

# SoK: How private is Bitcoin? Classification and Evaluation of Bitcoin Mixing Techniques<sup>\*,\*\*</sup>

Simin Ghesmati<sup>a,b</sup>, Walid Fdhila<sup>a</sup> and Edgar Weippl<sup>a,c</sup>

<sup>a</sup>SBA Research

<sup>b</sup>Vienna university of technology

<sup>c</sup>University of Vienna

## ARTICLE INFO

### Keywords:

Blockchain

Privacy

Mixing

Tumbling

Bitcoin

Distributed ledger

Anonymity

Deanonymization WGM

BEC

## ABSTRACT

Blockchain is a disruptive technology that promises a multitude of benefits, such as transparency, traceability, and immutability. However, this unique bundle of key characteristics has proved to be a double-edged sword that can put users' privacy at risk. Unlike traditional systems, Bitcoin transactions are publicly and permanently recorded, and anyone can access the full history of the records. Despite using pseudonymous identities, an adversary can undermine users' financial privacy and reveal their actual identities using advanced heuristics and techniques to identify possible links between transactions, senders, receivers, and consumed services (e.g., online purchases). In this regard, a multitude of approaches has been proposed to reduce financial transparency and enhance users' anonymity. These techniques range from using mixing services to off-chain transactions that address different privacy issues. In this paper, we particularly focus on comparing and evaluating mixing techniques in the Bitcoin blockchain, present their limitations, and highlight the new challenges.

## 1. Introduction

In recent years, there has been an increasing interest in blockchain technology. The first design was by Satoshi Nakamoto [52] in late 2008. The number of use cases and applications of blockchain technology beyond cryptocurrencies has increased exponentially. Examples of this include supply chains, industry 4.0, healthcare, and identity management. Unlike traditional systems that rely on centralized entities, blockchain technology uses a distributed shared ledger to permanently record transactions. In particular, in open blockchains such as Bitcoin, anyone can join, validate, and access the history of all transactions since the genesis block. Although in principle this is supposed to be one of the key characteristics of blockchain technology, such transparency can put the financial privacy of users at risk. This stems from the fact that all transaction details in Bitcoin are visible to everyone in unencrypted form. Such details include but are not limited to sender and recipient addresses as well as the exchanged amounts. Despite the use of pseudonymous identities in the form of public keys, it is still possible for an adversary to undermine the privacy of users. While a single transaction reveals very little information, research has shown that linking multiple transactions together can expose users' actual identities, interactions, and financial data. Having such information exposed can, in turn, lead to undesirable consequences, e.g., attract criminals, enable extortion or discrimination, and benefit competitors.

In the literature, several studies have focused on Bitcoin privacy and were able to analyze the chain of interactions

between users, identify relationships, and reveal users' real identities [56, 49, 35, 42]. This, in turn, has motivated research in both academia and industry to find solutions and methods to overcome privacy leaks. This has led to a plethora of either (i) new proposals for separate projects that have inherent privacy such as Zcash and Monero, or (ii) proposals for privacy improvement in Bitcoin. These two approaches are categorized, respectively, as (i) built-in data privacy and (ii) add-on data privacy [62]. In this paper, we consider only privacy methods proposed for Bitcoin, and more specifically mixing-based techniques [34]. In particular, we aim to evaluate and compare existing mixing approaches by analyzing their privacy, security and, efficiency and studying their applicability to the Bitcoin blockchain. The research questions investigated in this study are as follows.

**(RQ1)** How do existing mixing techniques compare in terms of privacy e.g., anonymity set, unlinkability, untraceability, and value privacy?

**(RQ2)** How resistant are mixing techniques to security attacks, e.g., theft, DoS, and Sybil?

**(RQ3)** How do existing mixing techniques compare in terms of efficiency e.g., no interaction with input users, no interaction with the recipient, Bitcoin compatible, sending the coins directly to the recipient, number of transactions, minimum required blocks?

The contribution of the paper is two-fold: a review of the literature, and the evaluation of mixing techniques. In Section 2, the main concepts are introduced and a selection of de-anonymization attacks are outlined. Section 3 discusses mixing techniques, while Section 4 evaluates them according to predefined criteria and provides a discussion. Finally, Section 5 concludes the work and summarizes the challenges.

\* This document is the results of the research project funded by COMET.

✉ sghesmati@sba-research.org (S. Ghesmati);

wfdhila@sba-research.org (W. Fdhila); eweippl@sba-research.org (E.

Weippl)

ORCID(s):

## 2. Background

In December 2008, Satoshi Nakamoto [52] published the Bitcoin white paper as a peer-to-peer (P2P) electronic cash system. Bitcoin users communicate over a P2P network, and exchange assets in the form of virtual currencies; i.e., bitcoins, that are assigned to cryptographic addresses and can be spent by providing the corresponding private keys [2]. In the following, we use the term bitcoin to refer to the cryptocurrency, and Bitcoin to refer to the underlying blockchain. Bitcoin consists of a sequence of chained blocks, each identified by a block header, and refers to a previous block (the block parent). This forms a chain of blocks that ties to the first block; i.e., the "genesis block" [2]. Transactions are recorded in a distributed, permanent, and verifiable manner. The ledger is immutable, and no data within it can be edited or deleted.

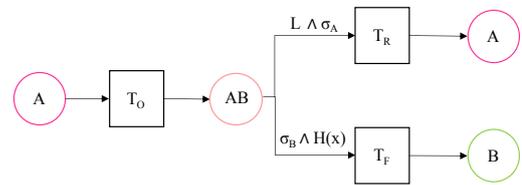
### 2.1. Fundamentals

**Address.** Asymmetric cryptography, in which public and private keys are used, is applied in Bitcoin. The addresses are the hash of public keys and users are able to unlock the coins associated with that address using the signature computed by the corresponding private key. Script hash is another type of address that allows transactions to be sent to a script hash [2]. Note that Bitcoin uses an elliptic curve digital signature algorithm (ECDSA) [11] to generate signatures.

**Transaction.** In Bitcoin, a transaction transfers the value from one user to another [14]. An unspent transaction output (UTXO) which belongs to a sender is used as the input of the transaction and the recipient address is the output. A new so-called "change address" is created and used as the output to transfer the remainder of the coins to the sender. A transaction contains attributes such as transaction id, version, inputs, outputs, and nLockTime (a parameter that specifies the time before which a transaction cannot be accepted into a block). Each input refers to the output of a previous transaction. To avoid double-spending [75], Bitcoin stores a list of UTXOs [77]. Once an output is spent, it is automatically removed from the list.

**Transaction fee.** To overcome flooding attacks [78], Bitcoin requires paying a transaction fee to miners to include the transaction in a block. This is calculated by subtracting the sum of the input values from the sum of the output values [79]. Note that large transactions often require higher fees in order to be confirmed [37].

**Transaction scripts.** Bitcoin script [73] is a Forth-like [76] stack-based language, called Script. Script words, which are also called operation codes, (opcodes) begin with "OP\_" as their prefix. A list of opcodes can be found in [73]. The Bitcoin script was designed to be simple and executable in most hardware while requiring minimal processing [2]. Transactions use scripts to specify the conditions under which the coins can be spent [37]. The vast majority of transactions in Bitcoin employ pay-to-public-key-hash (P2PKH) script [9]. Other forms of scripts can enable more complex conditions



$\sigma_A$ : Alice's signature.  $\sigma_B$ : Bob's signature. L: Lock time.  $T_O$ : Offer transaction.  $T_R$ : Refund transaction.  $T_F$ : Fulfill transaction.

**Figure 1:** Hash time locked contracts (HTLC)

for spending the coins, e.g., pay-to-script-hash (P2SH) [10] and multi-signature [70, 8].

**Timelock transaction.** A timelock transaction [71] restricts spending the coins until the specified time, which can be used for a refund. The transaction will be valid at the time set in the Bitcoin transaction either in block height or time.

**Hashlock transaction.** A hashlock transaction [72] is locked by a hash and can be spent by providing a pre-image of the hash. Pre-image is the data that was hashed and put in the condition of unlocking the output. Note that multiple transactions can be locked by the same hash. These transactions are not published unless a user behaves maliciously. Once one of the transactions is unlocked, the hash is revealed in the blockchain, and consequently, all the transactions that were locked with this hash can be redeemed [26]. To prevent redeeming the coins by other users in the blockchain, hashlock transactions are locked by both signature and the pre-image of the hash.

**Hash time locked contracts (HTLC).** HTLC [74] is a script that employs both hashlock [72] and timelock [71] transactions. The output is locked by a hash and if the recipient is unable to unlock it in a specific time, the coins are returned to the sender. Figure 1 illustrates an HTLC transaction, where Bob can fulfill the transaction by providing pre-image (x) as well as his signature, and Alice can receive the refund via  $T_R$  after the locktime.

### 2.2. De-anonymization in Bitcoin

The public availability of the Bitcoin blockchain introduces privacy issues for blockchain users. Indeed, a combination of heuristics along with information from other resources such as forums, online shops, etc., can effectively cluster the transactions and identify the users. In the following subsections, we will review the common heuristics used to de-anonymize blockchain users, and then examine recent research into the known attacks on the Bitcoin blockchain in which anonymity can be compromised. There is a degree of uncertainty around the term "anonymity" in the blockchain area. Consider the definition of anonymity proposed by [55]: "The subject is not identifiable within a set of subjects, the anonymity set". Bitcoin is not fully anonymous, and a multitude of studies [56, 49, 35, 22, 42] have demonstrated possible deanonymization by mapping Bitcoin addresses to their real entities. In the following, we present some of the most prominent techniques [15, 49].

**Common input ownership.** When a transaction has mul-

multiple inputs, each of the inputs should be signed by its associated signature. It is assumed that all the inputs in a transaction belong to the same user since it is not usual that multiple users join to create a transaction [15]. Considering this, the common input ownership heuristic considers all the inputs of a transaction to one user. According to [35], the heuristic is able to identify almost 69% of the addresses stored in the clients' wallets.

*Detecting change addresses.* When the sum of transaction inputs is larger than the sum of its outputs, a first use address called a change address is created, which returns the remainder of the coins to the sender [24]. This heuristic means that the change address is controlled by the owner of the input addresses [15].

*Transaction graph.* A transaction graph can effectively demonstrate the flow of bitcoins between users. In this graph, Bitcoin addresses represent the graph nodes, and transactions linking input addresses to output addresses are the graph edges [25]. As a transaction input is related to an output of a previous transaction, it becomes possible to identify relationships between the transactions [50]. Moreover, most of the transactions have change addresses which are under the control of the input entities (pseudonymous user) and, therefore, can be linked to the same entity in the transaction graph. By interacting with services that require users' real identities, their Bitcoin addresses can be linked to their identities, and consequently, the relationship between the previous transactions can be obtained in the transaction graph. Therefore, as also stated in [38], using a fresh address for every transaction cannot prevent a privacy leakage. Another issue is that the sender knows the recipient's address and consequently can identify further transactions performed by the recipient and discover with whom the recipient has transacted.

*Linking similar addresses / Address reuse.* The addresses that are reused in the transactions can be linked together in the blockchain belonging to the same entity.

*Side-channel attacks.* Side-channel attacks [15] such as time correlation, amount correlation, and network-layer [13] information can reveal the transactions and users.

*Auxiliary information.* Auxiliary information [15] from e.g., forums, merchants, search engines can be used to tag the addresses.

### 2.3. Related works on de-anonymization

In recent years, several works have addressed user de-anonymization in blockchain using the techniques in the previous subsection. Meiklejohn et al. [49] clustered Bitcoin wallets based on evidence of shared authority and then utilized re-identification attacks to classify the users of the clusters. They conclude that the information collected by Bitcoin businesses such as exchanges along with the ability to label monetary flows to those businesses curb the willingness to use Bitcoin for illicit activities. Reid and Harrigan [56] analyzed anonymity in Bitcoin by considering the topological structure of two networks derived from Bitcoin's public transaction history, showing how various types of informa-

tion leakage have the potential to contribute to de-anonymizing Bitcoin's users. They employed flow and temporal analysis in the research thereby identifying more than 60% of the users in the visualization and revealing their relationships. Harrigan and Fretter [35] explored the reasons for the effectiveness of simple heuristics in Bitcoin. They considered the impact of address reuse, avoidable merging, super-clusters with high centrality, and the growth of address clusters. Ermilov et al. [22] utilized off-chain information as votes for address separation and considered this together with blockchain information in their clustering model. They applied blockchain-based heuristics such as common input ownership, detecting the change address along with off-chain information for clustering. Jourdan et al. [42] defined features for classifying entities from a graph neighborhood perspective, as well as centrality and temporal features in the Bitcoin blockchain, classifying addresses into exchanges, gambling services, general services, and darknet categories. The results of the above-mentioned research heighten the necessity to enhance blockchain privacy through effective techniques. In the next section, we will explore existing mixing techniques.

## 3. Mixing techniques

Transactions consist of multiple inputs and outputs, which can be traced using sophisticated analytical tools. The mixing mechanism hides the correlation between inputs and outputs such that an attacker cannot trace an input by looking into the blockchain. The links between the recipient's addresses as well as the value of the transaction can also be hidden using enhanced techniques. Various mixing techniques exist and differ in terms of privacy, security, and efficiency. These techniques can be categorized into centralized mixers, atomic swap, CoinJoin-based, and threshold signatures. These will be discussed in the following subsections. Figure 2 illustrates the categorization of the techniques while indicating their evolution.

We first outline the research methodology adopted for the identification, selection, and synthesis of the research items included in this study.

### 3.1. Research methodology

In this study, we have followed common guidelines for research synthesis comprising (i) the identification of research questions, (ii) search and selection of the literature, and (iii) the analysis and synthesis of extracted data. Blockchain, privacy, mixing, tumbler, tumbling, and Bitcoin keywords were searched in IEEE xplora, Springer, and Science Direct databases to find related research items. Additionally, "arxiv.org" and "eprint.iacr.org" were used to identify unpublished papers. Moreover, we conducted a direct search on "Github" for existing implementations of mixing techniques. In total, we obtained 869 research papers. All the titles were read, and papers with no relevant content were dropped. Next, all abstracts were read and papers with no relevant content according to the abstracts were also removed. Only papers published between 2009 and 2020 were considered in our research. Duplicates, and items not focused

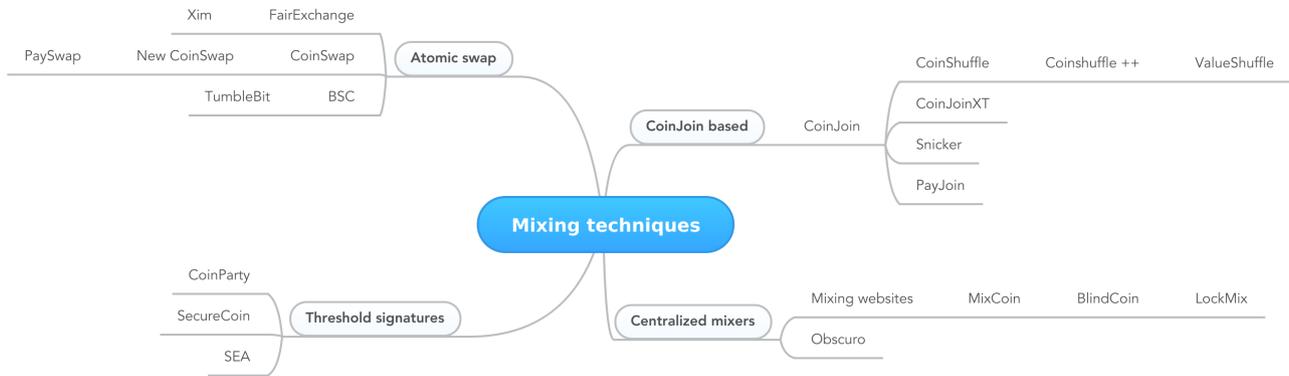


Figure 2: Evolution of mixing techniques

on mixing methods, or not related to Bitcoin were also excluded.

The literature for systematization is selected based on the following criteria, which are inspired by [1]: (i) **Scope:** The technique is compatible with the Bitcoin blockchain at least via soft-fork, and (ii) **Disruption:** The paper technique is a novel area that the community is investigating. (iii) **Merit:** The technique which explores privacy solutions is unique.

Our second contribution is the evaluation of the selected mixing techniques using the criteria defined in the following paragraphs. The mixing techniques have been evaluated over three main categories: Security, privacy, and efficiency. Several criteria have been proposed in the literature, and our selected criteria inspired by commonly used criteria in the recent research.

## 3.2. Centralized mixers

In this subsection, we investigate the mixing methods that rely on a centralized party, where senders forward their coins to a central mixer, which mixes and forwards them to the corresponding recipients.

### 3.2.1. Mixing websites

The mixing idea was initially proposed by Chaum [19] in 1981 to ensure anonymous email communication without relying on a universal trusted authority. Similar techniques have been lately employed to address anonymity in blockchain. The latter employ mix networks, e.g., mixing websites, to obfuscate the links between senders and receivers. For example, if Alice, Bob, and Carol want to send their coins to A', B', and C', respectively, then they will collectively use a mixer for their transactions (Figure 3). The latter receives senders' coins in equal amounts, mixes them, and forwards them to the recipients' addresses. Looking at the published transactions, one cannot distinguish whether Alice sent her coins to A', B', or C'. On most mixing websites, users are asked to fill out a form in which they fill out the recipient's address and select their preferred mixing delay. Subsequently, a fresh address is generated by the mixer to

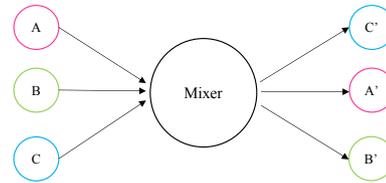


Figure 3: Mixing websites

receive the coins from the sender. The fresh address is revealed along with the mixing fee, transaction fee, and the conditions for the user.

### 3.2.2. MixCoin

MixCoin [15] was proposed by Bonneau et al. as a Bitcoin mixer to prevent theft in mixing services using the mixer signature as a warranty in case it acts maliciously. According to the protocol, the mixer has to sign the sender's mixing parameters (e.g., recipient address, preferred deadlines to transfer the coins, value, mixing fee). In case the mixer does not forward the coins to the intended recipients, the sender has the possibility to publish the warranty. This way, anyone can verify that the mixer acted maliciously, causing negative impacts to its reputation. In MixCoin, the mixing fee is all or nothing, as a constant mixing fee can reveal mixing transactions in sequential mixing. To improve anonymity, mixing transactions have a standard chunk size among all the mixers to make uniform transactions. Chaining multiple mixing together can provide strong anonymity.

### 3.2.3. BlindCoin

BlindCoin [65] adds Blind signatures to Mixcoin. Blind signature was proposed by Chaum [18] to sign a message without revealing the content to the signer. Using blind signature in BlindCoin hides the relationship between input and output addresses from the mixer itself, where the mixer gives her warranty by blindly signing the recipient's address. Later, the sender anonymously submits the unblinded recipient's address to the mixer using a new identity. The mixer will

send the coins to the recipient's address as it can see its signature on the recipient's address.

### 3.2.4. LockMix

LockMix [3] is a central mixer that improves BlindCoin [65] by preventing the mixer from stealing the coins using multi-signature. To run the protocol, the mixer announces parameters including user deposit, value, waiting blocks, and mixing fee in the network. Alice adds the desired times that are considered as the deadlines for the protocol's steps along with the blinded recipient's address and her address  $K_A$  to create a multi-signature address. The mixer creates a 2-of-2 multi-signature address ( $K_{AM}$ ) with Alice's address ( $K_A$ ) and its own address ( $K_M$ ). The mixer adds the multi-signature address and its escrow address, signs all the parameters, and sends it back to Alice. Then, Alice deposits an amount which is larger than the mixing value to  $K_{AM}$  as collateral, unblinds the recipient's address, and sends it to the mixer. The mixer sends the coins to the recipient, Alice waits for the agreed confirmation blocks and then sends the coins to the mixer escrow address. Finally, Alice creates a transaction transferring the mixing fee to the mixer and the remainder to herself from the 2-of-2 multi-signature transaction. Alice and the mixer should both sign the transaction to receive the coins. Although Alice and the mixer can abort the protocol at each of the aforementioned steps, this does not benefit any of them. Both may lose their coins or benefits if they misbehave, which is a lose-lose scenario.

### 3.2.5. Obscuro

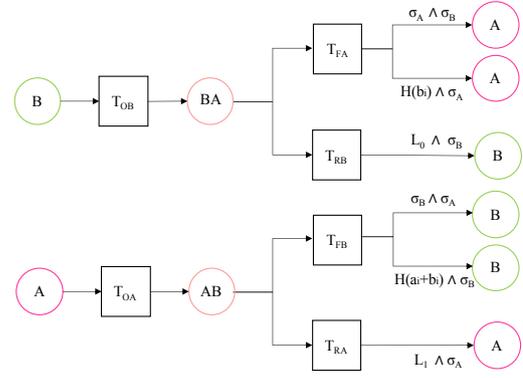
Obscuro [64] is a central mixer that employs a trusted execution environment (TEE). To run the protocol, the mixer generates the key in TEE and publishes the public key and Bitcoin address. All the users send their coins to a single address of the mixer and publish their transactions in the network. Encrypted recipient addresses along with a transaction refund script are included in the transaction to retrieve the coins in case that they are not spent by the lock time. Obscuro scans the blockchain and extracts these transactions. The mixer decrypts recipients' addresses, shuffles them, and transfers the coins to the corresponding recipients' addresses. A mixing transaction contains all users' deposit transactions as inputs and the shuffled list of recipient's addresses as outputs. The protocol contains maximum and minimum participants for the mixing set as well as the number of blocks to wait to perform the mixing transaction. Specifying the minimum number of participants assures users of the mixing set size before they participate in the protocol.

## 3.3. Atomic swaps

Atomic swap techniques enable users to exchange their coins with each other in a way that if one party is paid the other is also paid.

### 3.3.1. FairExchange

This protocol, which was proposed by Barber et al. [4], enables two users to swap their coins. As such, Alice sends



$\sigma_A$ : Alice's signature.  $\sigma_B$ : Bob's signature.  
 $T_{RA}$  &  $T_{RB}$ : Refund transactions.  
 $T_{OA}$  &  $T_{OB}$ : Offer transactions.  
 $T_{FA}$  &  $T_{FB}$ : Fulfill transactions.  
 $L_0$  &  $L_1$ : Lock times.

Figure 4: FairExchange

the coins to Bob's recipient address and Bob sends the coins to Alice's recipient address. In the first step, Alice and Bob create two key pairs to use in different transactions, and then generate secret values  $a_i$  and  $b_i$  and engage in a cut and choose protocol to provide  $H(a_i + b_i)$  and  $H(b_i)$ , which will be included in the offer transactions. As illustrated in Figure 4, in order to offer the coins, Bob creates  $T_{OB}$  that can be redeemed by either Alice's and Bob's signatures or Alice's signature and  $b_i$ . Bob also creates a transaction refund  $T_{RB}$  to ensure that he can retrieve his coin if the protocol is aborted. He waits for Alice to sign  $T_{RB}$  and publishes  $T_{OB}$  and  $T_{RB}$ . Alice does the same, where  $T_{OA}$  can be redeemed by both Bob's and Alice's signatures or Bob's signature and  $a_i + b_i$ . Bob fulfills Alice's transaction in  $T_{FB}$  by providing his signature and  $a_i + b_i$ . Alice then subtracts  $a_i$  and obtains  $b_i$  to fulfill Bob's transaction ( $T_{FA}$ ).

### 3.3.2. Xim

Xim [7] proposes a novel approach to finding a user to perform FairExchange transactions (3.3.1). The protocol uses blockchain to advertise mixing requests, in which Alice pays  $\frac{\tau}{2}$  coin to the miner to put her advertisement on the block including her location to contact the partners (e.g. Onion address or Bulletin board). She can then be contacted by several participants and choose one for partnership. The selected participant should then pay  $\tau$  coin to the miner (to prevent Sybil and DoS attacks) to start creating the transactions. Otherwise, Alice chooses another participant. Once the participant pays the fee, Alice pays another  $\frac{\tau}{2}$  to confirm the partnership. Afterward, they can swap the coins to their recipients' addresses using FairExchange.

### 3.3.3. CoinSwap

CoinSwap [48] prevents the coins being stolen by the intermediary. It requires four transactions in total, two for payments and two for releases. To run the protocol, Alice creates a 2-of-2 multi-signature transaction  $T_0$  that requires both Alice's and Carol's signatures ( $\sigma_A \wedge \sigma_C$ ) to spend the

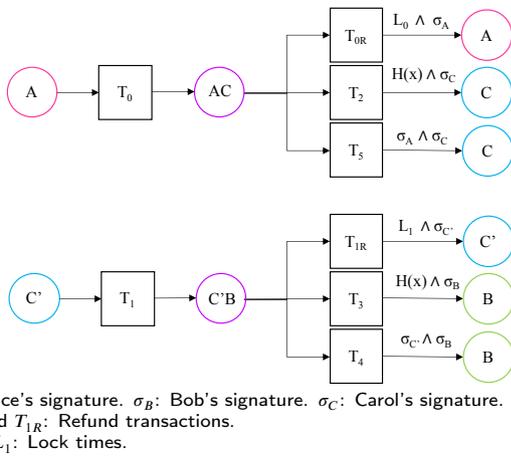


Figure 5: CoinSwap

coins (Figure 5). Carol creates a multi-signature transaction  $T_1$  that requires both Carol's and Bob's signatures ( $\sigma_C \wedge \sigma_B$ ). First, Carol and Bob create refund transactions  $T_{0R}$  and  $T_{1R}$ , which guarantee that the coins are sent back to Alice and Carol, respectively, if the protocol is abandoned. To ensure fairness, the timelock of  $T_{0R}$  should be longer than  $T_{1R}$ . To prevent the coins being stolen if anyone cheats, the protocol employs hashlock transactions. To create hashlock transactions, Bob chooses a random value  $x$ , computes the hash of  $x$  ( $H(x)$ ), and sends the hash to Alice and Carol. Alice creates a hashlock transaction  $T_2$  that can be redeemed by Carol's signature and  $x$ . Carol in turn creates a hashlock transaction  $T_3$  that can be redeemed by Bob's signature and  $x$ . Bob should publish  $x$  to claim  $T_3$ , which allows Carol to claim  $T_2$ . This hashlock is only to claim the coins in the case of misbehavior by users. Otherwise,  $x$  should not be published as it reveals the link between these transactions in the blockchain. Next, Carol waits to receive  $x$  from Bob and then creates  $T_4$  (to send the coins to Bob) and sends it to Bob to sign the transaction and then publishes it to the blockchain. When  $T_4$  is confirmed, Alice creates  $T_5$  (to send the coins to Carol) and sends it to Carol to sign and publish it to the blockchain. If Bob does not show  $x$  to Carol, she should redeem the coins from  $T_{1R}$  before the expiry of  $T_{0R}$ . If so, Alice can retrieve her coins from  $T_{0R}$ , while Bob can redeem the coins from hashlock simultaneously. Carol should use a different identifier for the transactions between Carol and Bob [51]. The key point in CoinSwap is that the transactions are in two different paths which makes it possible to have them in two different blockchains that support timelock and hashlock transactions (e.g. between Bitcoin and Litecoin) [28].

### 3.3.4. New CoinSwap

Since CoinSwap was proposed before the advent of check lock time verify (CLTV) [63] or check sequence verify (CSV) [16] Opcodes, it used the nLockTime feature to create refund transactions. The hashlock transactions were also kept as backouts and should not be published. New CoinSwap [28] uses CLTV to create hash time locked transactions as

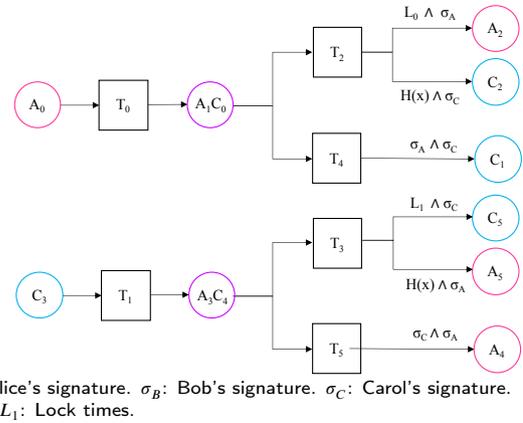


Figure 6: New CoinSwap

well as using segwit to prevent malleability and as a result, the protocol is theft-resistant. In this protocol, Alice takes the role of Bob in the original CoinSwap, which removes the interaction with the recipient. Instead, Alice first sends the coins to her fresh address and then can use mixed coins to send to the real recipient.

### 3.3.5. PaySwap / Design for a CoinSwap Implementation

PaySwap [6] is an improvement over CoinSwap. PaySwap utilizes two-party ECDSA to create 2-of-2 multi-signature addresses. These kinds of addresses are similar to regular single-signature addresses. Alice1 pays Bob1 and Bob2 pays Alice2 by CoinSwap transactions, such as Joinmarket wallet [5]). Alice can be a market taker and Bob can be a market maker. Alice pays a fee to Bob as a market taker. To prevent amount correlation in which an attacker can search the transaction values and find Alice2, PaySwap proposes multi-transactions in which Alice sends the coins to Bob and receives multiple transactions with different amounts with a total of those coins. To address internal traceability, in which Bob can trace Alice's coin flow, Alice can route her coins through many market takers (Bob, Carol, and Dave). In this route, a market taker only knows the previous and the next addresses. Alice will inform every market taker of the CoinSwap incoming address and outgoing address. Thus, none of the makers is able to distinguish whether the incoming address belongs to Alice or the previous market maker. Combining multi-transactions with routing is also proposed in order to enhance the privacy of the transactions. The combination of CoinSwap with PayJoin (3.4.7) is also considered as a possible solution to breaking multi-input transactions heuristics, in which Bob's input can be added to Alice's inputs in the transaction.

### 3.3.6. Blindly signed contract (BSC)

The protocol [38] is proposed in two schemes: (i) on-blockchain and (ii) off-blockchain. The former uses untrusted intermediaries while the latter utilizes micro-payment channel networks, where transactions are performed off-chain and

their confirmations on-chain. The on-chain scheme consists of two FairExchange transactions, which means four transactions should be submitted on the blockchain, where Alice sends the coins to Bob via an intermediary (Carol). To initiate the protocol, Carol posts public parameters including blind signatures parameters, a transaction fee, reward value ( $w$  is considered as a mixing fee in the protocol), and transaction time windows on the blockchain. Then, Bob chooses a fresh address to receive the coins. To create transactions, Alice selects a serial number and sends its hash to Bob. Bob sends this hash to Carol and asks her to create a transaction ( $TO_{C \rightarrow B}$ ) in which Carol offers one coin to Bob if she receives a voucher  $V = (sn, \sigma)$  that contains a serial number which has an equal hash to the one Bob provided beforehand. Carol posts  $TO_{C \rightarrow B}$  on the blockchain. Alice blinds the serial number ( $\overline{sn}$ ) and creates a transaction offering  $1+w$  coins to Carol if Carol provides a blind signature on ( $\overline{sn}$ ). Once it is confirmed in the blockchain, Carol fulfills the transaction  $TF_{A \rightarrow C}$  (which pays to Carol) by providing a blind signature ( $\overline{\sigma}$ ) on ( $\overline{sn}$ ). Alice can obtain  $\sigma$  and send the voucher to Bob to fulfill the transaction  $TF_{C \rightarrow B}$  that sends the coins from Carol to Bob.

### 3.3.7. TumbleBit

TumbleBit [37] uses the puzzle solution to provide privacy in the case of an untrusted central tumbler. The protocol requires four transactions confirmed in two blocks. As a high-level description, the tumbler gives the coins to Bob if he solves a puzzle and then the tumbler sells a solution to the puzzle provided by Alice for the same amount of coins. To run the protocol, the tumbler creates a Rivest–Shamir–Adleman (RSA) [57] puzzle for the solution  $\epsilon$ , and takes an ECDSA signature encrypted under the solution to the RSA puzzle, and creates a ciphertext  $c$ . This signature represents a transaction signature that allows Bob to spend one bitcoin out of the transaction escrow.

As illustrated in Figure 7, the tumbler sends the puzzle  $z$  and ciphertext  $c$  to Bob, who blinds it to obtain  $z^*$ , and sends it to Alice. Then, Alice creates FairExchange transactions with the tumbler to obtain the blinded solution by paying the tumbler the desired amount. Alice sends the blinded puzzle to the tumbler (who can solve the puzzle) and gets  $\epsilon^*$ . Alice then sends  $\epsilon^*$  to Bob. Bob unblinds it to  $\epsilon$  and receives the coins from  $T_{F2}$ .

## 3.4. CoinJoin-based

CoinJoin-based mixing employs techniques that do not require trusted third parties, thus eliminating a single point of failure. Furthermore, they can prevent theft and remove mixing fees in most of the proposed techniques.

### 3.4.1. CoinJoin

CoinJoin was proposed by Maxwell in 2013 [47]. Since the inputs of the Bitcoin transactions should be separately signed by the associated signatures, the users are able to jointly create one transaction with their inputs. In this man-

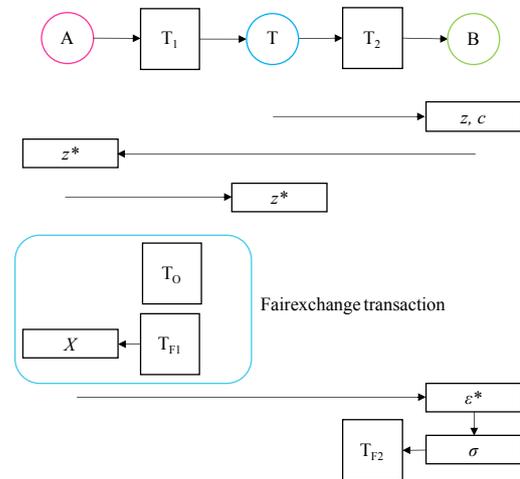
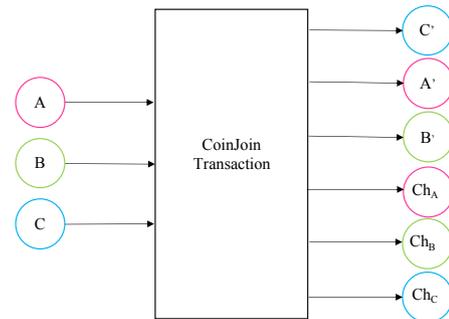


Figure 7: TumbleBit

ner, not only they can break the common input ownership heuristic, they can also hide the relation of inputs and outputs of a transaction if they send similar coins to the output addresses. To create the transaction (Figure 8), users provide their input, output, and change addresses, and separately sign the transaction. One of the users combines the signatures and broadcasts the transaction to the network.



$Ch_A, Ch_B, Ch_C$ : Change addresses.

Figure 8: CoinJoin

### 3.4.2. CoinShuffle

CoinShuffle [59] is an improvement over CoinJoin to reach untraceability against mixing users, inspiring Dissent protocol [20]. The users find each other via a peer-to-peer protocol, after which each user (except Alice) creates a fresh encryption-decryption key pair and announces the public encryption key. Alice creates layered encryption of her output address  $A'$  with all the users' encryption keys and sends it to Bob (Figure 9). Bob decrypts it and creates layered encryption of his own output address  $B'$  with the remaining keys. Afterward, he shuffles the outputs and sends them to the next one who repeats the same. The last user receives all outputs, adds her own output, shuffles them, and sends the shuffled list to all users. Each user is able to verify whether her/his output is on the list. Each user creates a transaction from all

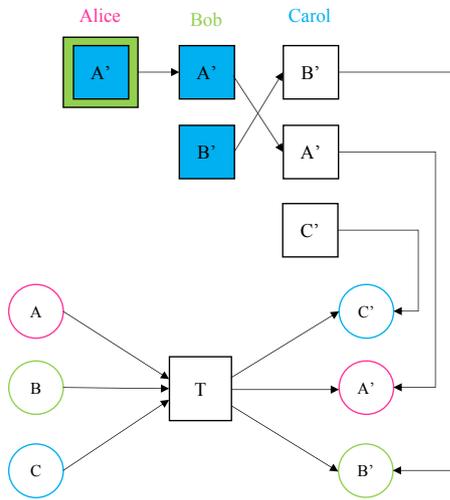


Figure 9: CoinShuffle

the inputs to the shuffled outputs, signs it, and broadcasts it to other users. Once all the users broadcast their signatures, one of them can create a fully signed transaction and publish the mixing transaction to the network. The protocol can enter the blame phase if at least one user acts maliciously. The malicious user will be excluded and the protocol will rerun.

### 3.4.3. CoinShuffle++

CoinShuffle++ [60] uses DiceMix protocol [60] which requires sequential processing. Its predecessor CoinShuffle [59] requires a number of communication rounds linear in the number of users. CoinShuffle++ utilizes DiceMix to process mixing in parallel, which is independent of the number of users and requires only a fixed number of communication rounds. A mixing transaction with 50 users in CoinShuffle++ can be performed within eight seconds.

### 3.4.4. ValueShuffle

ValueShuffle [58] is an extension of CoinShuffle++. The protocol combines CoinJoin with Confidential transactions and Stealth addresses. Using confidential transactions provides transaction value privacy, which is a great improvement in comparison with its predecessors in that it not only hides transaction value from prying eyes but also enables users to mix different amounts of coins with each other. Additionally, the recipient's anonymity (considered as unlinkability in our paper) can be guaranteed using Stealth addresses, which makes it possible to send the coins directly to the recipient's address. A Stealth address is a unique one-time address that is generated by the sender to improve the recipient's privacy.

### 3.4.5. CoinJoinXT

CoinJoinXT [32] proposes a form of CoinJoin transaction in which users first send the funds to a funding address that they jointly control using multi-signatures. Next, they sign a set of spending transactions from this address in ad-

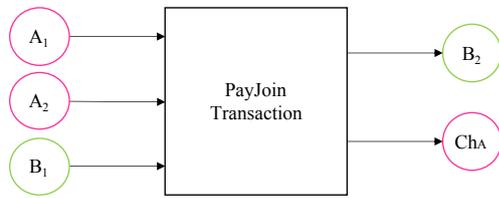
vance (using Segwit solves transaction malleability). All the spending transactions should be given a specified time lock to prevent publishing them at once. Spending transactions can be a chain or a tree. All the transactions should require the signature of both parties. Once they validate the signatures on the transactions, they can broadcast the funding address. It is also possible to add the UTXOs of each user in the subsequent transactions. However, to prevent double-spending, they should also create and pre-sign a backout transaction for each round, which has a specified time lock. To prevent subset-sum attacks, participants can use off-chain privacy e.g. channel in the Lightning network to send part of the outputs to the channel and shift the balance over time. Distinguishability of multi-signature in P2SH script can be solved by Schnorr signature [61] or Scriptless ECDSA-based Construction [45] to form 2-of-2 multi-signature transactions such as P2PKH transactions.

### 3.4.6. Snicker

Snicker [29] is a simple non-interactive CoinJoin where the keys for encryption are reused. It can be achieved without a server or interactions between participants. It is useful in a CoinJoin between two parties, in which one of the participants (Alice) encrypts the request by the public key of the other participant (Bob) to create a CoinJoin proposal. Alice should scan the Blockchain to find potential participants according to the amount and the age of his UTXO. Scanning the blockchain can be done by one of the block explorers and Alice only downloads the data to find the active users. Alice's message contains her UTXO, the desired recipient address, the amount, the transaction fee, the full transaction template by UTXOs of Alice and Bob, Alice's signature on the transaction, and Bob's recipient address, which is created by adding  $k'G$  to either Bob's existing reused public key (Version-1) or R value in one of Bob's signatures (Version-2). Alice includes  $k'$  value in the encrypted message to enable Bob to derive the private key of the newly generated public key. Alice sends the encrypted message in the network, e.g. a Bulletin board. Bob can decrypt the message, verify the ownership of the newly proposed public key, sign, and broadcast the transaction to the network.

### 3.4.7. PayJoin

PayJoin [31] (similar to Bustapay [36] and P2EP [12]) solves the distinguishability of CoinJoin technique by adding at least one UTXO of the recipient to the inputs of the transaction. It breaks the multi-input ownership heuristic as one of the most prominent heuristics in the de-anonymization of Bitcoin users. Moreover, it hides the true payment amount as the output will be more than the real payment amount. To run the protocol [30], Bob sends the recipient's address and amount. Alice creates and signs a transaction in which she sends the specified amount to Bob's address and provides her change address to receive the remainder and then sends the transaction to Bob. Bob checks the transaction and creates a new transaction by appending his inputs to the transaction created by Alice. Then he alters the output amount and adds



$Ch_A$ : Alice's change address.

**Figure 10:** PayJoin

up his inputs to the final amount. He signs his inputs and sends this new transaction to Alice. Alice checks and signs the transaction and broadcasts it to the network. Figure 10 illustrates a simple form of PayJoin.

### 3.5. Threshold signature

Threshold signature techniques use joint signatures, which can be signed by a specified threshold of the signatures to redeem a transaction.

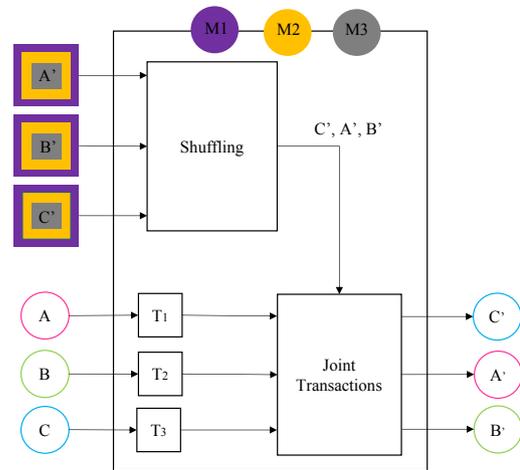
#### 3.5.1. CoinParty

CoinParty [79] employs mixing peers instead of group transactions in CoinJoin-based protocols to provide plausible deniability. It employs a threshold variant of ECDSA (inspired by [40]) using secure multi-party computation (SMC).

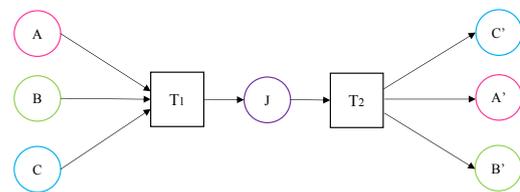
In the first phase, mixing peers generate a set of escrow addresses ( $T_1$ ,  $T_2$  and  $T_3$ ) from threshold ECDSA, which is under the joint control of mixing peers, and then send a different escrow address to each input peer. Input peers commit their coins to the escrow addresses (Figure 11). The coins in the escrow addresses can only be redeemed if the majority of mixing peers sign the transactions. In the shuffling phase, input peers utilize layered encryption to encrypt their output addresses by the public keys of the mixing peers. Then, they broadcast the encrypted output along with the hash of their output to the mixing peers (to be used in checking the final shuffled addresses by mixing peers). Each mixing peer decrypts the output and shuffles the address and sends it to the next peer. The last mixing peer shuffles the output addresses and broadcasts it to the mixing peers. All mixing peers check the shuffled addresses and seed a pseudo-random number generator (PRNG) to obtain a final permutation of outputs, which prevents the final peer from controlling the last permutation and ensures random shuffling. Finally, the mixing peers send the coins to the output addresses. As the private keys of the escrow addresses are shared among mixing peers, a threshold variant of ECDSA is applied to create and sign each of the transactions.

#### 3.5.2. SecureCoin

SecureCoin [39] uses the CoinJoin technique along with the threshold digital signature to mix the coins. In the first step of the protocol, a joint address ( $J$ ) is generated by users in the threshold bases. To do this, a public key should be jointly computed by the users. Once the address is generated, the users jointly perform a transaction ( $T_1$ ) to send their



**Figure 11:** CoinParty, inspired by [79]



**Figure 12:** Securecoin

coins to address  $J$  (Figure 12). In the next step, they generate fresh recipient addresses and shuffle the addresses. Address shuffling and blame phase (accusation in SecureCoin) are almost similar to CoinShuffle, in which the users encrypt their recipient addresses and send them to the next user to decrypt the message and add the recipient address. If a user misbehaves, the protocol enters the accusation phase, in which the malicious user is excluded and the shuffling is repeated by the remaining users. In the last step, users create a transaction ( $T_2$ ) from address  $J$  to the recipient addresses of honest users and the input addresses of kicked out users in the accusation phase. Therefore, a user retrieves her coins even if she behaves maliciously in the protocol. To complete the protocol, the majority of the users should jointly sign the transaction.

#### 3.5.3. Secure Escrow Address (SEA)

Secure Escrow Address (SEA) [68] is a decentralized protocol that employs distributed key generation as proposed by [27] to send the coins first to a temporary address in the joint control of the users and then to the recipients' addresses. To do this, all the users jointly create a public key of a joint address ( $J$ ), where each user has a share of the secret to redeem the coins from address  $J$ . Next, each user generates an encryption-decryption key pair similar to CoinShuffle [59], then they shuffle the recipients' addresses using layered encryption and the last user broadcasts the shuffle list. Each user checks the list to verify whether her recipient's address is included or not. If everything goes right, they create a transaction and transfer the coins from address  $J$  to the re-

ipients' addresses. To redeem the coins users should sign the transaction by their own share of the secret. The protocol has not been implemented in the proposed paper and there are no test results to see how the distributed key generation works in the ECDSA scheme.

## 4. Discussion and Analysis

In this section, the privacy, security, and efficiency properties of the selected mixing techniques will be discussed. Additionally, a review of their implementation in practice will be presented and future research explored.

### 4.1. Evaluation criteria

In this subsection, we evaluate the techniques in terms of privacy, security, and efficiency criteria. Figure 13 outlines selected criteria that were adopted from the most addressed criteria in the literature.

#### Privacy criteria.

*Anonymity set.* The set of participants in the mixing transaction that is required to enhance anonymity.

*Unlinkability.* "Given Two transactions with recipients X and Y, it is impossible (or at least computationally infeasible) to determine if  $X=Y$ , which means a user cannot receive coins from a different transaction to one specific address" [66].

*Untraceability.* "Given a transaction, all senders are equiprobable. One cannot figure out who is the sender among a transaction input addresses" [66].

*Payment value privacy.* The transaction value is protected from blockchain data analysis.

#### Security criteria.

*Theft resistance.* The coins cannot be stolen during the protocol execution.

*Dos resistance.* The participant cannot refuse to compute the transaction (considered only for decentralized peering to create a transaction).

*Sybil resistance.* The attacker cannot take part in the protocol with different identities to identify the recipient addresses with which it is paired.

#### Efficiency criteria.

*No interaction with input users.* There is no interaction with other participants for peering to create a transaction.

*No interaction with the recipient.* There is no interaction with the recipient to create the mixing transaction.

*Bitcoin Compatible.* The technique is compatible with the current Bitcoin blockchain and consequently leads to compatibility with blockchain pruning.

*Direct send to the recipient.* The ability to send the coins directly to the recipient, instead of receiving the coins first to a new address of the sender and then send it to the recipient.

*Number of transactions.* The minimum number of transactions to complete the protocol.

*Minimum required block.* The minimum number of blocks to complete the protocol (i.e., currently, 10 minutes for Bitcoin).

## 4.2. Evaluation of the techniques

In table 1, we evaluate the techniques into four main categories (centralized mixers, atomic swap, CoinJoin-based, and threshold signatures), which illustrates the comparison of the techniques in terms of privacy, security, and efficiency. In what follows, we investigate the techniques in detail, according to the criteria defined.

### 4.2.1. Privacy

**Anonymity set.** This criterion is compared based on three set sizes, namely, large, medium, and small. Most of the techniques, except for some of CoinJoin-based techniques, can provide a large anonymity set and be hidden among other transactions in the blockchain. In most CoinJoin-based techniques, the anonymity set is confined by transaction size; other than that coordination between a large set of users to create a CoinJoin transaction cannot be easily achieved in practice because large anonymity sets increase the risk of DoS and Sybil attacks and boost communication overhead. The reason why we assigned moderate size to CoinShuffle ++ and ValueShuffle is that peering was enhanced in these protocols using Dicemix, in which 50 participants can create a transaction in 8 seconds, which is considered a reasonable time for this size of anonymity set. Coinswap techniques can be hidden among all the transactions with the same value in the blockchain (implementation of two-party ECDSA, where multi-signature transactions look like single-signature transactions), can effectively provide anonymity for these transactions). Although the anonymity set in atomic swap techniques is large, the timelock transactions in these techniques curb the anonymity set [51].

**Unlinkability.** In all techniques, users should create fresh addresses to receive mixed coins. However, there is no guarantee that these addresses will not be used in the future. Therefore, those addresses and their transactions can be linked to each other, which consequently can be used in the transaction graph analysis. ValueShuffle can achieve unlinkability by using stealth addresses as one-time-use payment addresses. However, the stealth addresses can be applied in other techniques to improve those techniques over unlinkability. For instance, Darkwallet, which has not been updated since 2015 [17], was the implementation of CoinJoin that applied Stealth addresses. It should be pointed out that due to the unique structure of Stealth addresses, the anonymity of these addresses is confined to the set of the users that use such addresses [51].

**Untraceability.** All the techniques attempt to improve the untraceability of transactions in the blockchain. However, the techniques that have partial coverage of this feature are those that have internal traceability, in which the relationship between the inputs and the outputs is traceable among the participants in the mixing techniques. When a technique is internally traceable, the involved participants are able to store the other user's data, which can lead to information leakage. It should be mentioned that even if they are traceable among the participants, they provide privacy against blockchain an-



Figure 13: Mixing techniques criteria

		Anonymity set	Unlinkability	Untraceability	Value privacy	Theft resistance	DoS resistance	Sybil resistance	No interaction with input users	No interaction with recipient	BTC Compatible	Direct send to recipient	no. Tx	Min Block
		Privacy			Security			Efficiency						
Centralized mixers	Mixing websites	Large	○	●±	○	○	●	●	●	●	●	●	2	2
	MixCoin [15]	Large	○	●±	○	○×	●	●	●	●	●	●	2	2
	BlindCoin [65]	Large	○	●	○	○×	●	●	●	●	●	●	2	4△
	LockMix [3]	Large	○	●	○	○*	●	●	●	●	●	●	4	6
Atomic swap	Obscuro [64]	Large	○	●±	○	●	●	●	○	○	●	●	2	2
	FairExchange [4]	Large	○	●±	○	●	●	○	○	●	●	●	4	3
	Xim [7]	Large	○	●±	○	●	●	○	○	●	●	●	7	X*
	CoinSwap [48]	Large	○	●±	○	○°	●	●	○	○	●	●	4	2
	New CoinSwap [28]	Large	○	●±	○	●	●	●	○	○	●	●	4	2
	PaySwap [6]	Large	○	●	○	●	●	●	●	●	●	○	4	2
	BSC [38]	Large	○	●	○	●	●	●	○	○	○°	●	4	3
	TumbleBit[37]	Large	○	●	○	●	●	●	○	○	●	●	4	2
CoinJoin based	CoinJoin [47]	Small	○	●±	○	●	○	○	○	●	●	○	1	1
	CoinShuffle [59]	Small	○	●	○	●	○‡	○	○	●	●	○	1	2
	Coinshuffle++ [60]	Moderate	○	●	○	●	○‡	○	○	●	●	○	1	2
	ValueShuffle [58]	Moderate	●	●	○	●	○‡	○	○	●	○°	●	1	1
	CoinJoinXT [32]	Large	○	●±	○	●	○‡††	○	○	●	●	●	X	X
	SNICKER [29]	Small	○	●±	○	●	○	○	○	●	●	○	1	1
	PayJoin [31]	Large	○	●±	○	●	○	○	○	○	●	●	1	1
Threshold signatures	CoinParty [79]	Large	○	●	○	○⊕	○‡	○	○	●	●	○	2	2
	SecureCoin [39]	Moderate	○	●	○	○⊕	○‡	○	○	●	●	○	2	2
	SEA [68]	Moderate	○	●	○	○	○‡	●	○	●	●	○	2	2

● Full coverage ○ Partial coverage ○ No coverage  
 ± Internal traceability.  
 × Theft is detected, but it is not prevented.  
 \* It is possible in lose-lose or get nothing scenarios.  
 ° In the case of malleability of initial transactions.  
 †† In two-party cases.  
 ⊕ If 2/3 of users are honest.  
 ‡ Prevented by finding the malicious participant and excluding her.  
 ° Soft-fork is required.  
 △ Two blocks for public log messages plus two blocks for two transactions.  
 \* It is a two-party transaction, so needs many mixing transactions to achieve a large anonymity set.

Table 1: Evaluation of mixing techniques

alysts [51].

**Value privacy.** To prevent tracing of the transactions by precise value attacks, providing transaction value privacy, or hiding the actual payment value is one of the features that boosts transaction privacy. Among the techniques, ValueShuffle proposes using confidential transactions (CT) to

hide the values which require a soft-fork in Bitcoin. If CT is implemented in Bitcoin, all the techniques can benefit and there is no need for the fixed denomination in the proposed techniques, which consequently improves the usability and liquidity in other techniques where the users can mix their desired number of coins. [54] compares the implementa-

tion of CT in TumbleBit and CoinJoin and indicates that CT would decrease the mixing cost in the transactions with large values while increasing it in the transaction with small values. Furthermore, applying CT in Bitcoin transactions increases the transaction fee by a further factor of nine. Chaining the transactions by CoinJoinXT can provide a level of value privacy. However, the subset-sum may break this criterion. PayJoin also provides partial value privacy by hiding true payment amounts.

#### 4.2.2. Security

**Theft-resistance.** One of the most prominent criteria in payment networks is to prevent the coins from being stolen or lost. This criterion is crucial in blockchain as there is no practical solution to claiming the coins back (except hard-fork). While most of the techniques attempt to address this criterion, mixing websites are not theft-resistant. In MixCoin and BlindCoin, the mixer is accountable. Although theft can be detected in these techniques, it cannot be prevented. The previous exit scams in the mixing websites [37, 44] makes trusting those services hard. The techniques that are based on threshold signatures cannot significantly prevent theft as they need the majority of the users to be honest, which cannot be easily achieved in a peer-to-peer network. Atomic swap and CoinJoin-based techniques can provide this feature in the envisioned protocols.

**DoS-resistance.** According to our definition of DoS-resistance, most of the CoinJoin-based and threshold signatures techniques lack this feature as they need the users to behave honestly during the protocol. To prevent DoS attacks, finding and kicking out the malicious users and rerunning the protocol, and also locking the malicious user's UTXO, have been proposed in some of the techniques. However, this cannot perfectly prevent DoS attacks. Among CoinJoin-based techniques, PayJoin is DoS resistance as the recipient is able to broadcast the original transaction if the sender refuses to sign the PayJoin transaction. Centralized mixers and atomic swap techniques are DoS resistant as none of the participants can abort the protocol and affect others.

**Sybil-resistance.** In most of the techniques, Sybil attacks are prevented by receiving the fee upfront. The CoinJoin techniques that have no coverage in terms of Sybil-resistance (CoinJoin, CoinShuffle, CoinShuffle++, ValueShuffle) are those that do not propose preventing such attacks in their protocols.

#### 4.2.3. Efficiency

**No interaction between input users.** All the centralized mixers and most of the atomic swap techniques (except FairExchange and Xim) do not require interaction between input users. In most of the CoinJoin-based techniques (except Snicker which is a non-interactive creation of CoinJoin), input registration, creating the transaction, and signing require the availability of the users during the protocol. Even if the user is not malicious, the connection lost tends to the protocol

failure. This can effectively delay creating CoinJoin transactions while most techniques can be performed without interaction with the input users. Threshold signature techniques also require the interaction between input users to sign the transaction.

**No interaction with the recipient.** Obscuro, PayJoin, and PaySwap require interaction with the recipients, which means the recipient should be online to complete the protocol. Although CoinSwap, BSC, and TumbleBit require interaction with the recipient in their original protocol, the sender can play the recipient role with different identities in these protocols to omit interaction with the recipient. In this scenario, the sender receives the coins to her own new address and needs one more transaction to send the mixed coin to the desired destination address.

**Direct send to the recipient.** This criterion is intended to show that in some of the proposed techniques the user needs to first send the coins to her own address and next to the desired destination address. This problem exists in CoinJoin-based and threshold signatures techniques, where the participant should provide a new output if the protocol goes to the blame phase. Valueshuffle uses Stealth addresses to overcome this problem, however, the application of stealth addresses in other CoinJoin and threshold signature techniques can solve this problem in those techniques.

**Bitcoin Compatible.** Most of the techniques are compatible with the current implementation of the Bitcoin blockchain. However, ValueShuffle requires CT implementation via soft-fork. BSC also requires blind signatures to be implemented in the Bitcoin blockchain via soft-fork. Obscuro also requires some changes in the Bitcoin Core implementation.

**Number of transactions and Minimum required block.** The last two columns indicate the number of transactions and the minimum number of blocks to run one round of the protocol, which are great insights into delays and transaction fees that should be paid by the participants. It is really important to consider the cost that would be shouldered by the participants to do the mixing, and that, apart from the mixing fee, additional transaction fees in the mixing techniques would be the main barrier for the adoption of those techniques in practice. Even in CoinJoin-based techniques, the participants are required to pay at least one additional transaction fee for mixing the coins and then transfer the coins to the destination address. Considering the point that one round of CoinJoin is not sufficient to provide anonymity for the users, they need to perform multiple rounds of mixing to achieve their desired anonymity set, which consequently increases the number of transactions and blocks to be confirmed. Atomic swap techniques also require four transactions in at least two blocks, which in turn lead to additional costs and delays.

### 4.3. Implementation in practice

Most of the aforementioned techniques have not been implemented in practice, or there is a significant delay between the protocol and its implementation in practice. Table 2 lists

Centralized mixers Mixing websites adopted from [44]	CoinJoin based			Atomic swap
	CoinJoin	Coinshuffle	PayJoin	TumbleBit
ChipMixer.com BitMix.Biz Bitlock43blmhm.com Mixer.money MixTum.io Blender.io FoxMixer.com MixerTumbler.com CryptoMixer.io MyCryptoMixer.com tumbler.to	Joinmarket Wasabiwallet Samouraiwallet (Whirlpool) Darkwallet (until 23.01.2015) [17] Sharedcoin (until 02.09.2016) [67]	Shufflepuff NXT	Samouraiwallet (Stowaway) BTCpay Wasabiwallet Joinmarket Bluewallet	NTumbleBit Breeze

Table 2: Bitcoin mixing techniques adoption in practice

the implementation of the techniques in practice. As can be seen, most of the implementations are centralized mixing websites.

Dumplings [53] indicates an increment of CoinJoin transactions in the past two years (since 2018). It should be mentioned that the high number of CoinJoin transactions can occur as a result of multiple mixing rounds to achieve a better anonymity set. Joinmarket, Wasabi, and Samourai are the implementation of CoinJoin wallets. Joinmarket uses a taker-maker model where the taker announces her willingness to perform a CoinJoin transaction and makers participate with her in the CoinJoin transaction by receiving fees. In this approach, privacy is for the taker, who creates the CoinJoin transaction [33]. Wasabi uses Chaumian CoinJoin, where the participants register their inputs and blindly sign the outputs to the coordinator to create a CoinJoin transaction. Samourai proposes Whirlpool, which has specified pools where the users can join to mix their coin with other participants and create CoinJoin transactions. In Samourai, the wallet knows the xpub of the users, from which their Bitcoin addresses are derived, thus, and there is no privacy against the wallet [53]. SharedCoin as a CoinJoin service by Blockchain.info, in which Blockchain.info was able to find the inputs and outputs relationships, and also Darkwallet that created CoinJoin transactions and used Stealth addresses, were discontinued, probably due to legal reasons. In 2020, BTCpay implemented PayJoin to allow merchants to create stores that accept PayJoin transactions. At the time of writing, Wasabi, Samourai, Joinmarket, and Bluewallet support PayJoin transactions. However, creating PayJoin transactions between the users has been implemented before in Joinmarket and Samourai wallet (Stowaway). Shufflepuff [69] is an alpha version in Github and its last updates back to 2016 and Nxt [41] have been activated since block 621,000 (09.03.2020) on mainnet. However, at the time of writing, the CoinShuffle feature has been removed from the wallet feature list.

According to [51], Fairexchange transactions cannot be found in the blockchain. At the time of writing, there is no commercial implementation of atomic swap techniques. Recently [6], developing a new CoinSwap design/PaySwap wallet has been proposed. There are some alpha implementations of TumbleBit in Github (NTumbleBit and Breeze), which are not commercial at the time of writing.

#### 4.4. Future research

Future research could be done in three areas: usability, law enforcement, and practicality of the techniques.

**Usability.** Usable system can attract more users, and therefore it can achieve much anonymity [21]. In the usability area the following questions should be considered: (i) To what extent are the users aware of add-on and built-in privacy techniques and their implementations in practice? (ii) Do they trust third-party privacy-preserving services? (iii) Which would be preferred by users: using add-on techniques implemented by wallets and services or using built-in techniques such as privacy coins to achieve stronger anonymity? Do users accept the extra fees and delays to achieve stronger privacy in the blockchain? (iv) Is there any significant difference in paying for privacy between privacy-aware and privacy-unaware users? (v) Which privacy features are the users interested in (prevention of address reuse, hiding the amount, hiding the source, hiding source and destination, direct send to the recipient, no interaction with other users)? (vi) Do the current implementations of the techniques allow the users to realize what needs to be done and do they understand how to do it?

**Law enforcement.** Using privacy techniques in the dark web does create a footprint, which means that privacy-preserving techniques could be used for illicit activities. Privacy techniques may be employed by users who are aware of the catastrophic consequences as a result of de-anonymization in the blockchain. To the best of our knowledge, there is no research on the state of the art regarding categorizing the destination addresses of the CoinJoin transactions as one of the most implemented techniques (Table 2). Although the recipient of those CoinJoin transactions cannot be easily discovered, categorizing these addresses using ground truth can shed light on the usage of the CoinJoin techniques in the Bitcoin Blockchain. For this reason, the following research question would be a very useful starting point for further research. Is it possible to categorize the destination of the CoinJoin transactions to find the percentage of its application in illicit activities? There is always a trade-off between privacy and law enforcement rules in the cryptocurrency environment. Achieving privacy for most users while preventing the technology from being a good place for criminal activities and the dark market is still an unresolved problem in the field. [43] proposes a model to enable law enforce-

ment agencies to collaborate with involved parties in CoinJoin transactions to find criminals, which would be a good way of taking both privacy and law enforcement perspectives into account.

**Practicality.** Accepting the PayJoin technique into the market could effectively provide privacy for users as it has the ability to break the so-called common input ownership heuristic. However, these transactions should be implemented in a way that cannot tag the transactions as PayJoin. Unnecessary input heuristic and wallet fingerprinting should be considered in the implementation of the protocol, which needs further research to investigate their effectiveness in tagging the PayJoin transactions. Further research could also be conducted into non-equal amount CoinJoin transactions. As of now, the distinguishability of equal size CoinJoin transactions has the potential to create a problem for the users, since some of the exchanges refuse to accept the output of CoinJoin transactions. Knapsack, proposed in [46], and Wabisabi [23] would be a good basis for future work to improve the indistinguishability of these types of transactions.

## 5. Concluding remarks

The aim of current study is to review and evaluate mixing techniques in Bitcoin. The study has compared a multitude of selected proposals according to a set of criteria. These proposals offer different guarantees in terms of privacy, security, and efficiency. Strong privacy affects efficiency, scalability, and usability.

Among atomic swap techniques, New CoinSwap and its predecessors can meet most of the criteria while they require more transactions and consequently more time and fees. CoinJoin-based techniques have been commonly adopted in practice. Transaction distinguishability as a result of equal-sized outputs, and DoS attacks pose serious problems for these techniques. The recently proposed PayJoin method, which is based on CoinJoin can indeed resolve distinguishability and improve anonymity. One of the main advantages of CoinJoin-based techniques is the reduced number of transactions to run the protocol, which makes them affordable. Although multiple rounds of CoinJoin can provide better anonymity, it adds fees and delays. However, most CoinJoin techniques fail to provide a large anonymity set and plausible deniability. Confidential transactions to hide the UTXO amount, proposed in ValueShuffle, can efficiently solve this problem and provide indistinguishability for CoinJoin-based techniques. Mixing techniques often require a minimum number of transactions in order to hide the connection between senders and recipients. Although an increased number of transactions can improve anonymity, this also comes with a cost, i.e., transaction fees. Even though the mixing fee can be negligible, additional transaction fees may limit the technique's adoption by users. According to our results, except for centralized mixers and threshold-based techniques, theft resistance criterion is met by most of the techniques. We also investigated the adoption of the techniques in practice to indicate which protocols are most adopted and which features

were added for the implementation. Finally, we discussed future directions and challenges faced by mixing techniques. Although the initial intention of guaranteeing strong privacy was to prevent user details from exposure to malicious adversaries and criminals, such privacy-preserving techniques can be employed to conduct illicit activities. Therefore, new methods able to distinguish transactions used for illicit activities from regular mixing transactions (e.g., financial privacy) are required.

## Acknowledgments.

This research is based upon work partially supported by (1) SBA Research (SBA-K1); SBA Research is a COMET Center within the COMET – Competence Centers for Excellent Technologies Programme and funded by BMK, BMDW, and the federal state of Vienna. The COMET Programme is managed by FFG. (2) the FFG ICT of the Future project 874019 dIdentity & dApps. (3) the FFG Basisprogramm Kleinprojekt 39019756 Decentralised Marketplace for Digital Identity. The authors acknowledge TU Wien Bibliothek for financial support for editing/proofreading.

## References

- [1] Alrawi, O., Lever, C., Antonakakis, M., Monrose, F., 2019. Sok: Security evaluation of home-based iot deployments, in: 2019 IEEE Symposium on Security and Privacy (SP), IEEE. pp. 1362–1380.
- [2] Antonopoulos, A.M., 2017. Mastering Bitcoin: Programming the open blockchain. " O'Reilly Media, Inc."
- [3] Bao, Z., Shi, W., Kumari, S., Kong, Z.y., Chen, C.M., 2019. Lockmix: a secure and privacy-preserving mix service for bitcoin anonymity. International Journal of Information Security , 1–11.
- [4] Barber, S., Boyen, X., Shi, E., Uzun, E., 2012. Bitter to better—how to make bitcoin a better currency, in: International conference on financial cryptography and data security, Springer. pp. 399–414.
- [5] Belcher, C., (2018) Last accessed 24 March 2021. Joinmarket. <https://github.com/JoinMarket-Org/joinmarket-clientserver> .
- [6] Belcher, C., Last accessed 16 July 2020. Design for a coinswap implementation for massively improving bitcoin privacy and fungibility. [https:// gist.github. com/ chris-belcher/ 9144bd57a91c194e332fb5ca371d0964](https://gist.github.com/chris-belcher/9144bd57a91c194e332fb5ca371d0964) .
- [7] Bissias, G., Ozisik, A.P., Levine, B.N., Liberatore, M., 2014. Sybil-resistant mixing for bitcoin, in: Proceedings of the 13th Workshop on Privacy in the Electronic Society, pp. 149–158.
- [8] bitcoin.org, Last accessed 21 July 2020a. Multisig. [https:// developer. bitcoin. org/ devguide/ transactions. html#multisig](https://developer.bitcoin.org/devguide/transactions.html#multisig) .
- [9] bitcoin.org, Last accessed 21 July 2020b. Pay-to-public-key-hash. [https:// developer. bitcoin. org/ devguide/ transactions. html#p2pkh-script-validation](https://developer.bitcoin.org/devguide/transactions.html#p2pkh-script-validation) .
- [10] bitcoin.org, Last accessed 21 July 2020c. Pay-to-script-hash. [https:// developer. bitcoin. org/ devguide/ transactions. html#p2sh-scripts](https://developer.bitcoin.org/devguide/transactions.html#p2sh-scripts) .
- [11] Blake, I.F., Seroussi, G., Smart, N.P., 2005. Advances in elliptic curve cryptography. volume 317. Cambridge University Press.
- [12] Blockstream, Last accessed 20 September 2020. Improving privacy using pay-to-endpoint (p2ep). <https://blockstream.com/2018/08/08/en-improving-privacy-using-pay-to-endpoint/> .
- [13] Bojja Venkatakrishnan, S., Fanti, G., Viswanath, P., 2017. Dandelion: Redesigning the bitcoin network for anonymity. Proceedings of the ACM on Measurement and Analysis of Computing Systems 1, 1–34.
- [14] Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W., 2015. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies, in: 2015 IEEE symposium on security and privacy, IEEE. pp. 104–121.
- [15] Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J.A., Felten,

- E.W., 2014. Mixcoin: Anonymity for bitcoin with accountable mixes, in: International Conference on Financial Cryptography and Data Security, Springer. pp. 486–504.
- [16] BtcDrak, M.F., Lombrozo, E., 2015. Bip 112: Checksequenceverify. URL: <https://github.com/bitcoin/bips/blob/master/bip-0112.mediawiki>.
- [17] caedsvvv, 2015. Darkwallet. <https://github.com/darkwallet/darkwallet/releases/tag/0.8.0>.
- [18] Chaum, D., 1983. Blind signatures for untraceable payments, in: Advances in cryptology, Springer. pp. 199–203.
- [19] Chaum, D.L., 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 24, 84–90.
- [20] Corrigan-Gibbs, H., Ford, B., 2010. Dissent: accountable anonymous group messaging, in: Proceedings of the 17th ACM conference on Computer and communications security, pp. 340–350.
- [21] Dingleline, R., Mathewson, N., Syverson, P., 2005. Challenges in deploying low-latency anonymity. *NRL CHACS Report*, 5540–625.
- [22] Ermilov, D., Panov, M., Yanovich, Y., 2017. Automatic bitcoin address clustering, in: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE. pp. 461–466.
- [23] Ficsór, , Kogman, Y., Seres, I.A., Last accessed 3 Feb 2021. Wabisabi. <https://github.com/zkSNACKs/WabiSabi/releases/download/build-70d01424bbce06389d2f0536ba155776eb1d8344/WabiSabi.pdf>.
- [24] Filtz, E., Polleres, A., Karl, R., Haslhofer, B., 2017. Evolution of the bitcoin address graph, in: Data science–Analytics and applications. Springer, pp. 77–82.
- [25] Fleder, M., Kester, M.S., Pillai, S., 2015. Bitcoin transaction graph analysis. *arXiv preprint arXiv:1502.01657*.
- [26] Franco, P., 2014. Understanding Bitcoin: Cryptography, engineering and economics. John Wiley & Sons.
- [27] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T., 1999. Secure distributed key generation for discrete-log based cryptosystems, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer. pp. 295–310.
- [28] Gibson, A., 2017 (Last accessed 23 August 2020). New coinswap. <https://joinmarket.me/blog/blog/coinswaps/>.
- [29] Gibson, A., 2017 (Last accessed 31 August 2020). Snicker - simple non-interactive coinjoin with keys for encryption reused. <https://joinmarket.me/blog/blog/snicker/>.
- [30] Gibson, A., 2018 (Last accessed 23 August 2020)a. Basic payjoin / p2ep protocol for joinmarket wallets. <https://gist.github.com/AdamISZ/4551b947789d3216bacfb7af25e029e>.
- [31] Gibson, A., 2018 (Last accessed 23 August 2020)b. Payjoin. <https://joinmarket.me/blog/blog/payjoin/>.
- [32] Gibson, A., 2018 (Last accessed 31 August 2020). Coinjoinxt - a more flexible, extended approach to coinjoin. <https://joinmarket.me/blog/blog/coinjoinxt/>.
- [33] Gibson, A., Last accessed 3 Feb 2021. From mac to wabisabi. <https://joinmarket.me/blog/blog/from-mac-to-wabisabi/>.
- [34] Halpin, H., Piekarska, M., 2017. Introduction to security and privacy on the blockchain, in: 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE. pp. 1–3.
- [35] Harrigan, M., Fretter, C., 2016. The unreasonable effectiveness of address clustering, in: 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ ATC/ ScalCom/ CBDCom/ IoP/ SmartWorld), IEEE. pp. 368–373.
- [36] Havar, R., Last accessed 20 September 2020. Bustapay bip: a practical sender/receiver coinjoin protocol. <https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-August/016340.html>.
- [37] Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A., Goldberg, S., 2017. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub, in: Network and Distributed System Security Symposium.
- [38] Heilman, E., Baldimtsi, F., Goldberg, S., 2016. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions, in: International conference on financial cryptography and data security, Springer. pp. 43–60.
- [39] Ibrahim, M.H., 2017. Securecoin: A robust secure and efficient protocol for anonymous bitcoin ecosystem. *IJ Network Security* 19, 295–312.
- [40] Ibrahim, M.H., Ali, I., Ibrahim, I., El-Sawi, A., 2003. A robust threshold elliptic curve digital signature providing a new verifiable secret sharing scheme, in: 2003 46th Midwest Symposium on Circuits and Systems, IEEE. pp. 276–280.
- [41] Jelurida, Last accessed 11 August 2020. Nxt. <https://nxtdocs.jelurida.com/CoinShuffling>.
- [42] Jourdan, M., Blandin, S., Wynter, L., Deshpande, P., 2018. Characterizing entities in the bitcoin blockchain, in: 2018 IEEE International Conference on Data Mining Workshops (ICDMW), IEEE. pp. 55–62.
- [43] Keller, P., Florian, M., Böhme, R., 2020. Collaborative deanonymization. *arXiv preprint arXiv:2005.03535*.
- [44] LeGaulois, Last accessed 11 August 2020. 2020 list bitcoin mixers bitcoin tumblers websites. <https://bitcointalk.org/index.php?topic=2827109.0>.
- [45] Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., Maffei, M., 2019. Anonymous multi-hop locks for blockchain scalability and interoperability., in: NDSS.
- [46] Maurer, F.K., Neudecker, T., Florian, M., 2017. Anonymous coinjoin transactions with arbitrary values, in: 2017 IEEE TrustCom/BigDataSE/ICSS, IEEE. pp. 522–529.
- [47] Maxwell, G., 2013a. Coinjoin: Bitcoin privacy for the real world, 2013. URL: <https://bitcointalk.org/index.php>.
- [48] Maxwell, G., 2013b. Coinswap: transaction graph disjoint trustless trading (2013). URL: <https://bitcointalk.org/index.php>.
- [49] Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S., 2013. A fistful of bitcoins: characterizing payments among men with no names, in: Proceedings of the 2013 conference on Internet measurement conference, pp. 127–140.
- [50] Moser, M., 2013. Anonymity of bitcoin transactions.
- [51] Möser, M., Böhme, R., 2017. Anonymous alone? measuring bitcoin’s second-generation anonymization techniques, in: 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE. pp. 32–41.
- [52] Nakamoto, S., 2008. Bitcoin: A peer-to-peer electronic cash system.
- [53] nopara, ., Last accessed 3 Feb 2021. Dumplings. <https://github.com/nopara73/Dumplings>.
- [54] nopara73, Last accessed 3 Feb 2021. Tumblebit vs coinjoin. <https://nopara73.medium.com/tumblebit-vs-coinjoin-15e5a7d58e3>.
- [55] Pfizmann, A., Hansen, M., 2010. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management.
- [56] Reid, F., Harrigan, M., 2013. An analysis of anonymity in the bitcoin system, in: Security and privacy in social networks. Springer, pp. 197–223.
- [57] Rivest, R.L., Shamir, A., Adleman, L., 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21, 120–126.
- [58] Ruffing, T., Moreno-Sanchez, P., 2017. Valueshuffle: Mixing confidential transactions for comprehensive transaction privacy in bitcoin, in: International Conference on Financial Cryptography and Data Security, Springer. pp. 133–154.
- [59] Ruffing, T., Moreno-Sanchez, P., Kate, A., 2014. Coinshuffle: Practical decentralized coin mixing for bitcoin, in: European Symposium on Research in Computer Security, Springer. pp. 345–364.
- [60] Ruffing, T., Moreno-Sanchez, P., Kate, A., 2017. P2p mixing and unlinkable bitcoin transactions., in: NDSS, pp. 1–15.
- [61] Schnorr, C.P., 1989. Efficient identification and signatures for smart cards, in: Conference on the Theory and Application of Cryptology, Springer. pp. 239–252.
- [62] Tascia, P., Tessone, C.J., 2017. Taxonomy of blockchain technologies. principles of identification and classification. *arXiv preprint arXiv:1708.04872*.
- [63] Todd, P., 2014. Bip 65: Op checklocktimeverify. Github (accessed

- 18 October 2015) <https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki> .
- [64] Tran, M., Luu, L., Kang, M.S., Bentov, I., Saxena, P., 2018. Obscuro: A bitcoin mixer using trusted execution environments, in: Proceedings of the 34th Annual Computer Security Applications Conference, pp. 692–701.
  - [65] Valenta, L., Rowan, B., 2015. Blindcoin: Blinded, accountable mixes for bitcoin, in: International Conference on Financial Cryptography and Data Security, Springer. pp. 112–126.
  - [66] Van Saberhagen, N., 2013. Cryptonote v 2.0.
  - [67] Ver, R., 2016. The discontinuation of shared send at blockchain.info was due to threats of violence made by strangers in government. [https://www.reddit.com/r/btc/comments/50t0jf/roger\\_ver\\_the\\_discontinuation\\_of\\_shared\\_send\\_at/](https://www.reddit.com/r/btc/comments/50t0jf/roger_ver_the_discontinuation_of_shared_send_at/) .
  - [68] Wang, Q., Li, X., Yu, Y., 2017. Anonymity for bitcoin from secure escrow address. IEEE Access 6, 12336–12341.
  - [69] Weigl, D., 2016 (Last accessed 11 August 2020). Mycelium shufflepuff. <https://github.com/DanielWeigl/Shufflepuff> .
  - [70] Wiki, Last accessed 16 July 2020a. Multisignature. <https://en.bitcoin.it/wiki/Multisignature> .
  - [71] Wiki, Last accessed 16 July 2020b. Timelock. <https://en.bitcoin.it/wiki/Timelock> .
  - [72] Wiki, Last accessed 21 July 2020a. Hashlock. <https://en.bitcoin.it/wiki/Hashlock> .
  - [73] Wiki, Last accessed 21 July 2020b. Script. <https://en.bitcoin.it/wiki/Script> .
  - [74] Wiki, Last accessed 22 July 2020. Htlc. [https://en.bitcoin.it/wiki/Hash Time Locked Contracts](https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts) .
  - [75] Wikipedia, Last accessed 21 July 2020a. Double-spending. <https://en.wikipedia.org/wiki/Double-spending> .
  - [76] Wikipedia, Last accessed 21 July 2020b. Forth (programming language). URL: [https://en.wikipedia.org/wiki/Forth \(programming language\)](https://en.wikipedia.org/wiki/Forth_(programming_language)) .
  - [77] Wikipedia, Last accessed 21 July 2020c. Unspent transaction output. [https://en.wikipedia.org/wiki/Unspent transaction output](https://en.wikipedia.org/wiki/Unspent_transaction_output) .
  - [78] York, D., 2010. Seven deadliest unified communications attacks. Syn-gress.
  - [79] Ziegeldorf, J.H., Grossmann, F., Henze, M., Inden, N., Wehrle, K., 2015. Coinparty: Secure multi-party mixing of bitcoins, in: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, pp. 75–86.