Everything You Ever Wanted to Know About Bitcoin Mixers

(But Were Afraid to Ask)

by

Jaswant Pakki

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2020 by the
Graduate Supervisory Committee:

Adam Doupé, Chair
Yan Shoshitaishvili
Ruoyu Wang

ARIZONA STATE UNIVERSITY

May 2020

ABSTRACT

The lack of fungibility in Bitcoin has forced its userbase to seek out tools that can heighten their anonymity. Third-party Bitcoin mixers utilize obfuscation techniques to protect participants from blockchain analysis. In recent years, various centralized and decentralized Bitcoin mixing implementations have been proposed in academic literature. Although these methods depict a threat-free environment for users to preserve their anonymity, public Bitcoin mixers continue to be associated with theft and poor implementation.

This research explores the public Bitcoin mixer ecosystem to identify if today's mixing services have adopted academically proposed solutions. This is done through real-world interactions with publicly available mixers to analyze both implementation and resistance to common threats in the mixing landscape. First, proposed decentralized and centralized mixing protocols found in literature are outlined. Then, data is presented from 19 publicly announced mixing services available on the deep web and clearnet. The services are categorized based on popularity with the Bitcoin community and experiments are conducted on five public mixing services: ChipMixer, MixTum, Bitcoin Mixer, CryptoMixer, and Sudoku Wallet.

The results of the experiments highlight a clear gap between public and proposed Bitcoin mixers in both implementation and security. Today's mixing services focus on presenting users with a false sense of control to gain their trust rather then employing secure mixing techniques. As a result, the five selected services lack implementation of academically proposed techniques and display poor resistance to common mixer-related threats.

ACKNOWLEDGMENTS

I would like to express my appreciation to Adam, Yan, Fish, and Tiffany for their valuable support, guidance, and enthusiasm for my work. I would also like to thank my peers at the SEFCOM lab whose amazing research serves as an inspiration to me everyday. Lastly, I would like to thank my family and friends for their constant encouragement and support throughout my time at ASU.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

In May of 2019, EU authorities seized Bestmixer, a mixing service that advertised their ability to eradicate any criminal history associated with a user's Bitcoin. This marked the first public seizure of a Bitcoin mixer. After a year-long investigation, authorities were able to assert that the majority of the $200 million that traveled through the service had "a criminal origin or destination" [2].

Although their history paints a dark image, Bitcoin mixing services are not illegal by nature. Their guarantee to obfuscate a trail of funds appeals to a mass of benign users who seek anonymity. Various centralized mixing services are available to the public today and have an active user base within the Bitcoin community. The techniques implemented by these services have a direct impact on user privacy and security. For example, Bestmixer claimed to eradicate all "order history completely and automatically in 24 hours" [3]. This was proven to be untrue when authorities seized IP-addresses, transactions logs, wallet addresses, and chat messages stored on multiple Bestmixer servers. As a result, user location and permutations between inputs and outputs of the service were revealed.

To address the threats mixers and their users face, academic literature has proposed alternative mixing protocols. For example, Bonneau *et al.* [4] proposed a protocol that adds accountability to interactions with centralized mixers via PGP signed warranty agreements. Although these implementations depict ideal improvements to mixing technology, the majority of publicly available services are still associated with distrust and scam accusations.

Many studies of Bitcoin mixers focus on the proposal of new mixing techniques.

In this work, we explore the current public Bitcoin mixing service ecosystem and identify if these academically proposed solutions have been adopted. We rely on real-world mixer interactions with five public mixers to identify behavior indicative of their implementation and their resistance to common threats.

In Chapter 2, we provide background information related to Bitcoin and blockchain. In Chapter 3, we outline the various obfuscation features and present a threat model of mixing services. In Chapter 4, we discuss contributions from related work. Chapter 5 outlines decentralized and centralized mixing protocols proposed in literature. In Chapter 6, we categorize 19 public mixing services and choose six to conduct further analysis. In Chapter 7, we outline our experiments and perform both an implementation and security analysis on each mixer. Finally, in Chapter 8, we discuss our results and future work.

Chapter 2

BACKGROUND

2.1 Bitcoin and Blockchain

Bitcoin (BTC) is a decentralized digital currency that relies on a peer-to-peer (P2P) distributed network to store and check the validity of transaction data [5]. This data is stored on a public ledger where users are identified by pseudonymous addresses. We will discuss the security implications of these addresses further in Section 2.4. The blockchain is the underlying architecture of the public ledger. As seen in Figure 2.1, each block holds the hash of its predecessor and a merkle tree of transactions. Any change in transaction information would lead to a different merkle root hash and hash of the block itself.

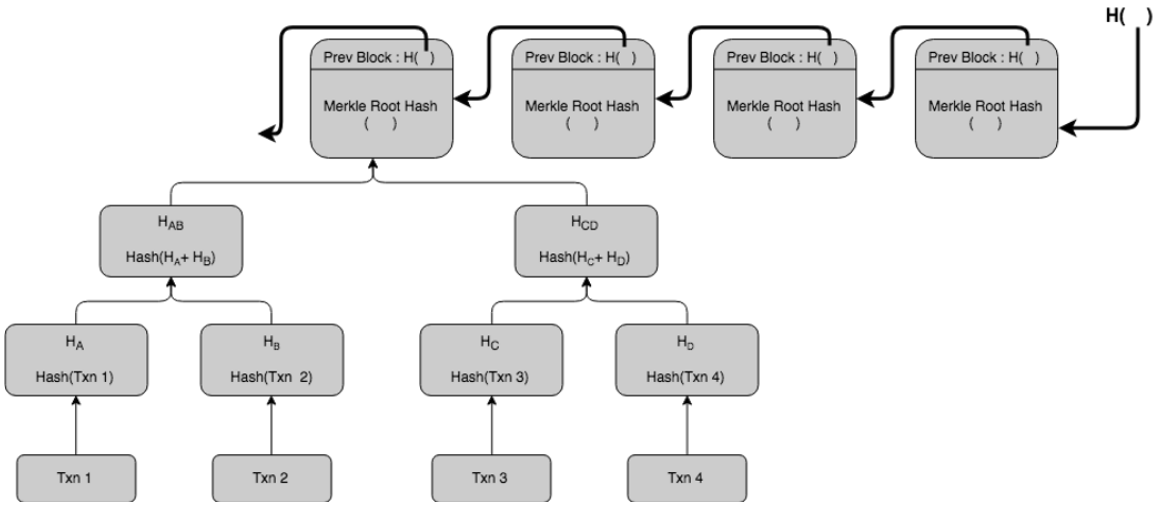

Figure 2.1: Blockchain and Merkle Tree Architecture [1]

Another integral part of Bitcoin's implementation is its use of the Elliptic Curve Digital Signature Algorithm (ECDSA). The pseudonymous addresses users create are

each derived from corresponding public/private key pairs stored in user's wallets. To prevent forgery of transactions, Bitcoin users sign created transactions with their private key. When transaction information is sent out to nodes in the P2P network, they use the sender's public key to validate that the transaction was signed by the correct corresponding private key. Each peer in the P2P network has their own copy of the blockchain which they use to share and verify new transactions in this manner. Ultimately, the blockchain limits excessive control by a single peer.

2.2 Anonymity in Bitcoin

As mentioned in 2.1, Bitcoin uses pseudonymous addressing to identify its users. While these users are capable of creating as many addresses as they'd like, they aren't required to do so. In turn, researchers have been able to use clustering, transaction analysis, taint analysis, and behavior analysis to track patterns and build relationships between public keys [6, 7, 8, 9, 10]. The official Bitcoin website highlights potential threats to user anonymity and clearly states that the currency is not anonymous [11]. Various solutions have been proposed to address this lack of anonymity, including Bitcoin mixing services.

2.3 Bitcoin Wallets

Bitcoin users store their funds in wallets. The four major forms of wallets are Web, Mobile, Desktop, and Hardware. While the best type of wallet is up for debate, their main function is to keep track of public/private key pairs and aggregate corresponding funds to give users a clear view of their total owned Bitcoin. Wallets also provide the ability to send and receive transactions. In this work, we make use of the desktop wallet, Electrum, to store, send, and receive funds.

2.4 Bitcoin Transactions

Bitcoin makes use of a transaction-based public ledger. Inputs and outputs of transactions are referred to as Unspent Transaction Outputs (UTXOs). Transactions consume UTXOs as inputs and create new ones as outputs. UTXOs can only be used in full or not at all. It is quite unlikely that a UTXO will match the exact amount needed to be spent. Thus, the majority of Bitcoin transactions have two outputs. While the recipient receives one output, the left over amount is sent back to the sender at a new address. This is referred to as the change output.

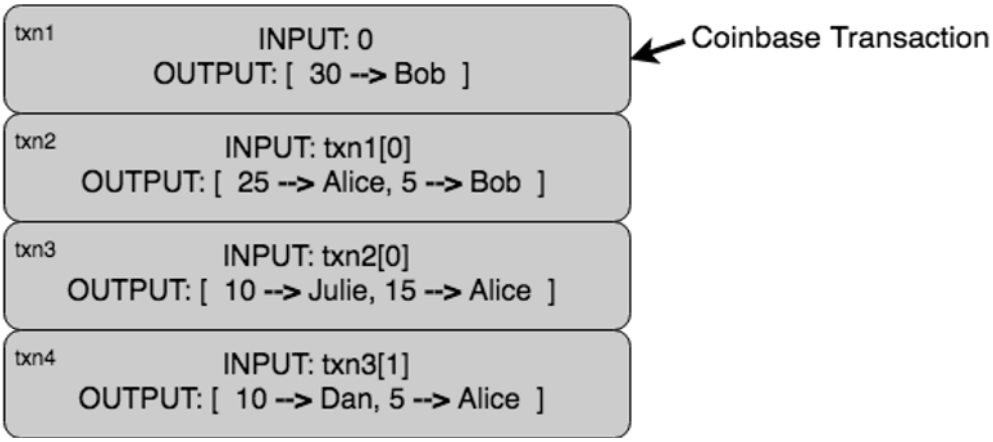

Figure 2.2: A Simple Transaction Ledger [1]

Figure 2.2 is an example of a simple transaction ledger. In this example, Bob is a miner and correctly solves a hash puzzle to create a new block in the blockchain. In turn, Bob receives a reward of 30 BTC. This is referred to as the coinbase transaction. Each transaction consists of an input and an array of outputs indexed starting at 0. Each input (excluding the coinbase transaction) refers to indices in a previous transactions output array. In txn2 of Figure 2.2, Bob intends to send 25 BTC to Alice. To do so, Bob must use index 0 of txn1 as the input. From here, 25 BTC is sent to Alice and 5 BTC is returned to Bob as the change transaction.

While Figure 2.2 only includes inputs and outputs of each transaction, publicly available blockchain explorers provide additional metadata. Transaction metadata includes: public keys, amounts of input and output UTXOs, size of the transaction in bytes, and hash of the transaction as a unique identifier. Transaction inputs also include signatures using the sender's private key; this allows anyone to use the sender's public key to verify the validity of the signed transaction.

2.5 Bitcoin Address Formats

There are currently three different Bitcoin address formats: P2PKH (legacy), P2SH (compatibility), and Bech32 (SegWit). The legacy format is Bitcoin's original format and begins with a *1*. Transaction size and fees are typically higher with legacy addresses. Compatibility addresses are similar to legacy but start with a *3* and provide more advanced functionality like multisig transactions. The SegWit address format was added to decrease the size of blocks, increase transaction speed, and decrease transaction fees. These addresses begin with *bc1*. Many services are not fully compatible with all three formats, however funds can be sent between all three.

2.6 Bitcoin Core

Bitcoin Core is an open-source Bitcoin client that offers full node capability. Users are able to validate transactions and blocks to help support the blockchain. Bitcoin Core also provides wallet functionality to create public/private key pairs and send/receive transactions without the use of third-party servers. The software comes bundled with Bitcoin Core daemon (bitcoind), which can be used for Bitcoin remote procedure calls (RPC).

Chapter 3

BITCOIN MIXERS

Bitcoin mixers, or tumblers, are services offering the ability to obfuscate user's funds. Figure 3.1 depicts the general functionality of a mixer with three users and the mixing operator. Each user sends their Bitcoin into the service and is returned another user's input to a different address. This output has a completely different transaction history associated with it. The mixer operator runs the service and is aware of all permutations between inputs and outputs. Although this high-level view may seem easily traceable, mixers use techniques that make it difficult to trace transactions and identify mixing service use on the blockchain.

3.1 Obfuscation Techniques

Since their inception, mixing services have adapted to threats stemming from transactional analysis. In this section, we outline potential characteristics used to trace transactions and the techniques implemented to eliminate traceability. The inclusion of obfuscation techniques varies between mixing services, as seen in Chapter 6.

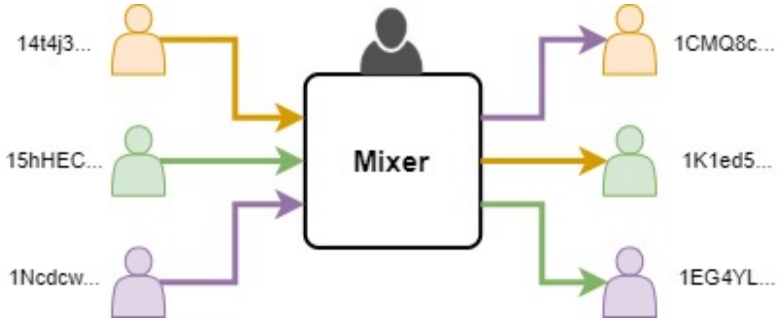

Figure 3.1: High-Level Mixer

Characteristic	Obfuscation Technique
Mixer input address	Newly generated mixer input address for each user of the service
User address	Multiple input and output addresses
Amount of input and output	Mixing fees and distribution over multiple output addresses
Time and date of input and output transactions	Mixing delays

Table 3.1: Traceable Characteristics and Obfuscation Techniques in Mixing Services

Table 3.1 outlines traceable characteristics of a transaction and its corresponding obfuscation technique. The mixer input address is presented to the user to send their funds to the service. If kept consistent for all users, it would be simple for anyone to identify mixing participants and the amount of Bitcoin the mixer has in its pool. To avoid this, mixers generate new input addresses for each user. Additionally, the user's address could be traceable if kept consistent throughout the mixing interaction. In turn, mixers allow their participants to specify multiple output addresses.

Patterns in amounts and timestamps of transactions could also indicate mixer use. Since network fees are public information, mixers add mixing fees to each transaction. In addition, mixing delays are used to make blockchain analysis more difficult. According to [12], there are more than 300,000 Bitcoin transactions every 24 hours. Thus, it is in mixing participants' best interest that delays are maximized. While the majority of services randomize fees and delays, some allow users to customize these features.

3.2 Threats

Trust is incredibly important for the success of a Bitcoin mixer. As third-party services, they must convince users that funds will be properly mixed and returned. Thus, mixers often offer features for users to check the status of their mix or proudly promote their forum posts. Still, Bitcoin mixers are continuously accused of scams and poor implementation.

While mixers may pose threats to their participant's funds and anonymity, users and external attackers also contribute to the threat landscape. Some of the threats posed by users and external attackers, like tracing transactions, are mitigated with obfuscation features. Others, like coin theft, can be mitigated by the proposed mixer implementations discussed in Chapter 5. The majority of current mixing implementations involve a centralized third-party run by an all-powerful operator. The threats posed by a this mixer operator are much more difficult to detect. In this paper, we focus our security analysis on the following threats presented by Tran *et al.* in [13]:

Permutation Leak : An adversary is able to access mixing logs or a database pertaining to the permutation between input and output addresses.

Coin Theft : An adversary steals the inputted coins by providing users with an alternative address or by compromising the mixer's address. The mixer operator can also steal user funds.

Dropping of Participants : A malicious mixer operator can deny participation to selected benign users to reduce the anonymity set.

Small Mixing Set Size : The mixing set size during each round is directly indicative of the quality of the mix. A large mixing set ensures anonymity and protection against blockchain analysis.

Join-then-abort : An adversarial participant disrupts the mix by aborting the mixing protocol before its execution.

Chapter 4

RELATED WORK

Related work regarding Bitcoin mixing can be divided into two categories: proposed mixing solutions and public Bitcoin mixer analysis. We discuss eight proposed mixing solutions in Chapter 5. In this chapter, we outline work related to Bitcoin mixer analysis.

In 2013, Moser *et al.* [14] explored Bitcoin Fog, BitLaundry, and the Send Shared functionality of Blockchain.info to attempt tracing their outputs back to their input accounts in a series of experiments. They identified that two of the services, Bitcoin Fog and Blockchain.info, successfully obfuscated their funds. They were successfully able to trace their BitLaundry outputs back to their original inputs using Blockchain.info's transaction graph functionality which has since been depracated. In 2015, Novetta [15] conducted experiments with BitMixer, BitLaunder, Shared Coin, and Bitcoin Blender to identify provable links in mixing schemes, identify fingerprints of individual mixers, and identify if mixing can be detected on the blockchain. The study found fingerprinting patterns in the services based on recurring addresses, fees, and branching patterns. Balthasar and Hernandez-Castro [16] interacted with DarkLaunder, Bitlaunder, CoinMixer, Helix, and Alphabay and identified security and privacy limitations in the services. Their work highlights the need for secure and privacy-aware protocols to improve the Bitcoin mixing ecosystem.

Chapter 5

PROPOSED MIXING SOLUTIONS

As outlined Chapter 3, Bitcoin mixing services have adopted techniques to evade detection and effectively obfuscate user funds. However, threats to these services and their user base exist. In response, the Bitcoin community and academic literature have proposed alternative methods to improve trust and eliminate these threats. In this chapter, we discuss the general architecture of four decentralized and four centralized proposed mixing protocols.

5.1 Decentralized

Decentralized mixing protocols strive to avoid the use of a third-party. While drawing inspiration from the mixing network presented in [17], many of these mixing protocols propose shuffling of inputs and outputs to ensure unlinkability. In most of the following protocols, a decentralized method for users to find other participants (bootstrapping method) is assumed. In general, decentralized protocols suffer from limited scalability and long wait times to find mixing peers.

5.1.1 CoinJoin

CoinJoin [18] is a method for multiple transactions, from multiple senders, to be combined into one. Without any modification to the current Bitcoin protocol, this technique makes it difficult for outside entities to identify the corresponding recipient for each input. Since Bitcoin's inception, it has been generally assumed that all inputs to a transaction belong to the same user. However, CoinJoin transactions prove that this isn't always the case. Users may collaborate to identify a uniform output amount

and combine their transactions into one. In turn, senders face lower transaction fees and lessen the load of transactions on the Bitcoin network. Additionally, participants of CoinJoin transactions do not face the risk of theft. This is due to the requirement that each participant must sign the transaction before it is considered valid.

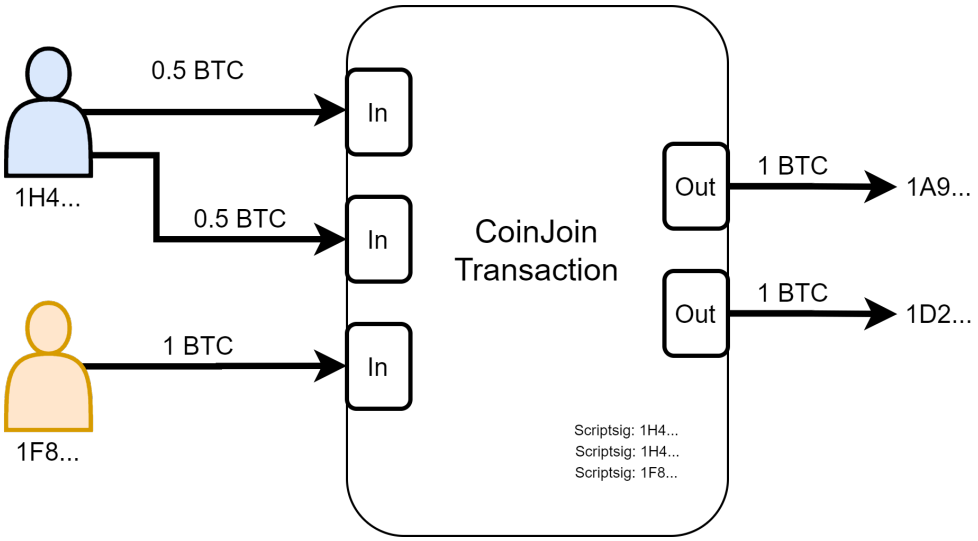

Figure 5.1: CoinJoin Example

Figure 5.1 depicts a simple example of a CoinJoin transaction. The transaction involves two users who must collaborate to identify a uniform output for the transaction. With a uniform output of 1 BTC, it is difficult to identify which output corresponds with 1H4 and which corresponds with 1F8. This becomes more difficult with a larger number of inputs and outputs. The challenges facing CoinJoin are scalability and avoiding centralization. There is currently no standard decentralized bootstrapping method users to collaborate and agree to execute a CoinJoin transaction at a large scale.

5.1.2 *CoinShuffle*

Ruffing *et al.* [19] presented CoinShuffle in 2014. The mixing protocol requires no third party, is compatible with the existing Bitcoin network, and uses CoinJoin to execute transactions. It is important to note that in Section 3.2 of [19] the authors state that the bootstrapping mechanism for users is a non-goal and the protocol assumes that users have a secure, decentralized method to express their interest in participation.

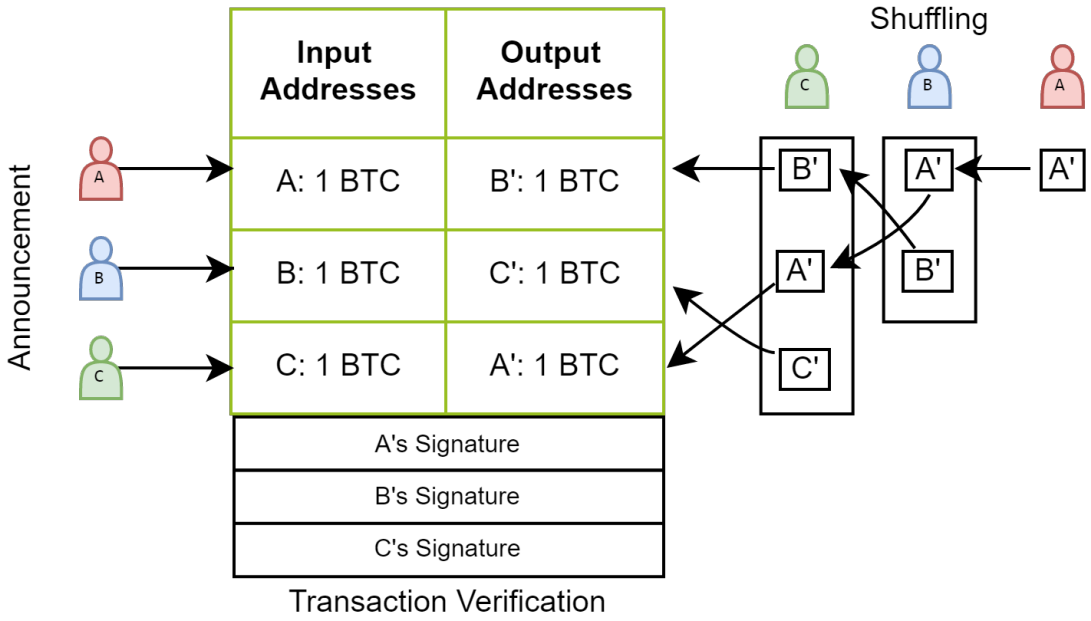

Figure 5.2: Three Phases of CoinShuffle

CoinShuffle is implemented in three phases: Announcement, Shuffling, and Transaction Verification. Figure 5.2 illustrates a high-level view of the CoinShuffle protocol. In the Announcement phase, each participant generates a short-term encryption-decryption key pair and broadcasts their encryption keys. In the Shuffling phase, the participants begin by generating a new Bitcoin address which serves as their output

address. Next, they shuffle these output addresses with a method similar to decryption mix networks [17]. The output of this phase is a shuffled set of output addresses, eliminating linkability between inputs and outputs. In the final Transaction Verification phase, each participant verifies that their generated output address exists in the shuffled list of output addresses. If so, a CoinJoin transaction is created including all of the inputs and the shuffled outputs. Each participant then signs the transaction and broadcasts it to the other users. Without signatures from each participant, the transaction is invalid. After all signatures are received, the transaction is ready to be sent to the network.

The use of a CoinJoin transaction allows for transaction fees to be split between all participants involved. The fee is less for each user and there are no separate mixing fees involved. Additionally, if a participant will require a change transaction, they may include a specific change address in the list of output addresses. Overall, CoinShuffle presents an effective decentralized protocol while making assumptions of methods for user communication. Due to these assumptions, the inclusion of a centralized, trusted authority is not completely ruled out.

5.1.3 CoinParty

Ziegeldorf *et al.* [20] proposed CoinParty in 2015. The mixing protocol is executed in multiple one-to-one transactions and offers decentralization and security. While being completely compatible with the existing Bitcoin network, CoinParty uses Secure multi-party computation (SMC) for users to collaborate.

CoinParty operates in three phases: Commitment, Shuffling, and Transaction. Figure 5.3 depicts the culmination of these phases with three participants. All n users have a specific, agreed upon amount of Bitcoin v at their input address (represented by $I_1 ... I_n$). Each participant's output address is represented by $O_1 ... O_n$. At the end

of the protocol, each participant receives v Bitcoin to their respective output address. However, only participant i can link I_i and O_i together. To achieve unlinkability, a secret permutation, π is used such that $I_i \xrightarrow{v} O_{\pi(i)}$.

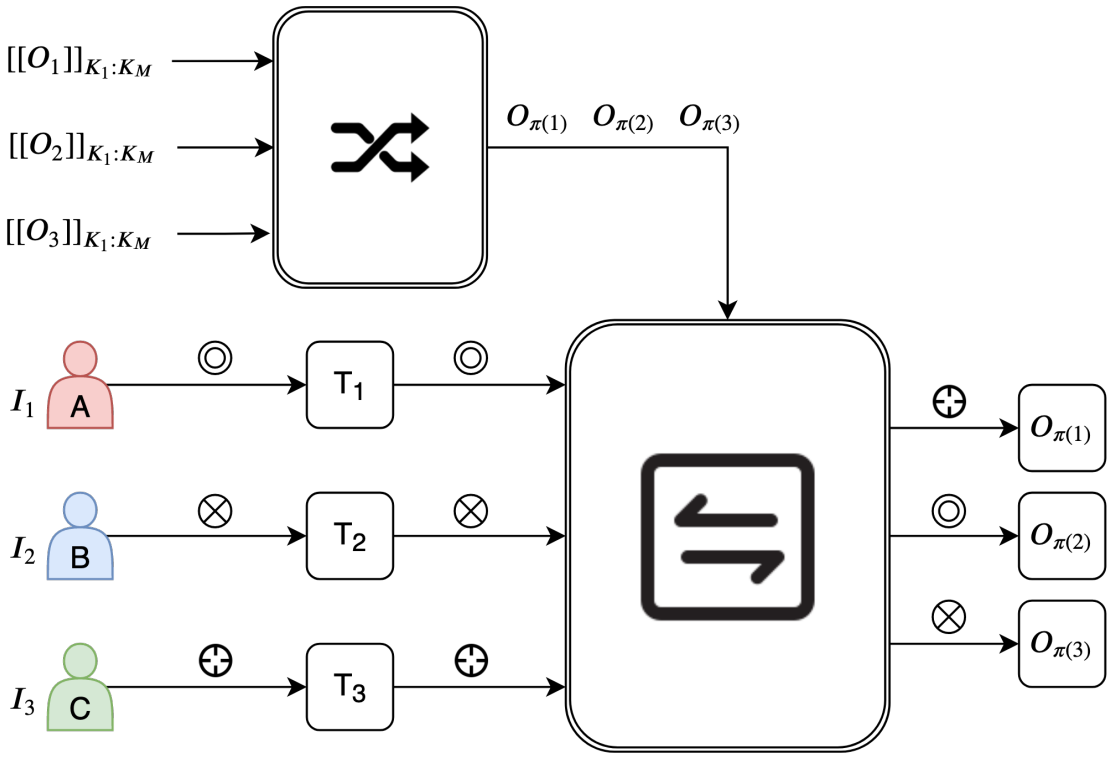

Figure 5.3: CoinParty Architecture

During the Commitment phase, the participants send the required funds to a temporary escrow address T_i. T_i is created using a threshold ECDSA key pair and is expected to be precomputed by the participants. In order for the funds to leave the escrow address, the majority of mixing peers must collaborate, protecting the funds from theft by a malicious user. Before the next phase begins, all transactions to escrow addresses must have atleast six confirmations.

In the Address Shuffling phase, the random, secret permutation π is used to shuffle the set of output addresses. Using a process inspired by decryption mixnets, CoinParty maintains anonymity against both outsiders and participants [21]. The

process requires that each mixing peer encrypts and broadcasts their output address O_i. Encryption is done with the public keys of other participants $K_1...K_M$ such that it results in a layered encryption $E_{K_1}(...(E_{K_M}(O_i))$. This layered encryption is sent to each mixing peer in which they decrypt a layer, apply a permutation, and send to the next participant. After successful shuffling, the link between participants and their corresponding output address should be erased. The addresses are then sorted lexicographically by the final participant and a pseudorandom number generator is used to identify a final random permutation. Checksums are used at each stage of the shuffling to ensure correctness.

In the Transaction phase, the participants create transactions $I_i \xrightarrow{v} O_{\pi(i)}$. These transactions must be collaboratively signed to be executed. A threshold variant of the ECDSA scheme is employed where transactions can be executed when 2/3 of the participants agree to do so.

5.1.4 Xim

Bissias *et al.* [22] explore the threats presented by Sybil-based denial-of-service attacks to Bitcoin mixing services. In response, they present Xim, a two-party mixing implementation. Unlike the previously described methods, Xim provides a decentralized method for mixing participants to find each other. The complete protocol requires no changes to the existing Bitcoin implementation. The goal of Xim is to provide a method for parties to discover, partner, and exchange funds while providing unlinkability and fairness.

The Xim protocol is done in two phases: Discovery and Fair Exchange. The second phase of the protocol can be done with either Barber's Fair Exchange Protocol [23] or SMC in Bitcoin [24]. Our analysis of Xim focuses on phase one, the discovery of a mixing partner. This phase depends heavily on an off-blockchain approach to

anonymous messaging. Joining a mix interaction requires both participants to spend τ funds. The requirement to pay to advertise and respond to desired mixing partners make Sybil attacks difficult.

In the Discovery phase, participants randomly take on the role of the advertiser A or respondent R. The advertiser A utilizes the TEXT field in a transaction of $\tau/2$ to post an ad stating the anonymous messaging platform where they can be reached, a unique nonce, and the amount they desire to mix. Any communication sent to A over the messaging platform are encrypted with the public key used for the ad. Any responses by R will include the ad, a nonce, and an anonymous method at which they can be reached. When a respondent is selected, A posts a signed message containing their nonce and the hash of R's nonce. This notifies R that they have been selected. R then posts an ad via a transaction of size τ with the TEXT field containing the message posted by A encrypted with A's public key. Finally, A publishes a response ad of $\tau/2$ with the hash of R's nonce. At the end of this phase, both parties have confirmed their interest in participating in a fair exchange using protocols outlined in [23] or [24].

5.2 Centralized

Centralized mixing protocols aim to secure a scheme in which an untrusted third-party exists. Mix participants meet and send their funds through these centralized services. The following mixing protocols attempt to address some of the threats discussed in Chapter 3 Section 3.2.

5.2.1 Obscuro

Tran *et al.* [13] present a centralized Bitcoin mixer using Trusted Execution Environments (TEEs). Obscuro addresses the threats posed by mixing operators to

lessen the control they have on the functionality and day-to-day activity of the service. To do so, the mixer codebase is isolated from the rest of the system. Users are given the ability to verify the isolated functionalities and are guaranteed a large mixing set size. OBSCURO's implementation requires no changes to the existing Bitcoin network and is generic such that it can be implemented with any TEE technique. In [13], the authors use Intel Software Guard Extension (SGX) [25, 26].

Intel SGX allows applications to be run in a special memory region called an enclave. This region is isolated from the rest of the system's software, including the operating system. Remote attestation is possible with the contents of the enclave to ensure they are performing the expected functionality. In runtime, the contents of the enclave are also encrypted. Thus, Intel SGX offers both integrity and confidentiality. However, simply including a mixer's functionality in an enclave does not protect the service from a malicious mixer operator. Mixer operators still have the ability to control the worldview of the service including participants and blockchain information. OBSCURO addresses this issue and presents a highly secure TEE-based protocol.

The main design goals for OBSCURO include indirect deposits, a guaranteed mixing set size, and state-rewind resistance. In addition, this architecture guarantees refunds and resistance to IP-level de-anonymization. Figure 5.4 depicts OBSCURO's four phases (represented by 1, 2, 3, 4 respectively): Bootstrapping, Indirect Participation, Block Scanning, and Final Announcement. The red fields indicate untrusted regions while green represent trusted regions of the architecture.

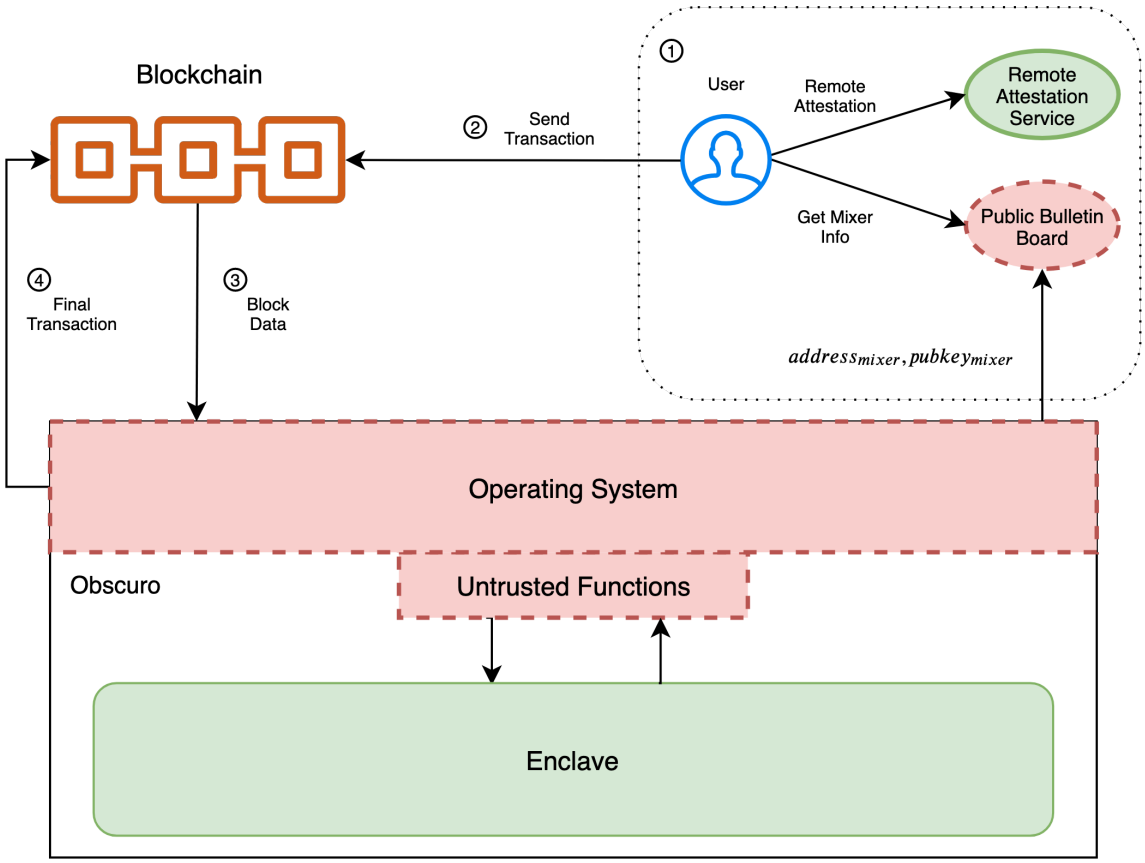

Figure 5.4: Obscuro Architecture

In the Bootstrapping phase, Obscuro generates a new public key ($pubkey_{mixer}$) and address ($address_{mixer}$) for the mixer. Then the service posts $pubkey_{mixer}$, $address_{mixer}$, and some metadata related to remote attestation onto public bulletin board(s). Next, the user retrieves this information from a bulletin board and uses a public remote attestation service to ensure that the correct functions are loaded into the enclave. Any tampering or false information posted to bulletin boards would lead to a failure in remote attestation and bootstrapping to the mixing service. For this reason, Obscuro prevents malicious mixing operators from refusing service or reducing the number of benign users. Any false information would be public and result a self-inflicted Denial of Service attack by the mixing service, since no one would be able to participate.

In the Indirect Participation phase, the user sends their deposit funds to $addr_{mixer}$. The transaction tx also includes the user's desired output address. In order to ensure the confidentiality of this output address, it is encrypted with $pubkey_{mixer}$. This deposit is covered by the guaranteed refund mechanism such that if the funds are not mixed by a specific waiting time, the user is guaranteed a refund.

During the Block Scanning phase, Obscuro downloads blockchain data. The service then scans for transactions depositing to $address_{mixer}$. For these transactions, the service decrypts the output address. In addition, Obscuro verifies the integrity of the scanned blockchain based on a recent checkpoint from the previous round's scan. This phase removes the need for interaction between the mixer and the user. In addition, the removal of benign users would require costly selectivity when scanning the blockchain.

The Final Announcement phase does not begin until Obscuro receives a specified minimum number of participants. When this minimum is reached, transaction TX is created including all of the deposits as inputs and output addresses as outputs. To ensure unlinkability, the output addresses are shuffled. Then, TX is broadcast to the network and the next mixing round begins.

5.2.2 Mixcoin

Bonneau *et al.* [4] propose Mixcoin, a Bitcoin mixing protocol that provides accountability to expose malicious centralized mixers. To do so, signed warranties are implemented between the participants and the service. If any wrongdoing occurs on the mixer's part, users have proof of an agreement between both parties to post on public forums. Warranties can be verified by publicly available information like transactions or public keys. Thus, Mixcoin provides an incentive for mixers to operate in a trustworthy manner. The protocol assumes there are various mixers M_i. Each

mixer has a warranty signing key K_{M_i} which is consistently used to sign warranties with each participant. Thus, the mixer's reputation relies heavily on the use of their key.

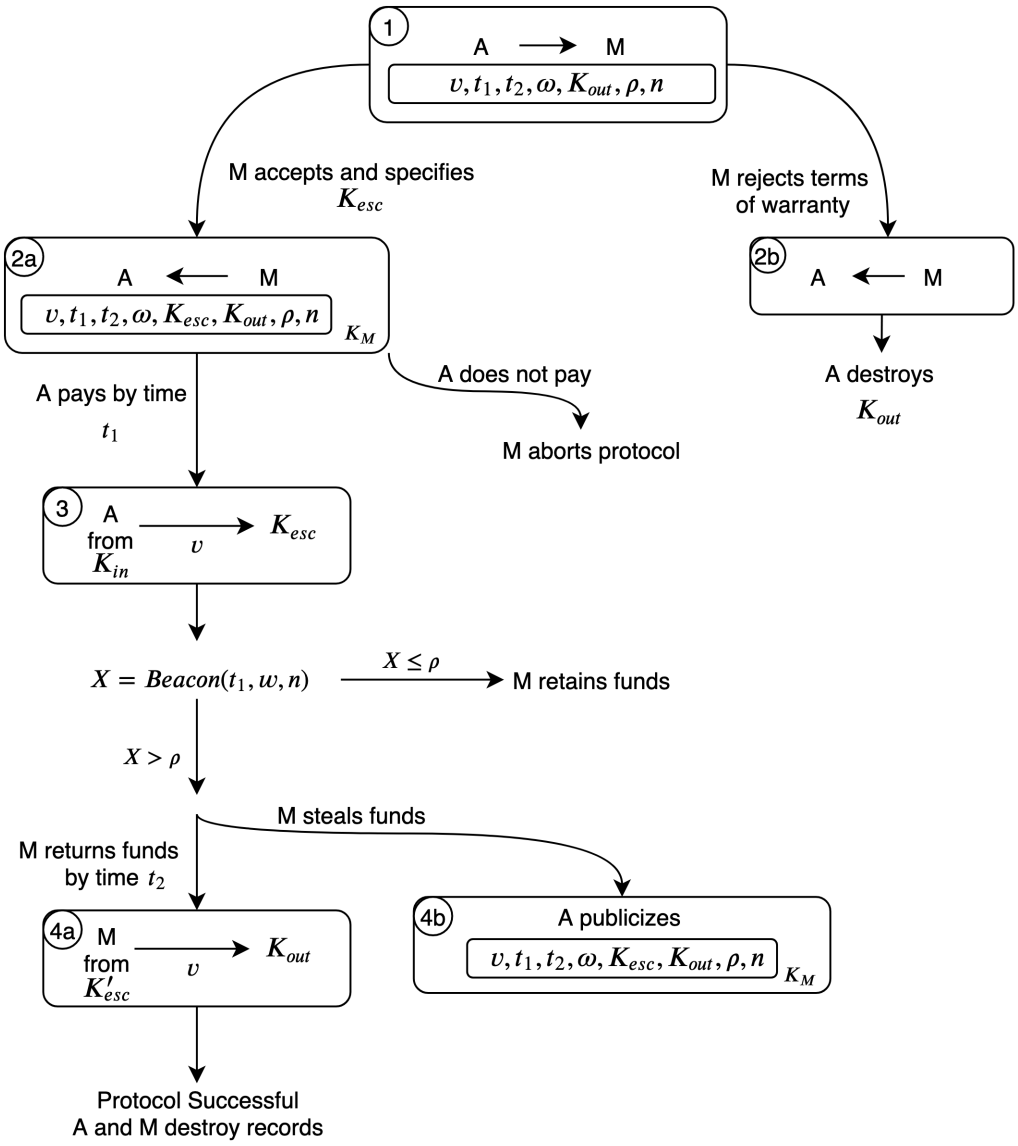

Figure 5.5: Mixcoin Protocol

Figure 5.5 outlines the Mixcoin protocol with Alice A and mixer M. The following parameters from [4] are in the initial warranty that Alice proposes:

v	value to be mixed
t_1	time by which A is required to send v BTC to M
t_2	time by which M is to return funds to A
K_{out}	output address where A wants to send her funds
ρ	mixing fee rate for A
n	nonce for randomized fees
ω	number of blocks required to confirm A's payment

If M agrees to the contents of the proposed warranty, it generates a new escrow address K_{esc} and adds it to the warranty. M then signs the warranty with K_M and sends it back to A (Step 2a). Step 2b depicts the case in which M rejects the terms of the warranty.

After receiving the signed warranty from M, A is expected to send funds to K_{esc} before time t_1. If A does not, M can simply delete its records. Since, A did not abide by the warranty, there is no proof that M was malicious. If A does send the funds before t_1 (Step 3), M is required to send an equal amount of funds to K_{out} by t_2. However, there is the possibility that the funds can be retained by M as a fee. This is done by calculating a random number via a Beacon function. If the random number is less than or equal to the agreed fee rate, M retains the funds. If the random number is greater than the fee rate and M sends the funds to K_{out} before t_2 (Step 4a), both A and M can destroy their records and carry on after a successful mix. However, if M does not send the funds to K_{out} before t_2 or steals the funds, A can publicize the warranty to prove that M is malicious (Step 4b). Although accountability is achieved, M is still able to steal funds from its users and potentially leak permutations between inputs and outputs.

5.2.3 Blindcoin

Valenta and Rowan [27] address Mixcoin's susceptibility to permutation leak attacks with Blindcoin. Without any changes to the existing Bitcoin protocol, a blind signature scheme and an append-only public log are added onto the Mixcoin protocol. Figure 5.6 depicts the Blindcoin protocol. The protocol includes three entities: the mixer M, Alice A, and an append-only public log. Parameters v, K_{out}, ρ, n, and ω are unchanged from Mixcoin (Section 5.2.2). The following is the list of parameters newly added or redefined from the Mixcoin protocol [27]:

K_{in}	address from which A sends their funds
K_{esc}	escrow address provided for A by M
K'_{esc}	escrow address M uses to send to K_{out}
A'	anonymous identity A uses to post to the public log
M_{pub}	public key of M
M_{priv}	private key of M
A_C	secret commitment/encryption function belonging to A
$A_{C'}$	secret decryption function belonging to A
t_1	time by which A is required to send v BTC to K_{esc}
t_2	time by which M must post token T to the public log
t_3	time by which A' must unblind the output address
t_4	time by which M must send v BTC to k_{out}
D	The mix parameters $\{v, t_1, t_2, t_3, t_4, \omega$, and $\rho\}$

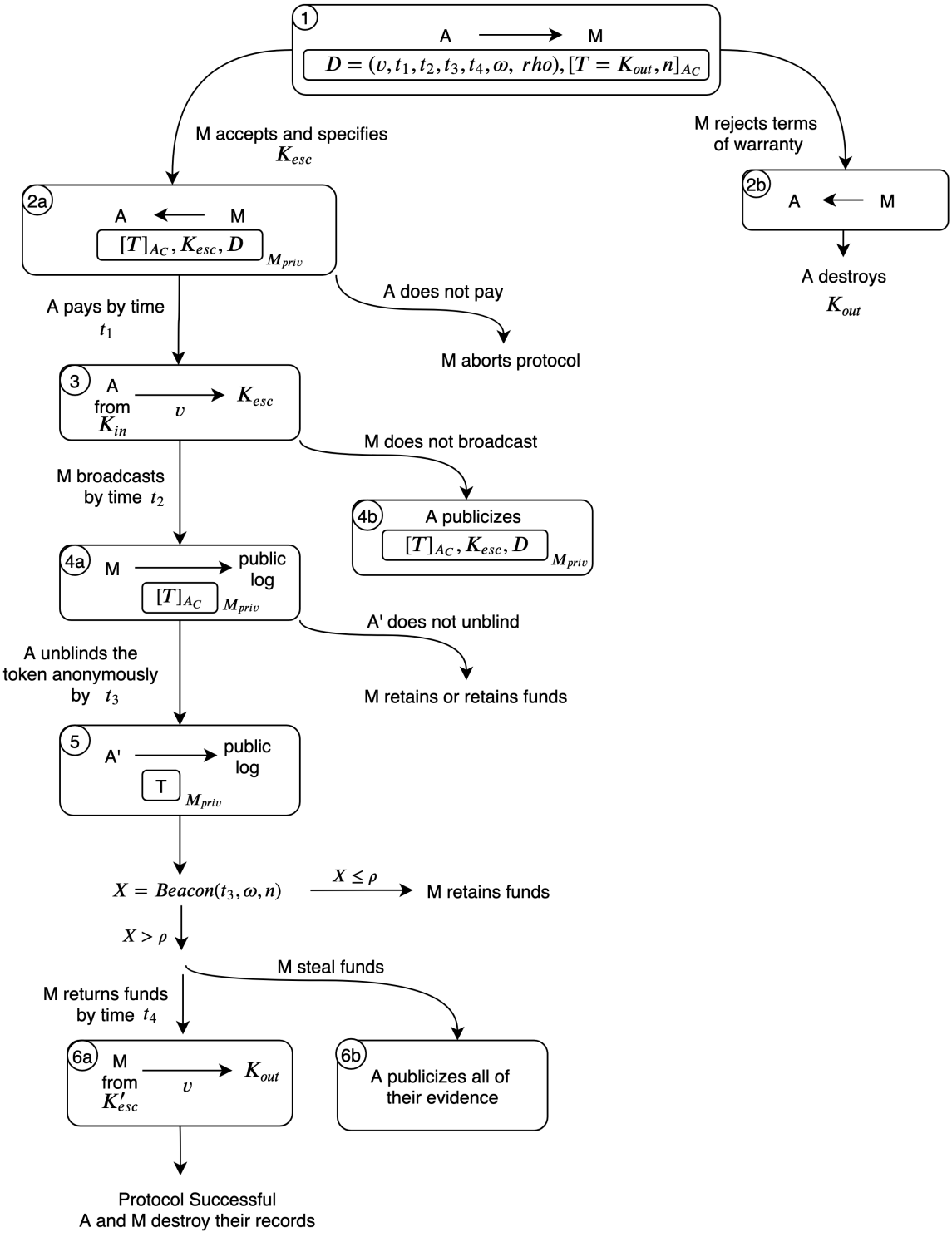

Figure 5.6: Blindcoin Protocol

In Step 1, A sends the mix parameters D to M. This offer includes v, t_1, t_2, t_3, t_4, ω, and ρ. In addition, this initial offer includes a blinded token T which consists of K_{out} and nonce n. The token is encrypted using A_C, a commitment function which can only be decrypted by A. In Mixcoin, this blinded token was not included and allowed the mixing service to see K_{out} and n.

After receiving the offer, M may accept (Step 2a) and send back a partial warranty. This reply consists of T, D, and a unique escrow address K_{esc} all signed by M_{priv}. In Mixcoin, the mixer sends the signed warranty back to A. If M rejects the offer (Step 2b), A can delete the output address and carry on. Steps 3a and 3b are unchanged from the Mixcoin protocol in which A is expected to send funds to K_{esc} before time t_1.

In Step 4a, M completes the warranty by signing the blinded token with M_{priv} and posting it to the public log by t_2. This serves as proof that A has sent funds to M on time and can be verified by any third-party. However if M fails to complete the warranty, A can publish incriminating information publicly including the partial warranty and the transaction before t_1.

Assuming that M successfully published the signed blinded token before t_2, A can use an anonymous identity A' to uncover the blinded token and repost it to the public log (Step 5). This action must be done by time t_3 or M is allowed to keep the funds. After this post by A', M now knows K_{out}. However, in this case all M has is a set of unblinded tokens available on the public log. M does not know their corresponding input addresses. Next, M can calculate the Beacon function for each to identify which inputs will be taken as fees. Fee calculation is the same in both Mixcoin and Blindcoin.

Finally, M is expected to pay to the output address before time t_4 if they have passed the Beacon function. If M does not send the funds back in time, A has solid

incriminating proof of a breach in protocol. This includes the partial warranty, the commitment function for the blinded token, the input transaction to K_{esc}, the signed token by M, and proof that the funds were not received by t_4. As a result, Blindcoin ensures accountability while keeping the mapping of input to output addresses secret. However, Blindcoin does not prevent coin theft since M can still steal funds from its users.

5.2.4 TumbleBit

In [28], Heilman *et al.* present TumbleBit, a unidirectional, unlinkable payment hub protocol. TumbleBit is completely compatible with the current Bitcoin protocol and relies on an untrusted centralized intermediary $\mathcal{M}$ to transfer funds between users. TumbleBit's transactions are sent off-blockchain and are not effected by the latency issues in Bitcoin. These payments are essentially off-blockchain puzzles generated through interactions with $\mathcal{M}$.

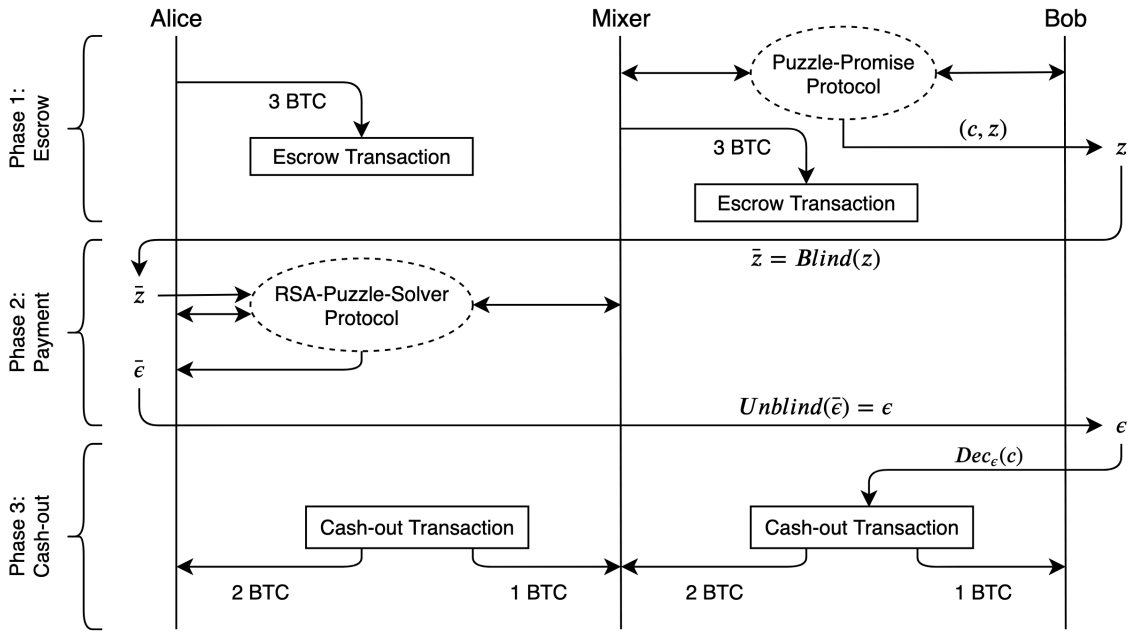

Figure 5.7: TumbleBit Protocol

TumbleBit consists of three phases (depicted in Figure 5.7). In this example, the protocol consists of three entities: Alice A, Bob B, and the Mixer $\mathcal{M}$. During the Escrow phase, A sends her Q BTC to an escrow address to open a payment channel with $\mathcal{M}$. Next, for B to create a channel with $\mathcal{M}$, $\mathcal{M}$ must first send Q BTC to another escrow address. These escrow transactions are 2-of-2 escrow smart contacts, meaning funds can only be accesses by transactions signed by both parties of the payment channel. Next, B and $\mathcal{M}$ make use of a puzzle-promise protocol. This protocol creates up to Q puzzles for B. In addition the puzzle-promise protocol is also a promise from $\mathcal{M}$ that B will receive one BTC for each solved puzzle. Each puzzle is an RSA encryption of a value.

In the Payment phase, A makes up to Q payments to B. These payments are off-blockchain meaning A must interact with $\mathcal{M}$ to learn the solutions to puzzles that have been provided by B. To do so, Bob first chooses a puzzle z and blinds it to be $\bar{z}$. Blinding z ensures that $\mathcal{M}$ cannot link z to $\bar{z}$. This blinded puzzle is then sent to A through $\mathcal{M}$. Next, A interacts with $\mathcal{M}$ to solve $\bar{z}$. The puzzle-solver protocol is used to execute a fair exchange such that A gives $\mathcal{M}$ one BTC if and only if $\mathcal{M}$ provides a valid solution $\bar{\epsilon}$. When obtained, A sends $\bar{\epsilon}$ to B through $\mathcal{M}$. This is repeated for Q' puzzles.

In the final Cash-out phase B uses his Q' solved puzzles to obtain Q' BTC from $\mathcal{M}$'s escrow address. $\mathcal{M}$ then obtains the remaining funds from its escrow and Q' BTC from A's escrow address, leaving the rest for A to keep. Throughout this scheme, the participants do not need to interact with one another and the mixer does not lose its initial amount of funds.

Chapter 6

PUBLIC MIXING SERVICES

6.1 Table of Mixers

To begin our analysis of the current mixing service landscape, we gathered a list of 19 centralized mixers. The majority of these mixers were posted as service announcements on Bitcointalk, Bitcoin's main forum created by Satoshi Nakamoto in 2009. Initial coin offerings, new service announcements, and technical discussions often originate from Bitcointalk. Next, our goal was to compile a list of characteristics that would provide insight into their implementations. Table 6.1 outlines the characteristics we collected for each mixing service. Our findings are displayed in Table 6.2 with some of these characteristics omitted for simplicity. A 'checkmark' signifies that the service offers the feature while an 'x' indicates that it is not supported. Any field marked with a dash was not found. The information is solely based on data available on each mixer's site and does not involve any transactions.

6.2 Popularity Analysis

Our next step was to identify a metric to rank mixing services based on popularity. As seen in Table 6.2, every mixing service has a highly recommended Tor mirror. These sites have a .onion extension and cannot be indexed by standard search engines. As a result, identifying the amount of traffic for each service is quite difficult. To address this obstacle, we first aimed to categorize mixers into two categories: Trusted and Untrusted. We based this categorization strictly off of service and user activity on Bitcointalk.

Characteristic	Description
Min	Minimum mixing amount allowed
Max	Maximum mixing amount allowed
Account	Is registration required to participate?
Fees	Mixing fees
Time	Time to finish mixing
Delay	Amount of delay on mixing output
Logs	Amount of time service keeps logs
Input Addresses	Number of input addresses given to user
Output Addresses	Number of output addresses user may specify
Distribution Control	Does the user have control of the distribution of funds across their specified output addresses?
Minimum Blocks	Number of network confirmations needed before mixing begins
Additional Features	Additional unique features (letter of guarantee, receipt, check mix function, etc.)
Tor	Hidden service URL
Clearnet	Clearnet URL
Established	Year established
Bitcointalk	Bitcointalk service announcement URL

Table 6.1: Mixing Service Characteristics

	Mixer	Year	Account	Mixing Fees	Distribution Control	Delay	Multiple Output Addr	Multiple Input Addr	Tor	Clearnet	Bitcointalk	Min. Blocks
Trusted	ChipMixer	2017	✗	✗	✓	✓	✓	✗	✓	✓	✓	1
	MixTum	2018	✗	✓	✗	✓	✓	✗	✓	✓	✓	1
	Bitcoin Mixer	2019	✗	✓	✓	✓	✓	✗	✓	✓	✓	1
	CryptoMixer	2016	✗	✓	✓	✓	✓	✓	✓	✓	✓	1
	Sudoku Wallet	2019	✗	✓	✗	✓	✓	✗	✓	✓	✓	3
	BitCloak	2016	✗	✓	✗	✓	✓	✗	✓	✓	✓	1
	BitMix.biz	2017	✗	✓	✗	✓	✓	✗	✓	✓	✓	1
	Mixer.money	2016	✗	✓	✗	✓	✓	✗	✓	✓	✓	-
	FoxMixer	2017	✗	✓	✓	✓	✓	✗	✓	✓	✓	6
Untrusted	Bitcoin Fog	2011	✓	✓	✓	✓	✓	✓	✓	✗	✓	6
	PenguinMixer	2017	✗	✓	✗	✓	✓	✗	✓	✗	✗	2
	SmartMix	2019	✗	✓	✓	✓	✓	✗	✓	✓	✓	3
	Blender.io	2017	✗	✓	✗	✓	✓	✗	✓	✓	✓	3
	DarkWeb Mixer	-	✗	✓	✗	-	✓	✗	✓	✗	-	-
	BMC Mixer	2017	✗	✓	✗	✓	✓	✓	✓	✗	✓	2
	Anonymix	2020	✗	✓	✓	✓	✓	✗	✓	✓	✗	1
	Mixer Tumbler	2019	✗	✓	✗	✓	✗	✗	✓	✓	✓	3
	AtoB Mixer	2019	✗	✓	✗	-	✓	✗	✓	✓	✓	-
	BlockMixer	2020	✗	✓	-	-	✗	✗	✓	✓	✓	3

Table 6.2: Trusted and Untrusted Public Mixing Services

Mixing services labeled as Trusted displayed consistent communication with users on the forum and had zero scam accusations. Untrusted mixers displayed a lack of communication with their users and had one or more scam accusations. Any mixer without a service announcement on Bitcointalk was also marked as Untrusted due to a lack of information from its users base.

After analysis of forum posts, nine of the mixers were labeled as Trusted and ten were labeled as Untrusted. We chose one Untrusted and five Trusted services to further our analysis. This includes PenguinMixer, ChipMixer, MixTum, Bitcoin Mixer, CryptoMixer, and Sudoku Wallet. In Chapter 7, we outline experiments conducted on the five Trusted services. In the following sections, we discuss each of the six selected mixing services.

6.3 PenguinMixer

PenguinMixer is an open-source mixing service and is only accessible via Tor. Although we categorized the service as Untrusted, an analysis of its implementation can be used to understand more complex, black-box mixing services like the ones interacted with in Chapter 7. Their site includes steps and necessary files to set up a mixer as a hidden service. In this chapter, we present an analysis of its source code and testing done on a local version of the service.

PenguinMixer offers limited user control for its obfuscation techniques. Participants are able to enter up to five output addresses and set a fast or slow delay. The mixing fee for the service is randomized between 0.5% and 1%. The maximum input increases as the service gains participants and is currently set to 5.0 BTC. The minimum input is 0.05 BTC multiplied by the number of output addresses used.

PenguinMixer's code base is written in PHP and Python. There are four files shared with the public related to account creation, mixing technique, the SQL database,

and the check mix feature. In the following sections, we will discuss the contents of each file.

accounts	
PK	account_id int NOT NULL auto_increment
	input_address varchar(35) NOT NULL UNIQUE
	required_confirmations int NOT NULL
	active_flag char(1) NOT NULL DEFAULT 'Y'
	secret_mixing_key char(64) NOT NULL UNIQUE
	created_datetime datetime NOT NULL DEFAULT CURRENT_TIMESTAMP

(a) Accounts

payments	
PK	payment_id int NOT NULL auto_increment
FK	output_address_id int NOT NULL references output_addresses(output_address_id)
	amount_gross decimal(16,8) NOT NULL
	amount_net decimal(16,8) NOT NULL
	transaction_id char(64) NULL
	created_datetime datetime NOT NULL DEFAULT CURRENT_TIMESTAMP
	payment_datetime datetime NOT NULL

(b) Payments

output_addresses	
PK	output_address_id int NOT NULL auto_increment
FK	account_id int NOT NULL references accounts(account_id)
	output_address varchar(35) NOT NULL

(c) Output Addresses

Figure 6.1: PenguinMixer SQL Database Tables [1]

6.3.1 Database Structure

The SQL database consists of three tables including user accounts, payments and output addresses. Figure 6.1 shows each of these tables and their attributes. In the figure, the green rows are primary keys (PK) and the yellow rows are foreign keys (FK).

The account table (Figure 6.1a) stores mixing participant information. Unlike the traditional user account, an account on PenguinMixer does not have a username and password. Instead, users are labeled with account IDs which are auto incremented and start at 1 with the mixer operator. Participants are also given an input address, a specific number of confirmations, an active flag, timestamp of account creation, and secret mixing key.

The output_addresses table (Figure 6.1c) consists of information related to the destinations specified by users. The primary key is an auto incremented output address ID and the foreign key is the account ID from the accounts table.

The payments table (Figure 6.1b) stores information related to outgoing transactions from the mixer. The primary key is an auto incremented payment ID and the foreign key refers to the output address ID from the output_addresses table. Additionally, the gross amount, net amount, transaction ID, timestamp of creation, and timestamp of each payment are stored.

6.3.2 Account Creation

An account on PenguinMixer is created after a user enters up to five output addresses and selects a fast or slow delay. The addresses are validated using the bitcoind call *validateaddress()*. When completed, the addresses are added to the database. User delay is then calculated by random number based on the selection of

fast or slow delay. A fast delay results in two to five confirmations and a slow delay is six to 24 confirmations. The result is stored in the database as *required_confirmations* in the accounts table. Next, the bitcoind call *getnewaddress()* is used to generate a new input address for the mixer to which the user is expected to send their funds within 24 hours.

6.3.3 Check Mix

Before the user account is finalized, a 32-byte, randomly generated key is provided for them to note. This secret mixing key can be used when communicating with support or when using the check mix feature. After entering the key onto the check mix page, the user is presented with information about the status of their mix including the account creation timestamp, input payment deadline, required confirmations, input address, and the minimum required payment to the input address.

6.3.4 Mixing Implementation

PenguinMixer's mixing method uses bitcoind functionality to ensure that users do not receive their own inputted funds. First the service checks which mixer addresses have received payments by using *listreceivedbyaddress()*. Next, the number of confirmations for each of these payments is checked. If the required number of confirmations has been reached, a random payment amount is calculated for each corresponding output address. Eq. 6.1 is the formula used to calculate this random amount for each output address. The variable *curr_due* refers to the total amount currently due for the participant and *min_per_output* is the minimum required per output address. *Num_output* refers to the number of output addresses for the participant and *curr_output_num* is the current output address number in the order which they were entered.

$$amount = curr\_due - (min\_per\_output * (num\_output–curr\_output\_num)) \quad (6.1)$$

In addition, each output address receives a random amount of delay (in seconds) and a randomized mixing fee between the percentages set by the operator. All calculated payment information is added to the database and the mixer begins the payment process for the particular user. To do so, *listunspent()* is called to return all unspent inputs to the mixer. Next, *listaddressgroupings()* is used to group all related wallet addresses with change addresses. The most important step occurs when the mixer uses *lockunspent()*. This temporarily locks any addresses related to the input address of the current participant. Thus, it is guaranteed that users will not receive their own funds from the mixing service. All interactions with the service are logged in separate files for each day. Although the service claims logs are deleted every seven days, it must be done manually by the mixer operator.

6.4 ChipMixer

ChipMixer was established in 2017 and operates on both the clearnet and Tor. With over 95 pages of Bitcointalk forum posts and no scam allegations, the service presents itself as the most popular of the Trusted services identified. ChipMixer presents a unique implementation with the introduction of "chips". It generates addresses and funds them with increments of 0.001 BTC up to 8.192 BTC. These addresses are previously funded and provided to ChipMixer's participants along with their corresponding private keys as outputs. Rather than executing on-blockchain transactions, users are expected to import the given private keys to their wallets off-blockchain. Thus, there is completely no link between funds deposited to ChipMixer and the chips given to participants. If a user deposits 0.005 BTC to ChipMixer, they

should expect to receive multiple chips adding up to the total deposited. For example, they may receive two separate chips of 0.002 BTC and one chip of 0.001 BTC.

Users may split, merge, bet, or donate the given chips before withdrawal. Splitting breaks one chip into two. Merging combines two same-sized chips into one. Betting allows users to bet a chip and receive one double the size with a 47% chance. Donating gives the chip to the service. Finally, withdrawing presents the private keys for the selected chips to be added to a personal wallet. These features can be used multiple times, in any order, and on individual chips. For example, a user may split their chips and choose to donate some of the resulting chips before withdrawing the others. ChipMixer claims these features increase user privacy and change the possible output value.

While ChipMixer does not require an account, users are given a session token and an input address which last for seven days. The service also gives users the option to destroy their sessions prematurely within this seven day period. Service logs are kept for the same length, however it is unclear whether logs can be destroyed prematurely along with the session. There is no predefined maximum input and the minimum input is 0.001 BTC. This input only requires one confirmation in the Bitcoin network before is accepted by the service. Any input less than the minimum is treated as a donation. Mixing fees are purely donation-based and users may choose to donate any amount of their given chips. On withdrawal, users are given a cryptographically signed receipt proving that the funds are coming from ChipMixer. In addition, they are given the option to receive a voucher code and use the non-withdrawn chips in other ChipMixer interactions.

6.5 MixTum

MixTum was established in 2018 and operates on both the clearnet and Tor. It is one of 11 mixers built upon the mixing platform Jambler.io. The service claims to have a separate pool of Bitcoin from cryptocurrency stock exchanges like Binance, OKEex, and DigiFinex. In turn, MixTum guarantees that participant funds are not mixed within a pool of other user's Bitcoin and instead outputs are primarily from exchanges. The service claims its implementation eliminates the possibility for users to receive previously inputted funds. Thus, an account is not required and session IDs are not provided. The site states that the service is completely automated and requires no manual interference from the mixer operator.

Like PenguinMixer, MixTum is a traditional Bitcoin mixer that sends on-blockchain transactions to return participant funds. Thus, the service implements similar obfuscation techniques to avoid blockchain analysis. Mixing fees are up to 5% (randomized) plus 0.00015 for the output network fee. Users can specify up to two output addresses which receive multiple payments when funds are returned. They are also given one input address which remains valid for seven days. The number of payments and distribution of funds between these addresses is randomized by the service. In addition, randomized delays of up to six hours are implemented on output transactions. MixTum provides users with a PGP signed letter of guarantee with information regarding the mixing interaction.

MixTum offers a free trial with the minimum required amount of 0.001 BTC, one output address, and no mixing fees. Any input less than the minimum is treated as a donation to the service. The recommended maximum input is 50 BTC. Although MixTum claims logs are not kept, they do keep data regarding participant interactions until the completion of the output transaction or until the session expires in seven

days.

6.6 Bitcoin Mixer

Bitcoin Mixer was established in 2019 and is accessible on the clearnet and via Tor. Much like PenguinMixer, the service provides its users with a Mix ID to check the status of their mix. The minimum input amount accepted is 0.0002 BTC. Anything lower than this minimum is treated as a donation. The maximum input amount is 80 BTC. Registration is not required and users are able to specify up to seven output addresses. When multiple output addresses are specified, users can control the distribution and delays for each. Delays for each output address range from less than one hour (rapid) to 12 hours. To send their funds, users are given one input address. Mixing begins immediately after one network confirmation on input amounts less than 20 BTC and six confirmations on any higher amount. The service keeps logs for up to seven days but gives users the option to manually delete their session details. Finally, the mixing fees for Bitcoin Mixer are 0.25% plus 0.000001 BTC per output address. The service does not offer a signed receipt or letter of guarantee.

6.7 CryptoMixer

CryptoMixer was established in 2016 and is available on both the clearnet and Tor. The service's initial announcement on Bitcointalk stated that it has over 2000 BTC in reserve. To prove this, the service gave reputable Bitcointalk members a list of their addresses with a signature from each. After verifying the signatures, each member posted a signed message on CryptoMixer's service announcement stating that they had verified the services pool of funds [29, 30, 31].

CryptoMixer allows a minimum input of 0.001 BTC. The maximum input changes based off of the amount of Bitcoin in the service's reserve. Accounts are not required

and instead users are given a CryptoMixer code to identify their sessions. This code can be used in future sessions to receive discounts and ensure previous inputs are not returned. CryptoMixer's site claims it has a 100% zero-logs policy but also states that transaction details are routinely deleted. The service allows users to specify up to 10 output addresses with custom delay for each from 0 minutes to 96 hours. In addition, the service offers distribution control for each output address. Mixing fees can be set by the user and range from 0.5% to 3% plus 0.0005 per output address.

Based on the fees, delays, distribution, and number of output addresses set, participants are given a security level for their mix. The Standard security level does not provide fast output transaction confirmation, allows for up to 24 hours of delays, and supports up to two output addresses. Service fees must be set anywhere between 0.5% and 1%. Silver security level guarantees fast output transaction confirmation, up to 48 hours of configurable delays, and up to five output addresses. Fees for the Silver level are required to be greater than 1% and less than 2%. The Gold security level is the highest level and guarantees fast output transaction confirmation, up to 96 hours of configurable delays, and up to 10 output addresses. For this level, the service fee must be set to 2% or higher.

Unlike the other discussed services, CryptoMixer allows users to generate an unlimited number of input addresses to send their funds. Each input address also comes with a verifiable, digitally signed letter of guarantee, proving that it was generated by the service. Each given address is only valid for 24 hours and deleted afterwards. The number of confirmations required for input transactions to be cleared for mixing varies based on the amount expected to be mixed. Less than 25 BTC, less than 250 BTC, and less than 1000 BTC require one, three, and four network confirmations respectively. Any input greater than 1000 BTC requires five confirmations.

6.8 Sudoku Wallet

Sudoku Wallet was established in 2019 and is available on the clearnet and Tor. The service is a single-use wallet which outputs private keys rather than on-blockchain transactions. These outputs are of two to four addresses funded from previously executed CoinJoin transactions. The distribution between these addresses is not customizable by the user. There is no minimum or maximum input enforced. Sudoku Wallet does not require accounts but provides users with a wallet key to access their session before it is automatically deleted in seven days. The service claims to have a strict "no logs" policy. To send funds to Sudoku Wallet, one input address is provided along with its corresponding private key. Mixing does not begin until the input transaction has three network confirmations. The mixing fee is randomized from 0.5% to 1% plus the CoinJoin fee which is described as the number of output addresses involved in the CoinJoin times the transaction fee.

Chapter 7

EVALUATION

In this chapter, we provide an overview of experiments conducted on ChipMixer, MixTum, Bitcoin Mixer, CryptoMixer, and Sudoku Wallet. We first describe the methodology, followed by the setup required for the experiments. Then, we outline the interactions made with each of the five services. Finally, we provide both an implementation and security analysis comparing the five selected services with the proposed mixing protocols discussed in Chapter 5.

7.1 Methodology

The experiments are based on real-world interactions with the five Trusted public mixing services: ChipMixer, MixTum, Bitcoin Mixer, CryptoMixer, and Sudoku Wallet. These five services were chosen based off of the popularity analysis conducted in Chapter 6. Our goal was to identify if these mixers have adopted implementation and security solutions provided by academic literature discussed in Chapter 5. During each interaction, we focused on the identification of behaviors indicative of the mixer's implementation. While doing so, we were also able to identify each service's resistance to the common mixing-related security threats discussed in Chapter 3 Section 3.2. Our security analysis expands on the work of Tran *et al.* in [13], which does not include public mixing services in its study. Overall, we use data from Table 6.2 and our experiments to compare implementation and security of the five services with the proposed mixing protocols from Chapter 5.

We conducted three trials of experiments. Each trial consisted of one transaction with each of five mixing services. We ensured that all five interactions during a trial

were finished before moving onto the next. To estimate the necessary amount of funds to execute all 15 mixer interactions, we set a constant network fee of 0.50 USD and calculated the worst-case mixing fees for each service. The total fees were estimated to be 57.25 USD. To account for changing network fees, unexpected mixer fees, or coin theft, we determined 100 USD would be sufficient to execute all three trials.

During the first trial, input amounts were set to the minimum required by each service. Inputs were gradually raised in the second and third trials. We heightened the intensity of the obfuscation techniques from trial to trial when customizable. This included longer delays, a higher number of output addresses, and higher fees. Changing these parameters and having multiple trials gave us the opportunity to explore a larger breadth of features for each service. The public nature of the blockchain allowed for comparison between interactions with a single service to identify unexpected behavior. We specify the exact parameters, input, and output values for each trial in Section 7.3. Eq. 7.1 was used to calculate the mixing fees for on-blockchain transactions. $Total\,Input$ and $Total\,Output$ refer to the total BTC sent to and from the mixing service including network fees. $Input\,Network\,Fees$ and $Output\,Network\,Fees$ are the network fees associated with $Total\,Input$ and $Total\,Output$.

$$(Total\,Input - Input\,Network\,Fees) - (Total\,Output - Output\,Network\,Fees) \quad (7.1)$$

Since all five mixers offer a Tor mirror, we used Tor Browser to interact with each service. This is the most popular method of using mixing services since it offers more privacy for users. To store, receive, and send Bitcoin, we used the desktop wallet Electrum. We maintained two separate wallets for legacy and SegWit functionality. Compatibility with SegWit results in lower transaction fees and provides insight into the mixer's implementation. All transactions were labeled according to their corresponding mixer and trial number. In addition to collecting screenshots of every

mixing interaction, the data described in Table 7.1 was recorded. The obfuscation parameter was manually changed for each trial and varies from mixer to mixer. Next, we will discuss the general steps taken and any special data collected for each service.

Data Field	Description
Obfuscation Parameters	Obfuscation features set (number of output addresses, delays, distribution, etc.)
Input Amount	Amount sent to mixer (before network fees)
Input Network Fee	Network fee on transaction to mixer (BTC)
Input Address	Address given to user by mixer to send initial funds
Time In	Date and time of input transaction
Input Transaction ID	Transaction ID of input transaction
Output Amount	Amount sent back to user's deposit address(es)
Output Network Fee	Amount of network fees on transaction(s) to deposit address(es)
Time Out	Date and time output transactions are sent from mixer
Output Transaction ID	Transaction ID of output transaction
Mixer Fee	Service fee collected
Additional Information	Information unique to service: Letter of Guarantee, Special Mixing Code, Receipt, etc.

Table 7.1: Data Collected During Each Trial

ChipMixer

There are five general steps in interactions with ChipMixer. During Step 1, users are given their session token and told to save it permanently to access their session for the next seven days. Step 2 is the Deposit step. Users are told to send at least 0.001 BTC in one transaction to a given input address, wait for one network confirmation on this transaction, and then refresh the page. During this step, users are also able to enter voucher codes from previous interactions to use funds that have not been withdrawn. At Step 3, users have a full view of their current chips grouped by value and have the ability to split, merge, commonize, bet and donate. On this page, they are also given the option to withdraw or receive a voucher for chips. These two options directly lead to Step 4, the withdrawal. Users are given the private key to their withdrawn chips and steps on how to import this key to Electrum, Bitcoin Core, or to a JSON file. As another option, they can sweep the chips to a desired output address. Before the final step, a signed receipt is offered for download. In Step 5, sessions can be destroyed.

We created a new session for each trial with ChipMixer. The session token was recorded to test its validity after the seven day period or after sessions were manually deleted. The given input address and the input transaction ID was noted to identify patterns in the movement of funds. Chipmixer's method of returning funds does not involve output addresses, so we used the SegWit wallet for all three trials. We considered the obfuscation parameters for ChipMixer to be the set of features used (split, merge, and donate) as well as the method of withdrawal. Commonize and betting were not given as options in all three trials. We attempted both sweep and private key transfer withdrawals to identify effects on traceability. Before destroying each session, we attempted to access each session's signed receipt to verify the signature.

MixTum

Interactions with MixTum consist of two steps. In Step 1, users enter up to two output addresses. In Step 2, users are given an input address along with its corresponding QR code. In addition, a signed letter of guarantee is provided for download.

Trials for MixTum were attempted with both legacy and SegWit addresses. The only customizable obfuscation parameter was the number of output addresses. On Step 2, all letters of guarantee were downloaded and signatures were verified using GnuPG. Transactions from MixTum were analyzed for their distribution and randomized delay. Mixing fees were also checked to see if they were accurately calculated. Input and Output transaction IDs were used to gain insight about the movement of funds.

Bitcoin Mixer

There are three steps in interactions with Bitcoin Mixer. In Step 1, users specify up to seven output addresses along with distribution (%) and delay (rapid to 12 hours) for each. In Step 2, the service provides a Mix ID and an input address. After delays have been reached, output transactions are executed. In Step 3, users are able to review their mix information and are given the option to delete their mix.

In Step 1, we attempted specifying both legacy and SegWit addresses to Bitcoin Mixer. The main obfuscation parameters for this service were the number of output addresses, the percentage distribution, and the delay. We heightened the intensity of these parameters from trial to trial and verified the accuracy of distributions and delays. Mix IDs for each session were noted to check their validity after deletion of the mix. After outputs were received, we calculated the mixing fees to identify

unexpected behavior. In all three trials, we deleted our mix information.

CryptoMixer

Interactions with CryptoMixer require two steps. In Step 1, users specify up to 10 output addresses and set the delay and distribution for each. Users can then specify their preferred service fee. The combination of these three obfuscation parameters determines the security level of the mix. On the same page, CryptoMixer's calculator displays the expected amount that each output address will receive. Before continuing to Step 2, the CryptoMixer code can be entered. In Step 2, a downloadable letter of guarantee is presented along with an input address. As input transactions are made the service displays the received amounts and their confirmations. If the amount is not sufficient, the service specifies the expected output as a negative value. Finally, users are also provided with a CryptoMixer code to use with future transactions.

Trials with CryptoMixer were conducted with both legacy and SegWit addresses. The customizable obfuscation parameters for this service include the number of input and output addresses, delay, distribution, and service fee. We originally set up our trials to test each security level. Due to lack of response from the service, we conducted Trial 2 and 3 with similar parameters and Silver security level. We recorded the output values displayed from the service's calculator to check for accuracy. The CryptoMixer code from Trial 1 was used in Trial 2 to test its effectiveness against receiving previous inputs. Finally, the letter of guarantee was downloaded for each input address in all three trials and both the signature and contents were verified.

Sudoku Wallet

Four steps are involved with Sudoku Wallet interactions. In Step 1, users are presented with a wallet key. In Step 2, an input address is presented along with its

corresponding private key. After three confirmations on the input transaction(s), the 'Mix my coins!' button can be pressed to proceed to the next step. In Step 3, two to four addresses with balances adding up to the user's input amount minus mixing fees are presented along with their corresponding private keys. The user then has the option to sweep these funds or import the private keys to their wallet. In Step 4, users are urged to delete their wallet and generate a new one to mix more funds.

We created a new wallet for each of our transactions and recorded the wallet key to check its validity after deletion. In Step 2, we noted the given input address and its private key. The obfuscation parameter for Sudoku Wallet is limited to the method of withdrawing the funds. In Step 3, we recorded the given output addresses and calculated the mixing fee to identify unexpected behavior. We also studied the history of these output addresses to ensure they were involved with CoinJoin transactions.

7.2 Setup

Before beginning the first trial, we obtained 100 USD worth of Bitcoin from the cryptocurrency exchange Coinbase. At the time, this equated to 0.01788742 BTC. Then, we created two separate Electrum wallets: Legacy and SegWit. The funds were then sent from Coinbase to the SegWit wallet to begin the first trial. In the next section, we outline the details of all three trials for each mixing service.

7.3 Experiments

7.3.1 ChipMixer

The results from each ChipMixer trial are displayed in Table 7.2. The fields include obfuscation parameters, total input, total output, output network fees, and mixer fees. For the sake of simplicity, transaction IDs, input addresses, output addresses,

timestamps, and session tokens have been omitted from the given data.

Trial	Obfuscation Parameters	Input (BTC)	Output (BTC)	Output Network Fees (BTC)	Mixer Fees (BTC)
1	sweep	0.001	0.000921	0.000079	0
2	split, donate, merge, withdraw	0.003	0.002	0	0.001
3	voucher, sweep	0.004	0.00381195	0.00018805	0

Table 7.2: ChipMixer Trials

Trial 1

In Trial 1, 0.001 BTC was sent in one transaction from the SegWit wallet. Within 30 seconds of the first confirmation on this input, we received one chip of 0.001 BTC. In Step 3, we were given the option to donate, withdraw, or receive a voucher. Options to split or merge were unavailable. We chose to withdraw our chips and proceeded to Step 4. We attempted to download the signed receipt but received an internal server error. Next, we chose to sweep the chip to the SegWit wallet with a network fee of 0.000079 BTC. The interaction resulted in 0 BTC mixing fees and our final output was 0.000921 BTC.

Trial 2

In Trial 2, 0.003 BTC was sent to ChipMixer in two separate transactions from the SegWit wallet. These transactions were 0.002 BTC and 0.001 BTC. The service provided one chip of 0.002 BTC (chip 1) and one of 0.001 BTC (chip 2). We split chip 1 into two chips of 0.001 BTC. Then, we donated one of these chips to ChipMixer and did not identify any movement of funds from the input address. Next, we merged the two remaining 0.001 BTC chips into one 0.002 BTC chip. On Step 4, we attempted to access the signed receipt but received an internal server error. We chose to withdraw our final chip by importing the private key into a new wallet. Importing resulted in 0

BTC network fees and 0 BTC mixer fees. The output to our wallet was 0.002 BTC.

Trial 3

In Trial 3, two separate sessions were created. In the first session, one transaction of 0.001 BTC was sent to ChipMixer and withdrawn for a voucher. The service provided a 53 character alphanumeric code. In the second session, one transaction of 0.003 BTC was sent to the given input address. The voucher code from the first session was also redeemed. In total, the service provided two 0.001 BTC and one 0.002 BTC chips. On withdrawal, the chips were swept into the SegWit wallet. This resulted in two on-blockchain transactions with outputs of 0.00190361 BTC and 0.00190834 BTC. The network fees associated with these transactions were 0.00009639 BTC and 0.00009166 BTC respectively. The total mixer fee was 0 BTC.

7.3.2 MixTum

Table 7.3 displays the obfuscation parameters, total input, total output, output network fees, and mixing fees pertaining to each trial with MixTum. To keep the data concise, transaction IDs, input addresses, output addresses, and timestamps have been omitted from the given data. For all three trials, signed letters of guarantee were successfully downloaded and verified. MixTum's calculator output displayed a smaller value than received on all three trials. In Trials 2 and 3, mixing fees were up to 5% plus 0.00015 BTC as advertised. However, Trial 1 charged a mixing fee of 0 BTC.

Trial	Obfuscation Parameters	Input (BTC)	Output (BTC)	Output Network Fees (BTC)	Mixer Fees (BTC)
1	1 Output	0.001	0.001	0.00024227	0
2	2 Outputs	0.002	0.001762	0.0004707	0.000238
3	2 Outputs	0.003	0.00276	0.00049838	0.00024

Table 7.3: MixTum Trials

Trial 1

In Trial 1, one legacy output address, O_1, was specified. A SegWit output address was attempted but was not accepted by the service. One transaction of 0.001 BTC was sent to a compatibility format input address provided by MixTum. Within five minutes, an output of 0.001 BTC was received by O_1. The network fee on the output was 0.00024227 BTC and mixing fees were 0 BTC.

Trial 2

In Trial 2, two legacy output addresses, O_1 and O_2, were specified. One input transaction of 0.002 BTC was sent to a compatibility format input address provided by MixTum. The first output of 0.001 BTC was received by O_1 in one hour and 14 minutes. The network fee on this transaction was 0.00024227 BTC. A second output of 0.000762 BTC was received by O_2 in four hours and 55 minutes with a network fee of 0.00022843. The overall mixing fee for this interaction was equal to 4.4% of the input plus 0.00015 BTC.

Trial 3

In Trial 3, two legacy output addresses, O_1 and O_2, were specified. Two input transactions were sent to a compatibility format input address provided by MixTum. The first transaction was 0.002 BTC and the second was 0.001 BTC. O_1 received two

output transactions of 0.0004 BTC and 0.001 BTC 47 minutes after the input. O_2 received 0.00136 BTC in 52 minutes. The network fees for these output transactions were 0.00017997 BTC, 0.00017305 BTC, and 0.00014536 BTC respectively. The overall mixing fee for this trial was 3% of the input amount plus 0.00015 BTC.

7.3.3 Bitcoin Mixer

Table 7.4 outlines the obfuscation parameters, input, output, and mixer fees associated with each Bitcoin Mixer trial. For the sake of simplicity, transaction IDs, input and output network fees, input addresses, output addresses, timestamps, and Mix IDs have been omitted. The distributions, mixing fees, and outputs associated with each trial were accurately calculated. Outputs were generally received 20 to 30 minutes early, indicating randomization of delays.

Trial	Obfuscation Parameters	Input (BTC)	Output (BTC)	Mixer Fees (BTC)
1	1 Output rapid delay	0.0002	0.0001985	0.0000015
2	3 Outputs distribution (%): 35, 35, 30 delay (hr): 1, 2, 2	0.0004	0.000396	0.000004
3	5 Outputs distribution (%): 13.3, 5.36, 21.98, 30.72, 28.64 delay (hr): 1, 2, 5, 10, 12	0.0006	0.0005935	0.0000065

Table 7.4: Bitcoin Mixer Trials

Trial 1

In Trial 1, one SegWit address, O_1 was specified with rapid delay. The service provided a compatibility format input address and a mix ID. One transaction of 0.0002 BTC was sent to this address. Within 30 seconds of the first network confirmation, a transaction of 0.0001985 BTC was sent to O_1. Overall, the interaction had a mixing fee of 0.0000015 BTC.

Trial 2

In Trial 2, three legacy output addresses O_1, O_2, and O_3 were specified. Delay and distribution among these addresses was set to be 1 hour with 35%, 2 hours with 35%, and 2 hours with 30% respectively. The service provided one compatibility format input address. One transaction of 0.0004 BTC was sent to this address. O_1 received 0.0001386 BTC in 43 minutes. O_2 received 0.0001386 BTC in 1 hour and 44 minutes. O_3 received 0.0001188 BTC in 1 hour and 44 minutes. The overall mixing fee for this trial was 0.000004 BTC.

Trial 3

In Trial 3, five SegWit output addresses O_1, O_2, O_3, O_4, and O_5 were specified. Delay and distribution was set to be 1 hour with 13.3%, 2 hours with 5.36%, 5 hours with 21.98%, 10 hours with 30.72%, and 12 hours with 28.64% respectively. The service provided one compatibility format input address. One transaction of 0.0006 BTC was sent to this address. O_1 received 0.00007894 BTC in 31 minutes. O_2 received 0.00003181 BTC in 1 hour and 26 minutes. O_3 received 0.00013045 BTC in 4 hours and 26 minutes. O_4 received 0.00018232 BTC in 9 hours and 26 minutes. Finally, O_5 received 0.00016998 BTC in 11 hours and 26 minutes. The overall mixing fee for this trial was 0.0000065 BTC.

7.3.4 CryptoMixer

Table 7.5 displays the obfuscation parameters, input, output, and mixer fees associated with each CryptoMixer trial. Transaction IDs, input and output network fees, input addresses, output addresses, timestamps, and CryptoMixer codes have been omitted for simplicity. the service's calculator displayed accurate outputs based on the set mixing fee for each trial. Overall, CryptoMixer displayed poor implementation and a lack of documentation, while providing the most user control of obfuscation parameters.

Trial	Obfuscation Parameters	Input (BTC)	Output (BTC)	Mixer Fees (BTC)
1	1 Output 2 Input 0.5060% fee 1hr 15min delay	0.001	0.00049494	0.00050506
2	CryptoMixer Code 3 Outputs 4 Inputs distribution (%): 20.05, 19.96, 59.99 delays: 3hr 7m, 9hr 1min, 15hr 2min	0.002	0	0
3	3 Outputs 3 Inputs distribution (%): 20.43, 19.85, 59.72 delays: 3hr 3min, 9hr 8min, 15hr 4min	0.002	0	0

Table 7.5: CryptoMixer Trials

Trial 1

In Trial 1, one SegWit output address, O_1 was specified. Additionally, the mixing service fee and delay were set to 0.5060% and 1 hour and 15 minutes respectively. This qualified for a Standard security level. The service provided a five character alphanumeric CryptoMixer code and one legacy format input address with its corresponding letter of guarantee. One transaction of 0.001 BTC was sent to this address. The service's calculator stated that the output would be 0.00049494 BTC. However, After one confirmation the service displayed an error stating the "amount is less than required." The error did not disappear and the number of confirmations on our original input did not update after the first detected confirmation. Assumin, the service expected an additional payment of 0.00049494 BTC, we generated a second input address and executed another input transaction. However, this was ignored by the service. After 1 hour and 21 minutes of the first input, O_1 received 0.00049494 BTC with a network fee 0.00007749 BTC. The overall mixing fee for this interaction was 0.00050506 BTC.

Trial 2

In Trial 2, the CryptoMixer code from Trial 1 was used and three legacy output addresses O_1, O_2, and O_3 were specified. Delay and distribution for these output addresses was 3 hours and 7 minutes with 20.05%, 9 hours and 1 minute with 19.96%, and 15 hours and 2 minutes with 59.99% respectively. The mixing fee was set to 1.0176%. These parameters qualified the interaction for a Silver security level. The service provided the same CryptoMixer code from Trial 1 and we manually generated four legacy format input addresses, I_1, I_2, I_3, and I_4. The letter of guarantee for each of these addresses was successfully downloaded. I_1, I_2, I_3, and I_4 were sent 0.001

BTC, 0.001 BTC, 0.0005 BTC, and 0.001 BTC respectively. The service's calculator stated that 0.00039386 BTC, 0.00039209 BTC, and 0.001178 BTC would be deposited to O_1, O_2, and O_3. However, no outputs were received.

Trial 3

In Trial 3, no CryptoMixer code was used and three legacy output addresses, O_1, O_2, and O_3, were specified. Delay and distribution for these output addresses was 3 hours and 3 minutes with 20.43%, 9 hours and 8 minute with 19.85%, and 15 hours and 4 minutes with 59.72% respectively. The mixing service fee was set to 1.0820%. These parameters qualified this trial for Silver security level. We received a new five character CryptoMixer code and manually generated two legacy format input addresses I_1 and I_2. The letter of guarantee for each of these addresses was successfully downloaded. I_1 and I_2 were each sent 0.001 BTC. However, we received the same error from Trial 1 stating "amount is less than required." For both inputs the service stated 0.00051082 BTC was pending. Thus, two transactions of 0.0005 BTC and 0.00001082 BTC were sent each input I_1 and I_2. However, the service did not identify these transactions and no outputs were received by O_1, O_2, and O_3.

7.3.5 Sudoku Wallet

Table 7.6 displays the obfuscation parameters, input, output, output network fees, and mixer fees associated with each Sudoku Wallet trial. Transaction IDs, input network fees, input addresses, output addresses, timestamps, and wallet keys have been omitted for simplicity. Mixing fees for each trial were inconsistent and unverifiable with any CoinJoin transactions.

Trial	Obfuscation Parameters	Input (BTC)	Output (BTC)	Output Network Fees (BTC)	Mixer Fees (BTC)
1	sweep	0.001	0.00087261	0.00012739	0
2	sweep	0.002	0.00171162	0.00024839	0.00003999
3	sweep	0.003	0.0000769	0.00022310	0.0027

Table 7.6: Sudoku Wallet Trials

Trial 1

In Trial 1, Sudoku Wallet provided a 25 character alphanumeric wallet key. The service then presented an input address with its corresponding private key. We sent one transaction of 0.001 BTC to this input address. After the service detected three confirmations on this input, we were able to view two output addresses funded with 0.00059025 BTC and 0.00040975 BTC along with their private keys. These funds were then swept to our SegWit wallet through an on-blockchain transaction. The network fee for this transaction was 0.00012739 BTC and 0.00087261 BTC was the final output. The overall mixing fee for this interaction was 0 BTC.

Trial 2

In Trial 2, Sudoku Wallet provided a new 25 character alphanumeric wallet key. The service presented an input address with its corresponding private key. We sent one transaction of 0.002 BTC to this address. After three confirmations, we were presented three output addresses with 0.00066667 BTC, 0.00064667 BTC, and 0.00064667 BTC. These funds were then swept to our legacy wallet through an on-blockchain transaction. The network fee for this transaction was 0.00024839 BTC and 0.00171162 BTC was the final output. The overall mixing fee for this interaction was 0.00003999 BTC.

Trial 3

In Trial 3, we received a new 25 character alphanumeric wallet key. We sent one transaction of 0.003 BTC to the given input address. After three confirmations, we were presented three output addresses of 0.0001 BTC each with corresponding private keys. These funds were swept to our SegWit wallet through an on-blockchain transaction. The network fee for this transaction was 0.00022310 BTC and 0.0000769 BTC was the final output. The overall mixing fee for this interaction was 0.0027 BTC.

7.4 Implementation Analysis

In this section, we use the data gathered in Chapter 6 regarding current public mixers and our experiments (discussed in Section 7.3) to identify the adoption of academically proposed solutions in ChipMixer, MixTum, Bitcoin Mixer, CryptoMixer, and Sudoku Wallet. The proposed solutions were discussed in Chapter 5 and include CoinShuffle, CoinParty, Xim, Obscuro, Mixcoin, Blindcoin, and TumbleBit.

Table 7.7 outlines which mixing services include key characteristics of proposed solutions in their implementation. The characteristics selected include CoinJoin, shuffling of output addresses in one transaction, multisignature escrows, TEXT field use to share data, signed warranties, blinding, and off-blockchain transactions. Each of these characteristics are used in at least one of the proposed solutions.

	CoinJoin	Output Address Shuffling	Multisig Escrow	TEXT Field Use	Remote Attestation	Signed Warranty	Blinding	Off-Blockchain Txns
ChipMixer	✓	✗	✗	✗	✗	✗	✗	✓
MixTum	✗	✗	✗	✗	✗	✓	✗	✗
Bitcoin Mixer	✗	✗	✗	✗	✗	✗	✗	✗
CryptoMixer	✗	✗	✗	✗	✗	✓	✗	✗
Sudoku Wallet	✗	✗	✗	✗	✗	✗	✗	✓

Table 7.7: Academically Proposed Implementation Adoption

ChipMixer

Through tracing our input transactions and outputs received by ChipMixer, we identified that funds sent to the service are routinely involved in the creation of chips ranging from 0.001 BTC to 8.192 BTC. For example, our Trial 1 input of 0.001 BTC was involved in the creation of five chips of 8.192 BTC. The creation involves a CoinJoin transaction with UTXOs sent to ChipMixer by users as its input set. The output is a set of chips of a uniform size. Unlike CoinShuffle, this CoinJoin is solely created with funds available in ChipMixer's wallet. Thus, the need for multiple signatures and shuffling of output addresses is eliminated.

ChipMixer incorporates off-blockchain transactions by giving users the option to split, merge, bet, commonize, and donate their given chips. These options have an impact on the amount and distribution of the mix without executing multiple on-blockchain transactions. The withdrawal of funds via importing private keys is also

done off-blockchain. Thus, a complete ChipMixer mixing interaction can be done with only one on-blockchain input transaction. This is comparable to TumbleBit and its incorporation of off-blockchain puzzles to send Bitcoin between two users.

ChipMixer claims to provide a signed receipt on withdrawal of chips. Although the service was unable to provide this receipt in all three trials, we do not believe it is comparable to the signed warranties produced in Mixcoin and Blindcoin. While ChipMixer's signed receipt aims to prove the origin of output funds, Mixcoin and Blindcoin's signed warranty outlines the terms of the mix before any input or output.

Overall, our analysis did not provide any evidence that ChipMixer implements signed warranties, blinding, remote attestation, output address shuffling, or multisignature escrow addresses.

MixTum

MixTum offers a PGP signed letter of guarantee before any inputs to the service. The letters for all three trials included the generated input address, the output address(es), the maximum mixing time, the deadline for users to send their input by, and the maximum service fee. This guarantee can be compared to the signed warranty provided in Mixcoin which includes the value to be mixed, the deadline for the input to be sent, the deadline for the service to return funds, the output address, the mixing fee rate, a nonce, and the number of confirmations required on the input. Mixcoin's protocol requires that users create the terms of the mix and provide them to the service. In the case of MixTum, the service creates the majority of the terms including the fee and deadline to return funds. The user is only able to set the output addresses. Overall, the PGP signed letter of guarantee from MixTum provides enough information to identify a breach in protocol and holds the service accountable.

We did not identify any evidence that MixTum incorporates CoinJoin, output ad-

dress shuffling, multisignature escrow addresses, TEXT field use, remote attestation, blinding, or off-blockchain transactions.

Bitcoin Mixer

Through our analysis and experiments with Bitcoin Mixer, we identified that the service does not implement any of the proposed mixing solutions found in CoinShuffle, CoinParty, Xim, Obscuro, Mixcoin, Blindcoin, or TumbleBit. The service does not implement CoinJoin transactions or shuffle output addresses of multiple users in one transaction. In addition, Bitcoin Mixer does not implement multisignature escrow addresses, TEXT fields in transactions, remote attestation, a signed warranty, blinding, or off-blockchain transactions.

CryptoMixer

CryptoMixer provides a signed letter of guarantee along with each input address. Unlike MixTum, CryptoMixer's letter of guarantee is signed using its Bitcoin private key. This letter provides confirmation of the origin of the input address, distribution of funds to each output address, delay for each output address, deadline for inputs, minimum and maximum input allowed, and mixing fee. This guarantee can be compared to the signed warranty provided in Mixcoin. In this case, the user specifies output addresses, delays, distributions, and the fees. Thus, CryptoMixer's letter of guarantee ensures accountability and can be used against the service in case of a breach of protocol.

Overall, the signed warranty was the only academically proposed solution adopted by CryptoMixer. We did not identify any evidence of CoinJoin, output address shuffling, multisignature escrow addresses, TEXT field use, remote attestation, blinding, or off-blockchain transactions.

Sudoku Wallet

Sudoku Wallet claims to provide funds from pre-mixed CoinJoin transactions. Blockchain analysis in all three trials revealed that inputs were not involved in uniform output CoinJoin transactions after being sent to the service. Additionally, outputs had not been involved in uniform output CoinJoin interactions in recent history. Thus, we do not believe the service uses CoinJoin transactions like CoinShuffle. However, Sudoku Wallet does make use of off-blockchain transactions on withdrawal. Like ChipMixer, the use of private keys as outputs ensures that outputs are not detectable on the blockchain.

Overall, we did not identify any evidence of CoinJoin transactions, output address shuffling, multisignature escrow addresses, TEXT field use, remote attestation, signed warranties, or blinding.

	Coin Theft Prevention	Relationship Anonymity	Participation Guarantee	Large Mixing Set Guarantee	Join-then-abort Resistance	Minimum On-Chain Txns
CoinJoin	✓	✗	✓	small set	✗	1
CoinShuffle	✓	✓	✓	small set	✗	1
CoinParty	✓	✓	✓	✓	✗	2
Xim	✓	✗	✓	small set	✓	7
Mixcoin	✗	✗	✗	✗	✓	2
Blindcoin	✗	✓	✗	✗	✓	2
TumbleBit	✓	✓	✗	✗	✓	4
Obscuro	✓	✓	✓	✓	✓	2

Table 7.8: Obscuro's Comparison of Proposed Mixers

7.5 Security Analysis

In this section, we build upon Obscuro's security analysis performed on CoinJoin, CoinShuffle, CoinParty, Xim, Mixcoin, Blindcoin, and TumbleBit in [13]. We expand on their academically proposed Bitcoin mixer comparison (seen in Table 7.8) by performing the same analysis on the five mixing services included in this study. Table 7.9 displays the results of this analysis. We compare the mixers solely based on their resistance to the threats outlined in Chapter 3 Section 3.2.

	Coin Theft Prevention	Relationship Anonymity	Participation Guarantee	Large Mixing Set Guarantee	Join-then-abort Resistance	Minimum On-Chain Txns
ChipMixer	✗	✗	✗	✗	✓	1
MixTum	✗	✗	✗	✗	✓	2
Bitcoin Mixer	✗	✗	✗	✗	✓	2
CryptoMixer	✗	✗	✗	✓	✓	2
Sudoku Wallet	✗	✗	✗	✗	✓	1

Table 7.9: Security Comparison of Five Public Mixing Services

Coin Theft

The five mixers in the study do not have protections in place against coin theft. ChipMixer, Bitcoin Mixer, and Sudoku Wallet provide no proof of origin for the provided input address, making it possible for adversaries or malicious mixer operators to steal funds. On the other hand, MixTum and CryptoMixer provide signed letters of guarantee, making it difficult for an attacker to inject their own address. However, the letter of guarantee is ineffective against malicious mixer operators. Although it sets accountability, users can still have their funds stolen. Mixcoin and Blindcoin suffer from the same protections against a malicious operator. Thus, six out of eight mixing services in Obscuro's analysis implement protections against coin theft. For example, CoinJoin, CoinShuffle, and TumbleBit use multisig addresses to ensure all parties are involved in the movement of funds.

ChipMixer and Sudoku Wallet provide private keys as outputs. Importing these keys to a wallet may be appealing because of its off-blockchain nature, however it leaves users susceptible to coin theft. The mixing service could still have access to the private key and sweep the funds to a separate address without user permission.

Relationship Anonymity

Relationship anonymity is not guaranteed in any of five mixing services studied. Malicious mixing operators can directly learn the permutation between inputs and outputs. Additionally, all five services store or log session data for a limited amount of time, providing a tempting target for adversaries. In comparison, five out of eight proposed mixing services in Table 7.8 provide a method to ensure relationship anonymity. For example, CoinParty and CoinShuffle use output address shuffling while Blindcoin and TumbleBit use blinding.

Participation Guarantee

All five public mixers lack resistance against dropping participants. This is common in protocols that involve a mixer operator who can control the mixer's worldview. In comparison, five out of eight protocols studied in Obscuro's analysis guarantee participation for all users. The only centralized protocol included in these five is Obscuro. In its implementation, selective dropping of participants results in a DoS attack because of the protocols dependence on public bulletin boards.

Large Mixing Set Guarantee

Of all five services, CryptoMixer was the only to guarantee a large mixing set size. For public mixing services, we view the mixing set to be the pool of UTXOs that the mixing service controls. To guarantee a large mixing set, CryptoMixer provided

reputable Bitcointalk users with access to a list of their owned addresses along with signatures for each. The users were able to confirm that the service had nearly 2000 BTC in their pool. In comparison, two out of eight proposed services provide a guarantee of a large mixing set. For example, OBSCURO refunds user inputs when a minimum number of participants is not reached. Mixcoin, Blindcoin, and TumbleBit do not include an agreement of a minimum mixing set size in their centralized protocols. In decentralized protocols like CoinJoin, CoinShuffle, and CoinParty, users are guaranteed a small set due to the communication overhead and long wait times with larger anonymity sets.

Join-then-abort Resistance

All five public mixing services provide resistance against join-then-abort attacks. This is common with all centralized mixing services. Users are unable to abort the mixing protocol after funds have been sent to the given input address. In comparison, five out of eight proposed protocols also provide resistance against this attack. In CoinJoin implementations, like CoinShuffle, users are able to disrupt the mix by disapproving of the final transaction.

Minimum On-Chain Transactions

The number of on-blockchain transactions for the five mixers in this study is similar to the proposed protocols in OBSCURO's analysis. Aside from Xim, which requires three ads on-blockchain before the four transactions in Barber's Fair Exchange, and TumbleBit, which uses two escrow channels, the proposed protocols require one to two transactions.

Chapter 8

DISCUSSION

8.1 Results

Overall, the implementation and security analysis show a clear disconnect between publicly available mixers and academically proposed solutions. Key characteristics of these solutions have not been widely adopted by today's most trusted Bitcoin mixing services. We found that all five public mixing services do not use output address shuffling, multisignature escrow addresses, TEXT fields in transactions, remote attestation, or blinding. The only three characteristics adopted include CoinJoin, signed warranties, and off-blockchain transactions.

All five mixers performed poorly in the security analysis. The lack of prevention against coin theft, permutation leaks, and dropping of participants in public services shows that current mixing services are not concerned with heightening security. Rather, most services seem to be focused on providing their users with the illusion of control over their mix. Centralized mixers displayed complete resistance against join-and-abort attacks, unlike the proposed decentralized solutions. CryptoMixer also leverages Bitcointalk to guarantee a minimum mixing set size.

To gain credibility and trust from their user's, today's mixers must employ a combination of key characteristics provided by proposed solutions. Public mixing services should advertise the use of proven solutions from academic literature, use trusted third-party remote attestation services, provide signed letters of guarantee, and adopt open-source practices. Mixers should also aim to leverage the solidified trust users have with reputable members of Bitcointalk and actively engage with

their participants. Output addresses can be encrypted with the mixer's public key and included in the TEXT field of input transactions to lessen the threat of selective dropping of participants. Although it would result in higher network fees, mixers should identify a minimum mixing set size and ensure outputs include multiple users rather than one-to-one transactions. The use of private keys as outputs must be eliminated from services to ensure safety against coin theft.

Our experiments on ChipMixer, MixTum, CryptoMixer, and Sudoku Wallet revealed additional, interesting behavior associated with each service. We believe these behaviors could be analyzed in a long-term study to learn more about underlying service implementation.

ChipMixer

Chip Generation

As discussed in the Implementation Analysis from Chapter 7 Section 7.4, ChipMixer generates new chips by creating CoinJoin transactions with uniform output chip values ranging from 0.001 BTC to 8.192 BTC. The set of inputs for these chip generation transactions is comprised of UTXOs adding up to the exact amount necessary to create the specified number of chips. In turn, chip generation does not include change transactions in its output. We identified this pattern in all four of our input transactions with ChipMixer. Additionally, we were able to trace these created chips to identify outputs to other users. It is possible that a large number of inputs could be sent to ChipMixer to gain a better understanding of their pool of chips. Table 8.1 provides some example chip generation transactions.

Chip Size (BTC)	Transaction ID
8.192	a3098c6d8961c6674ad4590a3b50c2ca213d833b49a2c774ce5248cabed135a2
0.256	5b7bfd2f60d6058344cdb59fe64d3c1402378c3489210de2a6d18a34e1c0bd5b
4.096	66c3429e06f5e8732717bbeba30d7df28f81a785c4018ad0a269959bbd37bce6

Table 8.1: Example Chip Generation Transactions

Session Token Expiration

Although ChipMixer claims logs and session information is deleted in seven days, we found that our session tokens for all three trials were still valid after 16 days. This could indicate that deletion of logs and session tokens is manually done by the mixer operator.

Illusion of Control

ChipMixer incorporates various features that focus on providing users a sense of control over their funds. However, off-blockchain transactions like split and merge essentially have no impact on the chips available in ChipMixer's pool. In addition, voucher codes carry no value outside of the service.

MixTum

Jambler.io

MixTum is built upon Jambler.io, a mixing platform that provides the source code to start a mixer. The letter of guarantee and the input address are generated from Jambler.io and the platform pays MixTum a commission on completion of each mixing interaction. Jambler.io claims to obtain funds from cryptocurrency exchanges and use a scoring algorithm to only mix with 'pure' funds. There is currently no method to

verify the effectiveness of this algorithm.

Minimum Input Amount and Fees

MixTum's typical mixing fees are up to 5% + 0.00015 BTC. However, in Trial 1 when an input of the minimum 0.001 BTC was sent to the service, we received an output of 0.001 BTC. The transaction fee on this output was 0.00024227 BTC. Thus, the service did not charge a mixing fee and lost money. This was tested twice with the same result. We believe the service does not charge mixing fees on inputs of the minimum.

CryptoMixer

Poor Implementation

CryptoMixer returned an output in Trial 1 even though the service stated that the input amount was less than required. In Trial 2, we identified that the service does not accept transactions less than the minimum 0.001 BTC. However, CryptoMixer's calculator still recognizes inputs less than the minimum and calculates accordingly. We believe CryptoMixer treats all inputs to a session as donations if an input less than the minimum is detected before an output transaction is scheduled. The first three inputs for Trial 2 were 0.001 BTC, 0.001 BTC, and 0.0005 BTC. All three received their first confirmation at the same time. We believe CryptoMixer recognized that one of these inputs was less than 0.001 BTC and treated all inputs as donations as a result. More testing will need to be done to confirm this assumption. In Trial 3, we learned that input addresses do not accept more than one transaction. Our second transactions were not recognized and CryptoMixer did not send an output. Overall, CryptoMixer has poor implementation without proper documentation.

Input Addresses

On the presentation of the input address, Sudoku Wallet also provides a corresponding private key. We believe this is done to give users the illusion that they still have access to their funds. However, in all three of our trials, Sudoku Wallet moved the funds associated with the input address before we obtained our output. For example, in Trial 1, we swept our outputs at 12:51 AM, however the input address funds had been moved to a separate address at 12:33 AM. This shows how simple coin theft is when mixers output private keys.

Inconsistent Fees

Sudoku Wallet's mixing fees are described as 0.5% to 1% (randomized) plus the CoinJoin fee (number of output addresses x transaction fee). However, mixing fees were inconsistent in all three trials. We were not able to identify any CoinJoin transactions to calculate the fees in each output's blockchain history. Thus, more transactions will need to be executed to understand the mixing fees.

Error on loadwallet()

During Trial 3, the provided wallet key was entered onto the Sudoku Wallet website. We received an error stating that the Bitcoin Client function loadwallet() verification failed. This error reveals that Sudoku Wallet creates a new wallet for each user to keep track of balances. This is the only implementation of separate wallet creation. Although Sudoku Wallet states that logs are not maintained, this is similar to logging transaction data for each participant.

8.2 Future Work

Bitcoin mixers are constantly evolving with new, unique techniques to obfuscate user funds. To build upon this work, a larger subset of these services can be tested with a high volume of transactions. In our case, the chosen number of trials proved to be a major limitation in identifying the reason behind certain mixer behaviors. A longer term analysis with more transactions could lead to better understanding of implementation and identification of mixer-specific behaviors. The study of these behaviors could also lead to the identification of new attacks towards these services. For example, chip generation transactions could be detected on the blockchain to identify the amount of funds in ChipMixer's reserve. We have outlined some behaviors we believe are worthy of exploration in the previous section.

Chapter 9

CONCLUSION

The Bitcoin mixing ecosystem attracts a wide range of users, many of whom simply wish to heighten their anonymity. The association of scams and poor implementation by these services has led to the proposal of secure protocols in academic literature. These proposed solutions provide methods to ensure accountability for mixing services and secure communication between participants without the leakage of input and output permutations. Through real world mixer interactions, we identified that there exists a disconnect in both implementation and resistance to common mixing threats between today's public mixing services and academically proposed solutions. For example, all five mixers analyzed in this study were found to be susceptible to coin theft while the majority of academically proposed solutions prevent theft from both malicious mixing operators and external attackers. We strongly believe that the disparities identified in this work represent an overall lack of regard for secure implementation. Although mixing services are often associated with criminal activity, the adoption of secure mixing methods could better their reputation and provide a foundation for future Bitcoin mixer research.

REFERENCES

[1] J. Pakki, A. Doupe, and Y. Shoshitaishvili, "Can you mix it? an analysis of bitcoin mixers," Master's thesis, Arizona State University, 12 2018.

[2] "Multi-million euro cryptocurrency laundering service bestmixer.io taken down," May 2019. [Online]. Available: https://www.europol.europa.eu/newsroom/news/multi-million-euro-cryptocurrency-laundering-service-bestmixerio-taken-down

[3] "Bestmixer.io the future of bitcoin mixing! technology is here," 2018. [Online]. Available: https://bitcointalk.org/index.php?topic=3140140.0

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

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[6] S. Delgado-Segura, S. Bakshi, C. Pérez-Solà, J. Litton, A. Pachulski, A. Miller, and B. Bhattacharjee, "Txprobe: Discovering bitcoin's network topology using orphan transactions," in *Financial Cryptography*, 2018.

[7] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: Characterizing payments among men with no names," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 127–140. [Online]. Available: https://doi.org/10.1145/2504730.2504747

[8] J. DuPont and A. C. Squicciarini, "Toward de-anonymizing bitcoin by mapping users location," in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 139–141. [Online]. Available: https://doi.org/10.1145/2699026.2699128

[9] N. Alsalami and B. Zhang, "Sok: A systematic study of anonymity in cryptocurrencies," in *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, Nov 2019, pp. 1–9.

[10] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using p2p network traffic," in *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 2014, pp. 469–485. [Online]. Available: https://doi.org/10.1007/2F978-3-662-45472-5_30

[11] "Protect your privacy," http://bitcoin.org/en/protect-your-privacy. [Online]. Available: https://bitcoin.org/en/protect-your-privacy

[12] "Bitcoin charts & graphs - blockchain." [Online]. Available: https://www.blockchain.com/en/charts

[13] M. Tran, L. Luu, M. Suk Kang, I. Bentov, and P. Saxena, "Obscuro: A Bitcoin Mixer using Trusted Execution Environments," in *ACSAC '18 (Annual Computer Security Applications Conference)*, ser. ACSAC '18, vol. 18. New York, NY, USA: ACM, 2018, pp. 692–701. [Online]. Available: https://dl.acm.org/citation.cfm?id=3274750{\%}0A

[14] M. Möser, R. Böhme, and D. Breuker, "An inquiry into money laundering tools in the bitcoin ecosystem," in *2013 APWG eCrime Researchers Summit*, 2013, pp. 1–14.

[15] L. Novetta, "Survey of bitcoin mixing services: Tracing anonymous bitcoins," McLean, VA, Tech. Rep., September 2015. [Online]. Available: https://www.novetta.com/wp-content/uploads/2015/10/NovettaBiometrics_BitcoinCryptocurrency_WP-W_9182015.pdf

[16] T. de Balthasar and J. Hernandez-Castro, "An analysis of bitcoin laundry services," in *NordSec*, 2017.

[17] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, p. 84–90, Feb. 1981. [Online]. Available: https://doi.org/10.1145/358549.358563

[18] Gregory Maxwell, "CoinJoin: Bitcoin privacy for the real world," 2013. [Online]. Available: https://bitcointalk.org/index.php?topic=279249

[19] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin," Tech. Rep.

[20] J. H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, and K. Wehrle, "CoinParty: Secure multi-party mixing of bitcoins," in *CODASPY 2015 - Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*. ACM, mar 2015, pp. 75–86. [Online]. Available: http://dl.acm.org/citation.cfm?id=2699026.2699100

[21] J. Brickell and V. Shmatikov, "Efficient anonymity-preserving data collection," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 76–85. [Online]. Available: https://doi.org/10.1145/1150402.1150415

[22] G. Bissias, A. P. Ozisik, B. N. Levine, and M. Liberatore, "Sybil-resistant mixing for Bitcoin," in *Proceedings of the ACM Conference on Computer and Communications Security*, ser. WPES '14. ACM, 2014, pp. 149–158. [Online]. Available: http://doi.acm.org/10.1145/2665943.2665955

[23] S. Barber, X. Boyen, E. Shi, and E. Uzun, "Bitter to better — how to make bitcoin a better currency," in *Financial Cryptography and Data Security*, A. D. Keromytis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 399–414.

[24] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Secure multiparty computations on bitcoin," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 443–458.

[25] Anati, S. Gueron, S. P. Johnson, and V. Scarlata, "Innovative technology for cpu based attestation and sealing ittai," 2013.

[26] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," in *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, ser. HASP '13. New York, NY, USA: Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2487726.2488368

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

[28] E. Heilman, L. AlShenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub," in *NDSS*. Internet Society, May 2017.

[29] "Cryptomixer.io fast, secure and reliable bitcoin mixer (since 2016)." [Online]. Available: https://bitcointalk.org/index.php?topic=1484009.msg15350012#msg15350012

[30] "Cryptomixer.io fast, secure and reliable bitcoin mixer (since 2016)." [Online]. Available: https://bitcointalk.org/index.php?topic=1484009.msg15256505#msg15256505

[31] "Cryptomixer.io fast, secure and reliable bitcoin mixer (since 2016)." [Online]. Available: https://bitcointalk.org/index.php?topic=1484009.msg15428183#msg15428183