Université de Montréal

Privacy in Bitcoin through decentralized mixers

par
Olivier Coutu

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

30 Avril, 2014

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

Privacy in Bitcoin through decentralized mixers

présenté par:

Olivier Coutu

a été évalué par un jury composé des personnes suivantes:

Louis Salvail,	président-rapporteur
Alain Tapp,	directeur de recherche
Neil Stewart,	membre du jury

Mémoire accepté le: 31 Décembre 2014

RÉSUMÉ

Dans les crypto-monnaies telles Bitcoin, l'anonymité des utilisateurs peut être compromise de plusieurs façons. Dans ce mémoire, nous effectuons une revue de littérature et une classification des différents protocoles existants pour anonymiser les usagers et analysons leur efficacité. S'appuyant sur certains critères désirables dans de tels protocoles, nous proposons un modèle de mixeur synchrone décentralisé. Nous avons ciblé deux approches qui s'inscrivent dans ce modèle, le plan de transaction et le réseau de transactions, le second étant une contribution originale de ce mémoire. Nous expliquons son fonctionnement puis analysons son efficacité dans le contexte actuel d'utilisation de Bitcoin.

Mots clés: Bitcoin, anonymité, vie privée, mixeur, argent électronique, réseau.

ABSTRACT

In cryptocurrencies such as Bitcoin, the anonymity of the users may be compromised in many ways. In this thesis, we review the literature concerning existing protocols used to increase anonymity by a method called mixing and produce a classification for such protocols. We propose a decentralized synchronous N-to-N mixing model that takes into account many considerations of mixers. We address two frameworks within this model, the transaction blueprint and the network of transactions, the second approach being a new contribution. We explain how it functions and analyse its efficiency in the current Bitcoin ecosystem.

Keywords: Bitcoin, anonymity, privacy, mixer, ecash, cryptocurrency, network.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION

lg	log_2
$H(D)$	Shannon entropy of distribution D
$H_\infty(D)$	Min-entropy of distribution D

LIST OF ABBREVIATIONS

AML	Anti-Money Laundering
btc	One unit of value in Bitcoin, can be divided
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
DOS	Denial of Service
ECDSA	Elliptic Curve Digital Signature Algorithm
FBI	Federal Bureau of Investigation
MPC	Secure Multiparty Computation
NSA	National Security Agency
P2P	Peer-to-Peer
TTP	Trusted Third Party
txout	Transaction output

ACKNOWLEDGMENTS

This thesis would not have been possible without the support of many people. I thank my adviser, Alain Tapp, who accepted to give me his guidance in an unorthodox project. I thank my committee members, Louis Salvail and Neil Stewart, who read this lengthy document and gave me important feedback. I also want to thank my colleagues and professors, especially Jeremy Clark, who allowed me to bounce ideas off them and gave me feedback. Of course, I would like to thank the Bitcoin community for creating and maintaining this amazing cryptocurrency as well as patiently explaining the intricacies of the protocol. My last acknowledgements goes to Catherine, who supported me, offered writing advice and proofread my many drafts.

INTRODUCTION

While computer security has been extensively studied since before the dawn of the Internet, privacy in the digital world has become more relevant in recent years, as individuals may wish to shield their private life from governments and private corporations. This thesis addresses one specific aspect of this phenomenon, that is the lack of privacy in the world of commerce, and analyses the performance in that regard of the Bitcoin cryptocurrency. To motivate this thesis, we initially survey both the positive and negative consequences of privacy, or the lack thereof, with regard to financial transactions.

The Case Against Privacy

Opponents of privacy within payment systems point to a number of problems that may stem from anonymous payment. Elias [37] argues that anonymous payments may act as a Ring of Gyges [1] on the Internet and allow nefarious users to commit crimes in an anonymous manner. The nefarious application that is arguably the most cited [39, 74, 78] in this context is money laundering, that is the concealment of the source of a large amount of money. This facilitates tax evasion, although some argue that it would only democratize it [38]. Additionally, criminals of all kinds may demand to be paid in some form of anonymous payment such as Bitcoin in order to avoid being identified. Widespread usage of an anonymous payment system for this purpose might force law enforcement agencies to adapt their means of identifying criminals.

Although published before the rise to prominence of Bitcoin, the *Financial Action Task Force*'s *Money Laundering Using New Payment Methods* [39] report examines the use of various anonymous or semi-anonymous payment methods and their usage for money laundering. The report illustrates not only the variety of ways that money can be laundered using these payment methods but also the various ways that criminals are caught even though the payments are anonymous.

Other alarming possible consequences of anonymous payment systems relate to ran-

1. In *The Republic*, Plato posits that no man could possibly be virtuous while possessing a ring that had the power to make the wearer invisible.

soms. Ransomware [48] such as *Cryptolocker* is a type of malware that has been used by criminals to encrypt victim's hard drives and offer to decrypt them in exchange for payment. Payment systems such as Bitcoin enable the attackers to receive the funds securely, anonymously and irreversibly, which would not have been possible with traditional payment methods. For instance, an anonymous poster claiming to own 2012 American presidential candidate Mitt Romney's tax return asked for a Bitcoin ransom [29] in order not to publicize the document, and one might imagine how not only information but also objects and even people might be ransomed through Bitcoin.

The Federal Bureau of Investigation (FBI) has taken notice of Bitcoin and assessed its use for illegal activities [74]. Among its conclusions, this US government organisation believes that cyber criminals would use Bitcoin as a payment option and possibly use it to launder money or make donations to illicit groups. The FBI also believes that law enforcement still have the ability to identify malicious actors if these actors convert their coins into fiat currency through enforcement of anti-money laundering laws concerning third-party Bitcoin services that exchange money for Bitcoins.

The Case For Privacy

In 1975, Foucault expanded Bentham's idea of the Panopticon in his book *Discipline and Punish* [40]. The Panopticon represents the idea that a citizen can be placed in a situation where they might constantly be spied upon without them knowing when it is the case. In 2013, National Security Agency contractor Edward Snowden leaked documents to *The Guardian*'s Glenn Greenwald and Laura Poitras that proved that governments around the world were collecting massive amounts of data on users and decrypting communications previously thought secure with limited or inexistant judicial oversight [3]. This massive collection of private information enables governments to potentially spy on both their own citizens and people abroad in a manner that would have been unimaginable even in Orwell's *1984* [76]. It also is a realization of Foucault's Panopticon, as one can never know when they are being spied upon.

One specific aspect of this breach of privacy comes in the form of financial trans-

actions, that is, information on who is transferring funds to whom and for what reason. This transfer of funds might happen when a product is bought, when money is deposited at the bank or when a donation is made. There are many aspects of money transactions that can leak information that users might prefer to keep private. For example, one might not want one's colleagues and friends to learn one's salary or where one spends their money. While this demand for privacy is often with respect to other people that one may know personally, some people demand that corporations or even the state or the bank not know how they are spending their money.

The lack of privacy with respect to corporations may have unexpected consequences for the users of the services they offer. Unrestrained access to a customer's purchasing history or financial data through the use of fidelity cards or credit cards enables these companies to paint a precise portrait of a customer's habits. One controversial consequence of this data collection might be some form of data redlining, that is discrimination between customers based on the data collected about them, e.g. denying a loan [52]. Another remarkable example of privacy breach happened when the marketing department of the Target stores was able to correctly guess that a young woman was pregnant by analysing her purchasing behavior [35]. This enabled the marketing department to send her coupons by mail for diapers and other baby items before the baby's birth. However, receiving these coupons forced the young woman to admit to her parents that she was pregnant before she wished to do so. This story illustrates the very real consequences of the lack of privacy that are already upon us today.

This transaction data may also be used by the state to collect information on individuals. One of the Snowden leaks [89] has shown that the NSA is already collecting financial information from major credit card companies with the avowed objective of "[collecting, parsing and ingesting] transactional data for priority credit card associations, focusing on priority geographic regions." Furthermore, donors and supporters of organizations that are deemed against the interest of the state may be harassed [43]. This enables a *1984*-esque social control and repression of political enemies by the state. If it is acceptable in so-called liberal democracies to detain people with connections to whistleblowers [4], one can only imagine what an authoritarian government with an all-

seeing eye may do.

Finally, we consider that the "illicit groups" that the FBI refers to may include groups that are considered subversive and do not align with the interests of some governments but whose actions are lauded by others. One such example is the whistle-blowing website *Wikileaks*. When governments forced major credit card companies to cease accepting payments for Wikileaks donations [6], the website turned to Bitcoin to enable donations that would not be censored by governments. While the way that Wikileaks handled the anonymity of the donations might be criticized [79], anonymous donations might be necessary to avoid the kind of repression that has come with being associated with the whistle-blowing organisation [43].

Similarly and quite prominently, *The Silk Road*[2] is a website that sells various, generally illicit goods and accepts only bitcoins. The website is only accessible through Tor to conceal IP addresses and relies on Bitcoin's anonymity to protect both the merchant's and customer's identities. While few condone some of the services offered on the website, such as assassinations, many libertarians involved in Bitcoin rejoice at the idea of buying drugs from sellers who can maintain a reputation on the website. Eventually, one might imagine such a market where goods that are not illicit can be b ought on the internet by people who are simply concerned about the privacy-breaching measures described above.

Although we realize that the downsides of anonymous payments are present, we believe that the increase in privacy not only acceptable but necessary, which is why we wish to encourage the development of more anonymous payment methods. Furthermore, we believe that Bitcoin's possible anonymity should be publicized and cherished accordingly.

Plan

This thesis is divided as follows. In chapter 1, we describe e-cash and some implementations of the concept. In chapter 2, we introduce Bitcoin and explain how it works

2. This website cannot is only accessible through Tor and its location changes over time.

and how it compares to traditional e-cash. Chapter 3 exposes the difficulties in being anonymous while using Bitcoin and defines our threat model. Chapter 4 surveys different ways that users mix their coins to become more anonymous. Chapter 5 describes one approach to mixing that has been explored before while in chapter 6 we propose a new framework to mix bitcoins. We conclude with some final remarks and avenues for future research.

CHAPTER 1

E-CASH

Before addressing the Bitcoin protocol *per se*, we briefly review the state of research on various means of anonymous financial transactions.

1.1 Cash and Modern Banking

Cash tends to be what people think of first for anonymous financial transactions. Upon closer inspection however, cash is not truly untraceable as in most currencies, each paper unit has a serial number that can be traced. Some collaborative [1] or law enforcement [60] efforts attempt to track or follow cash. Nevertheless, effectively tracing a large number of units to produce a complete map of transactions is considered impossible in most situations.

On the other hand, modern banking and the use of credit cards, debit cards, cheques and wire transfers is built to be traceable. Every time money changes hands this way, the bank knows the sender, the receiver, the amount and the time of the transaction. This information is necessary to the functioning of the system as the bank must know at all times the balance of its users' accounts to decide if transactions should be accepted or not.

The collection of this information represents a breach of the bank user's privacy. Furthermore, a bank can merge this data with the data provided by the bank users when opening an account to assemble a precise portrait of the users and their real-world habits, again compromising their privacy. These concerns motivate the study of e-cash, whose objective is to make all these convenient electronic transactions possible without violating users' privacy [80].

1. For example http://www.trackmycash.com/.

1.2 Setting

Electronic money (or e-cash) was first introduced by Chaum [26]. The goal of e-cash is to enable private electronic banking transactions between users in that the bank needs not know the identities of the users taking part in a transaction. An e-cash protocol involves three types of entities that exchange uniform value denominations called *e-coins* or simply *coins*.

- **Users** want to spend coins anonymously. They have accounts in banks with balances in other currencies.
- **Merchants** want to accept payments in e-cash that they can later redeem for another currency at the bank.
- **Banks** enable users to withdraw money and merchants to deposit it. It credits the accounts of both parties appropriately.

While a merchant might also be a user, we consider these roles differently as most protocols use coins that are not transferable without passing through the bank.

1.3 Security

Currencies, whether virtual or real, have a certain set of desired properties [58, 66]. A proper currency should be difficult to forge or duplicate. This presents a challenge for e-cash as the currency is just data that is easily copied from one device to another. Measures must be taken so that it is impossible for a single coin to be spent in two transactions in such a way that both transactions are simultaneously considered valid. Such an occurrence is called a *double spend*.

Different measures are taken by e-cash schemes to detect and prevent double spending. When a scheme is online [2], a double-spend can usually be immediately detected and blocked, as the merchant will not accept the coin. For offline schemes, the detection happens after the purchase and appropriate measures are taken by the bank to identify and punish the offending users. While most proposed e-cash protocols in the

2. A scheme is called *online* if the merchant and the bank are always communicating during the *Spend* phase of the protocol.

literature [21, 24, 28] have provided offline solutions, Bitcoin uses online verification to prevent double-spending. Finally, Aaronson [1, 2] proposes a protocol that would use the no-cloning theorem from quantum mechanics to guarantee that the money will not be copied, which enables his schemes to simply ignore double-spending.

1.4 Overview

While we do not do provide a complete description of any particular e-cash scheme, we describe an general overview that most e-cash schemes follow that is based on Chaum's first protocol with Fiat and Naor [28].

The idea of the protocol is the following. First, Alice creates an account at the bank where she is given an identity and a secret key. At this point, Alice's identity is known to the bank. When she withdraws funds, the bank blind-signs [3] a coin that acts as a prepaid cheque and withdraws the funds from Alice's account. Alice can then send the coin to a merchant, who can verify its validity since it is signed by the bank. When the merchant cashes the cheque at the bank, the bank accepts the cheque as it has signed it before but cannot tell it came from Alice. We now describe a more formal protocol.

In the first step, Alice creates an account at the bank where she generates public key pk_U and secret key sk_U and deposits some funds. The bank knows pk_U and each coin that Alice withdraws will contain some information about pk_U. Alice's withdrawal of a coin is illustrated in Algorithm 1. This protocol is *online* since it requires interaction between Alice and the bank.

3. A blind signature is a signature where the bank signs a message without knowing the exact content of the message but while knowing some information about the message. In this case, the bank knows that the coin is well-formed. For more information, see [28].

Algorithm 1: Withdrawal

1 Alice identifies herself to the bank with pk_U and authenticates by using sk_U to sign a random string given by the bank;

2 Alice chooses $2k$ random values $(a_0, ..., a_{2k})$ and produces $h(a_i)$ and $h(a_i \oplus pk_U)$ $\forall i$ where h is a collision-resistant hash function [a]. These hash values act as commitments [b] to values $(a_0, ..., a_{2k})$;

3 Alice gives the bank all of the previous hash values, thus committing to $(a_0, ..., a_{2k})$;

4 The bank asks Alice to reveal a_i for k values of i that the bank chooses. It then verifies that the committed hashes correspond to the values given by Alice for both values of each pair;

5 If all of the k opened commitments are valid, the bank knows that Alice can only have cheated with probability $O(\frac{1}{2^{k-1}})$ for each commitment pair that it has not opened. If even one of the commitments that the bank opens is invalid, the bank aborts the protocol;

6 The bank blind-signs the k remaining pairs to produce σ_i for each remaining commitment. These k hash pairs constitute k messages m_i. With their k signatures σ_i the set of k pairs $(m_i, \sigma_i) \forall i$ constitutes a coin;

7 Alice's account is debited by the bank;

a. A collision-resistant hash function h is a hash function where the probability that a probabilistic polynomial-time bounded adversary knowing h can find x and x' such that $x \neq x'$ and $h(x) = h(x')$ is negligible. A formal definition can be found in [56, p. 8].

b. A commitment can be thought of in the following way. Alice puts a message in a safe, locks the safe and gives it to Bob. At a later time, Alice may send the key to Bob so that Bob may discover the message. For more information on commitments, see [84].

The protocol that Alice uses to spend a coin is illustrated in Algorithm 2. The bank does not need to be involved in this step, making it *offline*.

Algorithm 2: Spending

1 Alice shows the merchant the coin composed of k messages and signatures (m_i, σ_i);
2 The merchant uses cut-and-choose to verify that the messages and signatures are properly formed. This means that for each message, the merchant asks Alice to either reveal a_i or $a_i \oplus pk_U$;
3 If all of the commitments revealed are valid and the messages are correctly signed, the merchant accepts the coin and keeps a record of the transaction;

Finally, the merchant deposits the coin at the bank in Algorithm 3.

Algorithm 3: Deposit

1 The merchant brings the transaction log to the bank;
2 If the coin is valid and was never spent, the bank credits the merchant's account and keeps a copy of the transaction log;
3 If the coin is invalid, the merchant should not have accepted it and the deposit is refused;
4 If the coin is valid but was already spent, the combined information related to both transactions from this coin enables the bank to obtain pk_U with high probability. Alice is then appropriately punished;

In online protocols [47], the merchant is communicating with the bank during the *Spending* phase and the *Spending* and *Deposit* protocols are combined into a single phase where the merchant knows immediately if the coin has already been spent.

1.5 Identifying Double-Spenders

Suppose Alice double-spends a coin. The first spending transaction to be brought back to the bank is accepted, the merchant's account is debited and the bank keeps a copy of the transaction log. When the second spending transaction is brought to the bank, it will try to identify the double-spending user. To do so, the bank looks at the

logs of both transactions. Each transcript will contain either a_i or $a_i \oplus pk_U$ for k values. If the two transcripts contain answers to a different query for even one i, the bank will be able to extract $pk_U = a_i \oplus a_i \oplus pk_U$ and thus Alice's identity. The probability of each merchant asking exactly the same queries is $\frac{1}{2^k}$ and thus negligible for large k. If this does happen, that bank can reliably infer that both merchants were colluding in asking the same questions, and can refuse to debit the second merchant. Once the offender has been identified, the bank may take appropriate measures to deal with her.

1.6 Analysing Anonymity

We briefly argue why an honest user Alice cannot be linked with her spend transactions while executing this protocol. This means that she can *withdraw* coins at the bank and *spend* them with a merchant without the bank and the merchant being able to know that both transaction involves the same person when the merchant *deposits* the funds at the bank.

1. *Withdrawal*: Alice creates her key pair on her own, thus the bank learns no information on it. When Alice registers at the bank, it learns her public key pk_U. When she comes back to the bank, Alice authenticates with sk_U but never reveals it. It is not necessary for the bank to know any information about Alice. When Alice withdraws coins, the bank learns that Alice, or specifically the person with public key pk_U, has withdrawn coins. The bank signs k messages m_i for Alice without ever seeing them, and only knows that the message is well formed except with a small probability. This lack of identifying information on m_i and σ_i is the consequence of blind-signing.
2. *Spending*: When Alice spends the funds, the merchant may learn a_i or $a_i \oplus pk_U$ $\forall i$ but never learns pk_U. Furthermore, the merchant verifies that the signatures are valid but the signatures reveal no information about pk_U.
3. *Deposit*:When the merchant shares the transcript with the bank, neither party learns Alice's identity as the bank has never seen the m_i or σ_i before.

1.7 Limiting Anonymity

Anti-money laundering (*AML*) laws in the U.S. and around the world are one of the political aspects that limit the adoption of e-cash. A number of adaptations to e-cash have been proposed to comply with such laws and strike a balance between accountability and anonymity. The simplest solution is to use a trusted third party (*TTP*) to make sure AML laws are respected [22, 50, 57, 90]. However, this approach goes against the aim of e-cash since it requires the TTP to trace transactions to particular users. Furthermore, the TTP would need to trace every transaction, which would be very expensive in this model. Another variant [49, 75] that doesn't use a TTP makes user's coins anonymous but linkable, so that excessive spending is detected. Hohenberger [47] introduced the *bounded-anonymity model* where each receiver has a publicly-known limit to the number of coins that may be received within a certain time window. A simpler variant [54] would be to divide time into short time periods and issue at most k coins to a user per time period. One consideration is that this variant might fail to fulfill certain transaction needs or might be vulnerable to Sybil attacks [4].

1.8 E-Cash Adoption

Although proposals for e-cash have been numerous [1, 21, 24, 28], none of the implementations have been commercially successful. In 1990, Chaum founded *Digicash*, an e-cash company that was to apply the scheme he had devised, but the company filed for bankruptcy in 1998[44]. Another cryptography-based anonymous payment system, *e-gold*, suffered a similar fate due to a lack of compliance with AML laws. This lack of adoption explains why e-cash has not been in the spotlight before the invention and relatively massive adoption of Bitcoin.

4. A Sybil attack happens when one party pretends to be many different parties, see [34].

CHAPTER 2

BITCOIN

In 2008, a pseudonymous developer known as Satoshi Nakamoto released a paper [72] that laid the foundation of a digital currency called Bitcoin. This currency represented a major departure from previous digital currencies as it is entirely decentralized, meaning there is no bank to issue the currency or verify the authenticity of the transactions. Bitcoin exists only as a peer-to-peer network where each peer may access the entire history of transactions. While the users themselves are pseudonymous [1], the balance of each user's account is public and every peer can verify that an outgoing transaction is valid and has not already been spent.

In 2009, Nakamoto released the first open source software implementation of his idea [73]. This original client [2] is now known as the Qt-client [3] and is the *default* client to manage Bitcoin wallets [4]. Other clients such as Bitcoin Armory [5] and Electrum [6] now offer different features based on the same protocol. As the userbase grew, more developpers started working with Nakamoto and using Bitcoin and the value of each bitcoin grew from under a penny in 2009 to over 1000 USD in 2013 [18]. The rising value of the currency has brought it to the spotlight in the media and in the eyes of security researchers. Nevertheless, even with the source code under scrutiny, a major breach in the security of the protocol has yet to be found. The anonymity of the protocol however has been called into question by many studies [5, 7, 79, 83].

1. Pseudonimity is the lack of a need to use personally identifying information as users are identified through pseudonyms.
2. A *client* is a program that is used to send and receive bitcoins.
3. `http://sourceforge.net/projects/bitcoin/`
4. A Bitcoin wallet is similar to a bank account.
5. `https://bitcoinarmory.com/`
6. `http://electrum.org/`

2.1 Bitcoin Overview

A bitcoin (abbreviated btc or coin) can be described as a chain of transactions from one user to the next. Each user owns both a public key and a private key that are used for signatures, the public key also serving as a pseudonym. To spend a coin, a user Alice must sign a hash of both the transaction in which she received the coin and the public key of the next owner, thus transferring ownership of the coin [7]. This signature then becomes a part of the coin. Because each transaction references the previous transaction, each coin includes a chain of transactions whose authenticity can be verified back to its original minting.

A newly signed transaction must be sent out to the network to be validated. There is no centralized issuer or verifier in Bitcoin so these duties are handled by *miners*. These particular users chose to dedicate some hardware to solving a difficult computational problem known as a *proof of work*. Every time a miner succeeds in solving a problem, they create what is known as a *block* that consists of a set of transactions that are then marked as validated as well as a reference to the previous block that was accepted by the network. Additionally, creating a block instantly mints a certain number of new coins for the miner to reward them for the effort of solving the problem. This is also how new coins are created as no centralized authority can issue them. This is illustrated in Figure 2.1 [8].

7. One can imagine Alice sending Bob a cheque, and then Bob signing under Alice's signature to enable Catherine to cash the cheque.

8. Image taken from `http://www.builtinchicago.org/blog/beginners-guide-bitcoin-part-one`.

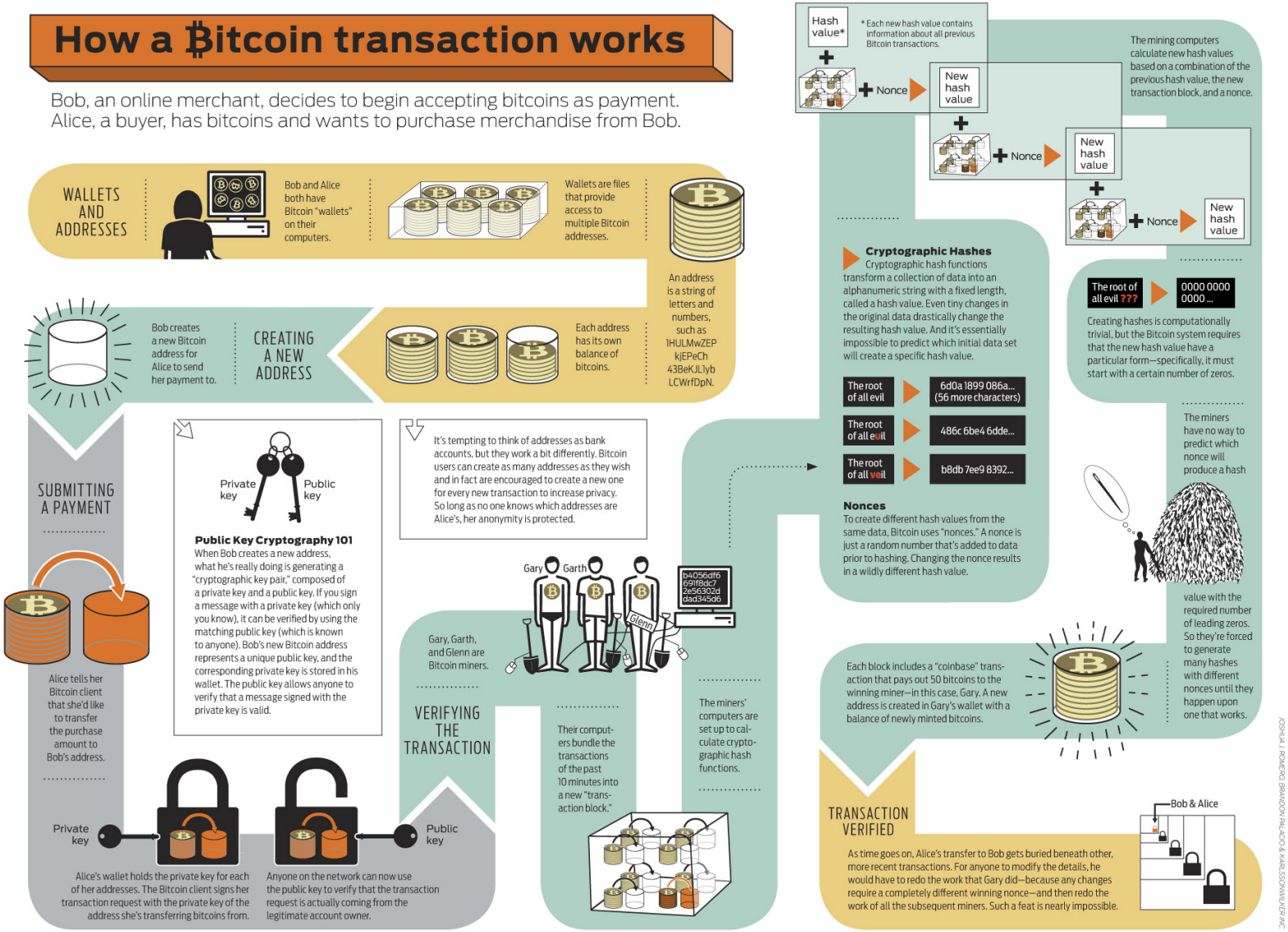

Figure 2.1: Bitcoin network overview

2.1.1 Bitcoin Developement

The core of the Bitcoin protocol follows what was initially proposed and implemented by Satoshi in the first Bitcoin client but there have been minor revisions to the protocol. These modifications influence not only how the clients work internally but how they interact. For example, in 2013, there has been some debate as to whether the maximum size of a block should be increased. If such a change were enacted, all of the Bitcoin clients would have to simultaneously switch to the new version of the protocol. If this were not the case, the clients using different versions of the protocol would not be able to communicate, and the network would split with disastrous results. Major changes in the protocol are called *hardforks* [17], as opposed to *softforks* that are changes in the way the protocol is handled that are compatible with clients that have not enacted these forks. While hardforks have been done in the past [9], they introduce stability risks and require overwhelming consensus. For these reasons, we concentrate in this thesis on exploring softforks as well as techniques that do not require any changes to the current Bitcoin protocol.

2.2 Transactions

In a traditional bank account, each account has a balance that lists the amount of money that it contains. The balance itself is independent of where that money came from, although banks might sometimes keep that information about the users. In Bitcoin, the balance of an account is determined by the number of coins that this account controls. In this context, the coins controlled by an address are called *transaction outputs* or *txouts* as they represent the output of a previous transaction. The total amount of coins carried by these *txout*s corresponds to the balance of an account. When Alice creates a spend transaction for Bob, she will always specify what *txout*s she is spending, although this will typically be done automatically by the client software.

The spending of *txouts* is similar to choosing coins in a purse to pay a merchant for

9. `http://siliconangle.com/blog/2013/05/13/bitcoin-blockchain-hard-fork-coming-may-15th-final-warning/`

a pack of gum. For brick-and-mortar stores, the merchant will give Alice change for her coin if she doesn't have the correct denominations, while in the case of Bitcoin Alice can give *herself* the change from the transaction. As in the physical case, Alice needs to fully spend every *txout* she uses as an input, but this change is returned to Alice as a new *txout*. It is recommended [15] that Alice use a new address for every change *txout* that she receives for privacy reasons.

2.2.1 Address Generation

Consider a user Alice that wants to create a new Bitcoin public key. She uses an ECDSA elliptic-curve based signature scheme [10] to generate a public-private key pair

$$ECDSA \rightarrow (sk_a, pk_a)$$

Here, sk_A is a 256-bit integer private key and pk_a is the corresponding 520-bit public key . This public key is then hashed through both $SHA256$ [11] and $RIPEMD160$ [12] and transfered into $Base58$ [13] for readability and error-checking reasons. The result is what is called an *address*. This transformation is illustrated in Figure 2.2 from Bitcoin forums user *itotheipi*.

Addresses are the equivalent of bank accounts. Alice can publicize her address and receive transactions to that address as she would receive cheques to her bank account. As address generation is free, fast and can be done offline, Alice may easily create any number of addresses. When they are created, these addresses do not control any funds, and are similar to an empty bank account.

2.2.2 Transaction Verification

Suppose Alice has produced and publicized transaction tx_{ab} that sends funds to Bob's address B. Suppose then that Bob wants to spend the funds he has received to Catherine's

10. ECDSA specification secp256k1 `http://www.secg.org/collateral/sec2_final.pdf`.
11. `http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf`
12. `http://homes.esat.kuleuven.be/~bosselae/ripemd160.html`
13. `https://en.bitcoin.it/wiki/Base58Check_encoding`

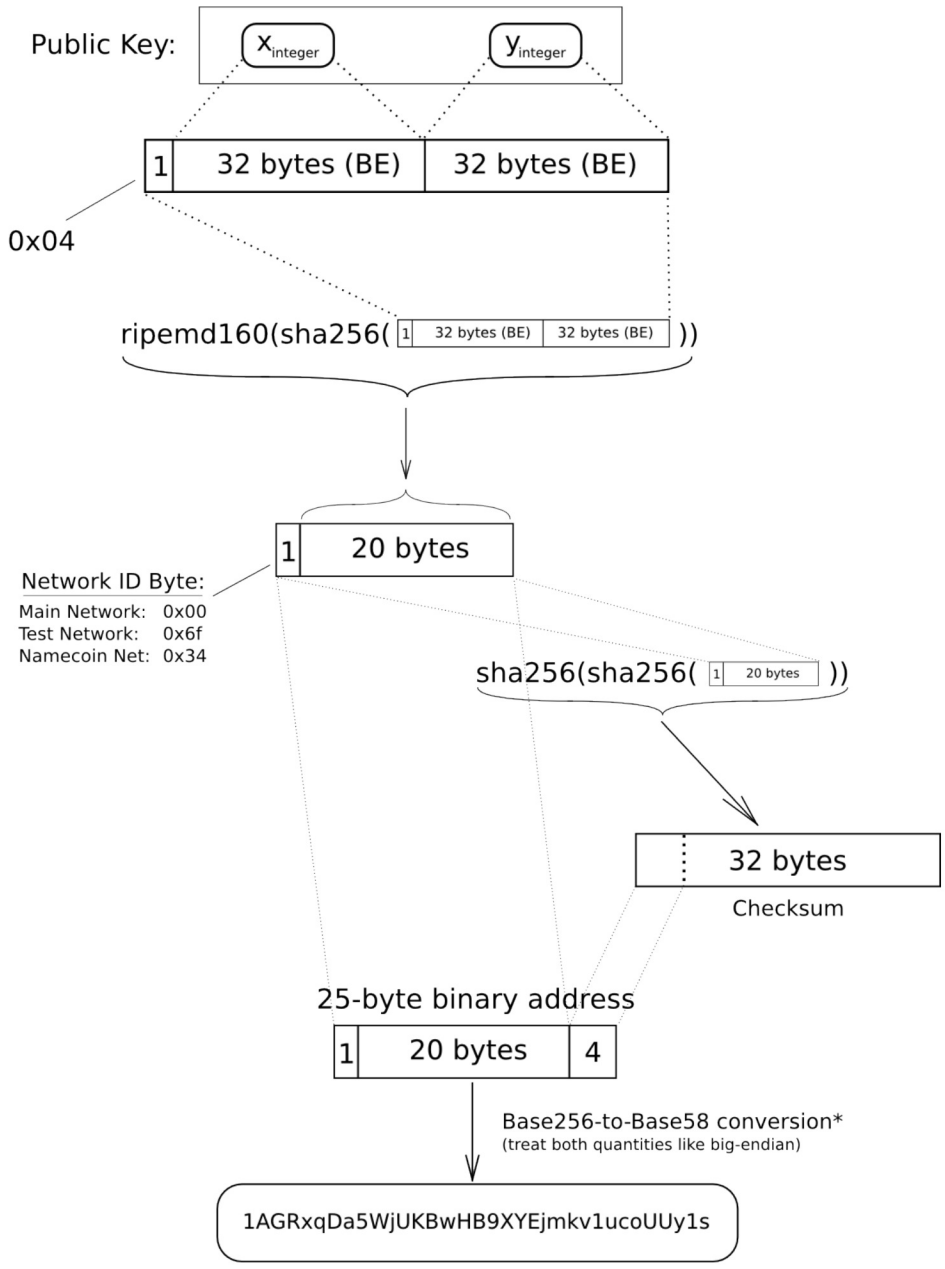

Figure 2.2: Public key to address

address C. He must produce a script that enables the transfer of $txout_{ab}$ to C. He then signs the hash of this script with sk_B which proves that he is indeed the owner of B. The resulting script and signature are collectively called transaction tx_{bc}. We note that tx_{bc} directly refers to tx_{ab}, which in turn refers to the transaction in which Alice got the funds. This chain continues until the first transaction where the coins were initially created, called a *coinbase* transaction. Each step in this chain is both public and publicly verifiable.

2.3 Mining

Due to the lack of a central authority, steps must be taken to validate transactions in a decentralized manner to prevent double-spending in the same way a bank must clear the checks of its customers. Bitcoin relies on *miners* who solve a computationally intensive challenge to be the gatekeepers to the network's integrity. Once a miner is able to solve the problem, they are granted a temporary authority to mark some transactions as valid and incorporate them with the proof of work into what is called a *block*. The miner is instantly granted a fixed number of new coins [14] to be spent by a predetermined address in what is known as a *coinbase* transaction. Additionally, the miner receives *transaction fees* from each transaction that they validate. These two incentives compensate the miner for the hardware and electricity costs of mining.

Whenever a miner creates a block, they immediately send it to the network to be

14. This amount is 25 coins as of 2014.

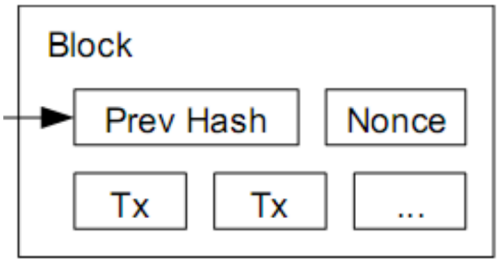

Figure 2.3: Block image taken from [72]

validated. A block that contains an invalid proof, an invalid transaction or any error will be considered invalid itself and thus ignored by the network.

2.3.1 Reasons for Block Creation

There are three reasons that we can see to justify the use of a proof of work within the Bitcoin network.

First, miners act as gatekeepers to the integrity of the network. They are necessary to keep the network coherent as they are the entities that stop double spending by validating transactions. These duties could be handled by a centralized entity in a much simpler and more efficient manner by simply publishing an official record of transactions that are valid, but this would go against the design goal of Bitcoin as a decentralized currency.

The second reason for the proof of work is its predictable outcome and lack of scaling that makes decentralization possible. It is designed in such a way that financial investments in solving it give a linear expected financial return. This makes it unlikely that a single party might get a monopoly on mining since even parties with limited resources can mine profitably. This conception has arguably been weakened by the emergence of application-specific integrated circuits [51].

Finally, since the expected time between blocks is known, the rate at which blocks are received acts as a proof of connectivity to the entire network for peers. This means that if an attacker were to attempt to split the network in two to enable the spending of a transaction in both halves simultaneously, users could realize that the network has been split through the slowdown in block creation and the attack could be thwarted.

2.3.2 Mining Specifications

Mining a block is similar to winning the lottery by randomly picking a number with the correct properties. To mine a block, a miner must produce a block header BH [13] such that

$$SHA256(SHA256(BH)) < target$$

where *target* is a 256-bit large number that determines the level of difficulty of mining at a specific time.

In addition to a reference to the previous block, there are elements in the block header that vary with each try. The difficulty of block generation, that is the target above, is adjusted dynamically. Details are in Appendix I.

2.4 The Blockchain

Since each block validates some transactions, two incompatible [15] transactions cannot be concurrently accepted in two different blocks. To prevent this from happening, each new block must refer to the preceding block that was published to the network by integrating its hash into the proof of work. This chain of blocks extends from the most recent block until the first block ever created, the *genesis block*. This chain is called the *blockchain* and acts as a distributed timestamp server. As each transaction that is accepted by the network belongs in a block, the blockchain represents the *exact state of the network* at any given time. Alice can download what she is missing from the blockchain at any time from the peer-to-peer network, which removes the need for any centralized entity to keep track of the transactions. As the blockchain is self-certifying, an external entity is not needed to verify its authenticity.

2.4.1 Blockchain Forks

Whenever a block is created, it is rapidly sent to the entire network for validation. Whenever a miner receives a block, they will start working on a new block that references it. As the expected time between the creation of two valid blocks is 10 minutes, it might happen that two blocks are created almost simultaneously, both referencing the same previous block. As only one block may directly follow another, the network must decide which of the two blocks will be considered valid.

This situation is known as a blockchain *fork*, at which point different parts of the

15. Two transactions are incompatible if each specifies the sending of a single *txout* to different addresses.

network accept different blocks as valid. As each miner works to build on top of the most recent block that they have received, one of the forks will become longer as more blocks are generated and added to that fork. The miners then abandon the shorter fork and the blocks within it. The transactions included in the abandoned blocks are no longer considered valid but may be validated in later blocks in the dominant blockchain. This is illustrated in Figure 2.4

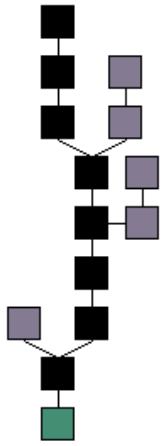

Figure 2.4: Blockchain image taken from [12]

Because of this uncertainty concerning the permanence of blocks, transactions that are included in a block that does not yet have another block following it are typically not yet considered fully *confirmed*. We name the number of blocks built over the block a transaction tx_A is in the *number of confirmations* of tx_A. Typically, a transaction is considered confirmed once it has six confirmations [16].

2.5 Hypotheses in the Bitcoin Network

The functioning of the Bitcoin network relies on some fundamental hypotheses. The first is that a majority of the miners are not colluding to cheat the protocol, as they are the gatekeepers of the validity of the transactions. We note here that this majority

16. Satoshi Nakamoto determined that with such a number the probability of the branch being later abandoned is extremely low [72]. Specifically, an adversary that could control 10% of the hashing power of the network would be able to catch up to 6 blocks with a probability lower than 0.0002429.

is not a strict $N > \frac{1}{2}$, but the chances of abusing the protocol grow very rapidly with the increasing number of colluding miners [72]. It is conceptually easier to consider the miners as discrete entities, but as the power to validate transactions comes from the ability to solve a computational problem, a miner's representation in these decisions is directly proportional to the amount of computing power they have.

The second hypothesis is that the computational problem solved by miners is hard, that is the SHA256 hashing function is collision resistant [56]. Furthermore, it is an unofficial requirement that mining must be able to be done in a decentralized manner. For this to be the case, the proof of work problem must be without memory so that increases in raw computing power increase the probability of solving the proof of work no more than linearly. As noted above, it can be argued that the emergence of application-specific integrated circuits has reduced decentralization of mining due to the investment required in order to mine profitably.

Finally, it is assumed that the network is fully connected that is every node can communicate with every other node and any update to the blockchain is rapidly known to the entire network. If a part of the network were to be separated from the rest, a *txout* could simultaneously be spent on both networks and the authenticity of the Bitcoin network would break down. One defense against this type of separation is to check the rate of block creation on the network. If it were to suddenly drop, one might suspect something is amiss and use extra caution. If an adversary had enough mining power to compensate for that hash rate, it would be more financially interesting to use that hashrate following the network rules than to enable a double-spend in most cases, that is whatever means could be used to cheat would be better used by a profit-driven adversary to cooperate with the network.

2.6 Differences Between Bitcoin and E-Cash

While traditional e-cash relies on a centralized authority, namely a bank, to issue and often verify the authenticity of transactions, Bitcoin operates in an entirely decentralized fashion. This means that the state of the currency is not determined by a bank but by

the state of the network. The authority over what happens to the network belongs to the majority of the miners, as they are the ones that eventually decide if a transaction is accepted or not, and any change to the protocol needs to be approved by a majority of miners. Furthermore, there is no distinction between a *user* and a *merchant* from the network's point of view. As nothing needs to be hidden on the Bitcoin network, there is no encryption of any kind that is used. Private keys used for signing are the only private component of the network and they are never transmitted over the network.

While most e-cash schemes in the literature are offline, Bitcoin is inherently online as a user must be aware of the state of the network to be able to assess if a transaction is valid. On the other hand, it is not required for a user to be online to receive funds from a transaction.

CHAPTER 3

ANONYMITY IN BITCOIN

Anonymity is generally not recognized as an explicit goal of Bitcoin by the core developers [11], and Bitcoin has often been called *pseudo-anonymous* [20]. Specifically, if Alice does not actively try to hide her tracks, public access to the blockchain makes it easy for anyone with an internet connection to link transactions, which is considered undesirable to the privacy seekers [15]. The level of anonymity granted by the use of pseudonyms in Bitcoin has been called into question by many researchers [5, 7, 79, 83]. Nevertheless, Bitcoin's use of pseudonyms only, or *pseudonymity*, has encouraged a number of users that want to use the currency as a way to conduct anonymous transactions. We briefly survey attacks on the anonymity of Bitcoin users.

3.1 De-Anonymizing Bitcoin Users

At first glance, Bitcoin might look like a truly anonymous protocol because Alice never has to identify herself with her true identity in order to spend or receive coins. Furthermore, Alice may have as many identities as she wants and can create a new one for every new transaction. However, there are many ways that Alice's identity could be known to some users on the network without her knowledge or consent. Once her real identity can be linked to an address A that she controls, then any address that A interacts with can be linked to Alice in some way.

3.1.1 IP Addresses

IP address leaking is always a risk to privacy within the Bitcoin network or any computer network. When Alice decides to publish a transaction to the network to be approved, she sends it to a few nodes on the network, who send it to more nodes and so on until the whole network of listening nodes is aware of the proposed transaction. If an adversary had a sufficient number of nodes listening in on the network, then that

adversary might be able to detect what IP the transaction is originally coming from. Kaminsky [55] proposed the idea and wrote a software called *Blitcoin* as a proof of concept.

Bitcoin is not the first electronic protocol where users may want to hide their IP address, and IP-anonymizing networks such as Tor [33] and I2P [1] can be used to conceal IP addresses. These protocols have known weaknesses [10, 46] that can be exploited in specific circumstances, but we do not concern ourselves with these weaknesses in this thesis. Since these solutions to IP address leakage are network-wide and not specific to Bitcoin, we do not treat them further but assume that in every protocol, users are using an encrypted, authenticated and pseudonymous connection to exchange information and publish transactions to the network.

3.1.2 Entering or Exiting the Network

Most users of the Bitcoin network are not miners. These users must get their coins before they spend them, which means someone has to send them coins. If Alice receives coins at address A from Bob [2] for whatever reason, it will probably happen that Bob has some information about Alice. This information can be as directly identifying as her name and home address or as remote as the IP address that she used to contact Bob. The same reasoning can be applied to exiting the Bitcoin network. As a financial transfer medium, bitcoins are only useful if they can be spent, that is exchanged for something outside of the Bitcoin network. In the vast majority of cases where Alice sends some coins to Bob, he will learn something about Alice. This could be what Alice looks like and in what city she lives if the transaction is done face to face, or her postal address if Bob is shipping something to Alice.

1. `http://geti2p.net/en/`
2. We note that Bob could be a bank.

3.1.3 Linking Addresses

Alice might be tempted to mitigate this information disclosure by sending the exact amount that she owes Bob from her *main* [3] address A_1 to a one-time spending address A_2 that is only used for this one transaction with Bob. Alice could then discard A_2 after the transaction with Bob as one would discard an empty subway card. However, since the entire history of transactions is public, Bob or anyone else could see that A_2's funds came from A_1 and thus reasonably infer that A_1 also belongs to Alice. The transfer of one address's information to another is called *taint*.

The most direct way that addresses can be linked is by appearing together in the same transaction. Suppose Alice wants to buy a car from Bob that costs 5 btc and owns a total of 6 btc divided in two addresses A_1 and A_2, each address controlling a single *txout* valued at 3 btc. In this case neither address controls enough funds to buy the car but Alice can combine the *txouts* of both addresses to pay Bob in a single transaction $tx : (A_1, A_2) \Rightarrow (B, A_3)$ or more precisely

$$txout_{A_1} \xrightarrow{3btc} B$$
$$txout_{A_2} \xrightarrow{2btc} B$$
$$txout_{A_2} \xrightarrow{1btc} A_3$$

Since *txouts* must be spent entirely to be spent at all, A_1 and A_2's balances go to zero and A_3 receives the change of the transaction [4], that is 1 btc minus the transaction fees. The transaction is broadcast to the network and eventually confirmed.

On the privacy side, such a transaction of course leaks some of Alice's private information to Bob, such as her ownership of addresses A_1, A_2 and A_3 but also leaks some information to any observer that watches the blockchain once the transaction is confirmed. For a transaction to be accepted, all parties inputting funds must sign the

3. In most wallets, there is no such thing as a *main* address.
4. It is called the *change address*.

transaction, which suggests that both A_1 and A_2 belong to the same entity. Furthermore, a blockchain observer may infer that one of $\{A_3, B\}$ is Alice's change address, and this address may be deduced with certainty if B is a public address. This enables a blockchain observer to link Alice's three addresses.

This linking enables the grouping of a large number of addresses contained in the blockchain, and has been the main method used to analyse ownership of addresses in the blockchain in the past [5, 79, 83]. More recently, the validity of this assumption has been called into question because while the owner of each *txout* used in a transaction must sign the final transaction, the signatures may be provided by different parties. This can be thought of as a business contract: The contract is only valid if each party signs it and if even one party does not sign, the contract is void. While the assumption that multiple signatures belonged to the same entity might have been valid in the early days of Bitcoin as one-to-one transactions were by far the most common kind, the emergence of many services that explicitly use the multi-signature property to save on costs and gain anonymity [64] has arguably made the assumption less valid. Furthermore, the hypothesis that some of the output addresses are change addresses is challenged by Meilkejohn [67] due to the prevalence of mining pool and gambling sites using multiple payout addresses simultaneously.

3.2 Getting Anonymity Back

The object of this thesis is the survey and development of protocols that enable a user Alice to transfer her funds from her original, tainted address A_i to a new, untainted address A_o in a way where it is hard for opponents to link A_i and A_o. These protocols are known as *mixing protocols* and are discussed in chapter 4. We start by describing our threat model and compare different measures for anonymity.

3.2.1 Threat Model

We consider an adversary whose aim is to de-anonymize Alice by linking A_i and A_o after a mixing protocol. Without loss of generality, we assume that it is always Alice

that the adversary is attempting to de-anonymize and that the adversary always knows Alice's position before the protocol. The adversary is computationally bounded, monolithic, static and active and may corrupt any number of parties involved in the schemes, including any trusted third party. Through the inherent pseudonimity of Bitcoin, the adversary might in fact just *be* one or many of those users simultaneously. Of course, the adversary always has full access to the blockchain. In some cases, we may restrict the resources of an adversary to enable simpler schemes, say by restricting the adversary to a simple blockchain observer or by making the adversary unable to corrupt a TTP.

3.2.2 Sybil Attacks

The concept of the *Sybil attack* was first introduced by Douceur [34] and defined by Levine [63] as *an attack against identity in which an individual entity masquerades as multiple simultaneous identities*. The very low cost of address generation combined with the pseudonimity of Bitcoin make Sybil attacks a substantial threat in any anonymizing protocol that relies on users hiding in a large group.

One special case we must consider is when an adversary floods a mixing service with requests to mix in such a way that all the users except Alice are colluding to de-anonymize Alice. In that case, since the inputs and outputs of the mixer are public, the attacker can deduce that the one output address that does not belong to them belongs to Alice. Thus, Alice does not gain any privacy at all with respect to the attacker. This is called an $(N-1)$ attack and cannot be remedied by any protocol [88]. While Sybil attacks cannot be eliminated, their threat can be mitigated by making either a small fee or a *CAPTCHA* [5] necessary to use a mixing service, thus making such an attack prohibitively expensive. For this reason, we consider the effects of a financially bounded adversary where a massive scale Sybil attack would be infeasible.

5. Completely Automated Public Turing test to tell Computers and Humans Apart.

3.2.3 Defining Privacy

Privacy has many definitions that all relate to a user's personal information being secret in some way. Due to the difficulty of quantifying privacy, we introduce the quantifiable concepts of *unlinkability* and *anonymity*.

Definition 1. *Pfitzmann defines [77]* ***unlinkability*** *as the inability for an adversary to distinguish if two items of interest* [6] *are related.*

In the context of e-cash and Bitcoin, *related* can be interpreted as follows: two addresses are said to be *unlinkable* when an adversary is unable to know if two spending transactions involve the same entities. Unlinkability is not a binary attribute and the level of unlinkability can vary from one protocol to another. Additionally, there is no single way to measure unlinkability.

Definition 2. *Pfitzmann defines* ***anonymity*** *[77] as the inability for an adversary to identify a subject within a set of subjects, the anonymity set.*

A user Alice is perfectly anonymous if an adversary is unable to identify her within the set of all users of the service with a probability higher than $\frac{1}{N}$ where N is the total number of users. As before, anonymity is not binary in general and can be measured in many ways.

Pfitzmann defines both *sender* and *receiver* anonymity. The former refers to inability to distinguish the sender of a message, and the later the receiver of a message. In the context of Bitcoin, both the sending and receiving addresses included in a transaction can be found in the blockchain and are thus public. Nevertheless, the *identities* associated with these addresses are not *a priori* known and we may then both consider sender and receiver anonymity in Bitcoin. In the mixing approach defined in the next chapter, we always assume that the identity of the sender is known [7] and the mixing protocol enables her to increase her *receiver* anonymity.

6. Items of interest are any object that are visible to the adversary, e.g. spending transactions.
7. We call this sender Alice.

3.2.4 Measuring Anonymity

The simplest way to measure anonymity is to measure the size of the anonymity set, in which case maximal anonymity is obtained when the anonymity set is the largest set possible. Within a protocol with N output addresses [8], the maximal anonymity set size would be N. However, Diaz [32] and Serjantov [87] point out that a mixing protocol that leaks information might have a reduced anonymity set. In particular, an adversary taking part in the protocol will know that her output address or addresses are not controlled by Alice, that is if there are k colluding adversaries in the protocol, the maximal anonymity set size is $N-k$. Additionally, other information might be leaked by the protocol that enables the adversary to determine that certain addresses have a higher likelihood of belonging to Alice. This can be formalized as a probability distribution D_E for Alice's position from the adversary's point of view. If the adversary has any information that helps her probability of correctly identifying Alice, D_E will not be uniform. To address this disparity between distributions, Diaz and Serjantov propose using Shannon entropy to quantify anonymity from the adversary's point of view.

Definition 3. ***Shannon entropy*** $H(D) = -\sum P(x_i) lg(P(x_i))$

Serjantov describes this as being the number of additional bits of information that the attacker needs in order to definitely identify Alice at the output of the mix.

3.2.5 Min-Entropy

While it is an improvement over set size anonymity, we believe that Shannon entropy is not the best measure of anonymity in situations such as Bitcoin mixing and propose using min-entropy instead. The reasoning for this deviation from a previously established measure is the following.

If the adversary were to try to guess which address belongs to Alice, one or many addresses would have the highest probability among the set of addresses to belong to Alice. We call this probability P_{max} and define it as follows:

8. Many of these users might in reality correspond to the same entity.

Definition 4. ***Maximal location probability*** $P_{max} = max(P_i) \forall i$ *where i is the position of each address.*

This measure may be represented by min-entropy.

Definition 5. ***Min-entropy*** $H_\infty(D) = -lg(P_{max})$

We now argue that min-entropy better represents anonymity than Shannon entropy within the context of Bitcoin mixing. First, we remark that this entropy can be used to quantify the probability that the adversary correctly identifies Alice if she uses the information she has in the best way possible, as $P_{max} = \frac{1}{2^{H_\infty(D)}}$. Furthermore, in the context of Bitcoin mixing, it is reasonable to believe Alice will probably be using multiple consecutive and unrelated mixers as illustrated in Figure 3.2.5. An adversary wanting to follow Alice through these mixers would likely be unable to follow Alice through all paths exiting the first mixer simultaneously, especially if the adversary wants to control adversaries in each mixing protocol. It is much more reasonable for the adversary to follow the most likely path for Alice, in which case min-entropy is the appropriate metric to quantify the adversary's probability of identifying Alice. This quantity may be added linearly for each consecutive mix, enabling accurate measurement.

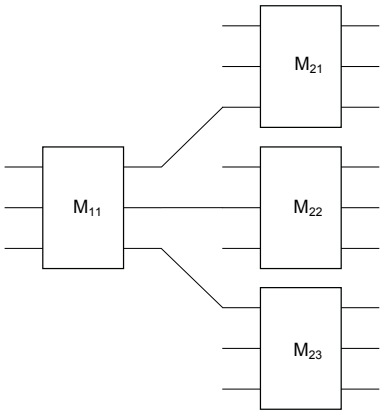

Figure 3.1: Multiple mixers branching out

We illustrate the intuitive sense in using min-entropy over Shannon entropy. Suppose we have the following two probability distributions:

$$D_1 = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad D_2 = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{9} \\ \frac{1}{9} \\ \frac{1}{9} \end{pmatrix}$$

Both of these distributions have the same min-entropy:

$H_\infty(D_1) = H_\infty(D_2) = -\frac{2}{3}lg(\frac{2}{3}) \approx 0.39$

However, the Shannon entropy of these distributions is not the same:

$H(D_1) = -\frac{2}{3}lg(\frac{2}{3}) - \frac{1}{3}lg(\frac{1}{3}) \approx 0.92$

$H(D_2) = -\frac{2}{3}lg(\frac{2}{3}) - \frac{1}{3}lg(\frac{1}{9}) \approx 1.22$

From the Bitcoin mixing point of view, both distributions are equally desirable since we assume that multiple mixers will be used, and only a single branch can be followed. Yet, while they have the same min-entropy, their Shannon entropies are different. This variance in the measure that is used for anonymity when comparing two situations where the intuitive notion of anonymity is the same leads us to using min-entropy over Shannon entropy for measuring anonymity in this context. Nevertheless, as this anonymity measure is not widely accepted, we provide both Shannon entropy and min-entropy for each proposed mixing protocol that we quantify.

3.2.6 Entropy and Mixing

The point of mixing for Alice is to increase the entropy of the position within a protocol of the address she controls at the end of a protocol. If D is the probability distribution of Alice's address's position and H is either Shannon entropy or min-entropy, then the objective of the protocol is maximizing H_D in order to minimize P_{max}. The next chapter formally describes the mixing model that we chose to study and the reasons behind that choice.

CHAPTER 4

MIXING

We return to our setting where a user Alice has an address A_i that is *tainted* with her identity and she wants to transfer her funds to an address A_o that is not tainted. The most naive attempt to do so would be to simply transfer all her funds directly from one address to the other in a single transaction.

$$\forall i, txout_{A_i} \rightarrow A_o$$

This approach fails at reliably anonymizing Alice since an adversary can reasonably infer from the blockchain that both addresses are controlled by the same person with a high probability. A variation might be for Alice to use many intermediate addresses and possibly split them on the way to A_o or keep the funds split. This does not work as Ron [83] has shown that it is possible to connect the addresses together with high probability simply by looking at the blockchain.

The solution to this problem may come in the form of mixing. Protocols that attempt to make user's coins more anonymous take different forms but all follow an idea that was introduced by Chaum in his paper on *mix networks* [25]. A basic mix network, also known as a mixnet or simply a mix, is a routing protocol where a server takes as input messages from multiple senders, shuffles them, and sends them back in random order to the receivers. The aim of such a mixer is to make an adversary unaware of the correspondence between inputs and outputs. Executing such a protocol many times in a row is called a mix-cascade [9] and may be used to enhance anonymity as well as reduce the need for trust in individual mixing servers by alternating between servers.

4.1 Mixing in Bitcoin

Mixing in Bitcoin uses a mixnet-like protocol to send funds instead of messages from tainted addresses to new, untainted addresses in such a way that an adversary cannot link

the input and output addresses. This is illustrated in Figure 4.1 where each input address with subscript i sends to an output address with subscript o with a one-to-one function. The mixing server, called a *mixer* in the Bitcoin literature, must route the funds such that the input and output addresses of a certain transaction that goes through the mixer do *not* belong in general to the same entity. The amount of btc used in each input is the same so that no party loses money in the exchange.

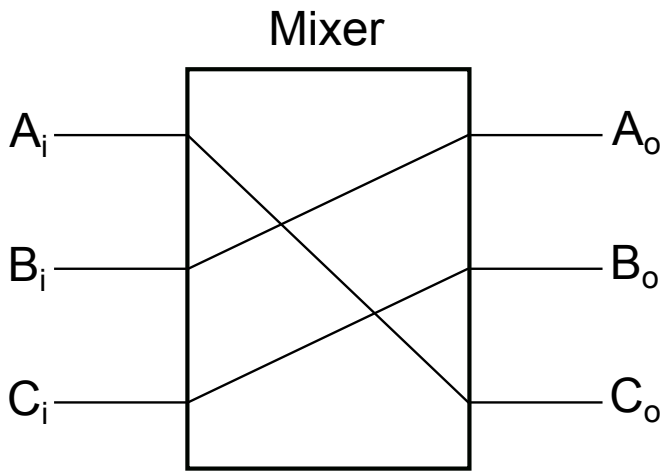

Figure 4.1: Possible permutation produced by a mixer

Mixers may be synchronous or asynchronous. In synchronous mixing, users must wait for other users to be willing to mix before a protocol may be started, while in asynchronous mixing, a user may send coins to a TTP at any time and get unlinked coins back some time later independently of how many people have wanted to mix within that period of time. Our research indicates that most centralized mixers that are used as of 2014 are asynchronous and support N-to-M operations where $N \neq M$ in general, that is the number of input and output addresses of a single user does not need to be the same. N-to-M synchronous transfers are difficult to implement in a private manner although some efforts worth noting have been made by Mike Hearn [45] to develop private protocols. These asynchronous mixers have usability advantages such as possibly faster operations but they suffer from a reduced anonymity set and are generally not

secure since the TTP controls the funds during some time.

The easiest way to mix coins is by actually using a TTP to implement the mixing. Such protocols are called *centralized mixers*. Websites such as Bitcoin Fog [1] or the Bitcoin Laundry [2] offer services where a user can send the service b coins with instructions to later transfer the same amount of coins to one or many output addresses. Of course, these services attempt to send outgoing funds that are not linked to the incoming funds, although as Möser [71] points out, not all such services are successful in doing so. These services generally work in an asynchronous manner, which enables more responsive operations but a smaller anonymity set. We note that although it is not their primary purpose, some online wallets [67] and exchanges may provide similar unlinkability properties.

There are some properties that make a mixing protocol more desirable in the context of Bitcoin. It is desirable that a protocol be decentralized and not use a TTP since doing so does not respect the decentralized nature of Bitcoin. Another consideration is that the protocol should be easy to implement within the actual Bitcoin network without resorting to a hardfork. The costs of the protocol in terms of money, computing power and communication should also be limited. Similarly, the protocol must not be a large burden on the Bitcoin network. Finally, we wish a protocol to be attack resistant, both in the sense of denial-of-service (DOS) resistance and in the sense that we want Alice to still gain some anonymity in the presence of adversaries.

The centralized approach to mixing suffers from serious security and privacy vulnerabilities if the TTP is not trustworthy after all.

1. Security: The TTP can fail to repay the mixed funds and might be hard to trace [67, 70]. A mixer that does not suffer from this vulnerability is said to be *secure*.
2. Privacy: Against an adversary colluding with the TTP, no anonymity is gained. A mixer that does not suffer from this vulnerability is said to be *private*.

We survey some mixing protocols that attempt to correct one or both of these vulnerabilities.

1. `http://www.bitcoinfog.com/`
2. `http://www.bitcoinlaundry.com/`

4.2 Secure Centralized Mixer

Caution: Mixing services may themselves be operating with anonymity. As such, if the mixing output fails to be delivered or access to funds is denied there is no recourse. Use at your own discretion. - The Bitcoin Wiki

The security vulnerability may be addressed to produce a secure mixer while ignoring the privacy side for the moment. This section describe various centralized protocols that may be used to implement a secure mixer. Some of these protocols use third parties but these parties are never in a position where they can steal coins without serious consequences.

4.2.1 Transaction Blueprint with a Facilitator

A curious third party that cannot steal coins can be used to produce a transaction blueprint. We name such a party a *facilitator*. A mixing protocol using a facilitator is illustrated in Algorithm 4. At no point does the third party have any control over the coins. This simple approach does not guarantee privacy and has disadvantages that we explain in chapter 5.

Algorithm 4: Mixing with a facilitator

1. Each party wanting to mix privately communicates their input address (A_i for Alice, B_i for Bob, etc.) and output address (A_o for Alice, B_o for Bob, etc.) to facilitator F;
2. Once F has received all of the expected addresses, they produce $\Pi(A_i, B_i, ...)$, a permutation of all input addresses illustrated in Figure 4.3;
3. F then produces a transaction blueprint B that spends the $txouts$ from the input addresses to the output addresses. This is illustrated in Figure 4.4;
4. F publishes B to all parties involved;
5. The parties sign B if it correctly mixes their funds;
6. The signed transaction blueprint is sent to the network as a valid transaction to be integrated into the blockchain;

4.2.2 Mixcoin

Another approach may be to force a traditional mixer to honour a contract or face the consequences. Mixcoin [19] makes this possible through the use of contracts. The mixer signs a contract that engages them to transfer a user's coins from an input address A_i to an output address A_o through intermediate address belonging to the mixer A_m. If the mixer does not honour the contract in a timely manner, the user may reveal the contract and prove that it has not been fulfilled, thereby damaging the reputation of the mixer. The authors argue that for certain realistic parameters it is better for a rational, profit-driven mixer to honour contracts rather than losing reputation and business.

4.2.3 Hashlock Transactions

Gregory Maxwell proposed *CoinSwap* [64], a cryptography-based asynchronous mixing protocol where users' coins are secure. We produce the first comprehensive description of this protocol.

CoinSwap uses a special type of transaction called a *hashlock transaction* and involves a first user Alice, a mixer Bob [3] and a second user Catherine. The protocol initially was proposed to enable transactions between Alice and Catherine that cannot be linked by looking at the blockchain alone but it can be transformed into a mixing protocol if Alice and Catherine are the same entity. In general, it is assumed that Alice and Catherine are collaborating even if they are not the same entity. The protocol is divided in three phases, uses two types of special transactions and is illustrated in Figure 4.2 and Table 4.I.

The first special transaction is called *2-of-2 escrow timelocked transaction*. This is a two-party transaction that is equivalent to putting some money in escrow where the funds may be later transferred if two parties agree to transfer them. If the funds are not transferred after a certain time, the funds are automatically refunded to their original owner.

In the first phase, Alice creates timelocked transactions $tx0$ that will be refunded at

3. Bob need not be a centralized mixer and can be a peer in a P2P network.

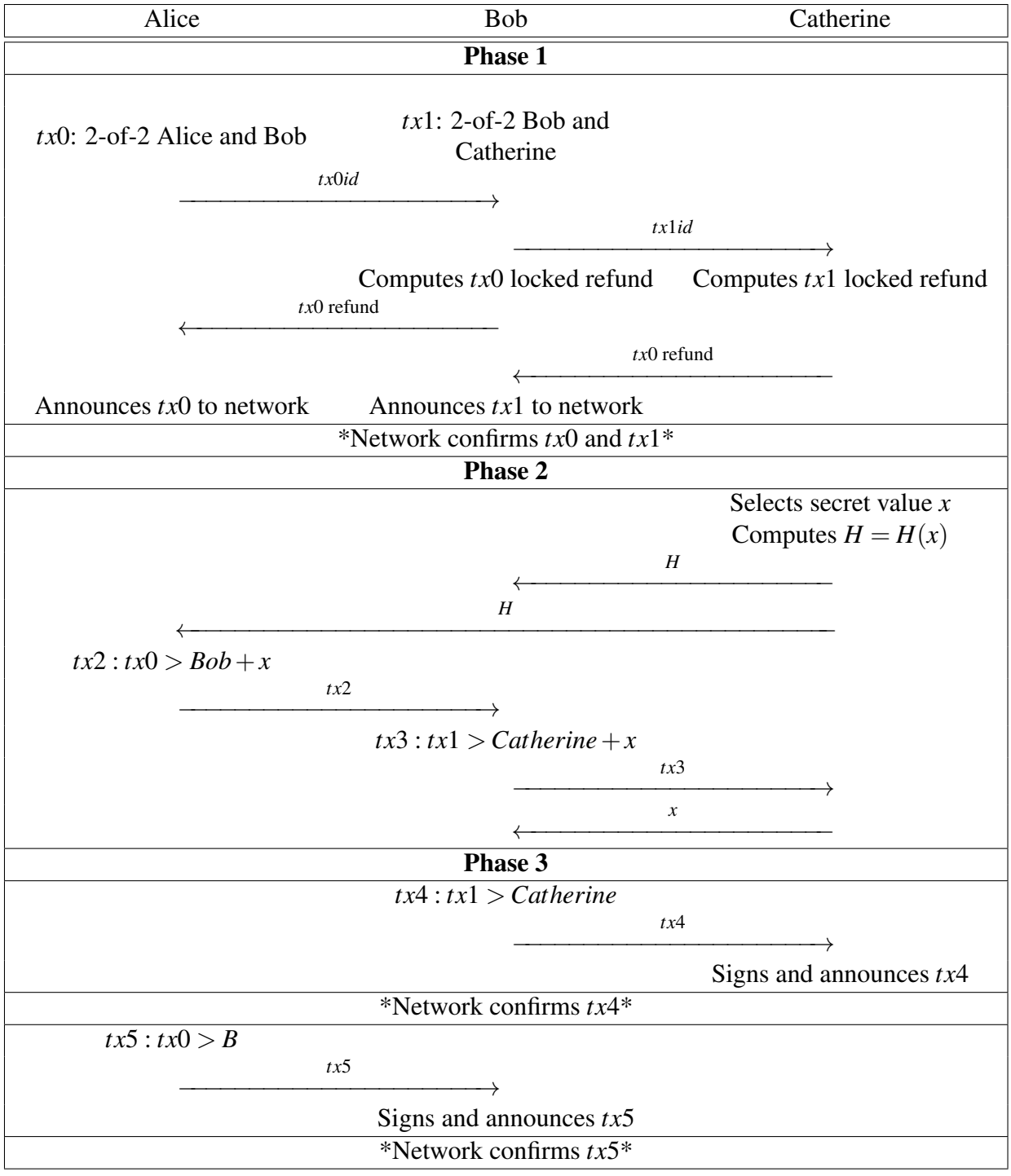

Figure 4.2: CoinSwap

Transaction	Origin	Destination	Requirements
$tx0$	Alice	Escrow	Signed by Alice and Bob
$tx1$	Bob	Escrow	Signed by Bob and Catherine
$tx2$	$tx0$	Bob	Includes x' such that $H(x') = H$ and signed by Bob
$tx3$	$tx1$	Catherine	Includes x' such that $H(x') = H$ and signed by Catherine
$tx4$	$tx1$	Catherine	Signed by Catherine
$tx5$	$tx0$	Bob	Signed by Bob

Table 4.I: Individual transactions in *CoinSwap*

time t_0. We note that such a locked transaction can be spent in another transaction if Alice wishes. Bob also creates a similar timelocked transaction $tx1$ that will be refunded at time t_1. Both these transactions are released to the network to be validated. It is necessary that t_0 expire later than t_1 for reasons that we explain below. The first phase ends when both transactions are validated by the network.

The second phase of the protocol uses a *hashlock* transaction. We note that any hiding commitment scheme could be used as the object of the hashing is a commitment to a value. At the start of this phase, Catherine produces a random x and sends $H(x)$ where H is a collision-resistant hash function to both Alice and Bob. Alice then creates a transaction $tx2$ that takes $tx0$ as input and can be redeemed by whoever can simultaneously sign with Bob's signature *and* produce a value that hashes to $H(x)$, that is produce x. Alice shares $tx2$'s description with Bob, after which Bob creates a similar transaction $tx3$ with $tx1$ as its input and needing a similar x that hashes to the same $H(x)$ and Catherine's signature. Bob sends $tx3$'s description to Catherine. These transactions are private to the users of the protocol and are not published to the Bitcoin network at large. Once she has received $tx3$'s description, Catherine releases x to Bob. At this point, both $tx2$ and $tx3$ could be sent to the network and claimed. However, this is not desirable because it would be easy for a blockchain observer to link them as they share the same hash.

In the third phase, Bob can transfer the funds locked in $tx1$ to Catherine unconditionally with a new transaction $tx4$ without fear of being defrauded. Once $tx4$ is confirmed, Alice can transfer $tx0$ to Bob through a new transaction $tx5$. The protocol ends when $tx5$ is accepted by the network.

We now argue that in each phase, neither party can cheat. The arguments used are

intuitive and could use a rigorous proof as further work. As Alice is attempting to send mixed coin to Catherine, we can always assume that Alice and Catherine are cooperating. We thus consider an adversarial relationship between, on one side, Alice and Catherine, and on the other, Bob. Each party is computationally bounded and may thus not break the commitments that are the hashed values. Each party has access to private communication channels and a broadcast channel. Each party is online and we ignore DOS attacks that would make one party unable to take an action before the timeout for a transaction expires.

During the first phase, if one party does not collaborate or does not respond, the other party just lets the timeout expire and the funds return to their original owners.

In the second phase, Bob is unable to spend $tx2$ before he receives x as he cannot find a collision in H. After he receives x, he has the power to spend $tx2$ but spending it reveals x and thus enables Alice to spend $tx3$, which is not to Bob's advantage. When considering Alice and Catherine's point of view, we have to assume that Alice may know x from her cooperation with Catherine. As Alice shares $tx2$ before she or Catherine learn $tx3$, any spending of $tx3$ will reveal x which will enable Bob to satisfy $tx2$. Catherine cannot wait for $tx0$ to expire and then validate $tx3$ since $tx1$ expires before $tx0$. At any point, any lack of action from any party will simply refund the initial transactions $tx0$ and $tx1$.

In the third phase, if all parties collaborate by signing their parts, $tx4$ and $tx5$ go through to the network and the protocol completes successfully. If $tx5$ goes through but Alice refuses to produce or sign $tx4$ within a reasonable delay, Bob can send $tx2$ to the network to be approved as he now knows x. Alice cannot counter by sending $tx3$ to the network as its input $tx1$ has already been spent in $tx5$. Bob cannot cheat Catherine by not releasing $tx5$ as Alice can release $tx3$ in a similar way. Both $tx0$ and $tx1$ cannot have been spent maliciously as both parties need to sign to spend them. As a result, it is in each party's best interest to collaborate until the end of the protocol.

The protocol requires four published transactions that look like regular 2-of-2 escrow transactions when everyone is honest. In this case, one cannot positively identify such transactions as belonging to a *CoinSwap* transaction by looking at the blockchain. This

might be advantageous if one of the mixing parties wishes not to be seen mixing for whatever reason. If one user does not act honestly however, the hashlocked transactions are used and the transactions can be identified as being used with *CoinSwap*.

We also note that Bob need not be a centralized entity or service and could be any party in a P2P network. Bob might be motivated to act as the intermediary because the coins that he receives from Alice are unlinked to his, so he gains some anonymity too. We note that Alice and Catherine are not anonymous with respect to Bob, and vice versa.

4.3 Decentralized Mixing: Secure *and* Private

In an effort to simultaneously address both the security and the privacy problem, we may chose to remove the third party entirely by making the protocol decentralized. This can be done in general for any functionality by using multi-party computation (MPC) [42]. However, most of the tools used in MPC are both too expensive to use in the context of Bitcoin as well as not supported in the scripting language used for transactions. Furthermore, some requirements of typical MPC protocols, such as the privacy of the inputs, are not necessary in Bitcoin. In addition, many assumptions that are often used in general MPC, such as threshold correctness [4], cannot be applied to Bitcoin because of the threat of Sybil attacks.

In the next section, we describe a synchronous, decentralized N-to-N [5] mixing model that is at the core of this thesis. Nevertheless, some mixing protocols do not fit this model but still deserve our attention. We describe *Zerocoin* and *fair exchange*, two decentralized secure mixing protocols that have been described in peer-reviewed literature but do not fit our model since they are asynchronous.

4. Threshold correctness is a property of a protocol where is a minimum number of parties must follow the protocol correctly for the output to be correct. A protocol is *correct* when the right output is given. If a Bitcoin protocol were not correct, coins could be transferred against the will of their owner for example.

5. N-to-N means that the number of input and output addresses in the mixing is the same.

4.3.1 Zerocoin

Zerocoin [68] is a cryptographic extension to the core Bitcoin protocol. Its interest is mostly academic as Zerocoin requires a hardfork in the Bitcoin protocol that breaks compatibility with the current protocol and requires every user, whether they use Zerocoin or not, to change the way they use the network. Zerocoin uses provably secure cryptographic tools to create a pool where coins can be deposited and withdrawn in an unlinkable manner at a later date, thereby offering the same functionality as an asynchronous centralized mixer without the security and privacy vulnerabilities.

Zerocoin achieves its goal by creating a separate anonymous currency, called *zerocoins*, that operates side-by-side with the Bitcoin network on the blockchain. Zerocoins act as hidden bitcoins and can be exchanged with their regular counterpart with a one-to-one ratio. Whenever a *zerocoin mint* transaction is included in the blockchain, a zerocoin is created and a bitcoin is put in the shared pool. This can be thought of as pawning a bitcoin in order to receive a zerocoin that may be traded back at a later time. The reverse operation can be done with a *zerocoin spend* where a zerocoin is destroyed and a bitcoin from the pool is sent to a new, unlinked address. This works because a spend operation only references the mint operation in a zero-knowledge fashion [6]. This is done using an accumulator that is incremented when a zerocoin mint is done, returning information to the minter. This information can be used to later prove in zero-knowledge that such a mint was done in the past. While zerocoin is cryptographically interesting, the need for a hardfork and for a trusted setup phase suggests that it and its variants, such as *Pinnochio coin* [31], will not be implemented in Bitcoin in the near future.

4.3.2 Fair Exchange

Barber [7] proposed *fair exchange*, a decentralized protocol that works in a similar manner to *Zerocoin* since it creates a pool of locked transactions that can later be redeemed by a user that has previously contributed to the pool. Instead of a large pool

6. To continue the pawn shop analogy, the shopkeeper could have blind-signed the zerocoin. When the zerocoin is redeemed, they would only know that this zerocoin is valid and can be exchanged. This analogy is limited because in the case of Zerocoin, the shopkeeper is a pool and not a centralized entity.

from which any coin can be redeemed at a spend transaction, *fair exchange* has pairs of parties exchange their coins in a way that cannot be linked within that pool. This means that while both parties are aware that they have exchanged coins, their inputs and outputs are indistinguishable within the pool of actively locked *fair exchange* transactions.

Fair exchange uses timelocked and hashlocked transactions similar to those used by *CoinSwap*. Each party in the pair commits to a hashlocked transaction in such a way that the other user may redeem it. One redemption automatically enables the redemption of the other as was done in *CoinSwap*. This enables the safe exchange of the coins between both parties while hiding the link between them. The main disadvantage of this protocol is its poor scalability and the non-zero chance of one party cheating and stealing coins.

4.4 Our Model: Synchronous N-to-N decentralized Mixing

We define a framework for synchronous, decentralized N-to-N mixing, which we use in the remainder of this thesis. As noted before, this model is not the only one that exists, but we choose it because of its interesting properties regarding guaranteed anonymity. Furthermore, this model can be described as a black box that executes an idealized function, enabling the comparison of different solutions.

We define an ideal mixing functionality as one that involves N parties who each control a public input address that has at least b btc available to spend and a private output address. Thus, Alice controls (A_i, A_o), Bob controls (B_i, B_o), etc. To describe our model, we use a TTP that gathers all the input and output addresses of the parties in a private manner. The TTP then knows the set of input addresses $I = \{A_i, B_i, ...\}$, which we can consider to be publicly known, and the set of output addresses $O = \{A_o, B_o, ...\}$, which is private at this point. This TTP then shuffles the output addresses to produce $SO = \pi O$ and publicizes it to the mixing parties without revealing what address was submitted by whom. This is illustrated in Figure 4.3. The TTP, or any other party, may then propose a transaction that transfers the funds from the ith input address of I to the i output address of SO, that is $\forall i, I(i) \rightarrow SO(i)$. This is illustrated in Figure 4.4. We call such a proposed transaction that has yet to be signed a *transaction blueprint*. This

transaction may then be signed by all the parties if it is fair and sent to the network.

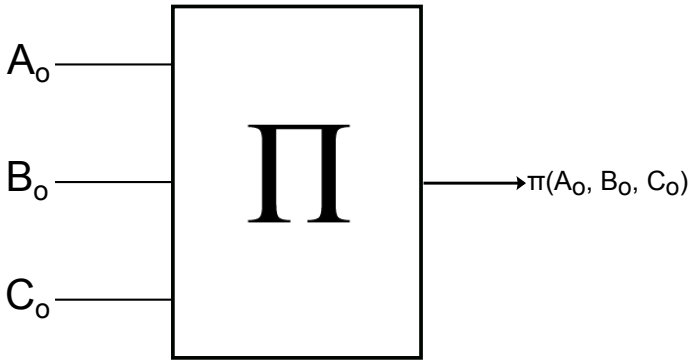

Figure 4.3: Ideal permutation functionality

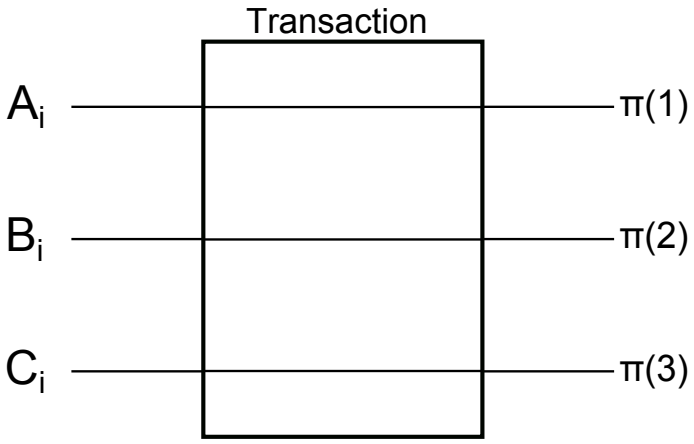

Figure 4.4: Transaction after a permutation

This is similar to the facilitator introduced in section 4.2.1. The aim of this thesis is finding ways to instantiate this idealized function without the need for any TTP. In this context, the privacy of a mixer depends on the way that the permutation π is implemented.

4.4.1 Chaumian Mixers

The permutation functionality described is quite similar to that of a Chaumian mixnet, but there are subtle differences. The first is that if the protocol aborts in Bitcoin mixers, the output addresses need not stay secret. This is in contrast to many applications of

Chaumian mixnets where the privacy of the sent message is of utmost importance even in the case of a protocol abortion. We call this property of Bitcoin mixers the lack of a need for *privacy on abort*.

The second difference is that as in the context of Bitcoin mixing, some information leakage is acceptable since a protocol may be repeated many times and the anonymity gain for each protocol execution is independent. This enables the relaxation in general of *privacy* requirements when compared to traditional mixnets, enabling simpler and more efficient protocols to be used in some situations.

Finally, the mixing protocols in Bitcoin need not be verified for output *correctness* in the same way mixnets are since a protocol whose output messages are not correct simply aborts at signing step and no user loses any privacy.

Nevertheless, we briefly describe some interesting possibilities to use Chaumian mixnets and their variants to mix bitcoins in section 5.2.

4.4.2 Model Limitations

There are some limits concerning what is possible with such a protocol. One of them is that Alice cannot gain any anonymity if all other parties are colluding adversaries in a $(N-1)$ attack. The Bitcoin protocol also imposes the limitation that any party may force the protocol to abort by not signing a transaction at any time.

Moreover, we the amount of btc input by each party must be the same. If it were not the case, either some parties would get less btc than they input, or it would be trivial to link and output input addresses simply by looking at the value of each input and output. It has been proposed by Yang [91] to use amounts of coins that are powers of 2 i.e. $\{\frac{1}{4}, \frac{1}{2}, 1, 2, 4...\}$ to anonymize the entire contents of an initial address by dividing the amount contained in the address and then anonymizing the contents of these new addresses [7]. Mixers could then advertise these standard sizes to ensure that the parties involved mix the same amount.

Alice may further enhance her anonymity by using such a protocol many times with

7. For example, 2.5 btc may be divided into 2 btc and 0.5 btc, both of which are powers of 2. These amount may then be mixed in different instances of a mixing protocol that take appropriate input sizes.

different people. However, this option is complicated by the fact that it is standard for users to pay fees for each transaction. As a result, the new address receives slightly less btc than the original address sent. The amount sent for transaction fees can be standard within a transaction to avoid leaking information on the correspondence between addresses, but that does mean that one cannot transfer the same amount many times in a row. To solve this problem, one solution would be for miners to accept a certain number of mixing transactions every block. This makes sense as we believe that these mixing transactions contribute to the health of the Bitcoin network. This modification could be implemented by requiring that every block includes a certain number of such transactions for free, but enforcing this rule might require a change to the Bitcoin protocol.

Bonneau [19] proposed using randomized mixing fees to pay the miners. This would enable the majority of mixing transactions to not have any mixing fees at all a very low proportion of inputs in a mixing transaction would be sent as mining fees. This can be done in a provably secure way without affecting the anonymity of the scheme.

4.4.3 Mixing in our Model

The next two chapters contain the main results of this thesis. We consider two categories of protocols to mix coins without a TTP. We named these approaches *blueprint mixing* and *networks of transactions*.

The first mixing framework we define is one where the parties instantiate the idealized function described in section 4.4 without the use of a TTP. This approach has been explored before both in the literature [91] and on the forums [85]. The parties collaborate to produce a transaction that mixes N inputs and N outputs in a single transaction blueprint that hides the link between the ownership of the input and output addresses, not only from a blockchain observer, but also from other parties. Once the blueprint is publicized, each party can sign the transaction if it is acceptable to that party, *i.e.* if that party's new address is included in the output for the correct amount. When all parties have signed the transaction, it is sent to the Bitcoin network to be confirmed. We explore various ways to produce a blueprint such as the one in Figure 4.5.

In Chapter 6, we propose a novel approach that uses networks of small, few-parties

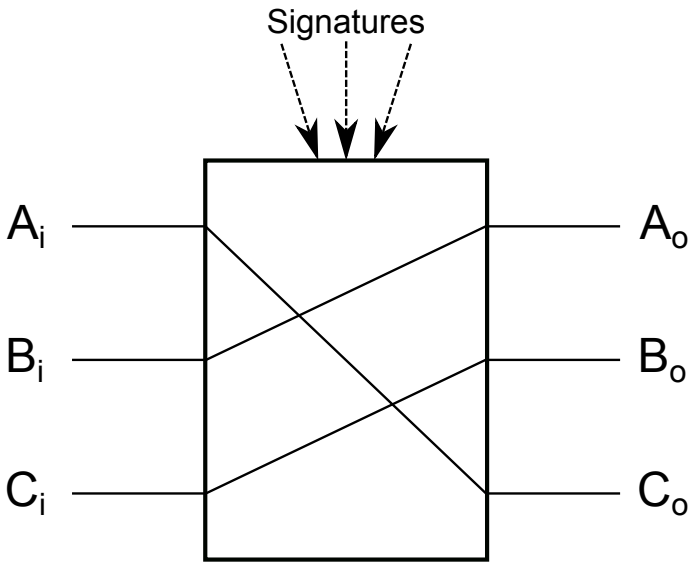

Figure 4.5: Multi-party transaction blueprint

mixing transactions to produce the same result as a larger mixing. The N parties do 2-party transactions in pairs and send them to the network at each step. This is illustrated by the example in Figure 4.6. Once each transaction has been confirmed, the parties can go on to do the next step of the network. This approach effectively produces an N-permutation that is unknown to non-colluding attackers in most cases.

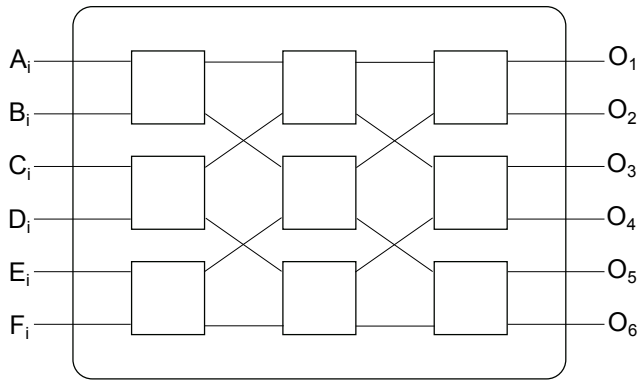

Figure 4.6: Network of transactions

CHAPTER 5

BLUEPRINT MIXING

In this chapter, we survey protocols that have been proposed to implement transaction blueprints and propose some of our own. Since creating a permutation with the output addresses while keeping these addresses private until the permutation is revealed is at the heart of decentralized mixing, we concentrate on ways to implement this permutation securely. Once the secure permutation is known, it is easy to construct a transaction blueprint from it.

The simplest way to create a permutation is for every party to reveal his or her output address to the group of people mixing and then create a provably random permutation [1]. This is similar to the facilitator approach of section 4.2.1 and is the approach used by some implementations of *Coinjoin* [65], such as *Sharedcoin* [2], which use a TTP to aggregate and publish transactions. Another implementation, Coinmux [3], has peers communicate with one another to set up the transaction blueprint. Although such methods are private against a blockchain observer, they are not with respect to the TTP for *Sharedcoin* and with respect to the other peers for *Coinmux*.

5.1 The Bulletin Board

One way to attempt to fix this is to use a private bulletin board where N users that want to mix may post their output addresses anonymously by using an obfuscated IP address. When N output addresses are posted in the bulletin board, a random permutation of these addresses is produced and publicized. The users may then post their input addresses on another board and a transaction blueprint is produced by a script on the bulletin board that links the input and shuffled output addresses. The transaction is then signed as usual. While this method may be private, it is exceptionally DOS-prone as any

1. We show how to do this in appendix II.
2. `http://sharedcoin.com`
3. `http://github.com/michaelgpearce/coinmux`

party included in the mixing may produce two output addresses on the bulletin board or a party not in the mixing might attempt to post an address on the board in order to receive coins. These attempts would be thwarted by the honest parties not signing the transaction in the end, but this illustrates the DOS-vulnerability coming from both inside and outside parties.

This vulnerability, as well as the requirement for a semi-trusted bulletin board, makes such a solution less interesting. The next protocols have the participants enter their obfuscated or encrypted output address in a protocol, and they are only made public once the permutation is complete.

5.2 Mixnets and Multi-Party Sorting

Any implementation of mixnets and the protocols that relate to them can be used to build Bitcoin mixers. Among others, blind signatures [16, 26, 61], ring signatures [82] and DC-nets [27] may be used to create a blueprint. These techniques do not leverage the reduced requirements of Bitcoin described in section 4.4.1. For a survey of these methods, see [86].

Yang [91] proposed producing a secure multi-party sorting of the set of output addresses. Since the addresses are randomly generated, their sorting is one type of permutation that achieves the required properties. The cost of the sorting as proposed by Jonsson [53] is $O(Nlog^2N)$ comparisons and $O(log^2N)$ rounds and requires a secret sharing scheme. As of this writing, the high cost of securely comparing the lengthy addresses is the main obstacle to the use of this technique.

5.3 Mixing Through Encryption

The following protocols rely Alice learning the output addresses of Bob and Catherine, B_o and C_o, without knowing which address belongs to whom. This is done through the use of encryption. Alice can then produce a permutation that includes both of these addresses as well as her own output address.

Some precautions must be taken when choosing the encryption scheme. Specifically, since Bob and Catherine encrypt each other's addresses and send them back, the encryption schemes must be secure against chosen-plaintext attacks [4]. An encryption scheme that realizes such a requirement is said to be CPA-secure. Furthermore, the shuffled plaintexts are eventually revealed so one must make sure that this does not enable, say, Bob to test all of the output addresses to find Catherine's. The protocol also needs the encryption scheme to be commutative. ElGamal encryption [36] or unpadded RSA encryption [81] could be used to achieve both the requirements of CPA-security and commutativity [62].

Since two parties colluding against a third would constitute an $(N-1)$ attack that cannot be prevented, we assume that the parties are not colluding. Furthermore, the lack of a need for privacy on abort enables us to ignore any party not collaborating at any point. As usual, access to authenticated, private communication channels between each party is assumed.

5.3.1 Mixing Through Public Key Encryption

Rosenfeld [85] proposed using commutative public key encryption to produce a secure 3-address permutation as illustrated in Figure 5.3.1. This protocol uses 8 rounds of communication and 4 encryptions and decryptions before the permutation is published. We refer to the original post for more detail.

The previous method may be generalized to N parties. We first extend the 3-party construction to another participant, Dave. The extension can then be done recursively for other parties. The intuition behind the extension is that Alice is to occupy a similar position to that of Bob in the previous protocol and Dave is to take Alice's place.

First, Dave creates a public and private key and publishes his public key. Alice, Bob and Catherine encrypt their addresses with Dave's public key. The protocol then unfolds in the same way as in the 3-party case with the addresses replaced by their counterparts encrypted with Dave's key. Since the encryption scheme is commutative, Alice can remove the layer of encryption that used her key from the two addresses she received from

4. These attacks are described in [56, p. 8].

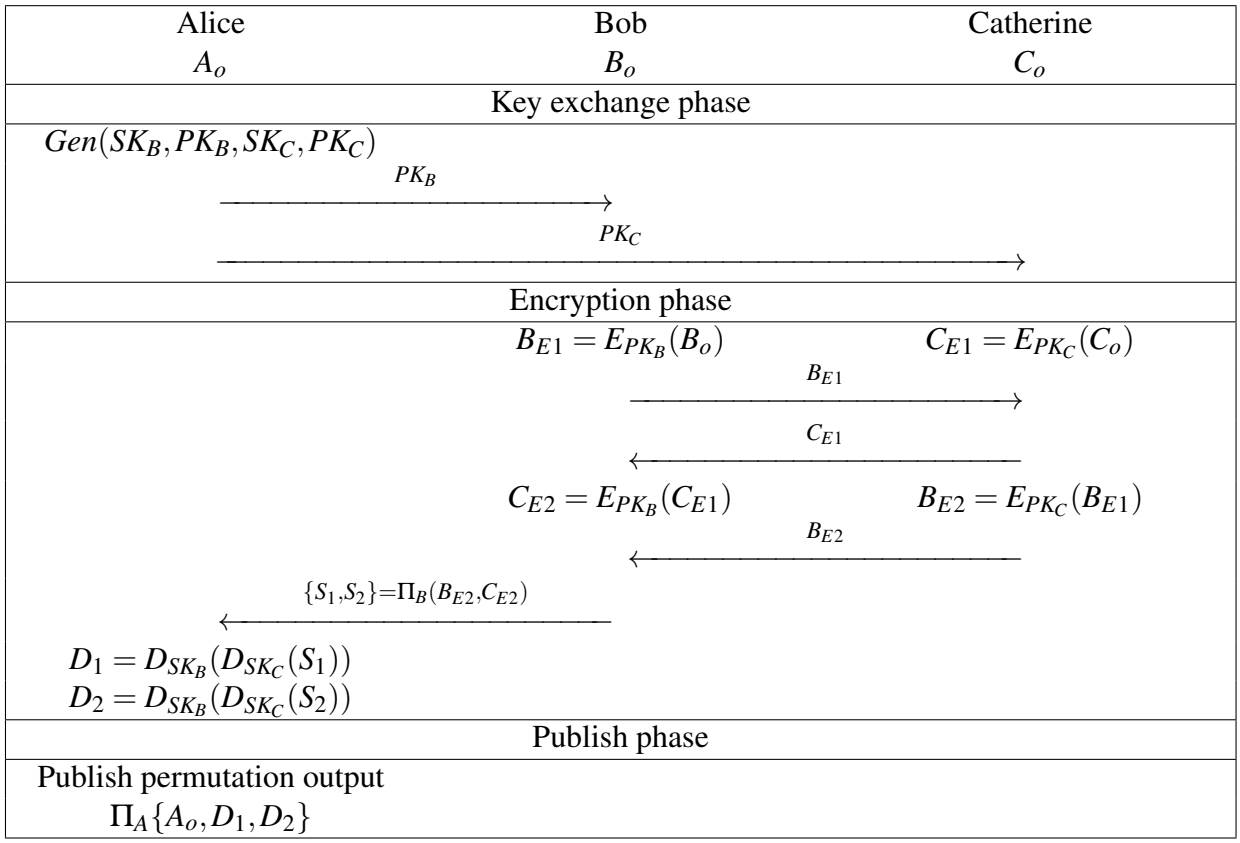

Figure 5.1: 3-party mixing through public key encryption

Bob. This leaves Dave's encryption on so that Alice cannot discover Bob and Catherine's output addresses. Alice then shuffles the two Dave-encrypted addresses she obtains with her own Dave-encrypted address and passes them to Dave. Dave decrypts them all, recovering the original address, shuffles them with his own address and publishes the output of the shuffling, which is a permutation of all the parties' addresses. We can add parties by doing this procedure recursively.

In terms of efficiency, the i^{th} participant will encrypt $O(N-i)$ times and decrypt $O(i)$ times for a total of $O(N)$ encryptions and decryptions. The number of amortized communication rounds is also $O(N)$.

5.3.2 3-Party Mixing Through Private Key Encryption

The greater efficiency of private key encryption over its public counterpart motivates our creation of a new, 3-party mixing protocol that uses private key encryption.

In this protocol, Alice produces two random keys $\{k_B, k_C\}$ that she shares with Bob and Catherine respectively. These keys are used with a commutative CPA-secure encryption scheme such as Algorithm 5 to encrypt the output addresses of Bob and Catherine. The encryption scheme we use is a block cipher uses a pseudorandom function F_k that could be instantiated with AES[5] or a keyed hash function.

We prove that encryption scheme of Algorithm 5 is commutative and CPA-secure as long as the random nonces used in each encryption are known and F_K is pseudorandom.

Algorithm 5: A commutative, CPA-secure symmetric block cipher

1. Alice and Bob share key k;
2. Alice encrypts m: $E_k(m) := F_k(r) \oplus m$;
3. Alice sends $\{r, E_k(m)\}$ to Bob;
4. Bob finds $m = E_k(m) \oplus F_k(r)$;

Lemma 5.3.1. *The block cipher of Algorithm 5 is commutative.*

Proof. $C = E_{k_1}(E_{k_2}(m)) = m \oplus F_{k_1}(r_1) \oplus F_{k_2}(r_2)$

5. *AES* stands for *Advanced Encryption Standard* and is a widely used symmetric encryption function [69]

$D_{k_1,r_1}(D_{k_2,r_2}(C) = m \oplus F_{k_1}(r_1) \oplus F_{k_2}(r_2) \oplus F_{k_2}(r_2) \oplus F_{k_1}(r_1) = m$
$D_{k_2,r_2}(D_{k_1,r_1}(C) = m \oplus F_{k_1}(r_1) \oplus F_{k_2}(r_2) \oplus F_{k_1}(r_1) \oplus F_{k_2}(r_2) = m$
Therefore the encryption is commutative. □

Lemma 5.3.2. *The block cipher of Algorithm 5 is CPA-secure.*

Proof. The proof of the CPA-security of the scheme can be found in [56, p. 90]. □

This encryption scheme can be used to build a 3-party mixing protocol based on private key encryption, as shown in the algorithm in Figure 5.2.

Proof. The above construction is private against a blockchain observer or a single adversary within the protocol.

Lemma 5.3.3. *Bob and Catherine's output addresses are indistinguishable by Alice.*

Proof. Suppose that Alice is able to differentiate Bob and Catherine's output addresses, that is she can differentiate between $\{B_{E2}, r_{B_2}, r_{C_1}\}$ and $\{C_{E2}, r_{B_1}, r_{C_2}\}$. We first note that Bob gives Alice the nonces in an order that enables her to know which nonce corresponds to whom but not which nonce corresponds to which encryption, the first or the second. Attempting to decrypt with a wrong key or in the wrong order will always produce a rubbish output that will be detected as such since it will fail the checksums used for Bitcoin addresses. Furthermore, Alice cannot infer any information from the nonces alone since they follow an entirely random distribution. The only information that Alice can extract from the nonces is that $m_1 = B_{E2} \oplus F_{k_B}(r_{B_2}) \oplus F_{k_C}(r_{C_1})$ and $m_2 = C_{E2} \oplus F_{k_B}(r_{B_1}) \oplus F_{k_C}(r_{C_2})$. This means that she can differentiate between the encrypted addresses, but our hypothesise specify the encryption scheme is CPA-secure. This is a contradiction.

Therefore Alice cannot differentiate between Bob and Catherine's output addresses.

□

Lemma 5.3.4. *Alice and Catherine's output addresses are indistinguishable by Bob.*

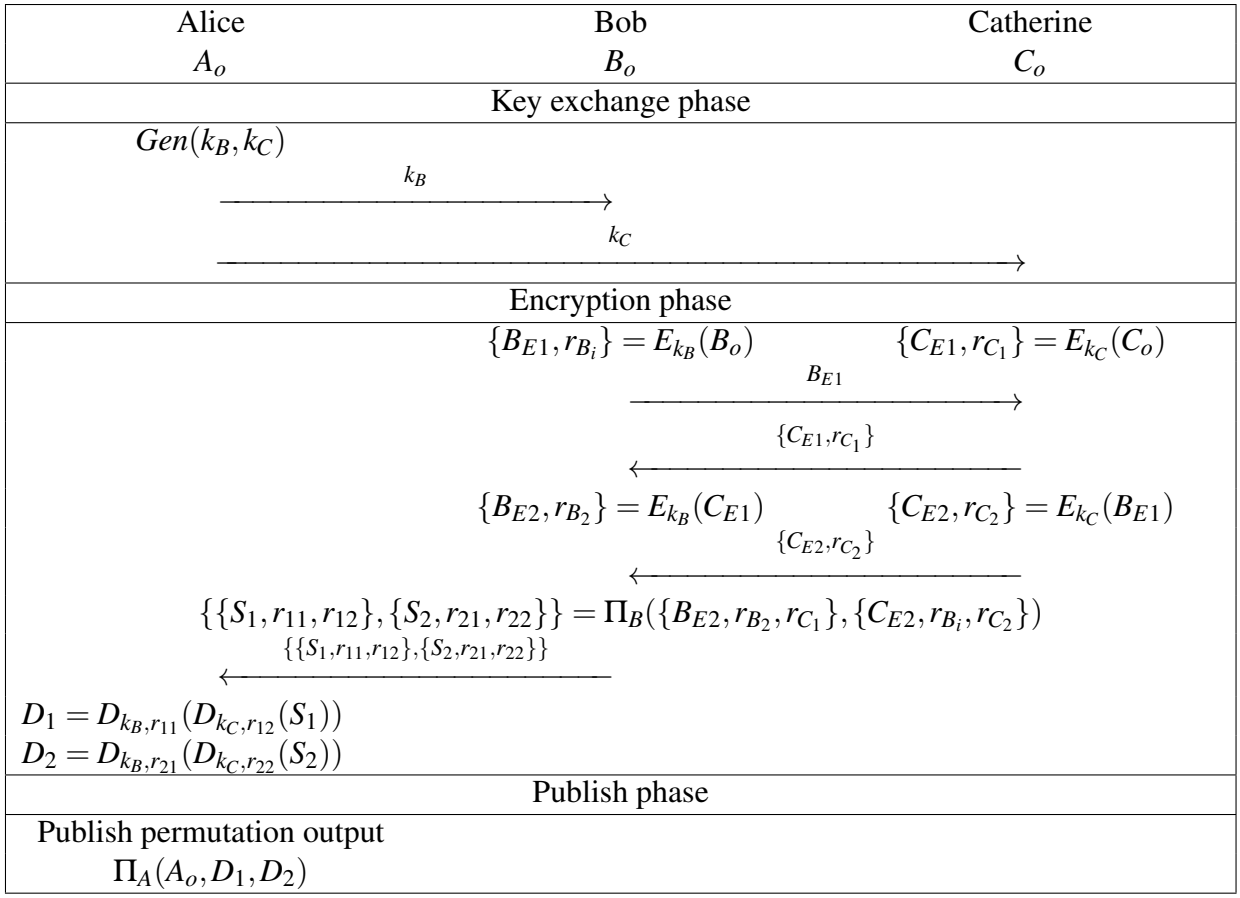

Figure 5.2: 3-party mixing through private key encryption

Proof. We first note that Bob never sees Alice's output address, encrypted or otherwise, before the addresses are revealed by Alice. At this point, Alice's address is indistinguishable from a random address. Thus, Bob can only distinguish Alice and Catherine's addresses if he can learn some information about Catherine's address.

When Bob receives $\{C_{E1}, r_{C_1}\}$, he cannot extract any information about C_o since the encryption is CPA-secure and thus secure against eavesdroppers. When Bob receives $\{C_{E2}, r_{C_2}\}$, he knows the message being encrypted but cannot extract any information on k_C since the encryption scheme is CPA-secure. Therefore, Bob has no information on Catherine's output address before the permutation is published.

Therefore Bob cannot differentiate Alice and Catherine's output addresses. □

The proof for Catherine's inability to differentiate between Alice and Bob's addresses is similar to Bob's. Therefore the protocol is private against non-colluding adversaries. □

5.4 Denial of Service Mitigation

In any protocol where the only transaction is at the end, if any party does not respect the protocol in a detectable way, the protocol is aborted without a loss of funds or anonymity and the party may be banned. We note that such banning is inherently difficult since the protocols are anonymous. Consequently, executing a small proof of work such as a CAPTCHA or paying a small fee before the protocol might be used as a token of goodwill and a deterrent to DOS attacks.

CHAPTER 6

NETWORK OF TRANSACTIONS

We now turn to the second approach we call *network of transactions* which was illustrated in Figure 4.6. This approach uses a number of small switchboxes combined in a structured network to achieve a permutation. Since there can be some information leakage during the protocol, we use the anonymity definitions of section 3.2.4. The objective of such networks is to increase the distribution entropy of Alice's output address H_D as much as possible even in the presence of adversaries.

6.1 Describing the Networks

This section uses many concepts that come from the study of networks and we choose to adopt the nomenclature used in this field. The nomenclature and methodology we use in this chapter comes from [30]. We describe networks with $N = 2^n$ ports or participants as a system made of *stages* and *link permutations* as illustrated in Figure 6.1.

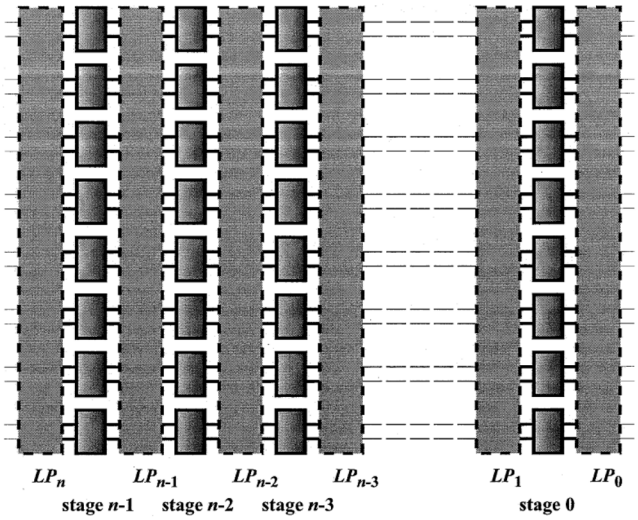

Figure 6.1: Generic multistage network using 2 x 2 switchboxes taken from [30]

The *stages* are where the two-party switching takes place between participants, and

the results of this switching are not known to anyone but the parties participating in a switchbox. The *link permutations* are predetermined, public permutations between the positions of the users. The link permutations can be mathematically described beforehand and determine the type of network being used. For efficiency concerns, we aim to reduce the number of switchboxes and the depth [1] of the network as much as possible.

6.1.1 Switchboxes

Each switchbox in the network consists in a transaction or transaction blueprint with a small fixed number of parties that produces some permutation of the funds to be transferred between input and output addresses of these parties. In this thesis, we focus on using two-party switchboxes.

Algorithm 6 illustrates how two parties may get together and mix their inputs. The result is shown in Figure 6.2.

Algorithm 6: Two-party mixing switchbox

1. Alice and Bob meet and decide to mix coins;
2. Alice and Bob each create an output address;
3. Each party communicates this address as well their input address to the other party on a private and authenticated channel;
4. The parties decide together at random on one of two scenarios: either Alice will send her funds to her own output address and Bob to his, or Alice will send her funds to Bob's output address and Bob to Alice's;
5. Either of the parties creates a transaction blueprint that enacts the transaction above;
6. Both parties sign the transaction;

This entire protocol might seem useless since Alice knows that the output address that isn't hers belongs to Bob and vice versa. Yet, an adversary who is colluding with neither Alice nor Bob cannot guess with probability higher than $\frac{1}{2}$ what situation, switch

1. The depth of the network is the maximum length number of switchboxes on the path from one input to one output.

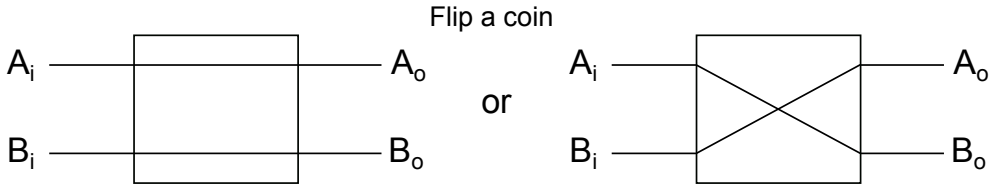

Figure 6.2: Switchbox

or not, was enacted. Additionally, each party's funds are secure as there is only one transaction that is signed by both parties. In terms of entropy, both the Shannon entropy and the min-entropy of Alice's position against a blockchain observer after the mix are equal to 1. We include the technical aspects of setting up such a protocol in appendix II. The protocol requires 3 rounds of communication between the participants and one bit commitment before the transaction can be sent to the network.

6.1.2 Link permutations

One might object that a 50% chance that Eve can successfully guess the positions of the parties may too low to be useful. The power of this protocol lies in using it as a building block to chain small permutations together in order to lead to a much lower probability of Eve successfully guessing Alice's position. In order to do this, these switchboxes must be chained together with *link permutations*.

Since there are $N = 2^n$ participants, the position of each participant can be labelled by n bits as $b_{n-1}b_{n-2}...b_0$. A link permutation LP is a bit-shuffling permutation that maps $b_{n-1}b_{n-2}...b_0$ to $b_{s_{n-2}}b_{s_{n-1}}...b_{s_0}$ with the condition that $s_j = s_k \iff j = k$. The value of s_j indicates the source of the bit in position j at after the link permutation. We introduce two bit-shuffling permutation. We first is called *butterfly permutation of scope* 2^k, denoted β_{k-1}, for which each output bit s_j follows the rule

$$s_j = \begin{cases} k-1 & \text{if } j = 0 \\ 0 & \text{if } j = k-1 \\ j & \text{if } 0 < j < k-1, j \geq k \end{cases}$$

The second bit-shuffling permutation is called a *perfect shuffle* permutation σ_n and

is described by

$$s_j = \begin{cases} k-1 & \text{if } j = 0 \\ j-1 & \text{if } 0 < j < k \\ j & \text{if } j \geq k \end{cases}$$

These two bit-shuffling permutations describe the link permutations of section 6.1.4.

6.1.3 Random Pairing

Random pairing or random mixing is the simplest way to build a network and instantiate the link permutations. We use it as a baseline for comparing various networks. The network is composed of d rounds where at each round, participants randomly get together in groups of two and execute the switchbox protocol above. We assume that each party anonymizes itself between each round, and one round's output addresses are the next round's input addresses. At the end of the d rounds, we are interested in knowing the average entropy gain of each party. We do not describe the link permutations associated as they are not fixed.

We note here that a blockchain observer has to follow Alice's many possible paths. If Alice mixes with Bob and Bob subsequently mixes with Catherine, a blockchain observer cannot tell if Alice was implicated in this second mixing and must assume that it is possible. The anonymity results for this type of mixing can be found in section 6.2.

6.1.4 Structured Networks

While participating in random mixing, if Alice is unlucky she might end up mixing with a number of people much smaller than N and thus gain very little anonymity. It is unlikely but possible that Alice could continually mix with the same person, in which case she would gain nothing more than she would have had she only have mixed with one other party. To avoid this, it is interesting to analyse structured networks that guarantee that Alice mixes with as many different participants as possible.

To avoid this situation situation we turn to networks with a fixed structure where ev-

ery input position can be routed to any output position for every execution of the protocol. One such network is the *butterfly network* illustrated in Figure 6.3. This network has $lg(N)$ depth and $\frac{N}{2}$ width and satisfies the address dispersion requirement. This means that from the point of view of a blockchain observer, Alice could be anywhere at the end of the mixing protocol. Consequently, both the Shannon entropy and min-entropy are equal to the number of stages in the network.

This network may be represented with link permutations before each stage k with $1 < k \leq lg(N)$. A butterfly network link permutation is defined as

$$LP_k = \begin{cases} I & \text{if } k = 0 \\ \beta_{k-1} & \text{if } 0 < k < lg(N) \\ \sigma_{n-1} & \text{if } k = lg(N) \end{cases}$$

LP_k represents the link permutation that transfers the output of the switchboxes at stage $k-1$ to the input of the switchboxes at stage k. The butterfly network is defined with a series of $k+1$ rows of $\frac{N}{2}$ switchboxes interconnected by the k link permutations described above.

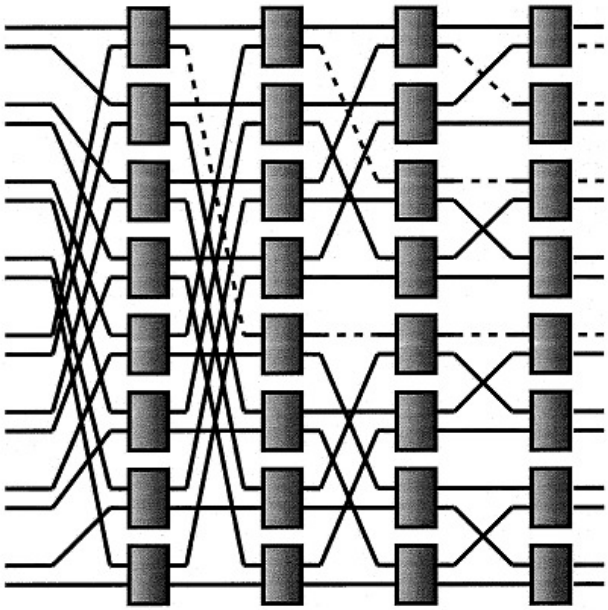

Figure 6.3: Butterfly network taken from [30]

There are other networks that are functionally equivalent to the butterfly network, that is, indistinguishable on the bases of the response at the output ports to any pattern of inputs. The *omega network* is one such network. In our simulations, we found that these networks indeed led to indistinguishable distributions both with and without adversaries in the mixing. For this reason, we did not include the simulation results for this network. The link permutations for an omega network are

$$LP_k = \begin{cases} I & \text{if } k = 0 \\ \sigma_{n-1} & \text{if } 0 < k \leq n \end{cases}$$

These permutations connect the rows of switchboxes as was the case for the butterfly network.

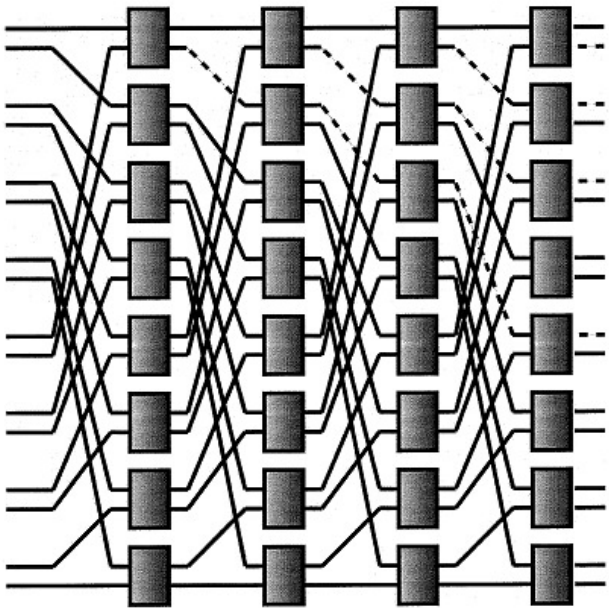

Figure 6.4: Omega network taken from [30]

6.1.5 Benes Network

While networks of depth $lg(N)$ such as the butterfly and the omega networks may enable Alice to occupy any output position at the end of the protocol, they cannot create all possible permutations. The total number of switchboxes in such a network is $\frac{N}{2}lg(N)$, which is smaller than the $lg(N!)$ switchboxes needed to produce the $N!$ different permutations. This is not a problem if the adversary is not colluding with any of the users but it becomes problematic if they are. In this case, the adversary learns a certain number of $\{\text{stage}, \text{position}\}$ pairs that do not belong to Alice. This knowledge enable them to rule out certain output positions for Alice. For example, if Alice mixes with an adversary at the first stage of a butterfly network, the adversary learn the output of the first stage for Alice and can directly rule out half of the output addresses as not belonging to Alice since she could not end up in these positions.

To fix this problem, we turn to the Benes network, which is composed of a butterfly network followed by an inverse butterfly network with the first step of the inverse butterfly removed. That gives a total depth of $2lg(N)-1$ and a total number of switchboxes of $\frac{N}{2}(2lg(N)-1) = Nlg(N) - \frac{N}{2}$. This network is illustrated in Figure 6.5.

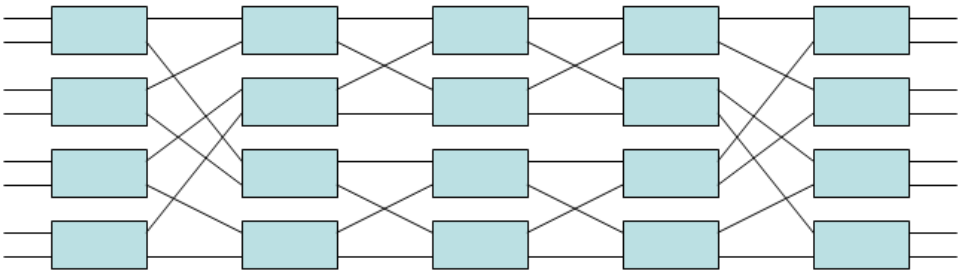

Figure 6.5: Benes Network from Wikimedia Commons

A Benes network's mathematical description requires the inverse of the permutation used in the butterfly network, which is called the *reverse butterfly permutation*: $\bar{\beta}_{k-1} = \beta_{n-k-1}$. The Benes network can then be described as follow:

$$LP_k = \begin{cases} I & \text{if } k = 0 \\ \beta_{k-1} & \text{if } 0 < k < n \\ \bar{\beta}_{k-1} & \text{if } n < k < 2n-1 \\ I & \text{if } k = 2n-1 \end{cases}$$

The Benes network is known in the networking field as a *shuffle-exchange network*, which means that the network can route various types of parallel processes. Cam [23] has shown that $(2n-1)$-stage shuffle-exchange networks are rearrangeable, that is they are able to produce all $N!$ permutations. It was proven by Benes [8] that the network that bears his name is optimal in producing all possible permutations in that its depth cannot be reduced. We offer the proof of an alternative, slightly weaker claim.

Lemma 6.1.1. *Any NxN network composed of two-party switchboxes that can produce any permutation must have at least* $Nlg(N) - 1.44N$ *switchboxes*

Proof. To produce the $N!$ possible permutations we need a minimum of $lg(N!)$ switchboxes since each switchbox only has two possible states. Using Stirling's approximation,

$$ln(N!) = Nln(N) - N + O(ln(N))$$

that is, transforming to base 2,

$$\frac{lg(N!)}{lg(e)} = N\frac{lg(N)}{lg(e)} - N + O(\frac{lg(N)}{lg(e)})$$

Consequently

$$\begin{aligned} lg(N!) &= Nlg(N) - Nlg(e) + O(lg(N)) \\ &\approx Nlg(N) - 1.44N + O(lg(N)) \end{aligned}$$

Therefore any network that achieves the conditions above must have at least $Nlg(N) - 1.44N + O(lg(N))$ switchboxes. □

Since this result is a lower bound on the number of switchboxes needed to produce the $N!$ permutations, any network that could produce all $N!$ permutations could only remove a maximum of

$$\begin{aligned}\text{Benes number} - \text{minimum number} &= (Nlg(N) - \frac{N}{2}) - (Nlg(N) - 1.44N + O(lg(N))) \\ &= 0.94N - O(lg(N))\end{aligned}$$

switchboxes from a Benes network. However, since each level in the Benes network uses $\frac{N}{2}$ switchboxes, no more than 1 level may be removed from this network without losing some of the possible permutations. Furthermore, it would be useless to have more than $\frac{N}{2}$ at any one level since there can be no fan-out in the network. Consequently, the depth of the network cannot be reduced by more than 1 while keeping all permutations possible.

The results for anonymity gained against a blockchain observer are the same as in the butterfly case for the first $lg(N)$ stages and the gain falls to zero afterwards. The Benes network's advantage comes in the presence of adversaries, although our results show that this advantage is not as good as we could have expected.

6.2 Anonymity Results

This section details the anonymity results that we have obtained from our simulations of networks of transactions. We decided to use a 64 party protocol as we believe it is reasonable that this number of parties may get together to mix. Furthermore, 64 is a power of 2 and thus suitable for the use of butterfly and Benes networks. Before the protocol is executed, the adversary has no information on the output addresses as one can imagine that they have not yet been generated. Whether this is the case or not is irrelevant.

We start by simulating the mixing protocol against an adversary that is only a blockchain observer. We then increase the number of adversaries present in the mixing to 8 and 32,

which represent respectively 12.5% and 50% corruption of the parties mixing. We take these values to represent respectively low and high percentages of collusion in the mixing protocol.

The results are illustrated in Figures 6.6 to 6.19. The purple horizontal line represents the maximum possible entropy gain in each situation, while the error bars represent the variance of each value for one hundred iterations. The methodology used to do these calculations is in appendix III.

6.2.1 Blockchain Observer

We initially calculate the average anonymity gained by Alice against a blockchain observer for random mixing in Figures 6.6 and 6.7. Against a blockchain observer, we observe that the Shannon entropy is almost linear in the number of stages with a slope just under its maximum value of 1 until stage 6, at which point it is at approximately 85 % of its maximum possible value. Min-entropy only keeps this slope up to stage 4, at which point the min-entropy has 50 % of its maximum possible value, before getting to approximately 80 % of that value at stage 9. We conclude that random mixing reaches 50 % of its maximum potential after about 4 rounds against a blockchain observer but takes many more rounds to reach full anonymity potential.

A butterfly network always produces a uniform probability distribution for Alice's position in the presence of a blockchain observer. The entropy gain for both entropies is one per level, up to the depth of the network. This illustrates an improvement of approximately 15 % over the random network at the same depth in the case of Shannon entropy. In the case of min-entropy, this gain after 6 rounds is approximately 20 %.

The results for Benes networks are similar to those for a butterfly as no gains are achieved after executing the first butterfly. The next section explores the effects of having adversaries that are present in the mixing protocol.

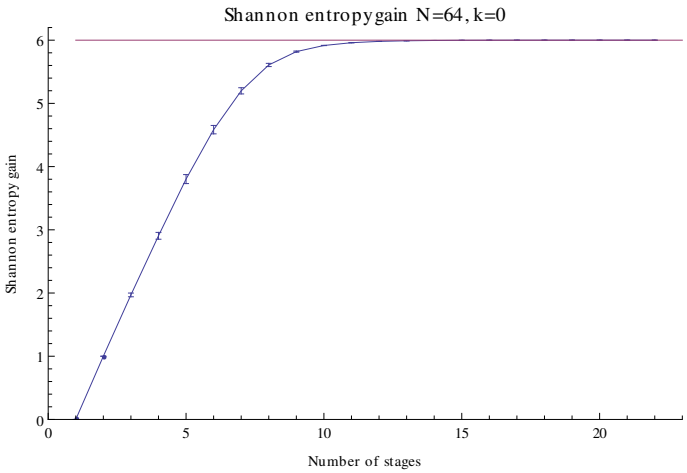

Figure 6.6: Shannon entropy for a random network N=64, k=0

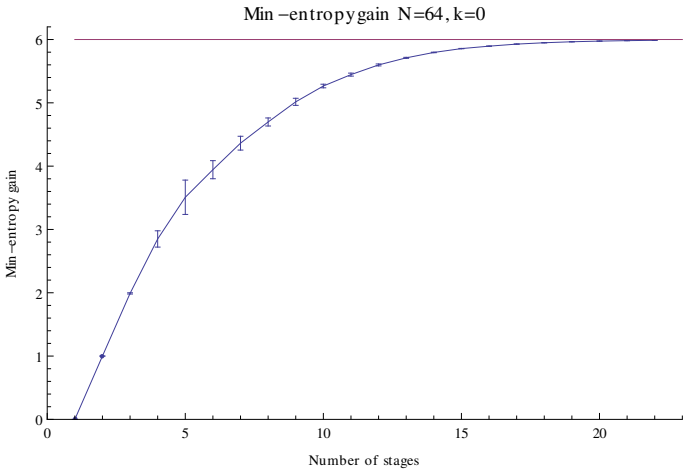

Figure 6.7: Min-entropy for a random network N=64, k=0

6.2.2 Mixing with Adversaries

Before simulating the results of the presence of adversaries, we must consider the importance of the initial positioning of the adversaries in the protocol. If an adversary is colluding with k parties in the mixing, they can identify Alice with probability at least $\frac{1}{N-k}$ as they know that Alice does not end up in the position of the parties they are colluding with. However, in most cases, Alice's anonymity loss compared to the blockchain observer case is much larger. As seen in section 6.1.4, if the adversary colludes with the first party that Alice exchanges with in a butterfly network, they can exclude exactly half of the output addresses. Conversely, if the adversary colludes with a party that Alice could not possibly exchange with until the last step, only one output address is eliminated. To illustrate the strong dependence of the anonymity gain on the position of Eve's collaborators, one can convince oneself that if Eve is colluding with $lg(N)$ parties, it is possible that Alice will mix with an adversary at every step of the network and thus gain no anonymity at all. This illustrates the necessity of randomness in specifying the parties' starting positions and in the coin tosses that decide the output of each switchbox.

One simple solution solution is to have participants commit to their input addresses. After the reveal phase, participants are ordered by alphanumeric order of addresses. Some parties may introduce bias by creating an address with certain properties [2] e.g. having a low alphanumeric address. This is especially problematic if adversaries know Alice or other participants' input addresses as they can determine their relative positions. This weakness can be addressed by using multi-party randomness to create the initial ordering. This can be done by using the protocol found in appendix II.

6.2.2.1 Interpreting the Results

We calculate the maximum possible entropy that can be gained in the presence of adversaries as $H_{max} = lg(N-k)$. This is not to be confused with the special type of entropy called *max entropy*[59]. With 64 participants, this maximum entropy is 6 with $k = 0$, 5.80 with $k = 8$ and 5 with $k = 32$. $\%max$ represents the comparison of each

2. Addresses with certain properties can be generated very rapidly and for free, e.g. `bitcoinvanity.appspot.com`.

gained entropy with this maximum entropy. We calculated confidence intervals (CI) for two sigma, which means that the probability that the result would be outside this bound is under 5%. These are illustrated as the blue vertical bars on the graphs. The confidence interval percentages CI pct $= \frac{\text{confidence interval}}{\text{entropy}} * 100$ illustrates the accuracy of the measure. This measure is not illustrated on the graphs but is contained in Tables 6.I to 6.XIV.

6.2.2.2 Random Network With k=8 and k=32

We first illustrate the results for the random network in Figures 6.8 to 6.11 as the baseline for comparing other networks. The results for 8 adversaries are quite similar to those of a blockchain observer, although the variance is greater and the curve falls off slightly earlier, meaning it takes more rounds to reach the same level of anonymity. With 32 adversaries, the anonymity gain is much slower, especially in the min-entropy case. Furthermore, the variance is much greater than in the previous cases.

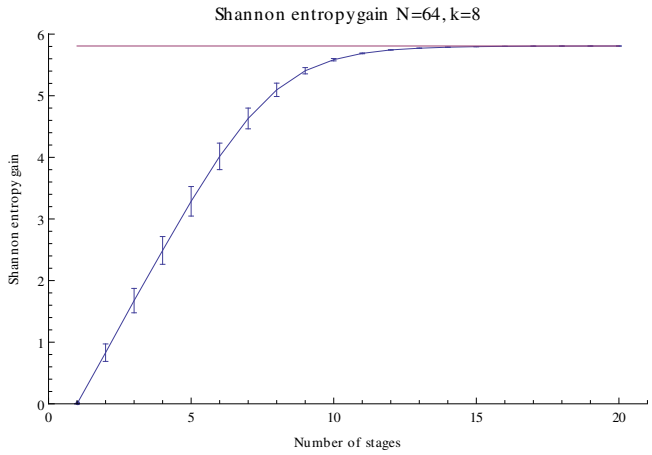

Figure 6.8: Shannon entropy for a random network N=64, k=8

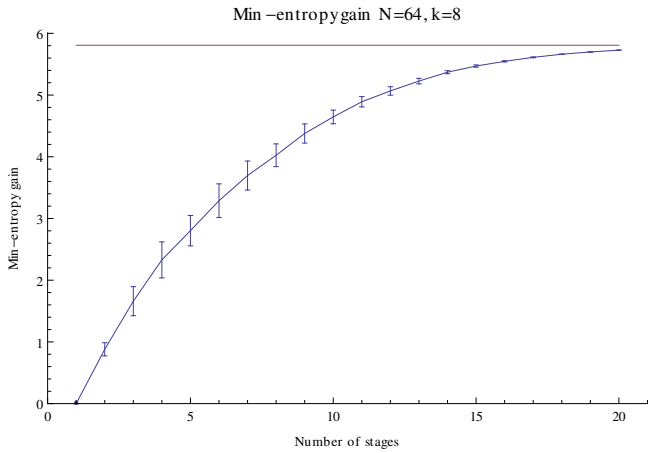

Figure 6.9: Min-entropy for a random network N=64, k=8

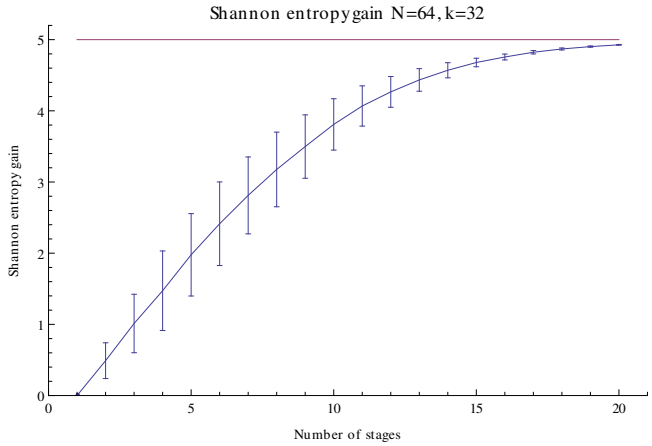

Figure 6.10: Shannon entropy for a random network N=64, k=32

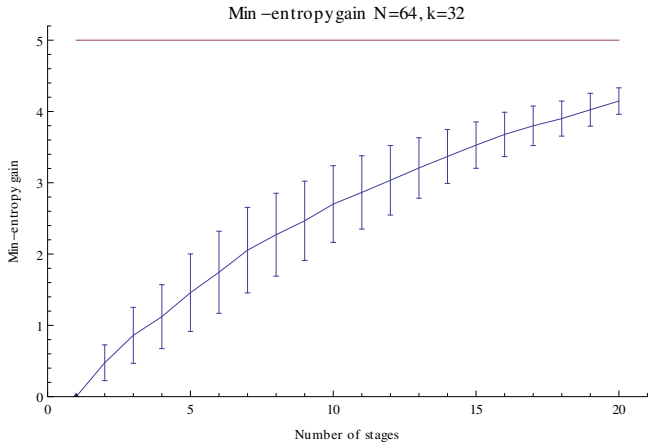

Figure 6.11: Min-entropy for a random network N=64, k=32

6.2.2.3 Random and Butterfly Networks With k=8

We now compare the butterfly network to the random network with $k = 8$ in Tables 6.I to 6.IV and Figures 6.12 and 6.13. The random permutations are illustrated with the blue curve, the structured networks with the purple curve and maximum possible entropy with the green line. We observe that the final benefit of structuring the network as a butterfly is a 10% and 9% improvement over the random network with the Shannon entropy and min-entropy metrics respectively.

Shannon entropy for a random network N=64, k=8						
Depth	1	2	3	4	5	6
Entropy	0.84	1.72	2.58	3.37	4.06	4.66
%max	14 %	30 %	44 %	58 %	70 %	80 %
Variance	0.13	0.21	0.26	0.28	0.28	0.21
CI	0.05	0.06	0.07	0.07	0.07	0.06
CI pct	6 %	4 %	3 %	2 %	2 %	1 %

Table 6.I: Shannon entropy for a random network N=64, k=8

Shannon entropy for a butterfly network N=64, k=8						
Depth	1	2	3	4	5	6
Entropy	0.86	1.74	2.60	3.50	4.38	5.25
%max	15 %	30 %	45 %	60 %	75 %	90 %
Variance	0.12	0.21	0.24	0.24	0.22	0.21
CI	0.05	0.07	0.07	0.07	0.07	0.07
CI pct	6 %	4 %	3 %	2 %	2 %	1 %

Table 6.II: Shannon entropy for a butterfly network N=64, k=8

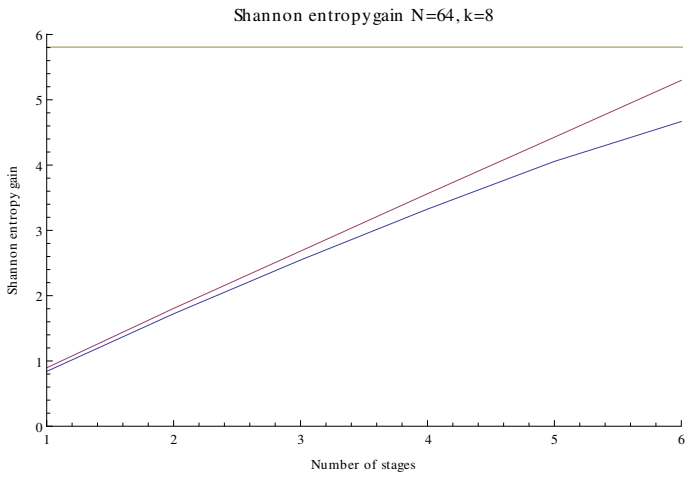

Figure 6.12: Shannon entropy for random and butterfly networks N=64, k=8

Min-entropy for a random network N=64, k=8						
Depth	1	2	3	4	5	6
Entropy	0.84	1.65	2.29	2.79	3.28	3.66
%max	14 %	28 %	39 %	48 %	57 %	63 %
Variance	0.13	0.27	0.38	0.31	0.34	0.31
CI	0.05	0.08	0.09	0.08	0.08	0.08
CI pct	6 %	4 %	4 %	3 %	3 %	2 %

Table 6.III: Min-entropy for a random network N=64, k=8

Min-entropy for a butterfly network N=64, k=8						
Depth	1	2	3	4	5	6
Entropy	0.86	1.66	2.29	2.95	3.55	4.18
%max	15 %	29 %	39 %	51 %	62 %	72 %
Variance	0.12	0.28	0.37	0.32	0.29	0.36
CI	0.05	0.07	0.08	0.08	0.07	0.08
CI pct	6 %	5 %	4 %	3 %	2 %	2 %

Table 6.IV: Min-entropy for a butterfly network N=64, k=8

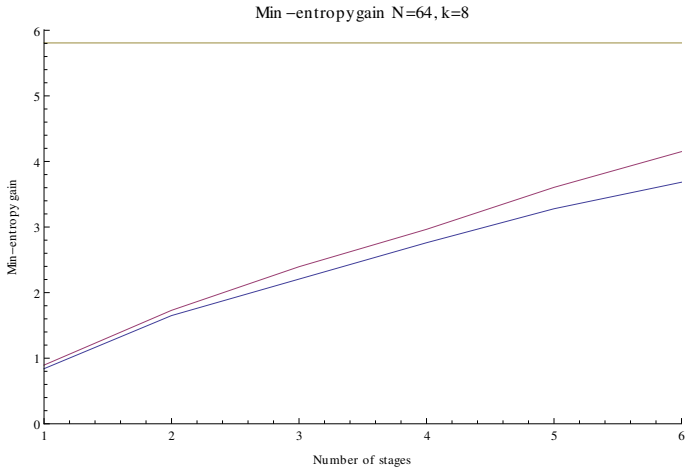

Figure 6.13: Min-entropy for random and butterfly networks N=64, k=8

6.2.2.4 Random and Butterfly Networks with k=32

Tables 6.V to 6.VIII illustrate random and butterfly networks with 32 adversaries. The entropy gains are superior for the butterfly network by 4 % and 2 % for Shannon entropy and min-entropy respectively. This figure is lower than we would have expected by looking at the structure of the network. We interpret this as signifying that if half of the participants are adversaries, the structure of the network does not matter much for short networks. In the next section, we analyse Benes networks in order to determine if the use of such networks improves the outcomes.

Shannon entropy for a random network N=64, k=32						
Depth	1	2	3	4	5	6
Entropy	0.5	1.01	1.53	1.98	2.42	2.81
%max	10 %	20 %	31 %	41 %	48 %	56 %
Variance	0.25	0.40	0.47	0.60	0.62	0.60
CI	0.07	0.08	0.09	0.11	0.11	0.11
CI pct	14 %	9 %	6 %	5 %	5 %	4 %

Table 6.V: Shannon entropy for a random network N=64, k=32

Shannon entropy for a butterfly network N=64, k=32						
Depth	1	2	3	4	5	6
Entropy	0.46	0.97	1.49	1.97	2.48	2.99
%max	9 %	20 %	30 %	39 %	50 %	60 %
Variance	0.25	0.46	0.57	0.69	0.74	0.72
CI	0.07	0.09	0.10	0.11	0.12	0.12
CI pct	15 %	10 %	7 %	6 %	5 %	4 %

Table 6.VI: Shannon entropy for a butterfly network N=64, k=32

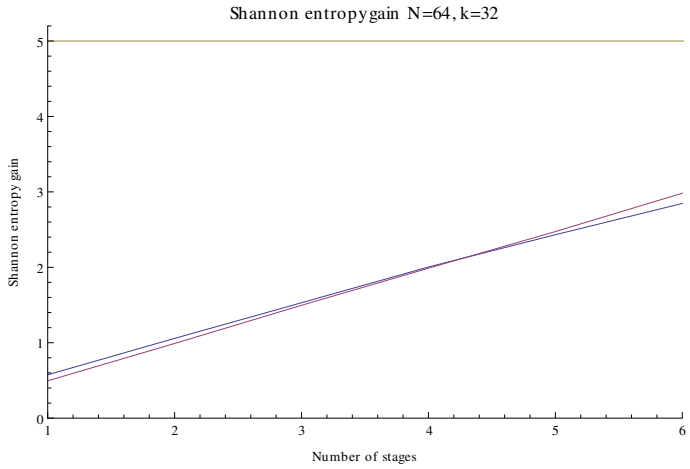

Figure 6.14: Shannon entropy for random and butterfly networks N=64, k=32

Min-entropy for a random network N=64, k=32						
Depth	1	2	3	4	5	6
Entropy	0.5	0.88	1.25	1.53	1.82	2.09
%max	10 %	18 %	25 %	31 %	36 %	42 %
Variance	0.25	0.31	0.40	0.45	0.49	0.51
CI	0.07	0.08	0.09	0.09	0.10	0.10
CI pct	14 %	9 %	7 %	6 %	5 %	5 %

Table 6.VII: Min-entropy for a random network N=64, k=32

Min-entropy for a butterfly network N=64, k=32						
Depth	1	2	3	4	5	6
Entropy	0.46	0.85	1.25	1.56	1.88	2.19
%max	9 %	17 %	25 %	31 %	38 %	44 %
Variance	0.25	0.38	0.45	0.50	0.53	0.47
CI	0.07	0.08	0.09	0.10	0.10	0.10
CI pct	15 %	10 %	8 %	6 %	5 %	4 %

Table 6.VIII: Min-entropy for a butterfly network N=64, k=32

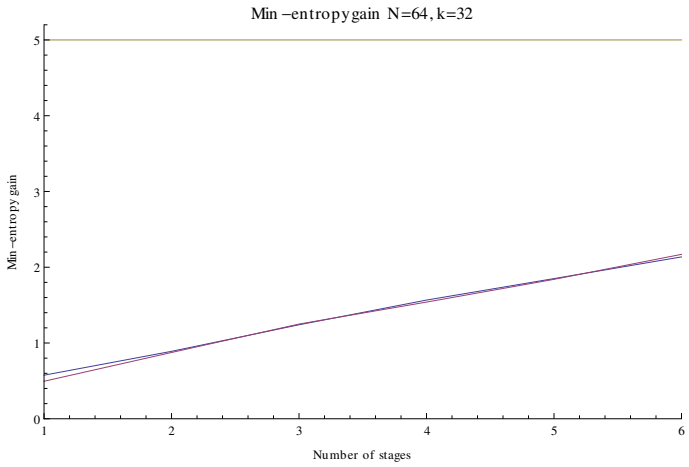

Figure 6.15: Min-entropy for random and butterfly networks N=64, k=32

6.2.2.5 Random and Benes Networks with k=8

Tables 6.IX to 6.XII compare the random and Benes networks with 8 adversaries. We show an increase number of rounds of random permutation to enable the comparison with Benes networks. We see from Figures 6.16 and 6.17 that past the first butterfly at step 6, very little is gained from the Benes network, to the point where random mixing actually passes the Benes network at some stages. The end results for Shannon entropy are indistinguishable since they are close to the maximum possible entropy. For min-entropy, the random mixing *exceeds* the results from the Benes network by 1 % at the last step, although this result is not significant.

Shannon entropy for a random network N=64, k=8											
Depth	1	2	3	4	5	6	7	8	9	10	11
Entropy	0.85	1.71	2.53	3.32	4.05	4.65	5.09	5.40	5.58	5.68	5.73
%max	15 %	29 %	44 %	57 %	70 %	80 %	88 %	93 %	96 %	98 %	99%
Variance	0.12	0.187	0.25	0.27	0.23	0.17	0.10	0.05	0.02	0.007	0.002
CI	0.04	0.06	0.07	0.07	0.06	0.05	0.04	0.03	0.02	0.01	0.006
CI pct	6 %	4 %	3 %	2 %	2 %	1 %	1 %	1 %	1 %	0.5 %	0.3 %

Table 6.IX: Shannon entropy for a random network N=64, k=8

Shannon entropy for Benes mixing N=64, k=8											
Depth	1	2	3	4	5	6	7	8	9	10	11
Entropy	0.82	1.71	2.58	3.45	4.33	5.20	5.27	5.37	5.49	5.59	5.76
%max	14 %	30 %	44 %	60 %	75 %	90 %	91 %	92 %	95 %	96 %	99%
Variance	0.14	0.19	0.27	0.27	0.26	0.24	0.25	0.24	0.18	0.14	0.003
CI	0.05	0.06	0.07	0.07	0.07	0.06	0.07	0.07	0.06	0.05	0.008
CI pct	6 %	3 %	2 %	2 %	1 %	1 %	1 %	1 %	1 %	1 %	0.1 %

Table 6.X: Shannon entropy for a Benes network N=64, k=8

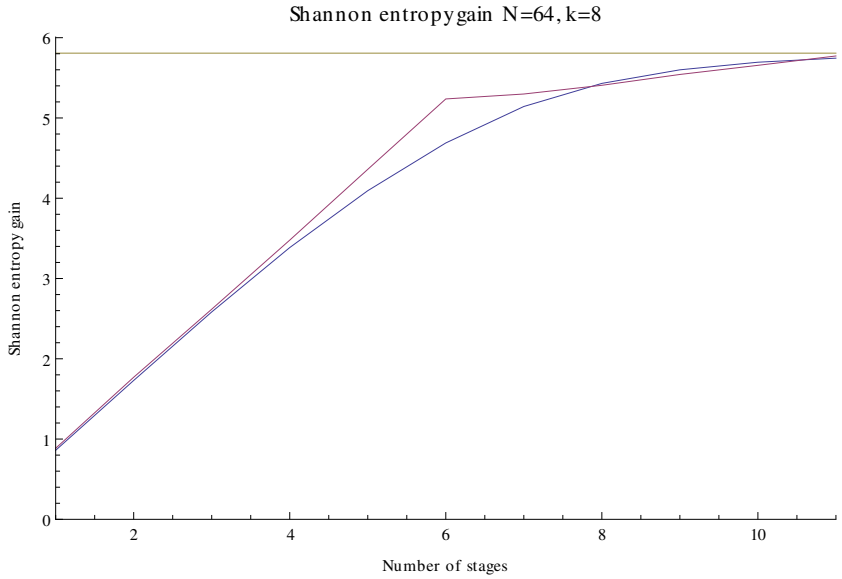

Figure 6.16: Shannon entropy for random and Benes networks N=64, k=8

Min-entropy for a random network N=64, k=8											
Entropy	0.85	1.61	2.21	2.74	3.24	3.64	4.00	4.34	4.60	4.85	5.04
%max	15%	28 %	38 %	47 %	56 %	63 %	69 %	75 %	79 %	84 %	87 %
Variance	0.12	0.26	0.37	0.37	0.33	0.25	0.21	0.15	0.12	0.10	0.07
CI	0.04	0.07	0.08	0.08	0.08	0.07	0.06	0.05	0.05	0.04	0.03
CI pct	6 %	5 %	4 %	3 %	3 %	2 %	2 %	1 %	1 %	1 %	1 %

Table 6.XI: Min-entropy for a random network N=64, k=8

Min-entropy for Benes mixing N=64, k=8											
Depth	1	2	3	4	5	6	7	8	9	10	11
Entropy	0.82	1.63	2.28	2.88	3.55	4.10	4.26	4.50	4.75	4.99	5.20
%max	14 %	28 %	39 %	50 %	61 %	71 %	73 %	78 %	82 %	86 %	90%
Variance	0.14	0.26	0.41	0.33	0.31	0.31	0.28	0.25	0.23	0.21	0.14
CI	0.05	0.07	0.09	0.08	0.07	0.07	0.07	0.07	0.06	0.06	0.05
CI pct	6 %	4 %	4 %	3 %	2 %	2 %	2 %	2 %	1 %	1 %	1 %

Table 6.XII: Min-entropy for a Benes network N=64, k=8

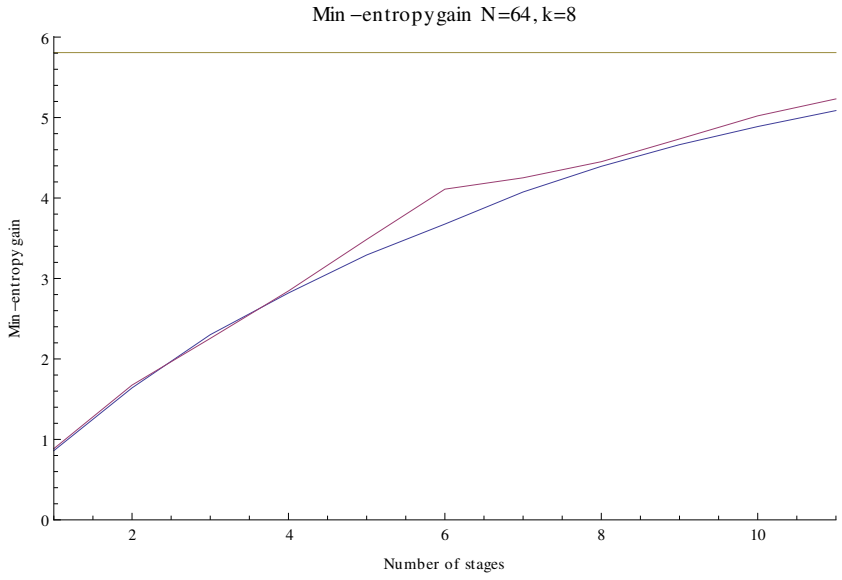

Figure 6.17: Min-entropy for random and Benes networks N=64, k=8

6.2.2.6 Random and Benes Networks with k=32

Finally, we compare the random and Benes networks with 32 adversaries in Tables 6.XIII to 6.XVI. The results are quite similar to the results of 8 adversaries and slightly advantage the random network over the Benes network, although again this result is within the margin of error.

Shannon entropy for a random network N=64, k=32											
Depth	1	2	3	4	5	6	7	8	9	10	11
Entropy	0.52	1.03	1.52	1.97	2.41	2.82	3.22	3.55	3.83	4.08	4.28
%max	10 %	21 %	30 %	39 %	48 %	56 %	64 %	71 %	77 %	82 %	86%
Variance	0.25	0.38	0.48	0.50	0.48	0.51	0.47	0.44	0.38	0.31	0.23
CI	0.07	0.08	0.09	0.10	0.09	0.10	0.09	0.09	0.08	0.07	0.06
CI pct	13 %	8 %	6 %	5 %	4 %	3 %	3 %	3 %	2 %	2 %	2 %

Table 6.XIII: Shannon entropy for a random network N=64, k=32

Shannon entropy for Benes mixing N=64, k=32											
Depth	1	2	3	4	5	6	7	8	9	10	11
Entropy	0.51	0.97	1.48	1.98	2.48	2.96	3.10	3.34	3.59	3.90	4.23
%max	10 %	20 %	30 %	40 %	50 %	59 %	62 %	67 %	%72	%78	85%
Variance	0.25	0.40	0.50	0.53	0.52	0.59	0.61	0.56	0.52	0.44	0.29
CI	0.07	0.09	0.10	0.10	0.10	0.10	0.11	0.10	0.10	0.09	0.07
CI pct	13 %	9 %	7 %	5 %	4 %	4 %	4 %	3 %	3 %	2 %	2 %

Table 6.XIV: Shannon entropy for a Benes network N=64, k=32

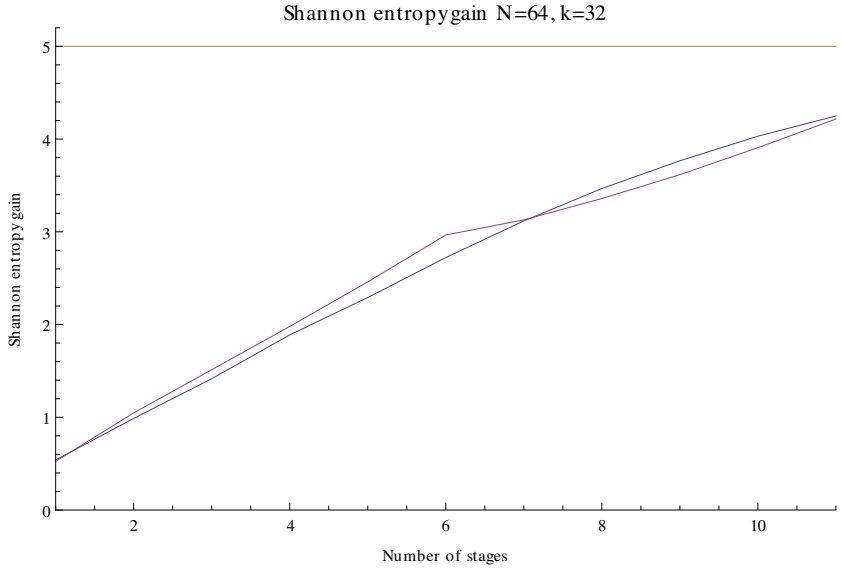

Figure 6.18: Shannon entropy for random and Benes networks N=64, k=32

Min-entropy for a random network N=64, k=32											
Entropy	0.52	0.91	1.22	1.51	1.79	2.09	2.32	2.51	2.76	2.91	3.09
%max	10 %	18 %	25 %	30 %	36 %	42 %	46 %	50 %	55 %	58 %	62 %
Variance	0.25	0.31	0.39	0.41	0.41	0.46	0.43	0.43	0.43	0.43	0.42
CI	0.07	0.07	0.08	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
CI pct	13 %	8 %	7 %	6 %	5 %	5 %	4 %	4 %	3 %	3 %	3 %

Table 6.XV: Min-entropy for a random network N=64, k=32

Min-entropy for Benes mixing N=64, k=32											
Depth	1	2	3	4	5	6	7	8	9	10	11
Entropy	0.51	0.85	1.20	1.52	1.85	2.15	2.30	2.53	2.68	2.93	3.13
%max	10 %	17 %	24 %	30 %	37 %	43 %	46 %	51 %	54 %	59 %	63%
Variance	0.25	0.31	0.39	0.38	0.41	0.46	0.44	0.41	0.44	0.51	0.43
CI	0.07	0.07	0.08	0.08	0.09	0.09	0.09	0.09	0.09	0.10	0.09
CI pct	13 %	9 %	7 %	6 %	5 %	4 %	4 %	4 %	3 %	3 %	3 %

Table 6.XVI: Min-entropy for a Benes network N=64, k=32

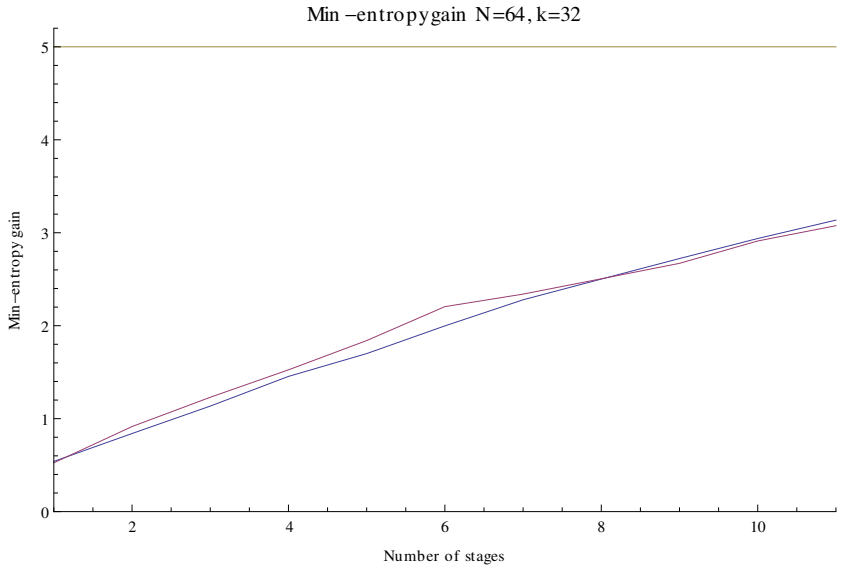

Figure 6.19: Min-entropy for random and Benes networks N=64, k=32

6.3 Network Interpretation

Such networks may be interpreted in various ways. The most direct interpretation of the network is the following.

Algorithm 7: Direct Network Execution

1. N parties get together to mix with new, untainted Tor identities;
2. The parties choose their initial ordering;
3. At the first stage of the network, parties execute the two-party switchbox shown in Figure 6.2 to create a blueprint transaction;
4. Parties sign the transactions and send them to the network to be confirmed;
5. When all transactions are confirmed, the parties refresh their Tor identities and proceed to the next step, openly declaring their new position in the mix. The new position is linked to the Tor identity. If two parties do not agree on the positions, the protocol is aborted;
6. The protocol finishes when all the switchbox protocols have been executed;

This protocol is more DOS resistant than most implementations of the blueprint approach. Although it may be interrupted by a single misbehaving user, the steps that have previously been completed are not wasted as some anonymity most likely already has been gained by most users. This can be seen in comparison to the blueprint approach where if even a single user refuses to sign the final transaction, the protocol must be aborted and the honest participants do not gain any anonymity. On the other hand, this protocol incurs important communication costs both for the Tor identity creation step and transactions step. The execution of a butterfly network above would create $Nlg(N)$ Tor identities and half that number of transactions on the blockchain. Furthermore, even if only two confirmations for each transaction were required before passing to the next step, the running time would be of the order of $\frac{lg(N)}{6}$ hours [3]. Additionally, since mining fees are expected for each transaction, the exchange amounts for every switchbox must decrease uniformly if only slightly, which may lead to problems if a user misbehaves

3. This is based on 10-minute average confirmation time.

and the protocol must be restarted.

Using the same networks as above, one may create a single blueprint transaction instead of a series of transactions. Step 4 in the protocol of Algorithm 7 may be replaced by both parties agreeing on their respective positions. After a Tor identity refresh, each party takes the position it has just agreed to and the network continues. Once the network is finished, each party can create a new identity and announce its position and output address, and a transaction blueprint can be constructed and signed as before. Thus, the number of transactions sent to the blockchain is reduced to one. This also reduces the problem of the transaction fees since they only need to be paid once and are $Nlg(N)$ times less for each party if we assume a fixed cost for any transaction on the blockchain. However, this approach suffers from all the disadvantages of the blueprint approach. We believe that this interpretation is not as interesting as others mentioned in the previous chapters as it is more complex and offers few advantages.

A third interpretation for the network is to send the results of each level of mixing to an untrusted third party who would create a transaction blueprint to be signed. Not only can the third party not leave with the money, that party does not gain any information that would not be accessible from the blockchain. This middle-of-the-road approach is less costly than the first interpretation but a denial of service attack halts the protocol at a specific stage. If this attack is not at the first stage, some anonymity is still gained. Similarly, it is more costly than the second approach but is less DOS susceptible. We believe that such interpretations may cohabit and satisfy different needs.

One can imagine excluding a misbehaving party and replacing that party with a new party that has not yet participated in the protocols. The operator of a mixing service may have a waiting list where people can join the protocol when a misbehaving party leaves. This suggests the idea of an endless protocol with a fixed number of parties participating in the protocol at any one time but being able to exit the protocol as the end of any step. Of course, such a modification would have privacy repercussions that are beyond the scope of this thesis. The modification could easily be integrated with random mixing in particular.

6.4 j-Party Switchboxes

The n-party mixing through encryption described in the previous chapter can be used to replace two-party switchboxes with n-party switchboxes. Each switchbox requires more computation and communication rounds but these switchboxes enable anonymizing networks with reduced depth. For example, we can build a butterfly-like network that uses j-input switchboxes and which has depth $log_j(N)$ and width $\frac{N}{j}$ for a total of $\frac{Nlog_j(N)}{j}$ switchboxes. The Benes network thus has a depth of $2log_j(N)-1$ for a total size of $\frac{2Nlog_j(N)-N}{j}$.

We have seen in section 5.3.2 that three-party switchboxes can be feasibly implemented through encryption. We believe that networks using these switchboxes could be worth investigating.

6.5 Discussion

Networks of transactions benefit from not needing a trusted third party and being more DOS-resistant than other approaches to anonymize but they suffer from increased financial cost and latency in most implementations, in addition to being more complicated to set up. The increase in efficiency of structured networks is minimal considering the increased complexity and lower DOS resistance of such networks compared to random mixing. Furthermore, doubling the number of steps to go from a butterfly network to a Benes networks yields very little improvement, and we believe that these extra steps should instead be used to mix with a new set of participants. It appears that our hypothesis that enabling all $N!$ would produce a better permutation was incorrect. We do believe however that continuous random mixing is an option that is viable in the current state of Bitcoin usage.

While we consider that these structured networks are not worth using in place of other solutions in the context of Bitcoin, we might ask whether other mixing applications might benefit from this analysis, especially those application where DOS-resistance and the lack of a central party are priorities.

CONCLUSION AND FUTURE WORK

In the introduction, we motivated the study and use of anonymous payment systems to hinder abusive privacy infringements by governments and companies. Our belief that Bitcoin could be part of the solution to these problems motivated our efforts to make the cryptocurrency more anonymous.

The existence of security and privacy vulnerabilities in traditional mixers used in Bitcoin motivated our study of decentralized mixers. The survey of existing mixing protocols enabled us to develop a framework to classify and compare the resulting anonymity of various protocols. In addition to contributing an efficient 3-party mixing protocol to the blueprint mixing approach, we have developed the network of transactions framework in order to address the high DOS-vulnerability of blueprint mixing. Although our findings indicate that this method will probably not be used for Bitcoin, we hope that such a construction might be used for other anonymizing problems unrelated to Bitcoin, possibly as a variant to mixnets in certain situations.

Our survey of mixers currently in use as well as proposed solutions indicates that there exist accessible solutions to the security problem. While it does not address the privacy vulnerability, blueprint mixing using a facilitator could be implemented at low cost and would address the security vulnerability of today's mixers. Fortunately, the emergence of protocols such as *Coinjoin* and *Sharedcoin* indicates that the community might be moving in such a direction. While more work would be needed to find a truly efficient large-scale private decentralized mixer, we believe that the small-scale solutions such as our proposed 3-party mixing through encryption could be implemented to produce secure and private mixing today.

In future work, we would like to find efficient protocols to produce mixing that involves more than three parties. Furthermore, an analysis of networks of transactions including switchboxes that take more than two inputs might enable us to discover some networks that are more efficient at anonymizing their users.

BIBLIOGRAPHY

[1] Scott Aaronson. Quantum copy-protection and quantum money. In *Computational Complexity, 2009. CCC'09. 24th Annual IEEE Conference on*, pages 229–242. IEEE, 2009.

[2] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In *Proceedings of the 44th Symposium on Theory of Computing*, pages 41–60. ACM, 2012.

[3] Al Jazeera America. Timeline of Edward Snowden's revelations, Jan 2014. URL `http://america.aljazeera.com/articles/multimedia/timeline-edward-snowden-revelations.html`.

[4] Aljazeera America. Journalist Glenn Greenwald's partner challenges UK detention, Nov 2013. URL `http://america.aljazeera.com/articles/2013/11/6/david-miranda-challengesukdetentioninday1ofhearing.html`.

[5] Elli Androulaki, Ghassan Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in Bitcoin. *IACR Cryptology ePrint Archive*, 2012:596, 2012.

[6] Charles Arthur. Wikileaks under attack: the definitive timeline. URL, January 2010. URL `http://www.theguardian.com/media/2010/dec/07/wikileaks-under-attack-definitive-timeline`.

[7] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better: How to make Bitcoin a better currency. In *Financial Cryptography and Data Security*, pages 399–414. Springer, 2012.

[8] Václav E Beneš. Optimal rearrangeable multistage connecting networks. *Bell System Technical Journal*, 43(4):1641–1656, 1964.

[9] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free mix routes and how to overcome them. In *Designing Privacy Enhancing Technologies*, pages 30–45. Springer, 2001.

[10] Alex Biryukov, Ivan Pustogarov, and R Weinmann. Trawling for Tor hidden services: Detection, measurement, deanonymization. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 80–94. IEEE, 2013.

[11] The Bitcoin Foundation. Protect your privacy, Nov 2013. URL `http://bitcoin.org/en/protect-your-privacy`.

[12] Bitcoin Wiki. Block chain, Sep 2013. URL `https://en.bitcoin.it/wiki/Block_chain`.

[13] Bitcoin Wiki. Block hashing algorithm, May 2013. URL `https://en.bitcoin.it/wiki/Block_hashing_algorithm`.

[14] Bitcoin Wiki. Transactions, May 2013. URL `https://en.bitcoin.it/wiki/Transactions`.

[15] Bitcoin Wiki. Anonymity, May 2013. URL `https://en.bitcoin.it/wiki/Anonymity`.

[16] Bitcoin Wiki. Blind Bitcoin transfers, August 2013. URL `https://en.bitcoin.it/wiki/Blind_bitcoin_Transfers`.

[17] Bitcoin Wiki. Hardfork wishlist, May 2013. URL `https://en.bitcoin.it/wiki/Hardfork_Wishlist`.

[18] blockchain.info. USD market price history, May 2013. URL `http://blockchain.info/charts/market-price?timespan=all&showDataPoints=false&daysAverageString=1&show_header=true&scale=0&address=`.

[19] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, and Joshua A Kroll. Anonymity for Bitcoin with accountable mixes. *Preprint*, 2014.

[20] Danny Bradbury. How anonymous is Bitcoin?, Jun 2013. URL http://www.coindesk.com/how-anonymous-is-bitcoin/.

[21] Stefan A Brands. An efficient off-line electronic cash system based on the representation problem, 1993.

[22] Ernie Brickell, Peter Gemmell, and David Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 457–466. Society for Industrial and Applied Mathematics, 1995.

[23] Hasan Cam. Rearrangeability of (2 n-1)-stage shuffle-exchange networks. *SIAM Journal on Computing*, 32(3):557–585, 2003.

[24] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *Advances in Cryptology–EUROCRYPT 2005*, pages 302–321. Springer, 2005.

[25] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), February 1981.

[26] David Chaum. Blind signatures for untraceable payments. In *Crypto*, volume 82, 1982.

[27] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[28] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Advances in Cryptology CRYPTO 88*, pages 319–327. Springer, 1988.

[29] Good Chris. Secret service investigating purported ransom of Mitt Romney's tax returns. URL http://abcnews.go.com/blogs/politics/2012/09/secret-service-investigating-purported-ransom-of-mitt-romneys-tax-returns/.

[30] Martin Collier. A systematic analysis of equivalence in multistage networks. *Lightwave Technology, Journal of*, 20(9):1664–1672, 2002.

[31] George Danezis, Cedric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio coin: building zerocoin from a succinct pairing-based proof system. In *Proceedings of the First ACM workshop on Language support for privacy-enhancing technologies*, pages 27–30. ACM, 2013.

[32] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *Privacy Enhancing Technologies*, pages 54–68. Springer, 2003.

[33] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[34] John R Douceur. The sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2002.

[35] Charles Duhigg. How companies learn your secrets, Febuary 2012. URL `http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html?_r=1&`.

[36] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18. Springer, 1985.

[37] Matthew Elias. Bitcoin: Tempering the digital ring of gyges or implausible pecuniary privacy. *Available at SSRN 1937769*, 2011.

[38] Tim Fernholz. How digital currencies democratize tax evasion, Jul 2013. URL `http://qz.com/92842/how-digital-currencies-democratize-tax-evasion/`.

[39] Financial Action Task Force. Money laundering using new payment methods, 2010. URL `www.fatf-gafi.org/.../ml%20using%20new%20payment%20methods.pdf`.

[40] Michel Foucault. *Discipline & Punish*. Random House of Canada, 1977.

[41] Oded Goldreich. Foundations of cryptography: Basic tools. 2000.

[42] Shafi Goldwasser. Multi party computations: past and present. In *Proceedings of the Sixteenth annual ACM symposium on Principles of distributed computing*, pages 1–6. ACM, 1997.

[43] Glenn Greenwald. Government harassing and intimidating Bradley Manning supporters, Nov 2010. URL `http://www.salon.com/2010/11/09/manning_2/`.

[44] Ian Grigg. How digicash blew everything, Jan 1999. URL `HowDigiCashBlewEverything`.

[45] Mike Hearn. Re:p2p mixing, July 2012. URL `https://bitcointalk.org/index.php?topic=93390.msg1036406#msg1036406`.

[46] Michael Herrmann and Christian Grothoff. Privacy-implications of performance-based peer selection by onion-routers: a real-world case study using i2p. In *Privacy Enhancing Technologies*, pages 155–174. Springer, 2011.

[47] Susan Hohenberger. *Advances in signatures, encryption, and e-cash from bilinear groups*. PhD thesis, Citeseer, 2006. URL `http://groups.csail.mit.edu/cis/theses/hohenberger-phd-thesis.pdf`.

[48] Nick Ioannou. *Internet Security Fundamentals: Practical Steps To Increase Your Online Security*. Boolean Logical Ltd, 2013.

[49] Stanislaw Jarecki and Vitaly Shmatikov. Handcuffing big brother: an abuse-resilient transaction escrow scheme. In *Advances in Cryptology-EUROCRYPT 2004*, pages 590–608. Springer, 2004.

[50] Stanisław Jarecki and Vitaly Shmatikov. Probabilistic escrow of financial transactions with cumulative threshold disclosure. In *Financial Cryptography and Data Security*, pages 172–187. Springer, 2005.

[51] Jay1337. Why ASIC's should not be the future of crypto currencies, Oct 2012. URL https://bitcointalk.org/index.php?topic=119460.0.

[52] Casey Johnston. Denied for that loan? soon you may thank online data collection., October 2013. URL http://arstechnica.com/business/2013/10/denied-for-that-loan-soon-you-may-thank-online-data-collection/.

[53] Kristján Valur Jónsson, Gunnar Kreitz, and Misbah Uddin. Secure multi-party sorting and applications. Cryptology ePrint Archive, Report 2011/122, 2011. http://eprint.iacr.org/.

[54] Ari Juels, David Molnar, and David Wagner. Security and privacy issues in e-passports. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 74–88. IEEE, 2005.

[55] Dan Kaminsky. Black ops of tcp/ip 2011, Aug 2011. URL http://dankaminsky.com/2011/08/05/bo2k11/.

[56] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007. ISBN 1584885513.

[57] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In *Advances in Cryptology-EUROCRYPT 2004*, pages 571–589. Springer, 2004.

[58] Nobuhiro Kiyotaki and Randall Wright. On money as a medium of exchange. *The Journal of Political Economy*, pages 927–954, 1989.

[59] Robert Koenig, Renato Renner, and Christian Schaffner. The operational meaning of min- and max-entropy. IEEE Trans. Inf. Th., vol. 55, no. 9 (2009), 2008.

[60] Jim Kouri. New report tracks ransom cash resulting from african piracy, November 2013. URL http://www.examiner.com/article/new-report-tracks-ransom-cash-resulting-from-african-piracy.

[61] Watson Ladd. Blind signatures for Bitcoin transaction anonyity, 2012.

[62] Alexander Lange. An overview of homomorphic encryption, May 2011. URL www.cs.rit.edu/~arl9577/crypto/alange-presentation.pdf.

[63] Brian Neil Levine, Clay Shields, and N Boris Margolin. A survey of solutions to the sybil attack. *University of Massachusetts Amherst, Amherst, MA*, 2006.

[64] Gregory Maxwell. Coinswap: Transaction graph disjoint trustless trading, October 2013. URL https://bitcointalk.org/index.php?topic=321228.msg3440187#msg3440187.

[65] Gregory Maxwell. Coinjoin: Bitcoin privacy for the real world, August 2013. URL https://bitcointalk.org/index.php?topic=279249.msg2983902#msg2983902.

[66] Gennady Medvinsky and Clifford Neuman. Netcash: A design for practical electronic currency on the internet. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 102–106. ACM, 1993.

[67] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. A fistful of Bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM, 2013.

[68] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from Bitcoin. In *IEEE Symposium on Security and Privacy*, 2013.

[69] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Advanced Encryption Standard*. Alpha Press, 2009. ISBN 6130268297, 9786130268299.

[70] Tyler Moore and Nicolas Christin. Beware the middleman: Empirical analysis of Bitcoin-exchange risk. *Financial Cryptography and Data Security*, 7397:455–466, 2013.

[71] Malte Möser. Anonymity of Bitcoin transactions. *Münster Bitcoin Conference*, 2013.

[72] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL `http://www.bitcoin.org/bitcoin.pdf`.

[73] Satoshi Nakamoto. Bitcoin v0.1 released, January 2009. URL `http://www.mail-archive.com/cryptography@metzdowd.com/msg10142.html`.

[74] Federal Bureau of Investigation. Bitcoin virtual currency: Unique features present distinct challenges for deterring illicit activity, April 2012. URL `http://www.wired.com/images_blogs/threatlevel/2012/05/bitcoin-FBI.pdf`.

[75] Tatsuaki Okamoto and Kazuo Ohta. Disposable zero-knowledge authentications and their applications to untraceable electronic cash. In *Advances in Cryptology CRYPTO 89 Proceedings*, pages 481–496. Springer, 1990.

[76] G. Orwell. *1984*. 1st World Library - Literary Society, 2004. ISBN 9781595404329. URL `http://books.google.ca/books?id=w-rb62wiFAwC`.

[77] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. *Version 0.34 Aug*, 10, 2010.

[78] Richard T Preiss. The consequences of anonymous access to the financial payments system. *Journal of Money Laundering Control*, 2(1):7–13, 1998.

[79] Fergal Reid and Martin Harrigan. An analysis of anonymity in the Bitcoin system. Pre-print. Available as arXiv: 1107.452 a4v2 [physics. soc-ph], May 2012.

[80] Consumer Reports. Big brother is watching, Sep 2009. URL http://www.consumerreports.org/cro/money/consumer-protection/big-brother-is-watching/overview/index.htm.

[81] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[82] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology ASIACRYPT 2001*, pages 552–565. Springer, 2001.

[83] Dorit Ron and Adi Shamir. Quantitative analysis of the full Bitcoin transaction graph. *IACR Cryptology ePrint Archive*, 2012:584, 2012.

[84] Burton Rosenberg. *Handbook of financial cryptography and security*. CRC Press, 2010.

[85] Meni Rosenfeld. Using mixing transactions to improve anonymity, December 2011. URL https://bitcointalk.org/index.php?topic=54266.0.

[86] Krishna Sampigethaya and Radha Poovendran. A survey on mix networks and their secure applications. *Proceedings of the IEEE*, 94(12):2142–2181, 2006.

[87] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies*, pages 41–53. Springer, 2003.

[88] Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In *Information Hiding*, pages 36–52. Springer, 2003.

[89] SPIEGEL. 'follow the money': Nsa spies on international payments, September 2013. URL http://www.spiegel.de/international/world/

spiegel-exclusive-nsa-spies-on-international-bank-transactions-a-922276.html.

[90] Markus Stadler, Jean-Marc Piveteau, and Jan Camenisch. Fair blind signatures. In *Advances in Cryptology EurocryptŠ95*, pages 209–219. Springer, 1995.

[91] Edward Z Yang. Secure multiparty Bitcoin anonymization, July 2012. URL http://blog.ezyang.com/2012/07/secure-multiparty-bitcoin-anonymization/.

APPENDIX I

BLOCK GENERATION DIFFICULTY

The expected amount of work for creating a block is adjusted dynamically by the network so that the expected time between new blocks is 10 minutes. The target of the hash function's result is stored in a packed representation in each block as 256 bit hexadecimal number. This packed representation can be expanded into the real target if multiplied by $2^{8*(0x1b-3)}$ [13]. The *difficulty* of mining a block as the expected amount of work required to mine the next block compared to the minimum amount of work that could possibly be used to mine a block. This corresponds to

$$\text{difficulty} = \frac{\text{maximum target}}{\text{current target}}$$

where *maximum target* is $0x1d00ffff$ in packed form. Difficulty is dynamically adjusted by the network every 2016 blocks, that is about every 2 weeks. When this happens, every Bitcoin client multiplies the target by the percentage difference between $2016 * 10min$ and the time it took to generate the last 2016 blocks. This modification is bounded by a factor of 4 to prevent large changes in difficulty. This enables a decentralized change in difficulty that every peer can agree to.

The probability distribution of the expected time between blocks follows a Poisson distribution. Since this distribution has no memory, elapsed time since the last block has no impact on expected time to the next block. The total number of hashes produced by a mining computer per second is called the *hashrate* and is typically measured in MH/s [1]. The total hashrate of the network can be inferred from the difficulty and the time between blocks. In the first half of 2014, it was of the order of 10^7 MH/s.

1. Millions of hash per second.

APPENDIX II

MULTIPARTY RANDOMNESS

This section details how to securely create 2-party randomness, that is a random string that cannot be influenced by one or the other of two non-colluding parties. We can call the party with the lowest alphanumeric input address Alice and the other party Bob. To produce a single bit of multiparty randomness, Alice can commit to a random bit b_A and send the commitment to Bob. Bob can transmit b_B to Alice, at which point Alice reveals b_A. We then define $b := b_A \oplus b_B$. This protocol's output is as hiding and binding [1] as the commitment scheme that it uses.

This choice of randomness may be used to implement 2-party switchboxes. If $b = 0$ then Alice and Bob each send the funds to themselves whereas if $b = 1$, they switch. Alice then creates the transaction blueprint, signs it and sends it to Bob to be signed. Bob signs it and publicizes it to the network. Alice cannot influence b as long as the bit commitment scheme is computationally binding, and Bob cannot influence it as long as the bit commitment scheme is computationally hiding. Any attempt to deviate from the protocol results in the protocol aborting.

This can naturally be extended to produce j bits of randomness by making b a string of j bits. One efficient way for Alice to commit to b_A is by sending the hash of the value with some random bits $H(b_A||r)$ to Bob. The random bits are necessary so that the commitment cannot be searched through a brute-force attack.

In order for N parties to produce a random permutation, each party can in turn commit to a random permutation and later reveal it, in which case the composition of these permutations in the same order will produce a truly random permutation that cannot be influenced by any non-complete subsets of the participants.

1. Intuitively, a protocol is *hiding* if it is hard for Bob to guess Alice's commitment and is *binding* if Alice may not change her commitment. For a more technical description, see [41].

APPENDIX III

NETWORK OF TRANSACTIONS SIMULATION

In this section, we describe the simulation we used to determine the Shannon entropy and min-entropy of networks of transactions. We assume that 2-party switchboxes are used but the methodology could be generalized to j-party switchboxes. We start by considering that the adversary is a blockchain observer.

III.1 Blockchain Observer

We consider 4-party mixing with a butterfly network. The state of the system from the adversary's point of view can be illustrated as a column vector representing the probability distribution of Alice being at various positions. The positions of the parties can be labelled from 00 to 11. Since we consider without loss of generality that Alice occupies position 00 at first, the probability distribution of Alice's position can initially be represented as

$$D_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

which the adversary can interpret as Alice being in position 1 with 100% probability. We can check that the Shannon entropy and min-entropy at this point are $H_\infty(D_0) = H(D_0) = lg(1) = 0$ as expected since the adversary does not need any extra information to know where Alice is.

At each stage, each participant mixes with their neighbour as illustrated in Figure 6.3. This can be represented with the following matrix that is the same at all stages against a blockchain observer:

$$S = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

The resulting probability distribution for Alice's position from the adversary's point of view after this first stage is

$$D_1 = D_0 S = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ 0 \end{pmatrix}$$

The adversary can interpret this as Alice's position being 00 with $P = \frac{1}{2}$ and 01 with $P = \frac{1}{2}$. The Shannon entropy and min-entropy have grown as expected as $H(D_1) = -2\frac{1}{2} lg(\frac{1}{2}) = 1$ and $H_\infty(D_1) = -lg(\frac{1}{2}) = 2$.

The link permutations in a network can be expressed by an NxN binary matrix of rank N. In the $N = 4$ butterfly case, the first link permutation is represented by the following matrix

$$LP_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The resulting distribution after this link permutation is

$$D_2 = LP_0 D_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{pmatrix}$$

The final distribution is thus

$$D_f = SD_2 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \end{pmatrix}$$

In this case, both the resulting Shannon entropy and min-entropy of the distribution

are $H(D_f) = H_\infty(D_f) = 2$. We realize that as expected, each stage yields an improvement in anonymity. Furthermore, we note that both entropies are maximal and cannot be increased.

III.2 Mixing in the Presence of Adversaries

We now consider the situation where each party that is not Alice might be colluding with the adversary. The randomness generation described in appendix II ensures that a corrupt party may not influence the result of a coin toss. We denote the positions of corrupt parties with bold characters.

As noted previously, no anonymity is gained if Alice mixes with a party colluding with the adversary. In this case, one of two situations will happen, but the adversary will know which is which. For example if Alice is only mixing with one adversary, the first stage would be one of $S_1 = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ or $S_2 = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ each with probability $\frac{1}{2}$ but the adversary would know which situation it is.

We may simulate a possible outcome of a 4-party butterfly with the adversary's collaborator starting at position 01. In this example, the inputs that the adversary is involved in switch for the first stage S_0 but do not for the second stage S_1.

$$D_0 = \begin{pmatrix} 1 \\ \mathbf{0} \\ 0 \\ 0 \end{pmatrix}$$

$$D_1 = S_0 D_0 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 \\ \mathbf{0} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$D_2 = L P_0 D_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$D_f = S_1 D_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

In this case, the adversary can rule out both the first and second positions. Multiple adversaries are treated similarly with different coin tosses.

APPENDIX IV

TECHNICAL SPECIFICATIONS

IV.1 Bitcoin's Scripting Language

Bitcoin clients are written in various programming languages but the Bitcoin protocol itself uses a Forth-like scripting language called *Script* to describe transactions. A *script* is a list of instructions recorded with each transaction that describes how the next person who wants to spend the coins can access them in order to spend them. For example, a simple script would not impose any conditions other than the signing of the transaction by the sender for the receiver to be able to redeem the coins and spend them. More complex scripts may enable a transaction to only be validated if certain conditions are met, e.g. a certain date has passed. Although *Script* is intentionally not Turing-complete [1], it enables the writing of transactions using a very large set of conditions. One notable functionality is the use of transactions with multiple senders and receivers. The only fixed requirement in all transactions is that all of the senders have signed the transaction.

IV.2 Transaction Example

This example transaction T is taken from the Bitcoin wiki [14]. It represents a *one input one output* transaction for the amount of 50 btc.

Input:

Previous tx: $f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04$ $470b9a6$
Index: 0
scriptSig: $304502206e21798a42fae0e854281abd38bacd1aeed3ee3738d9e1446618$

1. Turing-completeness is a property of a programming language in which anything that is computable can be computed.

$c4571d10$
$90db022100e2ac980643b0b82c0e88ffdfec6b64e3e6ba35e7ba5fdd7d5d6cc8d25c$
$6b241501$

Output:

Value: 5000000000
scriptPubKey: $OP\_DUP \quad OP\_HASH160$
$404371705fa9bd789a2fcd52d2c580b65d35549d$
$OP\_EQUALVERIFY \quad OP\_CHECKSIG$

The input specifies *txout* that is being spent in this transaction. The *Previous tx* is the hash of the previous transaction being referenced. The index represents the specific output number of the referenced transaction. ScriptSig is the first part of the script described later.

The output contains instructions for sending the coins. *Value* is the number of Satoshi [2] sent to the output address. The second part of the script is *ScriptPubKey*. If there were a second output address or a change address, it would be specified here. Any amount that is not redeemed in an output is automatically considered a transaction fee that is paid to the miner for validating the transaction.

The script is written in the Bitcoin-specific *Script* scripting language and consists of two parts, a signature (scriptSig) and a public key (scriptPubKey). The first is necessary to verify the integrity of T's inputs and is executed first. The values of different variables are left on the stack and when *scriptPubKey* is executed, it can use these values. The script *scriptPubKey* is where the signature of the sender takes place and where special conditions for the transaction may be coded. The transaction is accepted by the network if the execution of *scriptPubKey* returns the value *true*.

2. A *satoshi* is $\frac{1}{100,000,000}$ of a Bitcoin, the smallest denomination there is.