's address and verify in the Merkle Patricia Trie whether the user has enough balance to be withdrawn. Once verified, the API proceeds to withdraw funds into the destination address provided by the user. The destination address will then receive funds from a random one-time address that cannot be linked back to the original user.

5.3 Client Side

A React.js-based client has been developed to help the mixer users interact with the mixer server. The client has the following routes:

- **Deposit Funds**: The Deposit Funds route contains a form that accepts the amount the user wishes to deposit with the mixer. Once the user enters the amount, they will be prompted to sign a transaction with the specified amount on their Metamask wallet, as shown in Fig. 8. This signed transaction will then be sent to the backend. Figure 8 demonstrates this process for a particular user.

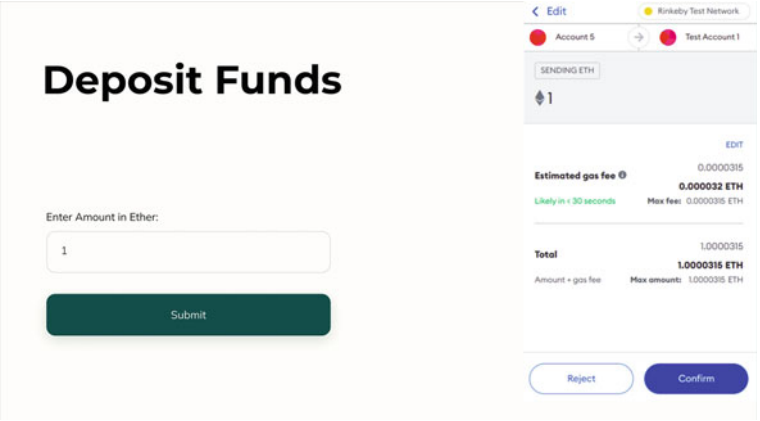

Fig. 8 An user trying to deposit 1 ETH to the mixer server

- **View Balance**: The View Balance route prompts the user to sign an empty transaction sent to the backend. Based on this, the route shows the balance of the user that will be returned by the server, as shown in Fig. 9.
- **Withdraw Funds**: Withdraw Funds contains a form that accepts the amount to be withdrawn and the destination address where the funds are to be received. The user will then be prompted to sign an empty transaction to verify their legitimacy, as shown in Fig. 10.

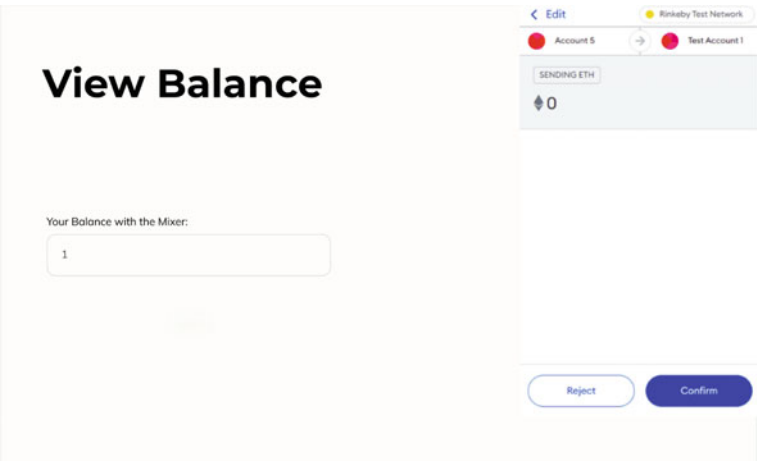

Fig. 9 An user is viewing the balance that was previously deposited into the mixer

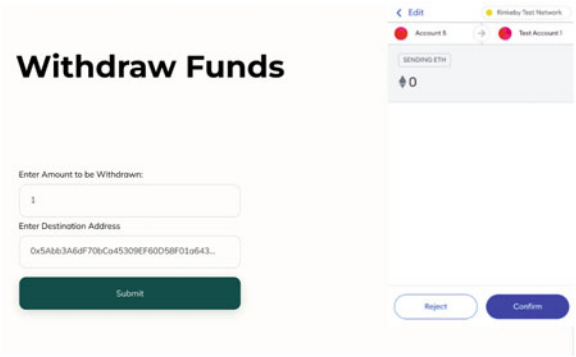

Fig. 10 An user trying to withdraw 1 ETH to their destination address

5.4 Quantitative Evaluation

After the deployment of mixer servers, it was deliberately taken down to test the robustness of the system, as shown in the *Status* property in Fig. 11. A test user was then made to deposit funds mixed by the protocol. Finally, the user withdrew their funds to a destination address. Figure 12 shows the transactions that were sent to the destination address. Since these transactions were received from servers with different locations, the location of the original user cannot be decoded.

Importantly, our model protects user anonymity even in cases of data leaks. The mixer server only stores the irreversible hashes of the user addresses in the form of a Merkle Patricia Trie and not the user addresses directly. Fig. 2 shows what an adversary would see if they were to attack the server and leak the data. It can be seen that the original user addresses are hidden and the adversary cannot decode the identity of the user from the Trie that contains the hashes of user addresses. An Ethereum address consists of 64 characters and is a hexadecimal string, i.e., a domain size of 16. Therefore, on average, it would take an adversary attempt to crack a hash of a user address

Fig. 11 Mixer Server stopped to test the robustness of the system

Fig. 12 Destination address receiving funds from one-time addresses

6 Challenges and Limitations

While the problem of user anonymity is solved using the mixers, the solution lags on a few other facets. Following are the drawbacks of using a mixer instead of a direct transfer:

1. **Additional Gas Fee**: The proposed solution uses the Rinkeby Test Network, one of the testing environments for the Ethereum Blockchain. The average gas fee while making a transaction on this network is 0.000045ETH. Assuming an average hop size of 3, it would take the user an additional 0.000135ETH to stay anonymous while sending the transaction to their destination.
2. **Higher Latency**: It would take additional time for the mixer to send the desired amount to the destination address compared to a direct transfer of funds, owing to the mixing protocol that makes the funds hop among different addresses. However, since the mixing is done at a stage earlier than withdrawal and the final transactions to the destination are sent in parallel, the latency is lesser than the architecture proposed by the Mixcoin [5] protocol. The latency for the direct transfer, the proposed mixer-based model, and the Mixcoin protocol were 8.62, 1.85, and 39.46 s, respectively.

7 Conclusion

While anonymity and unlinkability are supposed to be one of the primary features of the Blockchain, studies have proved that this is not necessarily the case. Though there were attempts to resolve these issues in the past, they contained flaws due to which they could not become permanent solutions to these problems. The proposed solution attempts to fix these issues by mixing coins to improve user anonymity and transaction unlinkability on the Blockchain. The simulation results have shown that the proposed solution successfully solves the issues identified in existing solutions and provides a layer of user anonymity and transaction unlinkability on the existing Blockchain.

References

1. Zhang R, Xue R, Liu L (2019) Security and privacy on blockchain. ACM Comput Surv 52(3). https://doi.org/10.1145/3316481
2. DuPont J, Squicciarini AC (2015) Toward de-anonymizing bitcoin by mapping users location. CODASPY '15, pp 139–141. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2699026.2699128
3. Jawaheri HA, Sabah MA, Boshmaf Y, Erbad A (2020) Deanonymizing tor hidden service users through bitcoin transactions analysis. Comput Secur 89(C). https://doi.org/10.1016/j.cose.2019.101684

4. Bonneau J, Narayanan A, Miller A, Clark J, Kroll JA, Felten EW (2014) Mixcoin: Anonymity for bitcoin with accountable mixes. In: Christin N, Safavi-Naini R (eds) Financial cryptography and data security, pp 486–504. Springer, Berlin, Heidelberg
5. Moniz H, Neves NF, Correia M, Verissimo P (2006) Experimental comparison of local and shared coin randomized consensus protocols. In: 2006 25th IEEE Symposium on Reliable Distributed Systems (SRDS'06), pp 235–244. https://doi.org/10.1109/SRDS.2006.19
6. Bonneau J, Narayanan A, Miller A, Clark J, Kroll J, Felten E (2014) Mix- coin: Anonymity for bitcoin with accountable mixes, pp 486–504. https://doi.org/10.1007/978-3-662-45472-531