UNIVERSITY OF TWENTE.

Department of Computer Science
Services and Cybersecurity (SCS)

Cyber Security & Robustness

Master Thesis

User privacy in cryptocurrencies

Arne de Roode

Supervisors Dr. E. Weitenberg (TNO)
R. Montalto, MA (TNO)
Dr. M. Everts
Prof. dr. R. van Rijswijk - Deij

January 13, 2022

Arne de Roode
User privacy in cryptocurrencies
January 13, 2022
Supervisors: Dr. E. Weitenberg, R. Montalto, MA, Dr. M. Everts and
Prof. dr. R. van Rijswijk - Deij

University of Twente
Services and Cybersecurity (SCS)
Department of Computer Science
Drienerlolaan 5
7522 NB Enschede

Abstract

Cryptocurrencies see growing interest, both in real-world adoption and research. In this study several gaps in current research on privacy and anonymity in cryptocurrencies are addressed. First, common and broadly acceptable definitions of privacy and anonymity are lacking. Therefore, we gather definitions from current literature, and devise definitions to be used. Second, there is no common standard for evaluating the level of privacy that is offered to a user in a cryptocurrency system. Therefore, in this work we develop a framework for evaluating and comparing the user-privacy preservation of cryptocurrencies. We then assess this framework using expert reviews. Furthermore, we apply it to multiple cryptocurrencies. Dash is one of these cryptocurrencies, and is sometimes considered a 'privacy-coin'. In this study, Dash is scrutinized to discover whether it lives up to its debated 'privacy-coin' status. Multiple issues with its privacy-feature, and resulting vulnerabilities, are reported. Throughout this work many avenues for future research are uncovered, spanning each of the addressed topics: definitions, an evaluation framework, and the Dash cryptocurrency.

Acknowledgement

Throughout creating this work I was supported by many. I hereby express my gratitude to everyone who contributed, although a few stand out.

First and foremost I want to thank my TNO supervisors, who both contributed a lot to this work. They invested time in weekly talks, guided me, came up with ideas, and provided feedback on my work and thesis. Meanwhile, Maarten kept me on the academic track, and also provided guidance, ideas, and feedback; which I am thankful for as well. Furthermore, I thank Roland for taking part in the examination committee.

I am also grateful to Thijmen Verburgh, Bart Marinissen and the cryptocurrency developer[1], who were willing to invest time in thorough reviews of the framework I developed. They all provided valuable insights which were incorporated in this work.

Finally, I want to thank my wife and family for supporting and motivating me through showing their interest, and being an occasional 'rubber-duck'. Also, I thank Lucas for the good time we had, and for sparking some new insights.

[1] Who rather stays anonymous.

Contents

1 Introduction

Distributed Ledger Technologies (DLT), and among them specifically blockchain, are growing fast in terms of applications and adoption. Countless potential blockchain applications have been explored [1, 2, 3, 4, 5], and especially for application to cryptocurrencies, distributed ledger technologies have attracted interest of many researchers and developers. This materializes in the manifold blockchain based cryptocurrencies, token systems, and smart contract platforms that have arisen, which all started with the development of Bitcoin, released in 2009 [6]. The real-world value of these applications is exemplified by the call for adoption of Bitcoin as an official currency in countries (e.g. El Salvador), and the usage of blockchain in huge markets such as logistics (e.g. TradeLens[1]).

Cryptocurrencies are a type of digital currencies that use cryptography, protocols and algorithms to fulfill the functionality of a regular currency, which includes storing value and executing transactions. As cryptocurrencies deal with transfer of value, and by now also have large monetary value, security is crucial. Moreover, for a system of value transfer, some level of privacy is also desirable. Privacy and anonymity have been topics of scrutiny in cryptocurrencies, notably blockchain-based cryptocurrencies have undergone various privacy analyses (e.g. [7, 8, 9, 10, 11]). Research on privacy and anonymity is also important for governments and law enforcement, since cryptocurrencies are used by criminals as well. Moreover, increasing adoption and usage in day-to-day life make privacy considerations relevant for regular users.

Many technologies to improve user privacy in cryptocurrencies have been developed, and, on the other hand, multiple vulnerabilities and analyses that reduce user privacy have been discovered. Much of the research in this area is covered in [7] and [8]. Moreover, for this study we generated an overview of the state-of-the art regarding technologies and attacks as well; which can be found in Appendix A.

A common problem in current research on privacy and anonymity in cryptocurrencies is the lack of common definitions and methods. In this research, we aim to fill this gap by gathering interpretations of these concepts from literature and proposing definitions to be used in future research. Moreover, we develop a method to analyze and compare cryptocurrency systems regarding privacy. This method, which

[1] https://www.tradelens.com/

will consist of an analysis framework, will contribute to developing systematic, meaningful and replicable comparability for cryptocurrencies regarding privacy. The proposed framework is subsequently evaluated using expert reviews, and applied to several cryptocurrencies. Application of the framework reveals the differences between cryptocurrencies regarding user privacy and anonymity, and can serve as a starting point for future research.

Dash is a cryptocurrency that is by many considered to be a so-called privacy-coin, a cryptocurrency with a focus on privacy. Therefore, Dash is one of the cryptocurrencies to which the framework is applied in this work, and Dash is found to only differ slightly from Bitcoin regarding privacy. As such, the framework triggers further research into the privacy features of Dash, to uncover whether Dash should be seen as a privacy-coin. In this study, first, BlockSci[12] is employed to get insight in the usages of Dash's privacy features. Second, various issues in Dash's privacy features are discovered, which are scrutinized and reported. Moreover, the cost and impact of privacy vulnerabilities resulting from these issues are discussed.

In the remaining part of this introductory chapter blockchain technology and its application in cryptocurrencies will be briefly explained. Furthermore, the objectives and contributions of this research will be summarized. Finally, an overview of the upcoming chapters will be presented.

1.1 Blockchain and cryptocurrencies

This section will introduce blockchain technology and some of its crucial components like hash functions and consensus mechanisms. Moreover, the application of this technology in cryptocurrencies will be elaborated.

1.1.1 Distributed Ledger Technology

Blockchain is a specific type of Distributed Ledger Technology (DLT). DLT has been defined in many ways, as elaborated in [13]. Following [14], a DLT is essentially a technology that enables a distributed, meaning (geographically) dispersed and decentralized, group of nodes to maintain and agree upon a shared ledger. The ledger is an append-only database of records, in which no previously added records can be altered or removed. The ledger is shared, meaning that it is copied and locally kept by each of the nodes partaking in the DLT system. To maintain the ledger, the nodes must share new records with each other. Therefore, the DLT will entail some type of (peer-to-peer) communication network. The nodes agree upon

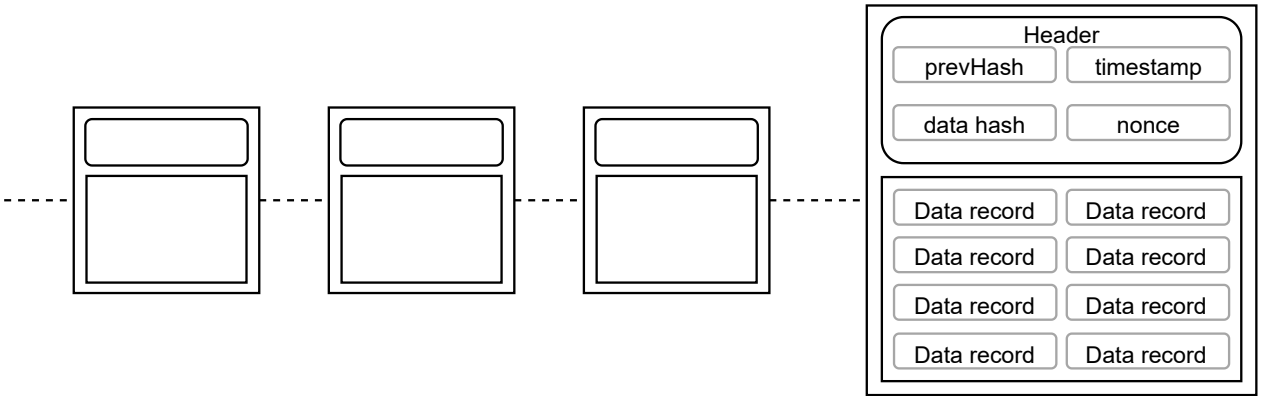

Fig. 1.1.: Adding a block to the blockchain entails filling the block and its header and connecting the block to the chain by including the hash of the most recent previous block (prevHash).

the state of te shared ledger by means of a consensus mechanism that the DLT must provide. The DLT must function well even if some participating nodes behave maliciously. Different DLTs have been introduced, e.g. blockchain[6], the tangle[15], and hashgraph[16].

Blockchain is a type of DLT, providing the characteristics that have been outlined above. In this case, the ledger is made up by a chain of blocks where each block consists of a number of records. The blockchain is locally stored by all participating nodes, which are interconnected in a peer-to-peer network. New records are shared among the nodes, and they are added to the ledger by aggregating them in a block, and then linking the newly generated block to the previous block, this is shown in figure 1.1. Note that header fields, as shown in figure 1.1, may vary per blockchain system; availability of certain fields is implementation-dependent. When a node adds a new block to the blockchain, it must check the validity of the block and its contents. Which of the nodes gets to introduce a new block to the blockchain depends on the rules of the DLT and its consensus mechanism. For example, when Proof-of-Work (PoW) is employed for consensus, the nodes race to find a valid block first, and whichever node finds it first appends it and broadcasts it to the others. How finding a valid block to be appended is done, will be elaborated in Section 1.1.4.

1.1.2 Blockchain types

A division is made in blockchain based DLTs depending on who has permission to read and write. Common divisions in blockchain systems are public versus private and permissioned versus permissionless systems. In a public blockchain, anyone

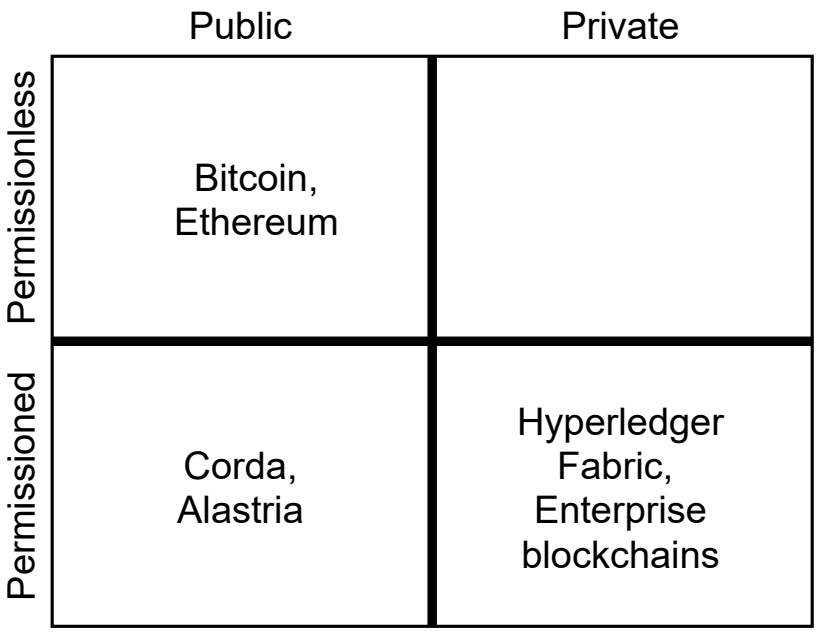

Fig. 1.2.: A classification of types of blockchain systems.

can access the blockchain, whereas in private blockchains only authorized users can. Similarly, in permissionless blockchain systems, anyone can contribute to the blockchain, whereas in permissioned systems only authorized users can. These two divisions can be used to construct a graph with four quadrants, in which blockchain systems can be plotted, this is shown in figure 1.2. Private permissionless seem somewhat counter-intuitive and systems are usually not meant for this combination. Still, they could be easily deployed by using public permissionless systems in a shielded environment, such as a corporate network. This happens in practice when test networks are deployed on private networks. For the other quadrants, a few example systems are noted.

1.1.3 Hash functions

The ordering of blocks in the chain is ensured by references between the blocks. Each new block must reference the previous block, which is done by including the hash of the previous block in the new block. A hash is computed using a hash function, which is a one-way function that takes some input bytes and produces a unique, fixed-length, irreversible output. Unique means that for each different set of input bytes, the probability that the same hash is produced is negligible. Irreversible entails that given a hash, it is computationally infeasible to compute what the input bytes were. A good hash function is collision resistant (computationally hard to find

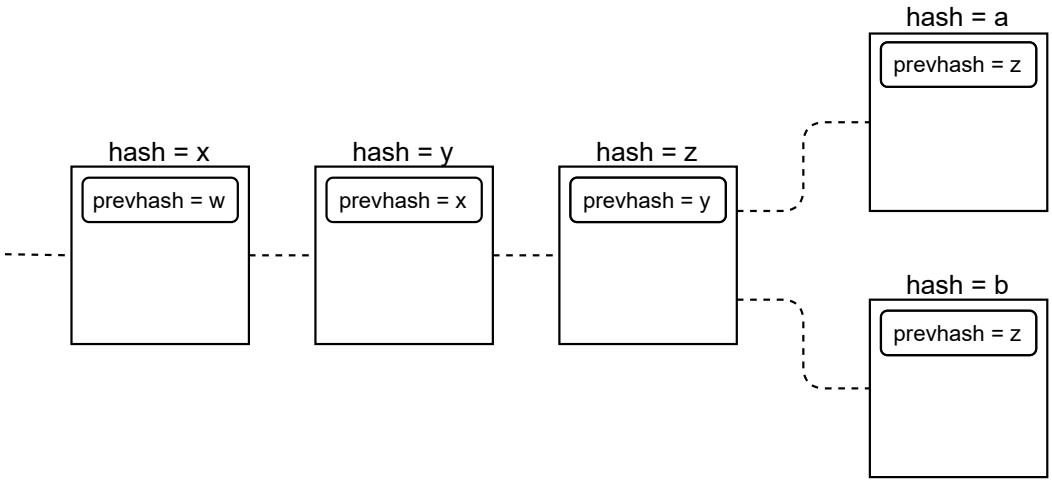

Fig. 1.3.: Blockchain fork: A fork occurs when multiple different new blocks extend the chain from the same previous block.

two inputs that hash to the same value), pre-image resistant (computationally hard to find the input given the output), and second pre-image resistant (computationally hard to find a second input that results in the same output). With these properties, it becomes infeasible to change a block after it has been hashed, because if one would alter anything in the content of the block, this would become immediately visible in a changed hash. A changed hash in a block that is not the most recent one would invalidate the chain, as such the network would not accept it. Moreover, a (valid) changed hash in the most recent block would cause a fork in the chain, because now there are two different most recent blocks that can be used as endpoints to further build the chain. Accidentally, or by malicious intent, forks may happen in blockchains. However, these conflicts are resolved using the employed consensus mechanism. A fork can happen when two blocks are generated and propagated at approximately the same time, then two blocks that both build on the same previous block may co-exist for some time. A blockchain fork is visualized in Figure 1.3.

Hash functions are also used to hash the data in the blocks, and the hash of the data is usually included in the block header. To construct the hash of all data records in the block, a so called Merkle tree is utilized. In a Merkle tree, data records are hashed, and their hashes are then combined (rehashed) in a binary tree structure, until only one hash remains: the Merkle root. This Merkle root is stored in the block header. Merkle trees provide storage efficiency and easier verification of data integrity.

1.1.4 Consensus mechanisms

Given there are competing sub-chains in a blockchain, there must be a mechanism to come to consensus over which chain is accepted as valid, since chain branches may include conflicting records. Several mechanisms and algorithms are used to achieve these, the two most well-known consensus algorithms will be briefly introduced.

Proof-of-Work

To avoid chain forks, only one node must be allowed to add a block at a time. When proof-of-work is used, the node that can first present a proof that they have done some computational work, will be able to add a new block to the blockchain. This means that for each addition, all nodes in the network that want to add a block have a competition to get that proof. In blockchain systems, proof-of-work is calculated by hashing the content of a block, together with a nonce (number used once), where the resulting hash must satisfy a certain threshold. Usually, the value of the hash must be lower than some target value, and therefore the block (+ nonce) hash must have a number of leading zeros. A node slightly changes the content of a block every time the hash does not satisfy the threshold. To get different hash values, the nonce or the contents of the block may be changed (e.g. incremented or shuffled respectively). This process is repeated until a valid block and hash are found. The repetition of slightly changing the block and updating the hash is the computational work (called mining), of which the resulting hash (that satisfies the threshold) is the proof. Nodes that are generating blocks and computing these hashes are called miners.

The target value, which determines the difficulty, must be chosen carefully such that it will not be too easy to find a block. Since, if it were too easy, many valid blocks could be generated simultaneously leading to undesirable forks. Many blockchain systems implement algorithms to adjust the difficulty value to the available hashing power. If available computation power increases, the difficulty does as well, and vice versa.

Still, accidentally forks may happen because blocks are found simultaneously. Therefore, nodes are incentivized to always build on the longest chain. Thus, when a fork happens, nodes will build on the branch they observed first, and as soon as one of the branches gets a new block, all nodes will switch to working on the longest chain. It is unlikely that blocks will be added to both branches simultaneously because of a likely asymmetry of computing power and because of the randomness involved in finding a correct proof of work. To be sure that records in a block are final, nodes

should wait for some time, until a number of blocks have been added on top of the considered block. At that point it becomes very unlikely that there is another branch that could take over the current one.

To be able to overtake the main chain with a conflicting branch a lot of computing power will be required. In fact, this may only succeed if at least 51% of the hashing power is controlled by the malicious actor trying to invalidate the main chain. All the work that is done on a branch that is not the main chain is wasted, although some blockchain systems, like Ethereum, try to avoid this wastage by linking these branches back to the main chain.

Proof-of-Stake

Another algorithm that is widely utilized to reach consensus is Proof-of-Stake (PoS). This mechanism addresses the problem of huge energy consumption required in PoW. Instead of a competition between nodes to add the next block to the chain, one node is picked randomly to complete this task. For nodes to be eligible for adding blocks, they must *stake* some value; in a cryptocurrency this would be (a portion of) their coins. This means they must prove their ownership of that value, which demonstrates they have an incentive to contribute to the blockchain in an honest manner, since they do not want that value to diminish. Among the nodes that have staked some value, a node will be picked in a pseudo-random fashion. The probability that a node is chosen depends on the amount of staked value. This process repeats every time slot, as such one block is added per time slot, which has an implementation dependent duration.

1.1.5 Cryptocurrencies

Blockchain is applied in countless cryptocurrencies to create a value exchange system. In this application, (often public) blockchains serve the purpose of a ledger to store transaction information. Users of the cryptocurrency can issue transactions, which are propagated through the peer-to-peer network to all nodes. These will validate transactions, and aggregate them in a block to be added to the blockchain. The nodes that work on extending the blockchain (e.g. miners for PoW or stakers in PoS) are generally rewarded by receiving some value for each block they add. This also creates an incentive for nodes to act legitimately, so their blocks will be accepted by others and as such they may claim their reward. Moreover, when issuing a transaction, users usually include a fee which can be claimed by the node including

the transaction in a block. This fee creates another incentive for block-generating nodes to support the system by including legitimate transactions.

By observing the blockchain and combining transactions involving a certain user (or their address), one can find information like the current balance. In practice a user is often represented by one or more addresses, which acts as a pseudonym for the user. A user's balance is then computed by adding the individual balances of these addresses together.

Blockchain was first applied as a cryptocurrency in Bitcoin[6], which to date is the most valued cryptocurrency in terms of market-capitalization[2]. In Bitcoin, users generate a cryptographic keypair (a public and private key), and can then receive bitcoins using a hash of their public key as an address. The funds they received on an address can be spent in a transaction by signing that transaction with the corresponding private key. Thus, to spend coins users must be able to generate a valid digital signature which proves ownership of the coins. Funds received at an address but not yet spent in a transaction are called Unspent Transaction Outputs (UTXOs). UTXOs must always be fully spent in a transaction. If only a part of the funds is required for a payment, the remaining funds must be sent to another address using an additional transaction output. It is possible to combine multiple inputs and outputs in one transaction. Inputs in a transaction are the UTXOs that are consumed by that transaction, and outputs to a transaction are the newly generated UTXOs. The total value of inputs of a transaction (minus some transaction fee) must be equal to the total value of the outputs. The concept of UTXOs can be used to generate a transaction graph, a graph based on blockchain data that can be used to trace where coins are going. In this graph, transactions can be represented by vertices and inputs and outputs by directed edges. An example of such a graph can be seen in Figure 1.4.

Multiple transaction types are possible in Bitcoin, paying the the hash of a public key as described here is one of the most commonly used types. It is also possible to make more complex transactions utilizing scripts.

In Bitcoin, all the transactions and their content is visible on the blockchain. Anyone can observe how many funds are at an address, however, only the owner can spend these funds with their private key. The publicity of transactions has been shown to affect the privacy and anonymity of users[17, 18]. On the other hand, many systems have been suggested to either improve Bitcoin-like blockchains or introduce completely new systems that are focused on improving privacy or anonymity (eg. [19, 20, 21]).

[2] `https://coinmarketcap.com/`, July 15, 2021

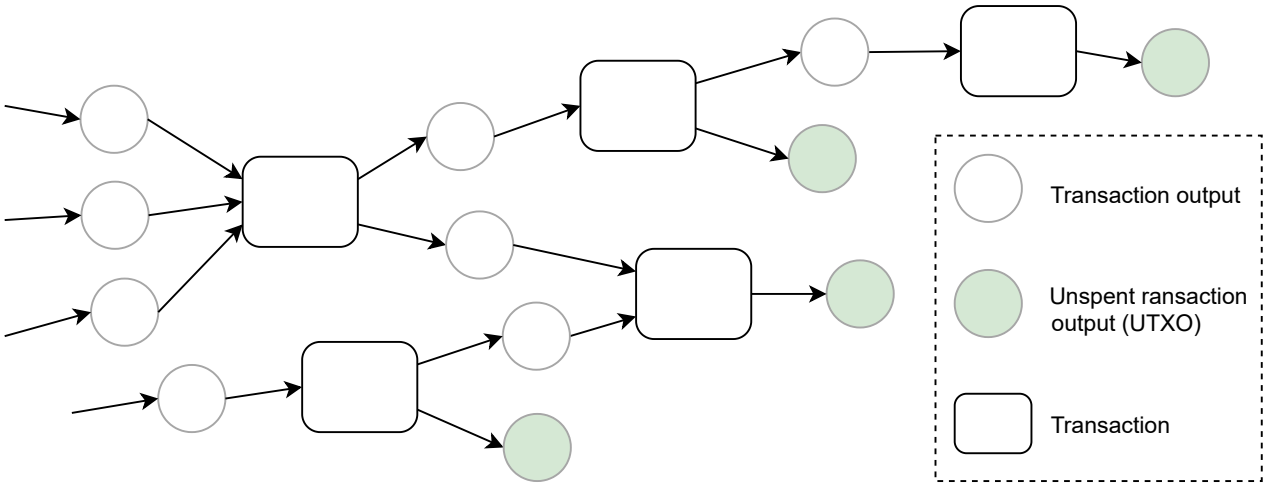

Fig. 1.4.: Transactions from the blockchain can be structured and visualized in a transaction graph, UTXOs are highlighted.

Examples of other blockchain based cryptocurrencies are Ethereum, Litecoin, Dash and Monero. Each of these have their own purposes and goals, however, they all use a public blockchain which affects the privacy of their users.

1.2 Objectives & contributions

The general objective of this study is to gain more insight in user privacy in cryptocurrencies through filling gaps in the literature on this topic. Therefore, we make an effort to develop definitions and a methodology for evaluating user privacy in cryptocurrencies, while also summarizing available literature on these topics. Moreover, Dash is analyzed since in the first part of this study it arose as a cryptocurrency of interest. The objective in analyzing Dash is to discover whether it can live up to the 'privacy-coin'-label that it has. These goals are summarized in the following research questions:

1. How are privacy and anonymity defined in the context of cryptocurrencies? Can a comprehensive definition be developed that is useful across disciplines?
2. How can privacy of users in cryptocurrency systems be evaluated? How may a framework to guide this evaluation be developed and validated?
3. How (well) does the Dash cryptocurrency protect the privacy of its users? Can the privacy features it offers be exploited?

The contributions that result from answering the questions above in this research are listed below.

- Common definitions for privacy and anonymity in the field of cryptocurrencies.
- A framework to evaluate cryptocurrencies regarding the privacy of their users and an evaluation of that framework through expert reviews.
- Description and analysis of the Dash cryptocurrency system and its privacy features.
- Exploits of the privacy feature implementation that the Dash cryptocurrency offers to its users.

1.3 Structure

After this introduction, we first develop a privacy and an anonymity definition in Chapter 2. When definitions are established, we design a framework for the evaluation of user privacy in cryptocurrencies in Chapter 3. After evaluating and applying this framework we dive deeper into the Dash cryptocurrency in Chapter 4. Moreover, we analyze Dash and the privacy feature it offers in Chapter 5, while we discuss vulnerabilities of Dash's privacy feature in Chapter 6 and Chapter 7. Finally, in Chapter 8, this study is concluded and recommendations for future research are provided.

2 Anonymity & privacy in DLT systems

Privacy and anonymity are related concepts that are sometimes hard to distinguish[22]. They are not the same thing, even though in the literature they are often mixed up. Some see anonymity as part of privacy (e.g. [23, 24]), some perceive privacy as part of anonymity (e.g. [25]), and some perceive them as equivalent (e.g. [26]). On the other hand, some studies also clearly differentiate between them: "one [Privacy"] is about hiding the content, and one [Anonymity] is about hiding who is saying it"[22]. The first aim of this chapter to get understanding of the used definitions (Section 2.1), and clearly distinguish privacy and anonymity. Secondly, we present definitions for anonymity and privacy to be used in this research in Section 2.2. And finally, the implications of these definitions for DLTs are examined in Section 2.3 and we conclude the chapter in Section 2.4.

2.1 Current literature definitions

Privacy and anonymity are interpreted in different ways in recent literature on DLTs; no comprehensive and common definition is employed for either. Availability and adoption of good definitions can help in communication about these topics, and will allow for better comparison between different technologies. In this section, anonymity definitions found in the literature will be compared by aligning them with anonymity terminology provided in [27]. Conversely, privacy is often not defined when used in literature on DLTs. Available definitions will be discussed in an attempt to find a common definition, although it has already been suggested no "unified formal privacy definition" exists[10]. First starting definitions based on [27] and the Cabmridge dictionary are introduced, which are used as a ground for comparison when analyzing the literature definitions. Then, studies considering mainly privacy in cryptocurrencies will be discussed, after which studies focusing mostly on anonymity are examined. Finally, resulting definitions for privacy and anonymity are proposed.

2.1.1 Initial anonymity and privacy definitions

Taken strictly, anonymity is *the situation in which someone's name is not given or known*[28]. Somewhat less strict and more applicable to real-life scenarios is the definition provided in [27], a proposal for terminology. This proposal defines anonymity and pseudonymity, among others, in the context of a communication network with senders and receivers, which is also a useful setting in the context of DLT systems. The anonymity definition is as follows:

Definition 1. Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.

An anonymity set is a group of subjects that exhibit similar behavior, and therefore can not be distinguished from each other based on their behavior. Thus, as an example, a distinction is made between sender - and receiver anonymity. A sender is only anonymous in the group of senders, conversely a receiver is only anonymous in a group of receivers.

For privacy, the definition provided in the Cambridge dictionary [29] will be taken as a starting point.

Definition 2. Privacy is someone's right to keep their personal matters and relationships secret; the state of being alone.

In a DLT system this entails that a user has the right to hide their actions and connections from other users. It should be noted that privacy is about the user, technology does not experience, enjoy or benefit from privacy, whereas its user does.

2.1.2 Privacy-oriented studies

In [30] the lack of a common privacy definition is noted, and it is suggested that privacy needs to be strongly defined using formal privacy definitions, such as differential privacy. Such a definition would make the 'privacy-preserving' property of a blockchain system more meaningful. The remainder of this section discusses some of the privacy and anonymity definitions encountered in current literature.

Rather early in the discussion on what privacy means in the context of cryptocurrencies, a study on privacy in Bitcoin was published [31]. In this work, privacy is defined using two notions: *activity unlinkability*, which means that an adversary who picks a target user cannot identify two (different) addresses or transactions that belong to that user; and *profile indistinguishability*, which means that an adversary cannot "reconstruct the profiles of all the users that participate" in public Bitcoin activity

(receiving and sending transactions). A profile in this context is the aggregation of addresses or transactions. The notions that are used in this study to characterize privacy are requirements for Bitcoin users to enforce their privacy as defined in Definition 2. Privacy in this study is not explicitly defined, although the notions and the way privacy is addressed suggest that it is not seen strictly as a right.

In [8], which is a survey on anonymity and privacy in systems like Bitcoin, definitions of privacy and anonymity are very brief. They are distinguished as privacy being about content, and anonymity being about the owner of the content. Privacy means no-one can see specific content, for example the emails in your secured account, whereas anonymity means others do not know a certain action or some specific information pertains to you, e.g. your voting ballot during elections. Anonymity is further defined as to entail "being unidentifiable and untraceable". Moreover, it is noted that anonymity is something often desired by criminals, as accountability becomes impossible when anonymity is guaranteed. Conversely, anonymity may have legitimate purposes as well, for example one might desire to be anonymous when voting in elections. The definition of anonymity in this research seems a bit more general than Definition 1. However, the definition in this research can be reformulated such that the resemblance is clear: "anonymity means others do not know certain actions or information pertains to you out of a set of people to who it could belong, and it entails being unidentifiable and untraceable within that set". It may seems like this is narrowing the original definition, however, it should be noted that a subject is always in some set. For example, this may be a set of senders in a communication network, or the set of all humans, or the set of all computers. As a result, the definitions of anonymity in [8] and [27] (Definition 1) are similar. The examples that the authors of [8] provide, highlight the dual-use aspect of anonymity, it is useful in morally good - as well as morally bad use-cases. The definition of privacy in [8] is somewhat different from Definition 2. Here, privacy is not defined as someone's right, rather it is defined as concealing things from others, which is an expression of the right to keep things secret (Cambridge, Definition 2). This is a regularly encountered issue in literature, where authors assume privacy to be equal to concealment of things, whereas actually concealing things is merely a tool to exercise privacy as a right.

A similar approach to defining privacy is taken in [7], which is a literature review on the privacy in cryptocurrencies. In this research, privacy is defined in the context of cryptocurrencies as "the ability to perform private transactions", where private transactions must have confidentiality (of amounts) and anonymity (hiding sender and receiver). Private transactions as defined here are in fact a tool to exercise privacy in the context of cryptocurrencies, thus the ability to perform these

transactions should be seen as a tool as well. Again, privacy is not defined as a right but rather in terms of an action that may protect that right, and as such it does not match well with Definition 2. Anonymity in [7] is only briefly mentioned to be hiding the senders and receivers. This is in line with Definition 1: a sender can hide in the set of senders and a receiver can hide in the set of receivers, where senders and receivers are then the respective anonymity sets. The anonymity definition in [7] can be seen as a more specific version of Definition 1.

In [24], several privacy protection mechanisms applied in blockchain systems are discussed. Although this research discusses privacy and anonymity, no definitions are provided. However, *identity privacy* and *transaction privacy* are two concepts that are introduced. Identity privacy is about the relation between a user's real identity and the address they own on the blockchain. If this relationship is revealed, then users lose privacy. Transaction privacy is about the transactions stored-on chain and the information they may reveal about a user's behavior. This differentiation defines the interpretation of privacy in this research, although an actual definition is lacking. A meaningful definition can be obtained when this differentiation is combined with Definition 2: in the context of blockchain systems, privacy is a user's right to keep secret their identity and transactions in a blockchain system. Furthermore, identity privacy is linked to anonymity, although it is stated that anonymity is not sufficient for identity privacy. Anonymity is not explicitly defined.

[9] researches privacy vulnerabilities in blockchain systems. This research discusses privacy extensively but does not define it until its conclusion, where it presents the following definition: *"Privacy in blockchain refers to safeguarding the identity of the user involved in a transaction and protecting the secrecy of transaction data"*. This definition takes two methods to protect privacy as a right, namely protecting user identity and protecting transaction data, and uses these to define privacy. This is a conflation of privacy as a right and ways to enforce this right. Although anonymity is mentioned in this research it is never defined. It is suggested that anonymity requires that there is no mapping between a user's identity and their public address, and, it requires that there is no personally identifiable information. These requirements are necessary to anonymity as defined in Definition 1, although they are not sufficient.

In [32], a recent survey on privacy enhancements in blockchain technology, the concept of privacy is not defined explicitly, although a set of requirements for privacy preservation is presented:

- Transaction confidentiality: access to blockchain data must be limited because it may cause privacy leakage.

- Anonymity: adversary cannot distinguish a particular individual from others.
- Transaction Unlinkability: transactions related to a user cannot be linked.
- Efficiency: acceptable efficiency in terms of communication, computation and storage is required for blockchain privacy-preserving mechanisms.
- Fairness: parties in a blockchain transaction are equal, no one may be harmed.
- Compatibility: compatibility with Bitcoin will boost acceptance.

Some of these are indeed necessary to enforce privacy in DLT systems (transaction confidentiality, anonymity, transaction unlinkability). The latter three are not, although according to the authors of [32] they are necessary for privacy preservation mechanisms to become successful. This research also provides a set of evaluation criteria for privacy enhancing techniques, including checking for the requirements above. Anonymity is defined based on [27], and is therefore well in line with Definition 1.

2.1.3 Anonymity-oriented studies

Several studies analyze DLT systems from the perspective of anonymity rather than privacy. In [22], an article that is mostly about what anonymity is and that mentions anonymity in the context of cryptocurrencies, privacy and anonymity are clearly distinguished. Throughout the article it becomes clear that anonymity is about not being known by name, while privacy is about hiding certain things (although little explicit attention is given to privacy). This principle also became clear from the quote at the start of this chapter. Anonymity is further associated with untraceability and pseudonymity, and the potential of metadata in deanonymization is recognized. Although an explicit definition of anonymity is not given, throughout the article it is interpreted literally, "not being known by name". This can be considered as a more specific definition compared to Definition 1.

An early study on Bitcoin ([33]) tried to set the stage for analyzing technologies that provide anonymity by proposing a framework for analysis and a definition of anonymity. In this study, anonymity is defined as *taint resistance*, which is about the capability of an adversary to get information about ownership of a bitcoin using its history. Transactions are taint resistant if an adversary can not feasibly decide which inputs taint which outputs in a transaction. Input A taints output B of a transaction when they are linked in the sense that (some of the) funds move from A to B in the transaction. The authors note that there is a need for an anonymity definition that is not based on unlinkability, because bitcoins can be easily distinguished on

the basis of their history. Conversely, they suggest to look at anonymity from an ownership perspective, and use a definition based on *taint resistance*, which is a property of transactions. In the end, the authors do not provide an explicit definition of anonymity, still they provide a novel way of looking at anonymity in Bitcoin-like currencies, together with a metric to evaluate anonymity (namely taint resistance). The practical and applied approach taken to anonymity is useful when trying to evaluate the anonymity supported by some technology; the notion or interpretation of anonymity will decide how well it can be evaluated. On the other hand, using something so specific as taint resistance may not be able to fully capture all aspects of anonymity. With regards to that, a more general definition like Definition 1 may be more accurate.

Another study that analyses various DLTs from the perspective of anonymity is [34], which exemplifies the lack of a common ground for evaluating and comparing the anonymity of different systems. In this research, anonymity in the context of a currency means that "any entity cannot be distinguished from any other entity". Although *entity* is not explicitly further specified, it seems to refer to stakeholders such as users and merchants. Moreover, it is suggested that a currency not providing this feature may violate privacy and will not be fungible. Fungibility in the context of a cryptocurrency says that each value unit is equal and that units cannot be distinguished from each other (e.g. on the basis of their history). It is noted that many virtual currencies employ a public ledger, and therefore to provide anonymity these systems must make sure transaction data can not be linked to the entities in the transactions. This requirement can be seen as a specification of their broad definition. The general definition mentioned above is quite close to Definition 1. Similar to the discussion about the anonymity definition in [8], when this definition is extended to "any entity cannot be distinguished from any other entity in a group of similar entities"; then it is identical to Definition 1. The study ([34]) also provides a theory of anonymity with characteristics of anonymity in the context of cryptocurrencies. In the proposed theory, anonymity is also defined using the definition from [27], in line with Definition 1. Moreover, a set of parameters is derived that are proposed for the evaluation of different aspects of cryptocurrency anonymity. The proposed parameters are:

- Unlinkability: any two transactions cannot be linked to the same user.
- Recipient anonymity: any transaction cannot be linked to their recipient, nor any recipient to a transaction (receiver anonymity).
- Untraceability: any transaction cannot be linked to its sender, this is equal to sender anonymity.

- Fungibility: value units cannot be distinguished from each other (e.g. on the basis of their history).
- Confidentiality: Transaction values must be hidden to avoid "behavior-based clustering".
- Unlinkability of Metadata: data such as user IP addresses can not be linked to blockchain data such as transactions or addresses.
- Deniability: a user can credibly deny they took part in some transaction.

Furthermore, the study suggests three parameters, namely the *anonymity set size*, *transaction processing time* and *transaction block size*, which could be used in combination with the parameters above to evaluate the level of anonymity of different anonymity-protecting mechanisms. However, it seems that especially these last two parameters aim to evaluate the usability of such mechanisms as well, not purely the anonymity. Still, the suggested parameters are useful and provide a basis for developing a common evaluation scheme for anonymity technologies. The authors of [34] do stress that "an acceptable framework to model anonymity in the context of cryptocurrencies" is something to be developed and would greatly contribute to understanding and evaluating the performance of cryptocurrencies in this regard.

In [26], a study that focuses mostly on anonymity, different tiers of anonymity in cryptocurrencies are defined. Privacy and anonymity are interchanged in this research, and privacy is not explicitly defined. Four tiers of anonymity are established, namely *pseudonymity*, *set anonymity*, *full anonymity*, and *confidential transactions*. Pseudonymity entails users being represented by pseudo-anonymous addresses; set anonymity means that the identity of a user is hidden within a set of identities; full anonymity is achieved when (in a transaction) anyone could be the sender and the sent value unit can be any (unspent) value unit; confidential transactions include hidden transaction values and is rather a separate feature than a higher tier. Interestingly, the *full anonymity* definition seems not to include receiver anonymity, although later in the study it is included. Set anonymity as defined in this study is similar to anonymity in Definition 1, although here it is specifically applied to users rather than the more general 'subjects' considered in Definition 1.

In [25], a short study on several cryptocurrencies that have a focus on privacy, an anonymous cryptocurrency is defined as "a cryptocurrency designed to have high levels of cryptographic qualities for providing anonymity for its users". Three requirements for such cryptocurrencies are given:

- Privacy: Origins, destinations and amounts in transactions are hidden.

- Untraceability: Coins cannot be traced or linked with their history.
- Fungibility: Coins are “pairwise indistinguishable and, thus, mutually interchangeable”.

As can be seen, privacy is interpreted as a requirement for anonymity. In fact, the way privacy is described suggests that it is about confidentiality or secrecy rather than privacy as in Definition 2, although hiding data can contribute to achieving privacy. Anonymity is not explicitly defined in this research, however, the requirements suggest that it is interpreted similar to Definition 1 since anything that would allow distinguishing between two ‘subjects’ in a cryptocurrency is denied by the anonymity requirements. The anonymity requirements presented here are a subset of the requirements discussed in [34].

A recent study on anonymity, presented in [11], first generally defines anonymity in the context of a group of entities, in which it is impossible to uniquely identify one entity of the group. In this general definition [11] is similar to [34]. In the study [11], the authors further discuss various approaches to anonymity, many of which overlap with our discussion in this work. Then, the authors provide and formalize a set of notions of anonymity, which aim to provide a ‘*fine-grained, formal* qualitative model of anonymity’. In the model, which is meant for modeling the anonymity in ‘massively decentralized systems such as modern cryptocurrencies’, two notions are used to characterize anonymity. First, indistinguishability (of senders, receivers, transaction values and metadata) which means that distinghuishing between two entities is impossible; for example, indistinguishability of senders means that given two possible known senders, it is impossible to distinguish which was the actual sender of a transaction. Second, unlinkability (of senders, receivers, values and metadata), which is the weaker notion of anonymity that is used, means that given two unknown entities it is impossible to link one of these to a known entity. “For example, value unlinkability refers to the inability to decide which of two transactions has the same value as a transaction of interest"[11]. The study furthermore creates a formal model of an adversary and an anonymity game (which is designed like security games in cryptography). These are subsequently used to devise anonymity notions based on parameterized adversary characteristics and an adversary goal which is (breaking) indistinguishability or unlinkability. This leads to a sophisticated formal model of anonymity, which can be used to evaluate the anonymity of cryptocurrencies by analyzing which of the anonymity notions is satisfied (i.e. which anonymity game does (not) result in adversary success). This study is similar to [34] in that it uses unlinkability to characterize anonymity, the notion of indistinguishability seems to be derived from concepts like fungibility and sender/receiver anonymity which

are noted previously in [34] and [25]. Moreover, in choosing unlinkability and indistinguishability this study is very similar to [31], which we discussed earlier in this section. [31] uses unlinkability and indistinguishability as notions of privacy, and uses these to quantify privacy of Bitcoin users. [11] generalizes some of the principles presented in [31], and applies these to anonymity rather than privacy in general. The approach in [11] is also similar to [31] because formal definitions are provided and game-based definitions of the used notions are introduced in [31].

2.1.4 Other studies

More studies on privacy and anonymity in blockchain systems have been done (e.g. [23, 35, 36, 37, 38, 39]), but these do not add clear/comprehensive definitions either. It can be concluded that the discussed literature confirms there is no common definition of privacy or anonymity. Interpretations differ, and often researchers fail to adequately define these terms in their work. No agreement on terminology poses a problem since it becomes hard to compare results or develop a common ground for evaluation of systems that enhance privacy or anonymity. Therefore, for this research both privacy and anonymity will be clearly defined as follows. Hopefully, these definitions can serve as a starting point for comparisons between studies and technologies in terms of privacy and anonymity, and can unify researchers in their approach to these concepts.

2.1.5 Summary

In table 2.1 the definitions that have been provided in previous literature are summarized. Moreover, the definitions that are introduced in this study are added, which we develop in the upcoming section.

2.2 A new definition for privacy and anonymity

Based on the literature discussed above privacy and anonymity are defined as follows.

2.2.1 Privacy

In the discussed literature privacy and confidentiality/secrecy are regularly confused. Privacy is more often perceived as “the hiding of things”, although in fact privacy

Study	Title	Concept	Definition
[29]	Privacy	privacy	Someone's right to keep their personal matters and relationships secret; the state of being alone.
[31] (2013)	Evaluating User Privacy in Bitcoin	privacy	No explicit definition but privacy is quantified using *activity unlinkability* and *profile indistinguishability*.
[8] (2018)	A Survey on Anonymity and Privacy in Bitcoin-Like Digital Cash Systems	privacy & anonymity	Privacy: hiding content. Anonymity: hiding the person behind an action or content, entails being unidentifiable and untraceable.
[7] (2020)	Privacy and Cryptocurrencies—A Systematic Literature Review	privacy & anonymity	Privacy: the ability to perform private transactions (confidential amounts and anonymous sender/receiver). Anonymity: hiding senders and receivers.
[24] (2020)	A Survey on Privacy Protection of Blockchain: The Technology and Application	privacy	Differentiates between *identity privacy* (hiding relation between real world identity and blockchain addresses) and *transaction privacy* (hiding behavior information resulting from visible transactions).
[9] (2020)	A Survey on Privacy Vulnerabilities in Permissionless Blockchains	privacy	Safeguarding the identity of the user involved in a transaction and protecting the secrecy of transaction data.
[32] (2020)	Privacy preservation in permissionless blockchain: A survey	privacy & anonymity	Privacy: no explicit definition but privacy requirements are given: transaction confidentiality, anonymity, transaction unlinkability, efficiency, fairness, compatibility. Anonymity: a particular individual cannot be distinguished from others.
[27] (2001)	Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology	anonymity	The state of being not identifiable within a set of subjects, the anonymity set.
[28]	Anonymity	anonymity	The situation in which someone's name is not given or known.
[22] (2014)	Anonymity and privacy: a guide for the perplexed	anonymity	Not being known by name, associated with untraceability and pseudonymity.
[33] (2015)	Privacy-Enhancing Overlays in Bitcoin	anonymity	No explicit definition, taint resistance introduced as a metric for anonymity.
[34] (2019)	A Survey of Anonymity of Cryptocurrencies	anonymity	Any entity cannot be distinguished from any other entity. Moreover anonymity parameters are given: unlinkability, recipient anonymity, untraceability, fungibility, confidentiality, unlinkability of metadata, deniability.
[26] (2019)	SoK: A Systematic Study of Anonymity in Cryptocurrencies	anonymity	Four tiers of anonymity are introduced, full anonymity is defined in the context of a transaction, where anyone can be the sender and the sent value can be any (unspent) value unit.
[25] (2019)	Rise of Anonymous Cryptocurrencies: Brief Introduction	anonymity & privacy	Anonymity is not explicitly defined, requirements are given: privacy, untraceability, fungibility. Privacy here means confidentiality or secrecy.
[11] (2021)	The Cryptographic Complexity of Anonymous Coins: A Systematic Exploration	anonymity	Impossibility to uniquely identify one entity in a group of entities. Anonymity is further formalized using two concepts: indistinguishability and unlinkability.
This study	User privacy in cryptocurrencies	anonymity & privacy	Anonymity: the state of being not identifiable within a set of subjects, the anonymity set. Privacy (in the context of DLT): the right of a DLT system user to keep secret any data stored or used within the DLT system that pertains to them. (See 2.2.)

Tab. 2.1.: Definitions of privacy and anonymity in current literature and our study.

is something that pertains to a person and is about the right to hide things as desired. Subsequently, hiding things is a tool to enforce this right. In available research, privacy is often seen as property of a system, a technology, or an object (e.g. transaction). However, a system itself does not directly have privacy, nor does a transaction, although they might help to enforce a user's privacy. Then these systems or objects should be named 'privacy-preserving', which is a property rather than a right.

Based on the definition from [29] and the way privacy is used in the literature, the definition of privacy in the context of DLT systems in this research is as follows:

Definition 3. Privacy is the right of a DLT system user to keep secret any data stored or used within the DLT system that pertains to them.

This means that privacy is not secrecy, it is not confidentiality and it is not anonymity; it is a higher level concept that is about users. However, this definition does not satisfy the use of the term *privacy* in current literature. In many instances, the term privacy in literature should be interpreted as *privacy-preserving*. In this work, an explicit differentiation will be made between privacy as a right, and the ability to enforce that right which is granted through *privacy-preserving* measures. Obviously, various levels of privacy preservation are possible, still, in general something that is privacy-preserving contributes to (at least in part) protecting a user's privacy.

Definition 4. Privacy-preserving is a property of systems, technologies or objects reflecting their ability to protect the privacy of their users (to some extent).

The dual use of the term privacy in literature, both as defined in Definition 3 and Definition 4, could be interpreted as some form of an abuse of notation. This provides basis to allow the use of the term *privacy* in this way. However, such an abuse of notation hinders the discussion on privacy. The privacy provided in cryptocurrencies is relevant to multiple parties; users, merchants, developers, regulators and law enforcement among others. To enable discussion and clear communication between these parties, which come from different backgrounds and disciplines, such abuse of notation should be avoided.

2.2.2 Anonymity

Definitions of anonymity have also been shown to vary in current research. However, there is more agreement on what anonymity is and what it is not, which is likely due to it being a more descriptive term. In this research the definition of anonymity presented in [27] will be used, which is also used in multiple other recent studies.

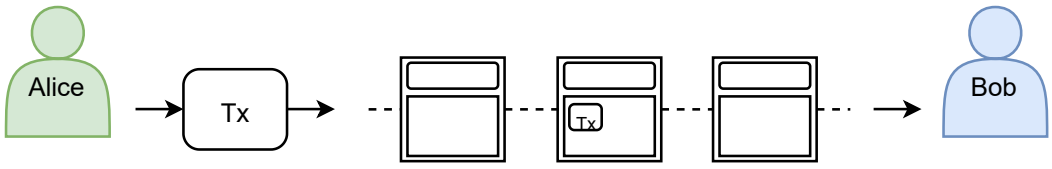

Fig. 2.1.: A cryptocurrency transaction between Alice and Bob via the blockchain.

As such, the definition of anonymity is the same as presented earlier in this section (Definition 1):

Definition 5. Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.

The relation between anonymity and privacy is such that to enable privacy enforcement, anonymity may be a measure to achieve some level of privacy-preservation. Moreover, anonymity can be used to assess the level of privacy-preservation that a system will allow a user to have, in that sense it can serve as an indicator.

2.2.3 Other definitions

There are several other terms that are often used in combination with anonymity in the context of DLT systems, which shall be briefly defined below; mostly in line with definitions provided in [27]. These definitions will be useful for understanding discussions on the anonymity provided in different systems and technologies later in this work. For clarity, in Figure 2.1 a setting is provided to which these definitions will be applied.

- **Pseudonymity**: "is the use of pseudonyms as IDs"[27]. If a DLT system user uses a pseudonym, then the pseudonym identifies that user; pseudonymity of the user results. In the setting in Figure 2.1, this would mean Alice and Bob are represented by some other data, for example a string of random characters. From the public eye, the transaction is then sent from a random string of characters (representing Alice), to another random string of characters (representing Bob).

- **Unlinkability**: this means that two objects in a DLT system (e.g. transactions) are not linkable to each other or to a user, even not after they have been (publicly) communicated in the system. Assume that in Figure 2.1, there are multiple transactions happening between Alice and Bob. Then, unlinkability means that an external party cannot link these transactions, for example by

concluding they are between the same users, or by either knowing they come from or go to the same user.

- **Unobservability**: "is the state of Items-Of-Interest (IOIs) being indistinguishable from any IOI at all"[27]; thus it should not be possible to know any event (e.g. when a sender sends or a receiver receives) because what is observed might as well be random noise. In the provided setting (Figure 2.1) this means that an external observer cannot distinguish this transaction from other things happening in the network. In short, an observer cannot observe this transaction between Alice and Bob happening.

- **Obfuscation**: Applying techniques to hide the content of public data such as encryption or hashing. Applied to the setting in Figure 2.1 this means that the content of the transaction, such as where it came from and what amount, will be unreadable. For instance, this may be made unreadable by encryption or using commitments.

- **Confusion**: Applying techniques to mix up public data in such a way that it becomes harder to extract privacy sensitive information. This can for example be done by aggregating multiple messages and permuting the order. In the given setting (Figure 2.1), now assume there are more parties who are also having different person-to-person transactions between them, like the one visualized. To introduce confusion, all these transactions could be aggregated in one transaction before including them in the blockchain, thereby confusing who is transacting with who.

2.3 Applying definitions to cryptocurrencies

Privacy and anonymity are concepts that have both often been applied to cryptocurrencies, but what really is a privacy-coin or an anonymous cryptocurrency? As discussed earlier, privacy is about a user, not about technology. Usually, when a cryptocurrency gets the privacy-coin label, it means that the cryptocurrency adopted some technique(s) to allow the users to hide (some of) the data they generate when using the system. Therefore, presumably, the term *privacy-coin* should be interpreted as a cryptocurrency that has some features to enable privacy enforcement for their users. Although having such features does, in practice, not guarantee the privacy-coin label: many counter examples are available. Similarly, an anonymous cryptocurrency should be interpreted as a cryptocurrency that has some features to enable its users to act anonymously within the system. Privacy and anonymity are

not properties of cryptocurrencies themselves, rather they are concepts pertaining to cryptocurrency users, which may, to some extent, be supported by the cryptocurrency features. For example, having features that make many of the things that happens in a cryptocurrency system confidential contributes to a user's privacy; conversely, explicitly stating someones name together with their transactions obviously hinders users' anonymity.

As a result, the anonymity and privacy of a user in a cryptocurrency system will consist of a spectrum. A user will have some anonymity between their actions being directly linked to their identity and their actions being linked to no one specific user in the group of all the users. Similarly, a user will have some privacy; meaning that some of the information pertaining to them will be visible to others, and some will not; depending on the what privacy-preserving mechanisms are available to the user. Defining these spectra further, and plotting available cryptocurrencies, and specifically privacy-coins, within the spectrum may further support understanding of how cryptocurrencies compare regarding privacy and anonymity.

2.4 Conclusion

In this chapter we explored the literature on privacy and anonymity in cryptocurrencies, to find how these concepts are defined. We summarized available definitions, and presented definitions that are to be used in this work. Moreover, we defined several other concepts that are relevant to privacy and anonymity in the context of cryptocurrencies. We also discussed how the presented definitions apply to cryptocurrencies. The results of this chapter can be used as a frame of reference for future studies on privacy or anonymity in cryptocurrencies, regarding how these concepts should be defined.

3 A privacy-preservation evaluation framework

To evaluate the preservation of user privacy in a cryptocurrency system, a repeatable and justified framework should be developed, as was also suggested in current literature [34]. Such a framework would allow for meaningful comparisons between cryptocurrency systems in a replicable manner. Moreover, it can serve as a basis for more sophisticated evaluation methods to measure cryptocurrency system performance regarding privacy-preservation. A framework that provides intuitive evaluation and comparison on the privacy-preservation of cryptocurrencies is also useful for communication (at the interface of different (scientific) disciplines).

In this chapter, such a framework is presented. First, relevant related work is discussed (Section 3.1). Then, the framework as suggested here is elaborated and justified (Section 3.2). Next, we apply the framework for a privacy-preservation performance comparison among Bitcoin, Dash and Zcash (Section 3.3). Moreover, the framework will be evaluated in Section 3.4 by using expert reviews. Section 3.5 discusses the framework and review results, and provides suggestions for future research. Finally, Section 3.6 concludes this chapter.

3.1 Related work

There exists some research which compares cryptocurrencies based on their privacy-preservation capabilities or the anonymity of their users. This section summarizes existing evaluation approaches, and highlights how the framework presented in this chapter differs from the existing approaches.

Several studies have compared cryptocurrencies based on their intrinsic features. Some of these features used as indicators for privacy-preservation are: *address hiding*, *untraceability*, and *hiding amounts*. This approach provides some insight into how cryptocurrencies differ, although it fails to capture similarities between those cryptocurrencies that take a different approach to privacy-preservation while achieving the same goal. For example, some cryptocurrency systems can anonymize users by encrypting their addresses; while other systems can apply mixing mechanisms to

achieve the same goal. Furthermore, these schemes also do not indicate to what extent cryptocurrency system features succeed in providing privacy-preservation or anonymity. Examples of this type of evaluation scheme can be found in [23, 24, 25, 30]. Some of the studies also compare different privacy-preserving technologies and discuss their advantages and disadvantages (e.g. [23, 24]), which similarly allow to compare cryptocurrencies on a feature-level but not on their overall privacy-preservation performance.

The study in [26] defines four tiers of anonymity (which can be seen as some form of privacy-preservation), and classifies cryptocurrency systems accordingly. The chosen tiers are: *pseudonymity*, *set anonymity*, *full anonymity*, and *confidential transactions*. The final tier is defined as having hidden transaction amounts, as such this is a feature that cryptocurrency systems may (not) have, rather than a separate tier. Subsequently, in this study cryptocurrency systems are categorized in these tiers and compared on the basis of what privacy-preservation techniques they have implemented. The tiers provide an initial step towards meaningful comparison of privacy-preservation performance across cryptocurrencies. It gives a basic indication of how significant the differences between systems are. Still, the classification is limited; only anonymity is considered and just three tiers are available. The framework introduced in this chapter aims to give clearer insight into the difference in privacy-preservation performance. The differentiation between cryptocurrencies will be more fine-grained, and aspects other than anonymity will be considered as well.

Some research looks at the performance of constructions that aim to improve privacy-preservation. One of these is [34], in which anonymity enhancing constructions are evaluated to determine how successful they are. Examples of the constructions considered in this study are mixing solutions like CoinJoin[19] and CoinShuffle[40] and cryptographic solutions like Zerocoin[41] and Monero[42]. In the study, a set of requirements for anonymity is presented, which are subsequently evaluated for each anonymity enhancing construction. A qualitative evaluation of these constructions results, which indicates how well a construction performs with regards to each requirement and allows for comparison between constructions. This is a methodological improvement over other studies, as evaluation is done on a higher level of functional abstraction; requirements of anonymity are evaluated instead of the presence of certain features, which makes comparison between cryptocurrency systems more meaningful. However, this study ([34]) does not focus solely on privacy-preservation but also includes usability aspects. Moreover, it focuses mostly on anonymity, whereas privacy-preservation entails more than anonymity alone. This chapter's framework includes more aspects of privacy, in addition to anonymity,

and disregards any non-privacy-related aspects. The proposed framework is similar to [34] in that privacy-preservation is evaluated based on requirements, and, it provides more insight on how privacy-preservation performances of cryptocurrency systems differ from one another.

While we were developing the framework that is presented in this chapter, a work was published that introduces a similar effort focused on anonymity. In [11], which we already mentioned in the previous chapter, a framework for evaluation of anonymity is presented which utilizes anonymity games (which are akin to security games in cryptography) to establish nuanced notions of anonymity. In the study, a formal theoretical framework is developed as a basis for the anonymity notions that are presented. First, the general functionality of a (distributed) payment system is modeled using a set of algorithms, the correctness of the model is subsequently verified. Second, an adversary model is created with parameterized capabilities and knowledge. Third, anonymity games are introduced in which a challenger tries to achieve an anonymity goal, which is either based on indistinguishability or unlinkability. For example, in sender indistinguishability, the adversary with some given knowledge and power may not have an advantage distinguishing between two potential senders of a transaction, of which one is the actual sender. For a set of useful anonymity notions, a game-based definition is presented. A system satisfies such a anonymity definition if the adversary can not win the corresponding game (or gain an advantage towards winning the game). Furthermore, the study also provides a structuring of all the anonymity notions, showing the hierarchy, relations and interdependencies between the different notions.

Our research aims at a similarly fine-grained framework, but on a higher level and for privacy-preservation rather than only anonymity. Moreover, the approach we take is less formalized, and not based on security games.

3.2 Framework basis

Thus far, cryptocurrencies have been compared in terms of features, classified in privacy tiers, and their privacy-features have been put to the test using anonymity requirements. The framework proposed in this section builds on the ideas presented in previous work. A logical next step is to design a framework for privacy-evaluation that provides meaningful results combined with comparability across cryptocurrencies. Meaningful results can be achieved through qualitative analysis that checks requirements of privacy-preservation. Comparability and replicability are obtained through somehow quantifying the differences between cryptocurrencies, which is

somewhat similar to putting them in classes or tiers, although we will aim to achieve a more fine-grained structure of privacy-preservation levels. We will present such a framework in this section.

The concept of privacy is subject to interpretation, and privacy-preservation can be approached from different angles. For example, some may choose to anonymize data, whereas others encrypt it or only publish commitments; both approaches have increasing privacy-preservation as their objective. Moreover, privacy-preservation can be about many subjects, such as identity, location, and financial records. To capture the different angles on privacy-preservation in cryptocurrencies, we decided to use a series of statements that capture characteristics of user-privacy on one hand and transaction characteristics on the other. These statements reflect how well the privacy of the users can be preserved in a cryptocurrency system, serving as a basis for evaluating user-privacy.

The statements are scored on a three-point scale; score 0 means the statement is not applicable to the cryptocurrency, score 1 means the statement is somewhat applicable to the cryptocurrency, and score 2 means the statement is fully applicable. These scores are subsequently used to determine overall privacy-preservation performance scores, which is, in turn, applied to compare cryptocurrencies. All statements are phrased negatively, such that when a statement is fully applicable the privacy loss is largest and when it is not applicable there is no privacy loss. Each of the statements will be further explained, and more details on what the scores mean will be given in coming sections, general criteria for how scores should be determined will result.

The statements that will guide the evaluation of privacy-preservation in a cryptocurrency system are divided in categories, which can also be used to highlight where the differences between cryptocurrencies regarding privacy-preservation originate. The statements within their categories are as follows.

- Sender privacy-preservation.

  S1. The sender's identity is revealed.

  S2. The sender's wallet balance is revealed.

  S3. The sender's location is revealed.

  S4. The sender can be linked to their transactions.

- Receiver privacy-preservation

  S5. The receiver's identity is revealed.

S6. The receiver's wallet balance is revealed.

S7. The receiver's location is revealed.

S8. The receiver can be linked to their transactions.

- Transaction characteristics

S9. Transaction inputs are visible.

S10. Transaction outputs are visible.

S11. Transaction amounts are visible.

S12. Unique transaction IDs are visible.

- Transaction privacy-preservation abilities

S13. Transactions can be linked to each other.

S14. Transactions can contain other information that reduces privacy-preservation.

S15. Transactions cannot be engineered to improve privacy-preservation

3.2.1 Adversary model

The privacy of cryptocurrency users is diminished when an actor can successfully attack that privacy. Opposite to privacy-preservations are the actions of an adversary who wants to uncover cryptocurrency users' information, which those users may not want to reveal themselves. The capabilities of such an adversary, who tries to attack the privacy or anonymity of cryptocurrency users, should be explicitly defined. This is crucial in privacy-preservation performance evaluation, since the capabilities of the adversary determine how well a user's privacy is protected. For instance, an adversary with infinite computing power can break many cryptosystems, that are not realistically breakable right now. The adversary model proposed here is applicable to public permissionless blockchain-based cryptocurrency systems, which have been defined earlier. And, this adversary model is used when applying the framework to cryptocurrency systems in this chapter. In any case, when the framework is applied to a cryptocurrency to evaluate privacy-preservation, an adversary must first be defined since it determines how the statements in the framework will be scored.

In [43], a classification of adversary types is presented. The characteristics that are used to differentiate between adversaries are listed below. We apply these characteristics to cryptocurrencies, which subsequently provides a basis to define the adversary used in this work. In the list below, the characteristics are explained

(based on [43]). If deemed useful, how these apply to an adversary who seeks to attack a user's privacy in a cryptocurrency system is also explored.

1. *Local - Global*

    - A *local* adversary has restricted access to the system. In the context of cryptocurrencies this could mean that the attacker does not have access to any nodes, or can only connect to a limited number of nodes. We define access to mean being able to 'connect to' nodes, not control them.

    - A *global* adversary has access to the entire system, in a cryptocurrency system this means that the attacker can connect to all nodes, the attacker has global presence. Generally, a global adversary is stronger (more likely to succeed in their attack on user privacy).

2. *Active - Passive*

    - An *active* adversary takes part in the system and is able to modify or interrupt the regular functioning of the system. In a cryptocurrency system this could, among others, entail that the adversary creates malicious transactions, relays invalid transactions or does not relay transactions at all. Normally, an active adversary is stronger than a passive adversary.

    - A *passive* adversary can only monitor ongoing activity. This means observing the public ledger in a cryptocurrency system and monitoring (network) communication within the cryptocurrency network.

3. *Internal - External*

    - An *internal* adversary is part of the system. In a cryptocurrency such an adversary may be a node operator or someone who is participating in transactions. Usually, an internal adversary is stronger than an external adversary.

    - An *external* adversary is not part of the system but can only attack it from the outside. Thus, in a cryptocurrency system the adversary would only be able to attack user privacy from the outside (e.g. using observable data).

4. *Static - Adaptive*

    - A *static* adversary chooses their strategy and resources before the attack and sticks with them for its duration.

- An *adaptive* adversary may adjust their strategy (and resources) while their attack is going on. This may be advantageous when they discover new strategies during the attack. Commonly, an adaptive adversary is considered to be stronger than a static adversary.

5. *Prior knowledge:* additional knowledge that may provide an adversary with and advantage in the target system. For example, knowledge about user properties or system parameters. In a cryptocurrency such prior knowledge may entail a dataset that links cryptocurrency wallet addresses to IP addresses or a dataset that links cryptocurrency exchange accounts with addresses (which a powerful adversary may have).

6. *Resources:* these are resources that an adversary has to execute their attack; for example, computational capacity or networking resources. These resources may vary, in a cryptocurrency they could entail the number of nodes that an adversary can operate. Resources affect what attacks are feasible for an adversary.

We use this classification to define the adversary that is assumed in this work; we define our adversary using the characteristics listed above, and explain how we interpret them. Moreover, we elaborate the implications of the choices that we make by giving concrete examples of the capabilities of the adversary used in this study. In this study we fix the adversary model, by picking one set of capabilities. We leave applying other adversaries and dynamic adversary models to future research.

The adversary used in this research for framework application is:

- *Global*: the adversary has access to the entire cryptocurrency system; they can connect to all other nodes.

- *Passive*: the adversary cannot take part in activity within the system, they can only observe what is going on. They can observe the blockchain and network traffic to retrieve information they can use to attack the privacy of users, but they cannot take part in transactions (which would unlock a whole new class of attacks).

- *Internal*: the adversary can be part of the system, they can, for example, become a miner full node as long as they do not interfere with the system. Although the adversary is internal to the system it is still passive. Therefore, it cannot interfere with the regular functioning of the cryptocurreny system. It must behave in accordance with the regular and expected behavior of normal

(non-adversarial) nodes. It is not a requirement that the adversary is internal, although in this work we assume the attacker has internal capabilities.

- *Adaptive*: the adversary can adjust their strategy during attacks, for example, they may incorporate newly gained knowledge on identities of specific users.
- *Prior Knowledge*: the adversary may already have some knowledge that can aid them in successfully attacking privacy. We assume that any data-sets that (are likely to) exist and are relevant for the adversary in attacking user privacy are obtainable for the adversary. For example they may have access to some deanonymized transactions (from existing research) or a database that links IP-addresses to identities. What this knowledge entails exactly is hard to define, since it will vary by cryptocurrency, and change over time. When access to certain data is relevant for the adversary in some attack scenario, then we assume the adversary has access to that data and otherwise specify it. In short, it is assumed the attacker can identify a user when they have some personally identifiable information of that user. Generally, when an attack requires some prior knowledge that is reasonably acquirable by an adversary with government-like powers; it is assumed the adversary will have this knowledge.
- *Resources*: just like prior knowledge, an attacker may have varying amount of resources to execute their attack. In the application of the framework we assume the adversary is computationally bounded, they cannot control or possess the majority of nodes or the majority of hashing power in a cryptocurrency network (which would also unlock a new set of attacks). Moreover, the adversary cannot retrieve users' private keys, thus decryption of securely encrypted data is infeasible.

Specifically in the context of cryptocurrencies, the adversary capabilities noted above result in the following concrete abilities in a cryptocurrency system:

- Participate in the cryptocurrency (peer-to-peer) network.
- Observe, send and receive network traffic (according to regular and expected behavior).
- Read blockchain data.
- Participate in consensus.

The adversary that is assumed in this chapter has the perspective of an observer who is trying to attack the privacy now (at the current point in time); can observe what

is going on presently and the publicly visible data on what happened in the past. For instance, they can monitor new transactions and the transaction history (if it is publicly stored in the blockchain), but not transaction relaying information that could have been obtained in the past; data that is generated but not permanently stored cannot be observed. Essentially, the adversary should be seen as a 'new' attacker, who has not executed attacks or obtained information until now.

Generally, the taken adversary model and perspective corresponds with a 'generally interested governement' perspective, that may want to obtain sensitive information about cryptocurrency users (in certain cases). the adversary can be seen as a powerful observer who can also participate in (but not interfere with) the cryptocurrency system. Governments, which are powerful since they can employ subpoenas to get information form institutions like banks or exchanges, are also one of the parties against which privacy-focused cryptocurrencies want to protect their users. The assumptions laid out in this section are interesting for various reasons which we elaborate further in the discussion (3.5).

3.2.2 Statements specification

The statements that are used for privacy-preservation evaluation are discussed in this section, together with details on how scores should be accredited.

S1 & S5: Sender/receiver identity When it is revealed who the sender of some transaction is (their identity is revealed), then that user completely lost their anonymity. Scores are to be accredited according to the criteria below, where sender is replaced by receiver when evaluating statement 5.

0. There is no (known) way to reveal the sender's identity using information obtainable in the cryptocurrency system.
1. There are some known ways to reveal a sender's real-world identity using information obtainable in the cryptocurrency system, with some degree of uncertainty.
2. There is at least one way to reliably obtain the identity of a transaction sender in a cryptocurrency system.

S2 & S6: Sender/receiver wealth Revealing the wealth of a user means there is a lack of financial privacy-preservation, therefore this should be hidden for others,

like it is in regular bank systems. By the wealth of a sender or receiver we mean the balance they have in their cryptocurrency wallet. Bitcoin-like cryptocurrencies, which are based on the UTXO model, can sometimes reveal someone's wealth. For example, when multiple addresses can be linked together or to the same user, then the sum of the balances in these addresses provides information about the wealth of the owner. This statement only addresses gaining additional information on the wealth of a sender or receiver, on top of the value of the transaction through which this sender or receiver was observed. Other statements (e.g. S9 and S10) cover the privacy-preservation loss caused by the revelation of one transaction input or output. As such, scores are to be accredited according to the criteria below, where sender is replaced by receiver when evaluating statement 6.

0. No information on the wealth of a sender can be revealed other than the value of the current transaction.

1. The wealth of a sender can be partially revealed or revealed under certain circumstances.

2. The total wealth of any sender can be derived from public information at any time.

S3 & S7: Sender/receiver location Location is another attribute that users may want to keep private, and an attribute that can be used by adversaries for deanonymization. By location in this context we consider the network location (IP-address) of the sender or receiver; from which transactions are broadcast. Location may be obtained via some known attacks, for example by monitoring where incoming transactions come from. At present, it is possible to spoof IP-addresses; which, if done, means that revealing the network location of a sender may not affect their privacy-preservation. We do not account for this here, although, if detectable, it could be included in scoring this statement. Scores are to be accredited according to the criteria below, where sender is replaced by receiver when evaluating statement 7.

0. There is no way to reveal the sender's location.

1. The sender's location may be revealed with some confidence but not with full certainty. Or, the sender's location is only revealed when some prerequisites are fulfilled.

2. The location of a sender can be revealed with high certainty using known methods.

S4 & S8: Sender/receiver transactions When transactions can be linked to a user, this information can be used to profile users, which may violate the privacy of those users. Therefore, users may want to keep secret which transactions belong to them, such that an adversary cannot link multiple transactions to them. Scores are to be accredited according to the criteria below, where sender is replaced by receiver when evaluating statement 8.

0. There is no way to link a sender's transactions back to them.
1. There are ways to link a sender and (some of) their transactions, when certain prerequisites are fulfilled.
2. All transactions made by a sender can be linked back to them. Such that, given a transaction, it can be attributed to a specific sender, or given a transaction sender, their transactions can be found.

S9: Visible inputs When inputs to transactions are visible it means that the source of the funds is publicly observable. In Bitcoin these would be the UTXO's that are consumed by a transaction. When inputs are visible, that can be used to deduce information that affects the privacy of users. For example, it enables clustering and transaction tracing. Inputs may not be visible when they are encrypted or when commitments are used. Scores should be accredited as follows:

0. Transaction inputs are invisible.
1. Transaction inputs are revealed under some circumstances.
2. Transaction inputs are always visible.

S10: Visible outputs Opposite but similar to transaction inputs, transaction outputs reveal the destination of funds. When outputs are visible (public and unencrypted) this enables the tracing of funds and may reveal information about the receiver. Scores should be accredited as follows:

0. Transaction outputs are invisible.
1. Transaction outputs are revealed under some circumstances.
2. Transaction outputs are always visible.

S11: Visible amounts Public and unencrypted transaction amounts can be used for (behavior-based) clustering of transactions. Clusters can then be linked to entities

such as users or companies, and they can be used for profiling. Also, transaction amounts show information about a user's wealth and the type of transactions they are performing. Generally, the transaction amount can be used as meta-data to obtain more information on a user and the context of a transaction. Scores for this statement should be accredited as follows:

0. Transaction amounts are completely invisible.
1. Transaction amounts are only visible under certain circumstances.
2. Transaction amounts are always visible.

S12: Visible IDs When each transaction is uniquely identified and distinguishable in public, this will enable certain attacks for adversaries, which are based on transaction analysis and tracing. Scores are to be accredited as follows:

0. Transaction IDs are not visible.
1. Transaction IDs are only visible under certain circumstances, or the IDs are not unique.
2. Transaction IDs are unique and visible for all transactions.

S13: Linkable transactions When transactions can be linked to each other, adversaries can trace transactions and follow the flow of funds. Linking transactions may be done because links are publicly visible or using meta-data, for example network information or transaction values. Linking transactions indirectly affects privacy-preservation, since the resulting information may be used for profiling and deanonymization later. Scores for this statement will be accredited as follows:

0. No methods to link transactions to each other are known.
1. Transactions can be linked to each other with some certainty using known methods.
2. Transactions can be linked to each other with full certainty.

S14: Other information In some systems, transactions can contain other data. This data, which may be placed in scripts or other extra transaction fields, can sometimes be used to decrease the anonymity of users. Scores for this statement will be accredited as follows:

0. There is no other information available in a transaction.

1. There is other information available in a transaction but this cannot (with current knowledge) be used to decrease privacy-preservation.

2. There is other information available in transactions can be used to decrease user privacy-preservation via known methods.

S15: Engineering privacy-preservation Sometimes, transactions can be engineered in various ways to improve their privacy-preservation ability. For example, many Bitcoin-like cryptocurrencies allow multiple-user transactions which can be used for mixing mechanisms like CoinJoin[19]. Scores may be given to currencies for this statement according to the following criteria:

0. There is functionality to improve user privacy-preservation built in the core software of the cryptocurrency.

1. Transactions can be engineered to improve privacy, although this is implemented or facilitated by third parties and not internal to the cryptocurrency's software and protocol.

2. There are no known available ways to improve user privacy-preservation through specific transaction types, transaction formats or usage methods.

3.2.3 Final score

The resulting scores for each of the statements can be combined to compute a *final score* for a cryptocurrency regarding their support for user-privacy preservation. This score can be computed by adding the individual statement scores for a cryptocurrency, and dividing the result by the number of statements multiplied by the maximum score, to obtain a normalized score. The resulting normalized score is somewhat counter-intuitive; since all the statements are phrased negatively, the final score now indicates how 'not privacy-preserving' a cryptocurrency is. This can be inverted by subtracting the final score from 1; then the score indicates the privacy-preserving performance of a cryptocurrency. Subsequently computing a percentage is useful to obtain an intuitive score. The privacy-preserving capacity of a cryptocurrency can then be summarized, using this framework, in a statement like "The user privacy-

preservation performance in [Cryptocurrency] is X% as evaluated by this framework". The formula for the final score results:

$$\text{FS} = \left(1 - \frac{1}{2 \cdot 15} \sum_{i=1}^{15} \text{Score}(\mathbf{S}i)\right) \cdot 100\%$$

The final score may also be made more meaningful through multiplying the individual statement scores by some weight value. However, this will also be more complex and requires further justification for these weight values. Still, this is useful to improve the accuracy of the framework and as such it should be a topic of future research.

3.3 Framework application on cryptocurrencies

When applying the framework to a cryptocurrency system, each of the statements has to be evaluated. The choices that are made in accrediting a score should be documented and justified, as will be done in this work for the addressed cryptocurrencies. This will allow future improvements to and discussion on the validity of the final score.

In this work, the decision was made to focus only on blockchain-based cryptocurrencies which were developed on top of the Bitcoin codebase. This was done to keep initial application of the framework relatively simple, by comparing cryptocurrencies that have an inherently similar codebase. Simplicity in application is desired since the framework is in its infancy and may need further improvements for more complex comparisons. Indeed, the framework may also be applied to cryptocurrencies that do not share the Bitcoin codebase (e.g. Monero), the general principle will be usable although the individual statements may have to be adapted. The cryptocurrencies hereby evaluated are Bitcoin, Dash, and Zcash; the choices that were made for each of these currencies are discussed in the coming sections. For baseline reference, a fictional perfectly private currency is added. Moreover, bank-to-bank fiat is added as a currency, since this is a currently established system that comes closest to cryptocurrency systems and serves as a frame of reference.

In evaluating the statements we employ the adversary described in Section 3.2.1. Can be briefly summarized as a 'powerful observer who can also participate in (but not interfere with) a cryptocurrency, with government-like powers'. The adversary is *global, passive, internal* and *adaptive*.

Perfect privacy and bank-to-bank fiat currencies

A currency that offers perfect privacy scores the best score regarding privacy-preservation for each statement, as shown in 3.1. This obviously results in 100% score for privacy-preservation.

Bank-to-bank fiat currency can be seen as a non-decentralized and currently available alternative to cryptocurrencies. Although it does not offer many of the functionalities that cryptocurrencies offer, it is useful for comparison regarding privacy-preservation, such that cryptocurrencies can be compared to currencies used day to day. By bank-to-bank fiat transfers we mean transfers that happen between a sender and receiver, where the sender has a bank account at some bank (bank A) and sends some fiat to the receiver who also has an account at some bank (bank B). In this scenario bank A and bank B are not necessarily the same bank. To avoid transnational complexity for now, we assume the banks are within the same jurisdiction.

The highly centralized nature of bank-to-bank fiat causes privacy-preservation to be much dependent on an attackers capability. In this evaluation, the adversary is assumed to be a powerful observer. Therefore, in this case we assume that the observer can have access to a bank's systems, for example because the attacker is a bank employee or a government with access capabilities. When a different adversary is chosen, the privacy-preservation evaluation of bank-to-bank fiat would change; since without access to bank's systems the privacy-preservation of this system is strong.

S1. The identity of a sender is revealed through access to the bank's data, since they know the identity of the sender. This statement is scored 2, since a powerful observer can obtain the identity.

S2. For similar reasons as the previous statement, the grade here is 2.

S3. The location of a sender can be discovered through access to bank systems, or potentially through man-in-the-middle attacks. However, the sender can also try to hide their location using proxies. The possibility to discover location exists but has certain prerequisites; as such this statement is scored 1.

S4. Linking transactions and senders together is doable with access to the bank's systems, for example by querying all transactions involving the sender's account number; this statement is scored 2.

S5. Similar to sender identity: grade 2.

S6. Similar to sender wealth: grade 2.

S7. Receiver location can be known to a bank if they monitor from where the receiver of a transaction connects to them. However, the receiver can also hide this connection, similar to a sender. As such, this statement gets score 1.

S8. Similar to sender transactions: grade 2.

S9. Inputs in this scenario consist of some funds from the sender's bank account, which is visible only to the sender themselves and the bank. As such, this may be uncovered and the statement is scored 2.

S10. This is similar to inputs, although outputs are only available to the receiver and the bank; score 2 results.

S11. Transaction amounts are visible internally, between sender, receiver and bank; a score of 2 results.

S12. Transaction can be uniquely identified by banks; a score of 2 results.

S13. A bank can easily link transactions to each other that involve the same sender or receiver, although this does require access to a bank's systems, transactions involving the same account number may be linked. This statement is scored 2.

S14. Many banks allow for a description or other information to be appended to a transaction. This may be used for data that negatively affects privacy-preservation; this statement is scored 1.

S15. Banks do not offer special features that make transactions more privacy-preserving, a score of 2 results.

The final score of bank-to-bank fiat currencies is 10%, which is a low score. However, it should be noted that this final score depends much the perspective one takes. Bank-to-bank fiat is not at all privacy-preserving against governments or the banks themselves. On the other hand, their privacy-preserving capabilities against outsiders are much stronger.

Bitcoin

Bitcoin, the first and largest cryptocurrency (in terms of market capitalization[1]) will serve as a basis, to which other cryptocurrencies can be compared. The scores for each of the framework's statements, which result from Bitcoin's properties, will be justified here.

[1] `https://coinmarketcap.com/`, accessed June 14th, 2021

S1. The identity of a sender of Bitcoin transactions is likely to be discoverable. Various deanonymization attacks exist (e.g. [17, 44, 45]), and the existence of companies such as Chainalysis[2] and CipherTrace[3] shows that methods to deanonymize Bitcoin users are available. On the other hand, when Bitcoin users do not leave any traces of a link between their pseudonym and their identity, then their identity is not necessarily revealed. Therefore, we score this statement 1.

S2. There exist ways to link Bitcoin addresses together with some certainty, which results in information about the wealth of the owner of some addresses; for example using transaction tracing or address clustering. Especially when a sender creates a transaction which consumes and/or combines multiple of their owned UTXOs this may reveal some information on their wealth. Since there are ways to at least get partial information on the wealth of a transaction sender, the score we pick for this statement is 1.

S3. Location of a sender may be revealed in some cases. For example, the first node that receives a transaction when it is relayed may know the IP address of the sender, which can be used to determine their location. Moreover, attacks have been suggested to obtain location information on the basis of relay patterns, as introduced in [46] and [47]. On the other hand, it is also possible to spoof a location, for example using proxies. Since the location cannot always be revealed we choose score 1 for this statement.

S4. Sender transactions may be revealed in some cases, for example when a sender creates a transaction that consumes multiple UTXOs then an observer can infer the transactions that resulted in these UTXOs also involved that sender. Moreover, sometimes the change address is clearly distinguishable and subsequent transactions involving that change address may also be attributed to the sender. Also, when transactions contain addresses which were publicly announced somewhere (e.g. on the Bitcoin forum) they are obviously linked. Conversely, there may not be observable links between a transaction and a sender, and then it will also be hard to link other transactions to a sender. As a result, we pick score 1 for S4.

S5. The identity of the receiver is, similar to the sender identity, discoverable using known deanonymization mechanisms. Again, this will not always be possible therefore we choose score 1.

[2] `https://www.chainalysis.com/`
[3] `https://ciphertrace.com/`

S6. The receiver's wealth may be revealed if they already used the address on which they receive funds to receive other transactions. Moreover, if the receiver subsequently combines some received funds with their other funds in a transaction, more information on their wealth will be available. As such, this also depends on the behavior on the receiver. We pick score 1, since some information is always available but how much will be dependent on the receiver behavior.

S7. The location of the receiver in Bitcoin is generally not revealed, except if they issue subsequent transactions where they become a sender; then it will be feasible to gain some information on their location. We pick score 0 for this statement, since the case of subsequent transactions is covered by S3.

S8. Linking transactions to a receiver and vice versa will depend largely on the behavior of the receiver. Given the receiver of a transaction, other transactions of that receiver that involve the same address may be easily discovered. Moreover, when the receiver uses that address in multi-input addresses (as a sender), other addresses and transactions may be discovered. Since other transactions may be discovered only in some cases, we pick score 1 for this statement.

S9. In Bitcoin, transaction inputs are publicly observable, as such we choose score 2.

S10. Similarly, transaction outputs are public, resulting in a score of 2 for this statement as well.

S11. Transaction amounts are also publicly visible in Bitcoin, score 2 results.

S12. Transaction identifiers are unique and public in Bitcoin, score 2 results.

S13. In Bitcoin, the flow of funds can be easily traced throughout the transaction graph, because inputs and outputs to transactions are public. As a result, transactions that are related can also be easily linked and we choose score 2 for this statement.

S14. Transaction mixing techniques (e.g. CoinJoin [19], CoinParty[48], etc.) are commonly applied in Bitcoin, although not incorporated in the core protocol. Moreover, some systems, such as the lightning network[49], can be used by Bitcoin users to improve the preservation of their privacy. However, Bitcoin has not implemented effective privacy-preserving technologies into their core protocol yet. We pick score 1 since mechanisms are available but not implemented in core software.

The final score for Bitcoin is 37%, as such it this framework states that the privacy-preservation of Bitcoin is 37% compared to 100% for a perfectly privacy-preserving currency and 10% for bank-to-bank fiat. Bitcoin mainly loses privacy because most of the transaction information is public. Compared to bank-to-bank fiat it is better in sender and receiver privacy-preservation; which indicates that it will be harder to link the public data to real-world individuals.

Dash

Dash, one of the cryptocurrencies of which the privacy features have been often debated (more on this in Chapters 4, 5). However, applying the framework to Dash will result in scores that are mostly similar to Bitcoin since the technology is much alike. Therefore, only the statements at which Dash differs will be explicitly addressed.

S2. Wealth of Dash senders may also be discovered in similar ways as in Bitcoin; at least partial information on their wealth is obtainable. Dash Evolution may impact what wealth information about users may be recovered, since in Dash Evolution accounts can be registered and transactions can be made to accounts. How well these may be tracked should be tested. The resulting score for this statement is 1.

S15. Dash contains a CoinJoin implementation (which used to be called PrivateSend), aimed at improving the privacy-preservation of users. This is implemented in the core wallet and facilitated by masternodes; as such score 0 applies for this statement.

Within this framework, it is clear that the main difference between Bitcoin and Dash, in terms of privacy-preservation, is in the privacy feature that Dash incorporated in its software. Otherwise Dash is similar to Bitcoin, resulting in a privacy-preservation score of 40%.

Zcash

Zcash is one of the privacy-coins that enable users to hide (part of) their transaction information. To do this, Zcash employs different types of addresses, namely z-addresses (which are shielded and privacy-preserving) and t-addresses (which are regular and public like Bitcoin addresses). Transactions can happen between any combination of these address types, resulting in different transaction types. Depending on the type of transaction, Zcash will reveal different amounts of information

that may affect preservation of user privacy. Because of the differences between transaction types in Zcash, providing an overall score for their privacy-preservation performance is complex. This would depend on which transaction types are used, and how often they are used. Therefore, each of the transaction types will be evaluated separately. To get an idea of the overall performance of Zcash, a (weighted) average could be used.

One could also score Zcash with an overall score using S15, and changing the weight of that such that it has more impact. This would be justifiable, since there the privacy-preserving transaction types in Zcash, are essentially built-in features fitting S15. However, it would be hard to capture the exact consequences of these transaction types, and the scores for many other statements would be debatable since they depend much on the transaction type. Moreover, it would fail to capture the relevance of the use-case of Zcash: if it is used to enhance privacy it does so, if it is not then it does not.

t-to-t transactions These transactions are the same as Bitcoin transactions, no extra privacy-preserving measures are taken. As such, the statement are all scored similar to Bitcoin. The final score for this type is thus the same as Bitcoin's: 37%.

t-to-z transactions Now the receiver address is shielded, the amount and sender address are still visible. It will now be very hard to find the receiver's identity, wealth or their other transactions, because the output is invisible. This results in value 0 for statement S5, S6, S8 and S10. It will also become harder to link transactions to each other, since subsequent transactions from the z-address of the receiver cannot be discovered. Therefore, we pick score 1 for S13. The resulting final score for t-to-z transactions is 57%.

z-to-t transactions This transaction type shields the sender address, and as such it will become harder to obtain information about the sender. This will affect statements S1, S2, S4; which now get score 0. Moreover, te transaction input will now be invisible resulting in score 0 for S9. Furthermore, because transaction linking will also be harder, S13 will have score 1. These changes have a similar effect as t-to-z transactions on the final score, which is 57%.

z-to-z transactions These transactions hide both sender and receiver information, affecting S1-3, S5-7. Moreover, inputs and outputs are invisible, as well as the

amount that is transferred in a transaction and linking transactions will now also be infeasible. As such, S9-11 and S13 also get score 0. As a result, the best possible privacy-preservation final score for Zcash is 83%.

Final score When, as an indication, a total final score is computed for Zcash, comprised of the average of the scores for each of the transaction types, then we obtain 58%. However, this can only be deemed a valid final score if all transaction types are used equally often, however, it has already been shown that this is not the case [50]. As such, a more justified final score for Zcash should be computed with weights on the transaction types, based on how often the types occur.

3.3.1 Summary

A summary of all scores and the resulting final privacy-preservation scores can be found in Table 3.1.

Statement	Perfect Privacy Coin	Bank-to-bank fiat	Bitcoin	Dash	Zcash (t-to-t)	Zcash (t-to-z)	Zcash (z-to-t)	Zcash (z-to-z)
S1: The sender's identity is revealed.	0	2	1	1	1	1	0	0
S2: The sender's wealth is revealed.	0	2	1	1	1	1	0	0
S3: The sender's location is revealed.	0	1	1	1	1	1	1	1
S4: The sender can be linked to their transactions.	0	2	1	1	1	1	0	0
S5: The receiver's identity is revealed.	0	2	1	1	1	0	1	0
S6: The receiver's wealth is revealed.	0	2	1	1	1	0	1	0
S7: The receiver's location is revealed.	0	1	0	0	0	0	0	0
S8: The receiver can be linked to their transactions.	0	2	1	1	1	0	1	0
S9: Transaction inputs are visible.	0	2	2	2	2	2	0	0
S10: Transaction outputs are visible.	0	2	2	2	2	0	2	0
S11: Transaction amounts are visible.	0	2	2	2	2	2	2	0
S12: A unique transaction ID is visible.	0	2	2	2	2	2	2	2
S13: Transactions can be linked to each other.	0	2	2	2	2	1	1	0
S14: Transactions can contain other information that reduces privacy-preservation.	0	1	1	1	1	1	1	1
S15: Transactions cannot be engineered to improve privacy-preservation	0	2	1	0	1	1	1	1
Final Score	100%	10%	37%	40%	37%	57%	57%	83%

Tab. 3.1.: Privacy evaluation of Bitcoin-like cryptocurrencies.

3.4 Evaluation

To further establish the viability of this framework, we decided to present it to multiple experts in the field of privacy(-preservation) in cryptocurrencies and request their opinion and feedback. This section discusses how this was done. First the questions that were asked to the experts are presented and the experts are introduced. Then, the results of this evaluation will be discussed, resulting in a set of improvements for the framework.

3.4.1 Expert review request

Each of the experts was asked to read this chapter on the privacy-preservation evaluation framework, and provide their insights. To guide this, a set of questions is used; which will help in structuring the results and covering several requirements that we deem relevant for evaluation of this framework. These requirements are:

- **Accuracy** in estimating privacy-preservation level. The framework is meant to give an indication of the privacy-preservation level of a cryptocurrency. This means that for experts usually at least an intuitively accurate score should result.
- **Completeness** of the included aspects in the framework. The statements reflect the aspects of which we think they are important to measure privacy-preservation. There may be technical or non-technical aspects of privacy-preservation that we miss in the current statements. For example, there may be privacy-preserving cryptocurrencies with a unique approach to privacy that we missed.
- **Comparability** of cryptocurrencies regarding privacy-preservation is a main goal. The framework should enable people who deal with privacy-preserving cryptocurrencies to obtain a general indication on how they compare and where their differences are
- **Durability** of this framework over time. Ideally, this framework should be applicable now and in the future, and it should be applicable to cryptocurrencies discussed in this chapter but also to many others alike.
- **Improvements** to this framework that were missed. We are aware this framework is not the final answer to evaluating cryptocurrencies regarding privacy-preservation. As such we try to gather, and if possible implement, potential improvements or alternative approaches.

Next to these aspects the experts can answer an open question to pose any comments or feedback they have on the proposed framework. Moreover, they are asked to write a short description of their experience and expertise regarding privacy-preserving cryptocurrencies. Finally, we request their permission to use their names and answers in this study. The resulting questions; which are asked to the experts are as follows:

According to your expertise and opinion:

Q1. To what extent does applying this framework provide an accurate estimation of the privacy-preservation level of a cryptocurrency?

Q2. Are there any technical or non-technical aspects of privacy-preservation that are not covered by the statements used in this framework? If so, which aspects were missed?

Q3. How well can this framework provide meaningful comparability across cryptocurrencies in terms of privacy-preservation?

Q4. How well will this framework be applicable in the future, with new developments in the field of privacy-preserving cryptocurrencies? Could this framework be used 20 years from now?

Q5. Other than the improvements discussed in the chapter; are there any improvements or approaches that were missed in this chapter but could provide additional value to privacy-preservation evaluation of cryptocurrency systems?

And:

Q6. Other than the points addressed by the previous questions, do you have any other issues or feedback you would like to share with us?

Q7. Can you briefly describe your past experience and expertise regarding privacy-preserving cryptocurrencies?

Q8. We would like to use your review in our work; do you give permission to use your answers and mention your name?

These questions, together with this chapter up to and including this section, are sent to the experts.

3.4.2 Experts

The experts who were asked to review the framework have varying background disciplines; through which we get a broad perspective on the viability of the frame-

work. Including multiple disciplines helps achieving a framework that is useful (for communication) across disciplines. The experts who took part in the evaluation of this framework are: Thijmen Verburgh, Bart Marinissen, and a cryptocurrency developer who prefers to stay anonymous.

Expert 1. Thijmen Verburgh - a social scientist who studied the use of cryptocurrencies by criminals.

Expert 2. Bart Marinissen - a researcher who studies privacy coins in the context of projects for Public Prosecution.

Expert 3. Cryptocurrency developer - an experienced developer of a cryptocurrency that also implements privacy features[4].

Although we cover a wide range of backgrounds in the experts that reviewed the current framework, further iterations of this framework and subsequent review may include more reviewers to gain more confidence in the framework.

3.4.3 Results

In this section, we first summarize each review and then discuss the implications of the reviews for the framework. The full reviews can be found in Appendix E.

Expert 1 Generally, this reviewer interprets the framework as an accurate method to estimate the privacy-preservation level of a cryptocurrency, while some remarks are given. In terms of missing aspects, the reviewer notes that some statements may have non-technical aspects (e.g. legality or availability of privacy services). Including these in scoring the statements, which are largely technical, may be too complex. Moreover, aspects like decentralization and management within a cryptocurrency affect the privacy-preservation potential but are not covered in the statements.

Regarding comparability and durability, the reviewer states that it is good that the framework is not limited to technical methods, and is able to provide an easy indicative comparison of cryptocurrencies. Although, the weight of statements and the impact they have on the final score should be studied and included. As an alternative approach to privacy-preservation evaluation the reviewer suggests to look at features rather than cryptocurrencies; since it may be interesting to see how features impact the statements.

[4]The name of this developer is known by the authors of this work. The developer preferred to stay anonymous.

The reviewer also mentions that theoretical possibilities (which are often not practically achievable) may pose a problem in the way we evaluate statements. A theoretical possibility would affect the score in the current framework, but this may not be realistic nor representative. Moreover, the world changes over time, and we should at least mention the impact of this in the chapter.

An important point that is noted is that the adversary that is used is unclear and not well defined. Its capabilities and knowledge need to be well specified. Furthermore, the framework's usability would benefit from different adversary types.

Finally, the reviewer suggests comparison with cash flows as well, next to bank-to-bank fiat, which we included.

Expert 2 Overall, the second expert considers the framework to be fundamentally flawed. We did also further discuss the feedback with the expert and will include the results from that discussion in this summary. The expert's core issue with the framework has two aspects:

- Reducing the results to a single score
- Fixing the threat model

The level of privacy-preservation in a cryptocurrency depends on the scenario in which the cryptocurrency is used. Different scenarios (and technologies) will impact the scores in different ways, this information is lost when subsequently the scores are combined into a single score. Moreover, how the evaluation of a cryptocurrency changes depending on the attacker capabilities (threat model) is interesting since it provides insight in the threats and how realistic those are.

To fix the core issues, the reviewer suggests to drop the single score and use individual scores for each statement, which are to be interpreted by an expert. The statements should also be improved to include more aspects (e.g. developer disposition, best-practices and common practices, anonymity set size). Furthermore, the statements should be evaluated over a range of attacker capabilities (thus varying the threat model) to see how the scoring changes and reveal weaknesses in given scenarios.

The reviewer notes there are many aspects to privacy-preservation that are not covered in the framework. Therefore the system is, according to the expert, inaccurate as well as incomplete. Some aspects are listed:

a. Size of anonymity sets

b. Ease of managing multiple pseudonyms

c. Behavior of user-base

d. Impact of attacker capabilities

e. Effect of available tools, guides, and best practices

f. Development team responsiveness & priorities

As per the arguments that have already been mentioned, the reviewer argues that the presented framework is not useful for cryptocurrency comparison. Although, if these issues were fixed the framework may give an interesting basis for comparison. Regarding durability, it is noted that the framework is focused on the Bitcoin model of transactions, and there will likely be new classes of attacks and cryptocurrencies, which are not covered in the current framework.

As mentioned before, the scenario in which the privacy-preserving capabilities of a cryptocurrency are evaluated has a lot of impact on the result. As such, the threat-model (or adversary model) and common usage mode should be included in (and revealed by) the framework, according to the expert. Moreover, conclusions should, generally, not come from just a final score but be based on an expert opinion. The framework should reveal the privacy-preservation performance of a cryptocurrency regarding given aspects of privacy, under given conditions. The reviewer argues this will allow comparison between currencies, under given conditions with respect to given aspects.

Finally, the reviewer critiques the fact that no weights are used in the framework, and notes that the use-case of the framework is unclear. Upon discussion, the reviewer does see some potential in this framework, in that it is a start to something that may be useful but it is not useful in its current form.

Expert 3 The third expert states that the framework provides a good starting point for evaluation, but to make it a meaningful comparison tool there are also many nuances to consider.

Regarding the accuracy of the framework, the reviewer notes that a qualitative approach may not be accurate for measuring the effectiveness of technologies. The argument that is presented boils down to that the framework is not granular enough to capture how effective one technology might be over another. Furthermore, the expert notes a lack of coverage of the user experience and the whole environment of a cryptocurrency (including third-party tools/services) in the framework. Aspects

like usability, user behavior and best-practices impact the privacy-preservation of a cryptocurrency, but are not included. As an example, Zcash transactions may be very private (z-to-z transactions) and will be if (and only if) this is the user's goal and they have the required knowledge and tools, otherwise they may just use regular transparent transactions. Other than these issues, the framework includes the relevant aspects of privacy-preservation in cryptocurrencies.

The expert notes that comparison results on the basis of this framework may be distorted, and supports this with an example of Dash versus Bitcoin. Although Dash scores better (as seen in the framework application in Section 3.3), it provides little advantage over Bitcoin if one considers the whole environment (including third-party mixers etc.). In fact, then Bitcoin has better alternatives in terms of privacy-preservation. It is argued that Dash is (somewhat) centralized, since Dash masternodes can know input-output pairs in mixing sessions, while better functionality exists in Bitcoin third-party services, where the mixing service will not gain any information on users of the service (e.g. solutions based on Zerolink[76]). Also, it is noted that whether functionality is built-in or not, does not mean it neccessarily gives better privacy, as seems to be assumed in S15. Moreover, S15 leaves a lot of room for interpretation, for instance, it is not clear what functionality is classified as improving user privacy-preservation.

Finally, the durability of the framework is hard to evaluate. The used statements may be general enough, however, the way users interact with cryptocurrency systems may and probably will change. New ways of interaction, for example through Layer 2 solutions[5], will require new statements as well.

3.5 Discussion & Future research

Based on the expert reviews and the application results, we will discuss the framework in this section. Through this discussion, a set of improvements are developed, which are to be made to this framework in the next iteration of its development. This section will be split up in subsections, each addressing a part of the framework. We will also regularly refer to the requirements that we used for evaluating the framework (Section 3.4.1), to highlight the impact of certain changes on the framework.

[5] Second-layer solutions (L2 solutions) are additional (network) layers on top of an existing blockchain network, which use that blockchain but aim to improve scalability and add other innovative features. An example is the lightning network[49] which operates on top of the Bitcoin network.

Overall, the framework is a good start for evaluating and comparing the privacy-preservation performance of cryptocurrencies, although there are many considerations to be made. As was found in the expert reviews, the approach can be valuable and is intuitive. However, the framework in its current form seems to miss some nuances of privacy-preservation in cryptocurrencies. Moreover, several improvements were suggested to the structure of the framework, in particular to its (lack of) flexibility and granularity. In this sense, we also found our framework to be in contrast with the anonymity framework presented in [11], which we found after developing this framework and we can also use this as inspiration for further development.

3.5.1 Adversary Model

The adversary model is introduced in Section 3.2.1, and is summarized as a 'powerful observer who can also participate'. As we note in Section 3.2.1, it is important to specify the adversary carefully. However, as was also noted in the received feedback, the adversary model is currently not well specified. Therefore, to improve the adversary model, a clear structure that can be used to specify the adversary's goal, knowledge and power should be introduced. In this study the adversary's goal is not explicitly stated, knowledge (prior knowledge) and power (capabilities and resources) are mentioned but all of these should be made more concrete. This is important for accuracy as well as comparability of the framework. In [11] the adversary knowledge and power are formalized using parameters; this creates a much more concrete model of what the adversary can(not) do and know. A similar approach could be used to improve the adversary model for this framework. This would also allow for flexibility in the adversary model while the differences between adversaries are specified, which was desirable according to the consulted experts.

The framework will be improved in multiple ways when different types of adversaries can be modeled:

- Completeness of the framework would be improved, since it can now reveal which aspects of privacy-preservation are much dependent on the adversary model, and how these aspects respond to a changing adversary.

- The framework can provide more comprehensive comparability, since it can highlight the differences between cryptocurrencies regarding their response to a changing adversary model.

- What is considered a realistic adversary will change over time, since new attacks and analyses that affect privacy will be discovered. Therefore a flexible adversary model is valuable for the durability of this framework.

We conclude that further development of the framework should include a flexible adversary model, which is concretely specified using a clear structure, for example by parameterizing the adversary knowledge and power similarly to [11].

Further research should also apply different adversary models, and observe how privacy-preservation scores change. In this study, the adversary has an observer-perspective, and the observer is able to observe public data that is available in the cryptocurrency system. Regarding knowledge, the observer may have access to data from the *present* and the *past*; each with their consequences:

- *Past*: When an observer looks to the past to obtain information in a cryptocurrency system; then they will rely purely on blockchain data to attack privacy of users.

- *Present*: When an observer observes what is currently going on in the cryptocurrency system's network they can obtain all relevant network traffic and relaying information, new transactions, and blockchain data since that is also always available.[6]

A new adversary model should include these types of knowledge, and be able to differentiate between them. For example, the parameter that specifies the knowledge of the adversary (regarding data that exists within the cryptocurrency system) may have the value past, present or both, indicating to what data the adversary has access. Moreover, the adversary could also have a different perspective, where it is not observing but participating, or both observing and participating, as a transaction sender or receiver. Effects of changing the perspective (and corresponding knowledge) on the adversary should be captured in the adversary model, such that future research can study the impact that adversarial knowledge (and capabilities) has on privacy-preservation performance of a cryptocurrency.

[6]Within the *present* perspective a further differentiation can be made, since what an adversary can achieve with data gathered in the present depends much on their capabilities. For example, if the adversary is able to monitor a specific user's network traffic, rather than just connecting to a cryptocurrency network and monitor what's incoming, then they will be able to more accurately link transactions to users. Such a scenario is not unrealistic, a government may decide to monitor someone because of suspicious activity. In this research we only considered an adversary who observes through regularly connecting to and participating in the cryptocurrency network. Having adversaries with other capabilities, corresponding to real-world scenarios, will provide more insights in privacy-preservation of cryptocurrencies.

3.5.2 Statements

For this framework we designed 15 statements that aim to cover privacy-preservation in cryptocurrency systems from multiple perspectives. However, we are aware that there may be aspects of privacy-preservation in this context that we do not cover. Moreover, some of the statements are closely related. For example, often when S13 is applicable then profiling can be done which can lead to S1 and/or S5 being applicable as well. Similarly, when S13 is not applicable, S4 and S8 are very likely to be not applicable either. The multiple-perspective approach that is taken is (part of) the cause of potential interdependence and relationships between statements. Further scrutiny on how these statements overlap, and what consequences this should have, would be valuable. Overlap could be compensated for using weights, or, if necessary, statements could be rephrased or changed.

The relevance of S12 is a point of discussion. Many blockchain-based cryptocurrencies utilize unique IDs for each transactions, although not all of them publicly show them (e.g. MimbleWimble implementations). In fact, these systems all use transactions, which can be timestamped and through that get an unique ID. As such, the score for this statement will be 2 for each cryptocurrency considered, except if a system does not employ (public) distinguishable transactions (e.g. like cash). Previously in this chapter it is stated that transaction IDs enable certain attacks, although these attacks may also be implemented when transaction IDs are not present, as long as transactions are publicly visible, only that will introduce more complexity. If weights are applied, the decreased relevance of this statement could also be accounted for using a low weight.

Although we did aim to keep the statements somewhat generic to support the durability of the framework, some statements are still too specific. On the other hand, some of the statements are too generic, and leave too much room for interpretation which will hinder comparability. Moreover, the statements do not cover all aspects that are relevant to privacy-preservation in cryptocurrencies. As such, there is a number of improvements to be made on the statements.

- The meaning of S4 and S8 is not clear, and, depending on how they are interpreted, they overlap with S1 and S5 or S13. S4 and S8, it aim to capture whether there are two-way links between transactions and their sender/receiver in a cryptocurrency. Thus, whether given a transaction it is possible to find the sender/receiver, AND, given a sender/receiver it is possible to identify their transactions. Rephrasing the statement to "There is a two-way link between the sender (receiver) and their transactions" and providing this explanation solves unclarity. However, there is still overlap with S13, which

looks at a similar issue of linkability but from a different perspective. This overlap can be accounted for using weights.

- S9 and S10 should be made somewhat more generic, they are currently focused on Bitcoin-like transactions. They could be rephrased to:

  S9. Transaction sources are visible.

  S10. Transaction destinations are visible.

- The availability of unique transaction IDs is not relevant, it is easy (for an adversary) to uniquely identify transactions, for example by hashing a transaction with a timestamp. Statement S12 should cover the public visibility of individual transactions, and should therefore be changed to something like:

  S12. Transactions are publicly visible.

  However, such a statement will generate much overlap with other statements. Another option is to remove it completely, or to account for this using weighted scores.

- S13 is too broad, it leaves a lot of interpretation room for how transactions may be linked. This may be fixed by introducing separate statements for different types of linking. The overlap with S4 and S8 can also be fixed by using more explicit statements.

- S15 currently suggest that having built-in privacy-preserving features provides better privacy-preservation than third-party features. As pointed out by the reviewers this is not necessarily true. Therefore, this statement should be split in two, one for internal and one for external privacy-preserving features.

- To account for missing aspects of privacy-preservation, which are not necessarily technical, new statements should be added. Through that we can capture more aspects of the cryptocurrency's environment that impact privacy preservation. Examples of factors that can impact privacy-preservation are: third-party tools and services (and their availability and legality), best practices, common practices, common behavior, usability (of privacy-preserving functionality), decentralization, developer disposition/priorities (with respect to privacy).

Another topic of future research is to find which of the used statements cause the biggest differences in scores. These statements are the most impactful, and are the best at differentiating between the cryptocurrencies. This information may subsequently be used to simplify the framework, by removing statements that have very little impact.

3.5.3 Scoring & weights

The statements are currently evaluated on a scale of 0 to 2. This obviously limits representing nuances and slight differences in performance between cryptocurrencies with regards to a statement. Therefore, to enable more precise scores and detailed comparison, a larger score range should be used. However, increasing the range also requires more detailed data, which is necessary for the user of the framework to more precisely differentiate between scores. This data may not be available, or may require additional research on certain cryptocurrencies. Additionally, if precise data is available for some cryptocurrencies but not for others, then using this precise data may hinder comparability of cryptocurrencies. Therefore, availability of (similar) precise data on the cryptocurrencies that are to be compared should be taken into account when evaluating cryptocurrencies.

When new statement scores are introduced, justification and explanation on what these scores mean should also be provided. Obviously, changing the score range also requires adjusting the formula for final score.

In this research we decided to not further complicate the framework by adding weights to the statements. However, to make the resulting scores more meaningful weights should be used; especially since some statements (e.g. S15) have more impact on privacy-preservation than others. These weights would alter the final score, given that the weight of a statement is a value between 0 and 1 ($w_{\mathbf{Si}}$), then the new formula for the final score is:

$$\text{FS} = \left(1 - \frac{1}{2 \cdot 15} \sum_{i=1}^{15} \left(w_{\mathbf{S}i} \cdot \text{Score}\left(\mathbf{S}i\right)\right)\right) \cdot 100\%$$

How to add weights to statements and what these weights should be are topics for future research. We expect that, when accurately justified, adding weights will be an improvement to the framework's accuracy. The importance of adding weights to the statements also resulted from the expert reviews.

Whether the final score is something that should be retained, should be a topic of further research as well. In the reviews we obtained, the opinions on this are mixed. A single final score could provide an intuitive, quick-glance comparison between cryptocurrencies. On the other hand, it oversimplifies the complexity of privacy-preservation, and many nuances are missed when only the final score is considered. There are various alternatives that may be valuable, a promising solution is to use a radar (spider) chart[7], which can help to visualize the differences between

[7] https://en.wikipedia.org/wiki/Radar_chart

cryptocurrencies in an intuitive manner. To avoid making the graph too complex, if there are many statements some could be aggregated in a category, while not losing as much complexity as with a single score. An example with the data that we generated in applying the framework is shown in Figure 3.1 (data in Table 3.1).

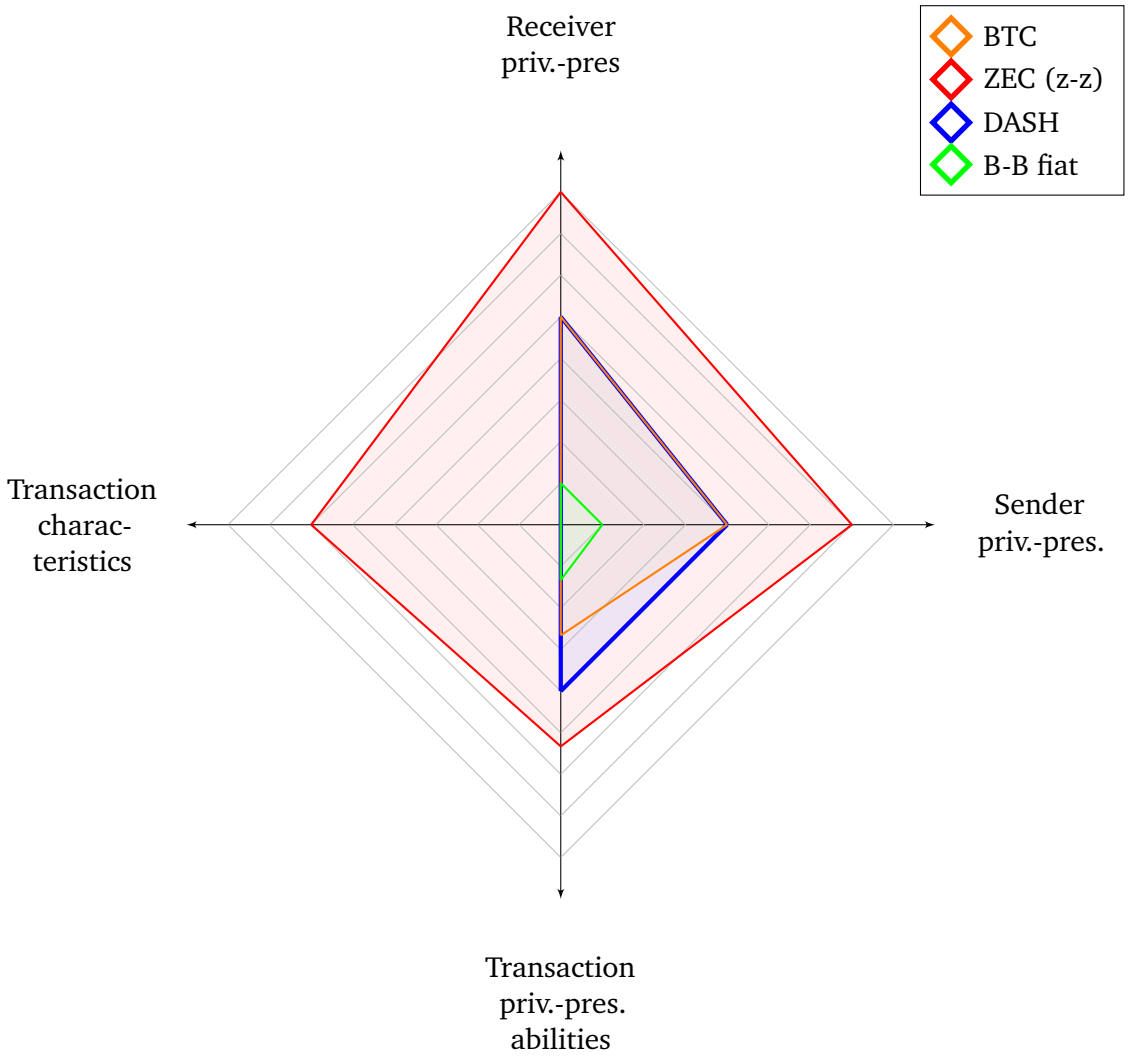

Fig. 3.1.: Privacy-preservation of different cryptocurrencies: framework application results in a spider chart.

In the example in Figure 3.1, we used the categories that were introduced in Section 3.2.2. Moreover, we assume that the categories have an even weight (which is not necessarily a right assumption but this is just for illustration). Since there are only 3 statements in the last category, we multiply the score in that category by $1\frac{1}{3}$. This will improve intuitiveness and readability of the graph.

3.5.4 Application

When applying the framework to cryptocurrencies, thorough research has to be done on what an adversary can achieve. This was also confirmed by expert feedback. Such research can consist of gathering available attacks presented in current literature, but can also include implementing new attacks. However, this also indicates that the framework results may change over time, since new attacks may be developed, resulting in changed scores for certain cryptocurrencies. Regularly repeating application of the framework to cryptocurrencies allows including new developments and monitoring privacy-preservation performance over time.

Time constraints limited this study to three cryptocurrencies for application of the framework. Further research should add more cryptocurrencies, which will contribute to proving the framework's viability and improve its usefulness since more comparisons can be made. For future research it is interesting to focus on cryptocurrencies that implement privacy measures. Examples of this are Decred (which implements CoinShuffle++[51]), Litecoin (which will soon release MimbleWimble[52]) and Bitcoin Cash (which uses CashShuffle and CashFusion[8]). These examples are limited to Bitcoin-like cryptocurrencies, many more cryptocurrencies with privacy features exist and are interesting. It is useful to extend the framework to cryptocurrencies that are blockchain-based but less similar to Bitcoin. For example, Monero, Beam, Firo and PIVX are interesting privacy-focused cryptocurrencies. Extending the framework to cryptocurrencies which are less comparable to Bitcoin might require adapting existing statements or adopting new statements.

Feedback that we obtained on the application of the framework confirms the necessity of thorough research and shows that we cannot evaluate the privacy-preservation capabilities of a cryptocurrency accurately without considering the 'environment' of the cryptocurrency. There are many non-technical factors that affect privacy-preservation performance, which must be included for the sake of completeness and accuracy of the framework. We previously listed a number of factors that are to be considered. The reviewers gave examples of various cryptocurrencies in which these non-technical factors play a (large) role. One of these examples is the comparison between Dash and Bitcoin, where it is argued that if the environment is taken into account Bitcoin could be seen as more privacy-preserving than Dash.

Expert 2 provided several comments on the application of the framework to the bank-to-bank scenario. They note that there are several methods that are used by money launderers to create privacy, such as using money-mules and bank-hopping. We did not consider this, and there is little room for this under the current statements,

[8] https://github.com/cashshuffle/spec/blob/master/CASHFUSION.md

although we can incorporate it under S15. Future application of the framework to a similar scenario should account for such methods as well, including them in the evaluation. Furthermore, the expert notes that the General Data Protection Regulation (GDPR) may impact this scenario, especially since it blocks banks and potential attackers from gathering and sharing financial data about users.

3.5.5 Evaluation

In this research a number of experts from various disciplines were consulted. The evaluation was done on a small scale to gain initial confidence on the framework's viability and gather feedback for improvements. A further iteration of this framework can be solidified by requesting feedback from more experts, including multiple per discipline. Future research may include this, or develop other forms of evaluation for this framework. Further evaluation is useful to find more improvements and to develop support and adoption for the framework.

3.5.6 Summary

The main takeaways from this discussion are summarized here. The general conclusion of the evaluation is that a next iteration of development of this framework would be much beneficial. In this iteration, the following points should be considered:

1. Improve the adversary model by providing a concrete structure for specifying the adversary and allowing for different types of adversaries.
2. Review and modify multiple statements to become more suitable.
3. Add statements to cover (non-technical) 'environment' factors that impact privacy-preservation.
4. Increase the granularity of the scores to improve framework accuracy.
5. Study how statements may be weighted to improve framework accuracy.
6. Reconsider the value of a single final score.

3.6 Conclusion

The primary goal of this chapter was to introduce a framework that establishes a basis for meaningful evaluation and comparison of cryptocurrencies regarding

user-privacy preservation. This was achieved by representing different perspectives on and aspects of privacy-preservation in statements, which, in the framework, are scored on their applicability to a cryptocurrency. Scores for each statement result, which can be combined to obtain a final score indicating privacy-preservation performance. The final score, and individual scores, which are also categorized, can be used to compare cryptocurrencies and uncover their differences.

The framework introduced in this chapter is built on ideas from previous literature, and was verified through applying it to several cryptocurrencies and obtaining expert reviews. Reviews were based on multiple questions posed to the experts. Applying and evaluating the framework led to new insights regarding privacy-preservation evaluation, and a number of improvements for the framework resulted. We conclude that the framework introduced here provides a good starting point for development of a privacy-preservation evaluation framework. Through this work we gathered the necessary literary background, created a general setup for a framework, and, through applying and evaluating it, uncovered many nuances to consider. This led to multiple suggestions for improvement and further research, which will ultimately lead to a viable framework for privacy-preservation performance evaluation in cryptocurrencies.

Applying the framework also highlighted the (lack of) differences between cryptocurrencies that are considered 'privacy-coins' and ones that are not. Specifically, we noted that there is little difference between Dash and Bitcoin, although the first is considered a 'privacy-coin', while the latter is not. This observation was also made by one of our reviewers, who stated that Bitcoin arguably supports privacy better than Dash. To settle this debate, further research into Dash's privacy-preservation performance is required, which motivated the research presented in upcoming chapters.

4 The Dash cryptocurrency

In the previous chapter the introduced privacy framework was applied to Dash, among several other cryptocurrencies. It became clear that the differences between Dash and Bitcoin are small, regarding user privacy-preservation. In this chapter, the Dash cryptocurrency will be introduced, which will set the stage for further analysis of its privacy features in upcoming chapters. This research can be used to enlighten the debate on the rightfulness of Dash's 'privacy-coin' label.

4.1 History

Dash was introduced to the market as Xcoin in 2014, which was soon rebranded to DarkCoin. For DarkCoin a whitepaper was published outlining its features[53]. The Xcoin (and thus DarkCoin) codebase was forked from Litecoin originally, although soon it was rebased off Bitcoin[54]. DarkCoin introduced some new features of which DarkSend and the masternode network were the most impactful. DarkSend allowed for peer-to-peer mixing, which is a technique to anonymize transactions. The masternode network was then developed to facilitate mixing. Over time, the masternode network's functionality has been extended to include governance, improving consensus, and Dash Evolution. After about a year DarkCoin was rebranded to Dash in 2015, which is a combination of the words *digital* and *cash*. DarkCoin was linked to dark web markets, negatively impacting DarkCoin's reputation and motivating this rebranding[55]. DarkCoin was interesting for dark web vendors because of the increased privacy it offered. The rebrand also aimed at shifting the focus of the coin somewhat away from privacy to being a usable cash-like currency.

Originally, Dash was developed to improve privacy and to address the fungibility issue of Bitcoin, as explained by the initiator Evan Duffield [56, 57, 58]. The coin was also marketed as privacy-centric. However, as time progressed the focus of Dash has shifted and the coin is increasingly marketed as a payments-focused coin. This means the focus is on transaction speed, accessibility and usability. This shift is especially expressed in the Dash whitepaper titles, which changed from *Dash: A Privacy-Centric Crypto-Currency* to *Dash: A Payments-Focused Cryptocurrency*[59] in 2018.

Still, Dash is dealing with its privacy-focused background. As governments more often require exchanges and services alike to implement Know-Your-Customer (KYC) and Anti-Money-Laundering (AML) policies, some of these services chose to delist privacy focused cryptocurrencies. Dash also faced issues in this regard, although it claims to be no different than Bitcoin in terms of available privacy features[60, 61].

Throughout its history, Dash has also faced some controversy, especially about its launch[62]. Some have regarded Dash a scam because a lot of coins were mined in the initial hours when the coin went live (called an instamine or fastmine), supposedly mainly by the initiators of the coin. This is addressed by Evan Duffield and others various times, and although it is admitted mistakes were made, it is made clear that there was no intentional fastmine or scam[63, 64, 65].

There have also been various (online) discussions about the actual privacy provided by Dash, however no (empirical) scientific published research is used to back up arguments about the effectiveness of Dash's privacy features[1].

4.2 Dash governance and masternodes

The Dash network and implementation is governed by a so-called Decentralized Autonomous Organization (DAO). This is an entity that is able to fund proposals for changes to or new features of Dash. Funding is awarded to proposals that pass a vote of the Dash network participants. Nodes can vote on proposals, and if a proposal reaches enough votes it will automatically be awarded funding by the DAO. A vote is passed if the net total of votes (positive minus negative votes) is positive, and, the net total is at least 10% of the maximal number of votes. As such, there must remain at least 10% of the maximal number of votes when all negative votes are subtracted from the positive votes. Only so-called masternodes are eligible to vote for proposals, therefore, the maximal number of votes is determined by the number of masternodes. Anyone can submit a proposal, although a fee of 5 Dash must be paid which avoids spam and stimulates well-thought proposals. Usually, proposals are discussed in various community channels before they are submitted.

Masternodes are nodes in the Dash network that provide services for some of Dash's features. They are incentivized to do this by a reward, which is made up of 45% of the block reward, which is split up according to the following rule:

- 45% for miners.

[1] Recently, a study has been published on Dash CoinJoin, which will be discussed next chapter.

- 45% for masternodes.
- 10% for the DAO for funding proposals.

To become a masternode, nodes must provide a collateral of exactly 1000 Dash. This creates a hard limit on the number of masternodes (total supply / 1000) and a soft limit because of the high cost and since the collateral is locked for the duration of running the masternode. The collateral also stimulates decentralization, since obtaining control over a large portion of the network is very expensive. Moreover, anyone trying to obtain many masternodes will raise the price in obtaining the required Dash, making it even more infeasible. For illustration, currently there are 4700+ active masternodes. An attacker that controls half of these would require over 2 million Dash (equivalent to 243.6 million USD[2]), where the current total supply is about 10 million.

4.3 Chainlocks

Resistance against 51% attacks is one of the challenges of blockchain based cryptocurrencies. Such an attack is possible when one entity possesses more than 50% of all hashing power in the network. In that scenario, an attacker can create his own chain by only adding blocks on top of his own blocks. This will succeed because the attacker generates the majority of the blocks. As a result, the attacker can censor transactions, ignore blocks and even double-spend[3]. Dash presents Chainlocks to improve protection against this attack and provide faster definitive confirmations of blocks. When a block is mined, a so-called quorum of masternodes sign the block if it was indeed the most recent block they received. This quorum is a set of masternodes that has been chosen at some point, and will be kept alive for some time to verify and sign consensus related messages. Multiple quorums are active at the same time, and one of them is picked to sign a chainlock and another to verify the signature. If a majority of the nodes in the quorum sign the block, a message is generated that 'locks the chain' by confirming that block as the most recent block. Up till the locked block, the chain can not be reorganized after a chainlock is issued. Moreover, future blocks must extend on that block. A locked block guarantees consensus over that block, and thus transactions in it are final. Furthermore, since that block is final, all blocks previous to that block can also be considered final. This means one usually does not have to wait for 6 confirmations (6 confirming blocks) as is common in other cryptocurrencies (e.g. Bitcoin).

[2] 1 Dash = 121.80 USD, `https://coinmarketcap.com/currencies/dash/`, date: 14-07-2021
[3] As an example, this recently happened in Firo, just before they implemented Chainlocks[66]

4.4 InstantSend

One of the unique features of Dash is its ability to generate consensus on transactions within seconds. This is enabled by the masternode network, which can validate transactions even before they have been confirmed in the blockchain. This is also done using masternode quorums that validate a transaction. If there is consensus within the masternode network on a transaction, then it is locked and cannot be changed anymore.

4.5 PrivateSend/CoinJoin

To improve fungibility and user privacy, Dash has implemented a peer-to-peer mixing system based on CoinJoin, called PrivateSend. Recently, while this research was in progress, the PrivateSend feature has been rebranded to CoinJoin. In short, users who want to make their coins fresh and remove any (easily) traceable history can initiate a mixing session for these coins. The funds are then split up in denominations, and a request to mix is sent to a masternode. The masternode then finds available peers, who want to mix the same denomination. Then, a transaction is formed that mixes the funds of the participating users, and this transaction must then be signed by all the users. Once it is signed, it is propagated to the network and will be included in the blockchain. Multiple rounds of mixing are chained together, to make sure the coins were mixed with many other coins, and as such traceability becomes very hard. Dash users can pick the number of mixing rounds they desire. Further description and scrutiny of Dash's CoinJoin functionality is given in Chapter 5.

4.6 Dash Evolution

Currently, Dash Evolution[4] is being developed, which consists of functionality to improve the usability of Dash. It is extensively described in [67]. Its core components are the Decentralized API (DAPI), second layer storage called Drive (off-chain) and blockchain accounts (rather than addresses). The DAPI will be hosted by masternodes and is accessible over HTTP, like a normal API. This should make it easy for external applications to include Dash, for example as payment functionality. The DAPI can be used to access accounts and make transactions. Blockchain accounts will improve usability and abstract away complex addresses. Moreover, second layer storage can be used in many ways by applications. This data storage will still be

[4]Recently also named Dash Platform.

secured by the blockchain, by including hashes of data in the blocks. Second layer storage can be used by Dash applications.

5 Dash privacy-feature analysis

The main privacy-preserving feature offered by Dash, which is meant to enhance the anonymity of its users, is called *CoinJoin*[1]. This is an implementation of the CoinJoin protocol, which was introduced in [19]. This chapter discusses Dash's CoinJoin implementation and its potential problems, which could weaken the anonymity gained by its users. To do so, first some relevant related research is discussed (Section 5.1). Then, the CoinJoin protocol steps as well as how these are implemented in Dash will be elaborated in Section 5.2 and Section 5.3. In Section 5.4, we explore the adoption of Dash CoinJoin. Next, in Section 5.5, the potential anonymity-weakening problems with the CoinJoin protocol will be discussed, while the issues that currently affect Dash's CoinJoin implementation will be scrutinized in future chapters. Finally, the potential impact of these issues will be discussed, and the chapter will be concluded in Section 5.6.

5.1 Related work

Previous studies have analyzed various mixing services, although most are not specifically focused on Dash. A recent study on Bitoin mixing services categorizes available mixers from multiple perspectives[68]. First, a distinction is made between centralized, decentralized and cross-chain mixers. For centralized mixers the users are dependent on a central party (e.g. a special node or a server) to mix their funds. Obviously this poses a single-point-of-failure threat, and requires trusting the central party. Decentralized mixers do not rely on a central party, although they do require some form of (secure) communication between mixing participants; causing extra overhead. Cross blockchain mixing services (offered by exchanges like Bisq[2] or (previously) ShapeShift[3]) allow a user to swap their coins with coins of another currency. This leads to the second distinction in mixing mechanisms, namely some operate using *swapping* functionality; whereas other employ *obfuscation*. In the case of *swapping*, the mixing service swaps the coins of one user with the coins of another

[1] This functionality used to be called *PrivateSend*. Recently most of the *PrivateSend* code has been refactored such that the functionality is now named CoinJoin.
[2] `https://bisq.network/`
[3] `https://shapeshift.com/`

user. Thereby, the coins of one user are mixed with another user's coins, and the user will receive coins that were not theirs. Obfuscation aims to hide a transaction sender in an anonymity set by employing transactions that have multiple inputs, and outputs, with the same value. In such transactions, it is hard to uncover the links between inputs and outputs. Through that, an anonymity set is created: only one transaction output relates to a sender, but which is unknown. In the study ([68]), the mixing mechanisms of several specific mixing services are empirically analyzed to determine whether obfuscation or swapping is used. Moreover, methods to find mixing transactions (from the blockchain) and estimate mixing profit and usage are presented.

Another study on cryptocurrency mixers focused on the feasibility and impact of so called sybil attacks on mixers [69]. In a sybil attack, which was formalized in [70], the attacker creates many identities with which it interacts with a system to undermine it. In [69], sybil attacks on cryptocurrency mixers are split in two categories: sybil-based deanonymization and sybil-based denial-of-service (DoS). Moreover, mixers are also split in multiple categories: they can be centralized or decentralized and obfuscation-based or zero-knowledge-based. Obfuscation is similar to how it was defined in [68]: it aims to hide the links between addresses by creating anonymity sets. Zero-knowledge-based mixing utilizes cryptography, which can completely erase visible links between addresses or transactions.

In sybil-based deanonymization attacks, the attacker creates many sybil identities with which they subsequently hope to be paired to a user who wants to mix their funds. Obfuscation-based mixers often use decoy addresses to hide the mixing users' addresses, if the attacker controls a (significant) portion of the decoy addresses then they can deanonymize users at best and decrease the anonymity set size at worst. Sybil-based DoS attacks allow the attacker to disturb the mixer's functioning by participating in the mixing with many identities. Then, in a decentralized mixing service, the attacker can decide to not cooperate when paired with certain users (deny mixing). Centralized mixers do not face this problem, since they can exclude non-cooperative users, although they may face regular DoS attacks.

Defenses against sybil attacks are also presented in [69]. Economic measures against misbehavior are suggested as the only viable option. Four methods, named *burning*, *time locking*, *fidelity bonds*, and *coin-age* are introduced; which all introduce some monetary cost to an attacker who wants to execute a sybil attack.

In the preparation of this study we did much more exploratory research in the field of mixers and mixing mechanisms. The results of this research can be found in Appendix B.

During the final stages of this research, a study was published in which Dash's CoinJoin is subjected to address clustering to achieve deanonymization of transactions[71]. By developing further heuristics and a deanonymization attack, the authors are able to link up to 43.8% of the transactions that spend CoinJoin outputs but are not mixing transactions back to the transactions in which the CoinJoin denominations were created. By linking these transactions the deanonymization that was supposedly introduced by CoinJoin mixing rounds that happened in between is undone.

5.2 CoinJoin protocol

One of the earliest transaction mixing ideas was introduced by Maxwell in [19]. In this forum post CoinJoin is introduced, which is a mixing mechanism that could be implemented in a decentralized as well as a slightly centralized version. The idea is that multiple cryptocurrency users meet via some (anonymous) platform (which is assumed to be in place) and then decide that they will construct a transaction together. The users decide on one common output amount and then all users will provide inputs and outputs of that amount and potential change outputs. These are combined together in one transaction, which is subsequently signed by all participants of the transaction. As a result, a valid transaction has been constructed of which it will be hard, for an observer, to link the outputs to specific inputs. The construction of a CoinJoin transaction is visualized in figure 5.1. In the figure, the users all only provide one input and output, which will be possible if they have the amount they want to mix readily available in an unspent output (which is always the case in Dash, but not necessarily in other implementations). Moreover, users may also provide multiple inputs and outputs, although usually there is a limit.

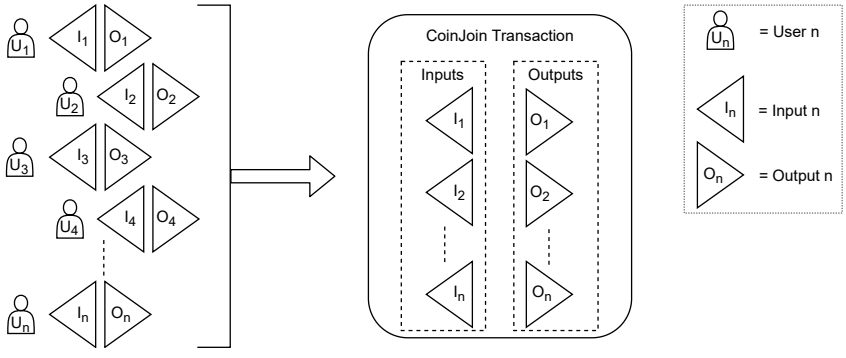

Fig. 5.1.: n users construct a CoinJoin transaction where each user provides one input I_n and one output O_n.

For the users who want to participate in a mixing session to find each other, as well as constructing the transaction, a central party could be used, although this is not strictly necessary. Commitments, blinding, and zero-knowledge systems could be used to even avoid that transaction participants (or the central party constructing the transaction) can link inputs to outputs[19]. With regular CoinJoin, users may learn (some of) the other participants' input-output pairs during construction of a CoinJoin transaction. If a central party coordinates the construction, then they will learn the input-output pairs corresponding to participants of a CoinJoin transaction.

This way of mixing transactions together creates an anonymity set based on how many users participate in one mixing transaction, and how many times a user decides to mix the same funds in a mixing transaction. Simply put, given some amount of funds of a user, if the user mixes this with n unique other users in a total of l rounds then their anonymity set has size n^l; assuming that there is no overlap in the other participants across mixing rounds. In fact, if the other participants are (known to be) the same in each round, then the anonymity set is n. The CoinJoin mixing mechanism is categorized as obfuscation mixing, as introduced in previous research, since an anonymity set is created through the CoinJoin transactions[68, 69].

5.3 Dash CoinJoin implementation

The Dash implementation of CoinJoin is discussed and described in [59], [72], and [73]. This section discusses what it entails and how it works.

Within Dash, the masternodes facilitate CoinJoin sessions. They connect users who want to mix their funds by creating the CoinJoin transaction from received inputs and outputs. In the context of existing research, Dash's CoinJoin implementation may be considered decentralized, since there is no central party that ever controls fund's or acts as a receiver or sender of funds. However, it can also be considered centralized, since a masternode will facilitate the construction of the CoinJoin transaction, and thereby also learn the input-output pairs.

The Dash documentation presents a series of steps that occur in a run of their CoinJoin implementation. To prepare for a CoinJoin session, the user must denominate their funds, which is done through a denomination transaction which is generated by the wallet when a user decides to mix coins. Moreover, the user must also have collateral transactions ready to join a mixing session; these are also generated by the wallet. Denomination transactions split up the funds that the user wants to mix into amounts that are used in Dash CoinJoin, the denominations. The denominations used in

Dash are 10.001, 1.0001, 0.10001, 0.010001, and 0.0010001 dash. As an example, 3.3333333 dash would be split in $3\times1.0001+3\times0.10001+3\times0.010001+3\times0.0010001$ dash. Denomination transactions are necessary since Dash CoinJoin requires common input amounts. Collateral transactions prepare some collateral that is to be submitted to the masternode that facilitates a CoinJoin transaction. The collateral is consumed by the masternode when a CoinJoin participant misbehaves, or randomly (with a probability of 0.1) to pay the miners for CoinJoin transactions.

A user who seeks to mix their funds, activates the CoinJoin feature in the Dash Core wallet, which does the preparation described above and then the following steps will be executed, as listed in the Dash documentation[72].

Step 0 **Pool selection.** To join a CoinJoin session, the wallet checks whether it knows of any existing open mixing pools, which may have been announced by masternodes recently through queue messages (*dsq message*).

Step 1 **Pool request.** The wallet joins an existing mixing pool through sending a *dsa message*, and otherwise requests a masternode to create a pool using the same message. The *dsa messsage* contains a collateral transaction that can be used by the masternode to make the user pay a fee (in case of misbehavior).

Step 2 **Pool response.** In response to the *dsa message*, the masternode sends a status update (*dssu message*), indicating a mixing session is started which is now in the queue state.

Step 3 **Queue** When a queue is started, the masternode broadcasts a *dsq message*, which wallets can reply to with a *dsa message* to join that queue. Once the queue is full, or a timeout is reached whilst enough participants are in the queue, the masternode sends another *dsq message* with a ready flag, indicating that the queue is ready to start creating the CoinJoin transaction.

Step 4 **Inputs.** Now the wallets that joined the queue must respond to the *dsq message* with a *dsi message*, in which they provide their inputs and outputs that must be included in the CoinJoin transaction. They must also again provide a collateral transaction, although this may be the same as the one provided previously, as long as it is not yet spent.

Step 5 **Status update.** In response to the received inputs and outputs the masternodes sends back a status update to the wallet, using a *dssu message*; which usually confirms that the inputs and outputs were added (to the transaction).

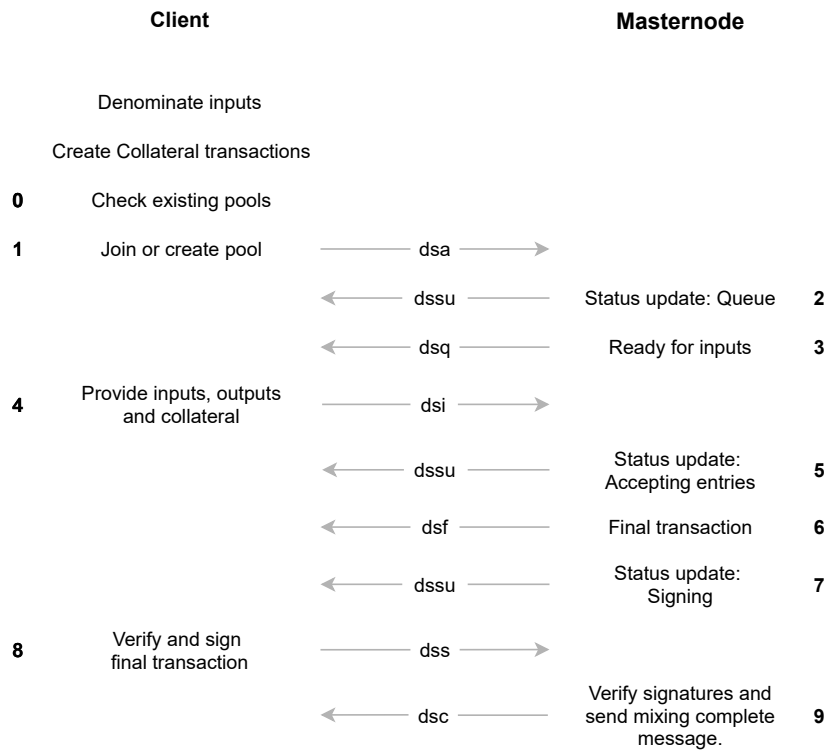

Fig. 5.2.: A successful run of Dash's CoinJoin protocol. (See steps explanations in 5.3 for message meanings.)

Step 6 **Final unsigned transaction.** After the status update, once the masternode has received the inputs and outputs from all participating wallets, the final transaction is created and sent to the participants using a *dsf message*.

Step 7 **Status update.** Moreover, the masternode sends another status update (*dssu message*) that indicates the wallets must now sign the transaction.

Step 8 **Sign transaction.** Subsequently, the wallets verify their in and outputs and if those are correctly included they sign their inputs. They communicate the signed inputs back to the masternode with a *dss message*.

Step 9 **Final transaction broadcast.** The masternode then verifies these signatures and finally creates the signed final transaction (*dstx*) which it broadcasts as well (using an *inv message*). Moreover, the masternode sends a *dsc message* to the participants stating that the CoinJoin session is complete.

A successful run of the CoinJoin protocol, with the corresponding messages[4], is also depicted in figure 5.2

[4]The message names in Dash CoinJoin always start with *ds*, this refers to *DarkSend*, the original name of Dash's CoinJoin functionality.

If all goes well, the *dstx* is broadcast and accepted, after which a wallet successfully completed mixing the provided inputs with one round of CoinJoin mixing. Now these inputs can mixed further in additional rounds, for increased anonymity, or remain in the user's wallet as mixed funds. The number of rounds to mix can be configured by the wallet user. By default, the wallet will aim to mix coins 4 rounds, the minimum of rounds is 2 and the maximum 16. Recently a feature called Random Round CoinJoin was added, to avoid attacks that assumed that all inputs provided by one wallet (for mixing) will be mixed the same number of rounds. Random Round CoinJoin will add extra rounds (with some probability), to decrease viability of such attacks[5].

5.4 Dash CoinJoin adoption

To gain knowledge on how much Dash's CoinJoin is being used we employed BlockSci[12] for analysis of Dash's blockchain. The BlockSci tool allows us to execute queries on Dash's blockchain and filter certain transactions. We use the tool to filter and count the CoinJoin transactions, and analyze and visualize CoinJoin usage over time. Analyzing how much Dash CoinJoin is used provides insight in the success of this privacy-feature and is indicative of their privacy-preservation performance. Moreover, it provides a basis for analyzing the cost and impact of potential exploits of this privacy-preserving feature.

5.4.1 Methods

With BlockSci, we can query the blockchain and filter transactions. To do that we have to provide heuristics which BlockSci applies to transactions to filter them. CoinJoin transactions are distinguishable on the Dash blockchain for various reasons, which we can utilize to build an accurate heuristic. First, they exist of at least three inputs and three outputs. Second, they must have an equal number of inputs and outputs. Third, CoinJoin transactions pay no fee to the miners, thus the total input amount must be equal to the total outputs amount. Fourth, the input and output values must be of one of the denominations which were listed earlier. These characteristics are transformed into a heuristic in code, which is sent to BlockSci, which subsequently filters out the CoinJoin transactions. These transactions are then counted over a block interval, and we can use the resulting data to visualize CoinJoin usage. The heuristic that filters CoinJoin transactions is noted below, and the

[5]Random round CoinJoin: `https://github.com/dashpay/dash/pull/3661`

(documented) Python[6] script used for extracting the data via BlockSci is provided in Appendix C. The heuristic is fed into BlockSci using a lambda function that operates on a transaction object (*tx*). For the transaction (*tx*) all the requirements described above are checked; the *.all()* function checks that all inputs of the transaction (*tx*) satisfy the requirement of being equal to one of the CoinJoin denominations.

```
lambda tx: (tx.fee == 0)
  & (tx.input_count == tx.output_count)
  & (tx.input_count >= 3)
  & (tx.inputs.all(
    lambda i: (i.value == 1000010000)
      | (i.value == 100001000)
      | (i.value == 10000100)
      | (i.value == 1000010)
      | (i.value == 100001)
    )
  )
```

Next to counting the CoinJoin transactions, we also count the total number of transactions in a block interval, which is then used to compute the ratio of CoinJoin transactions over regular transactions. This will allow us to see the development of CoinJoin adoption over time without the impact of over-all increase of Dash usage. The resulting block intervals with their CoinJoin counts, transaction counts and CoinJoin rates are stored in a CSV-file[7], which we then use for further analysis.

To create visualizations and to do further analysis on the gathered data we use a Jupyter notebook[8] in which we execute Python code to process and visualize the data. The Python code used in the notebook is also listed in Appendix C.

5.4.2 Results & Discussion

This section describes and visualizes the data that was gathered using BlockSci and processed through a Python Jupyter notebook. We instructed BlockSci to include all blocks from the Dash blockchain until block height 1512300 in this analysis. At the time of gathering the data, block height 1512300 was close to the most recent block.

[6] `https://www.python.org/`
[7] Comma-Separated Values file: a text file in which each line represents a data record and the value fields of that record are separated by commas.
[8] https://jupyter.org/

The main goal is to gather insight in the usage of CoinJoin within Dash. Therefore, we first visualize how often CoinJoin transactions are included in the blockchain, which serves as an indicator for CoinJoin adoption. We counted the number of CoinJoin transactions in each interval of 100 blocks, which results in 15123 data points each representing a number of CoinJoin transactions in a certain block interval. To improve visualization the number of data points is decreased by aggregating the number of CoinJoin transactions over 1000 blocks, and computing the average number of CoinJoin transactions per block. This results in over 1500 data points, each representing the average number of CoinJoin transactions per block over a block interval of 1000 blocks. (1000 blocks equates to about 43 hours.) To visualize the resulting data, the intervals are represented on the X-axis by the block height of the start of the interval, and for each interval the number of CoinJoin transactions per block are plotted on the Y-axis. The result is shown in Figure 5.3.

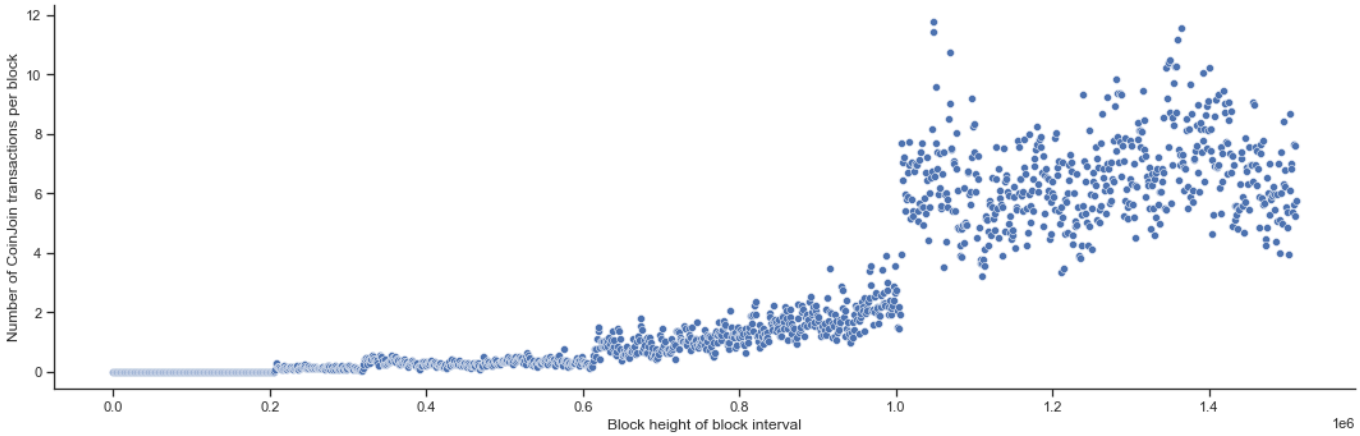

Fig. 5.3.: The number of CoinJoin transactions per block interval for all 1000-block intervals in the Dash blockchain.

As can be seen, throughout the history of Dash, the number of CoinJoin transactions per block has increased significantly. At various points in the graph (e.g. around block height 300000 and a little after 1000000) we see a sudden increase in the number of CoinJoin transactions per block. However, since the latest 'jump' in the number of CoinJoin transactions per block (around block height 1006000), the average number of CoinJoin transactions has not increased much, although the variance in the number of CoinJoin transactions per block has visibly increased. We can clearly see this if we zoom into the last part of the graph (block height > 1006000) and plot the CoinJoin counts averaged over 100 blocks (thus including more datapoints). We also plot a trend-line through the resulting graph, showing a minimal increase of CoinJoin transactions over time, the result is shown in Figure 5.4

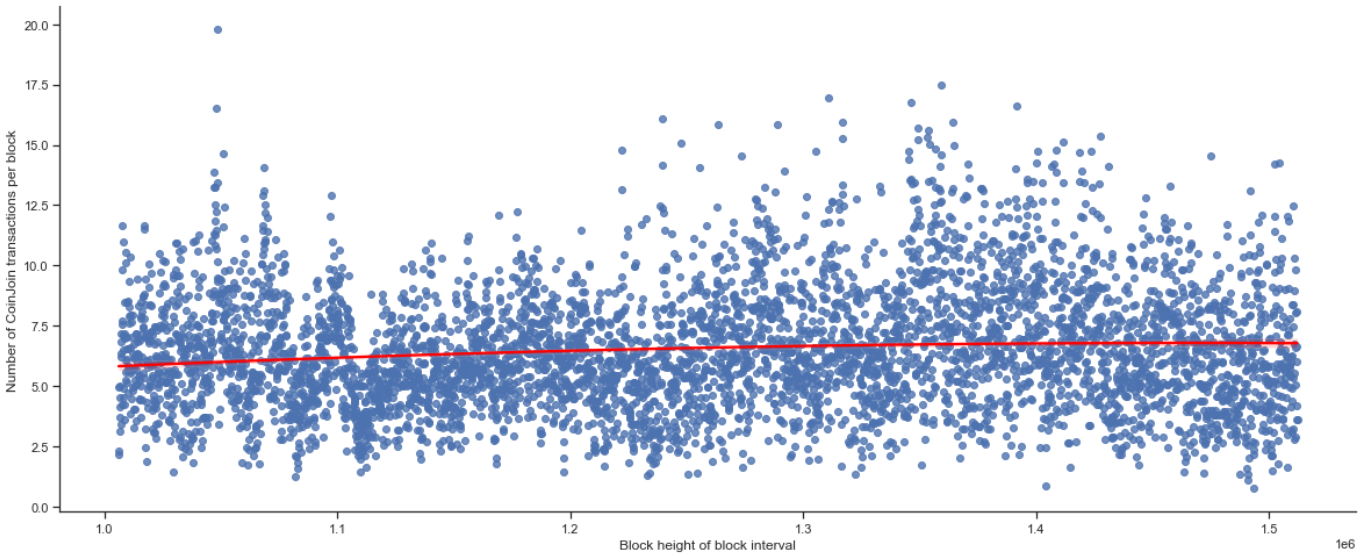

Fig. 5.4.: The number of CoinJoin transactions per block interval for all 100-block intervals in the Dash blockchain where the starting block height of the interval is larger than 1006000. The red line is a trend-line.

We also compute the mean and the standard deviation[9] for the data used in Figure 5.4, to get a range of how many CoinJoin transactions will usually be in a block. The resulting mean is 6.49 CoinJoin transactions per block; the standard deviation is 2.50. From this we can also conclude that most blocks will contain between 4 and 9 CoinJoin transactions, the maximal number of CoinJoin transactions per block that we observe in the graph is about 20. Although, it should be noted that the CoinJoin counts per block in the graph are averages over a 100 blocks (as one block interval contains 100 blocks). Therefore, individual CoinJoin counts per block could be higher than 20, although overall the estimate of 4 to 9 CoinJoin transactions per block is accurate.

To show how the number of CoinJoin transactions relates to the total number of transactions in a block we also compute the ratio of CoinJoin transactions to the total number of transactions. We compute this rate over 1000 blocks through dividing the number of CoinJoin transactions in a 1000 block interval by the total number of transactions in that same interval. We then also plot the block intervals on the X-axis (by the block height of the start of the interval) and the ratio on the Y-axis; showing the development of this ratio over time. The result is shown in Figure 5.5.

In the graph we see that the rate has generally increased over time. There are also some large fluctuations visible in the graph in Figure 5.5. These can be explained by (sudden) raises or drops in the amount of CoinJoin transactions or the total amount

[9] `https://en.wikipedia.org/wiki/Standard_deviation`

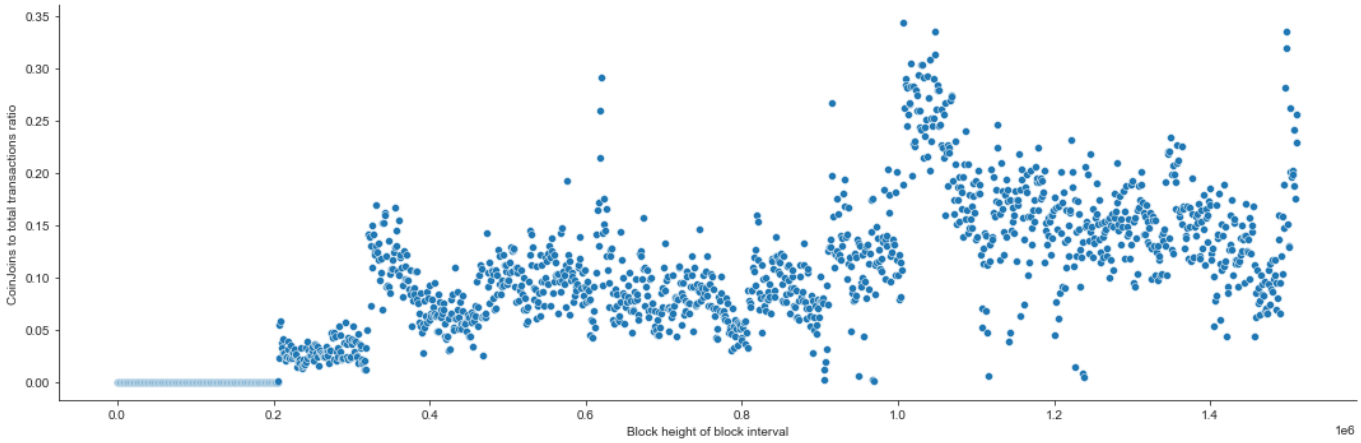

Fig. 5.5.: The ratio of CoinJoin transactions to all transactions for all 1000-block intervals in the Dash blockchain.

of transactions in a block. In Appendix D we have added graphs that visualize the total number of transactions over time, these graphs, together with Figure 5.3, can be used to explain the changes in rate.

As noted in the previous section, throughout the history of Dash CoinJoin there have been fluctuations in its use. Overall, usage seems to go up, although this is largely due to sudden increases in the number of CoinJoin transactions per block. Most of these sudden increases can be linked to new releases or updates of the Dash core software. For example, the first sudden increase in the number of CoinJoin transactions per block occurs near block height 20500, the block with block height 205000 was issued on January the 18th[10], close to the release data of Darkcoin core 0.11.0. This version fixed some CoinJoin issues (which was called DarkSend back then), even though, interestingly, this was not the earliest version of Dash that introduced the CoinJoin (DarkSend) functionality (which was 0.10.x) [54, 74]. Similarly, the sudden increase in CoinJoin transactions near block height 620000 happens close to Dash's 0.12.1 release which lowered cost and improved usability of CoinJoin (called PrivateSend then) [54].

5.5 CoinJoin issues

CoinJoin is not a perfect solution, it is found to be vulnerable to DoS attacks, users can behave maliciously by not following the protocol. For example, they can join a session but not present valid inputs or outputs, or purposely refuse to sign their inputs and outputs in the CoinJoin transaction. Such behavior will invalidate the

[10] `https://dashblockexplorer.com`

mixing session and other participants will have to mix again, as such they are denied access to the mixing service. This type of DoS is also feasible in Dash, although to discourage this Dash utilizes collateral payments that are consumed when this is detected. In Chapter 6 we implement a DoS attack on Dash's CoinJoin and analyze its cost.

Moreover, in [75] it was shown that having CoinJoin transactions where users do not settle on a common output amount, which did occur in practice, negates anonymity gains. Obviously, this will allow to link certain inputs and outputs based on their amounts. It is concluded that output splitting could be used when arbitrary amounts are desired, although following CoinJoin's original design of common output values is more effective. Output splitting ensures that for a given a set of inputs in a mixing transaction, multiple output sets may correspond to that input set. On the other hand, having confidential transaction amounts, such as available in Monero and Zcash, will also allow for CoinJoins without a common output amount. Since Dash requires common input and output values, the findings of [75] do not apply there.

Since CoinJoin requires some way for users who want to mix their funds to find each other, it may result in centralization. Dash is an example of this, the masternodes play a central role in CoinJoin sessions and can even unveil the individual input output links. This centralization problem is inherent to the design of CoinJoin. Since masternodes have knowledge on input-output links, they form a central point of failure in Dash's CoinJoin implementation. This is somewhat mitigated, since masternodes are randomly selected (under some restrictions) by the mixing clients to facilitate their mixing. However, there still seem to be ways to exploit the extra knowledge that masternodes can gain, which we further explore in Chapter 7.

Finally, as we already discussed in Section 5.1, in [71] an analysis and attacks are presented which indicate that even when CoinJoin is used, deanonymization may still be possible. This is largely because of mistakes by the users. If they combine outputs from denomination transactions with outputs from mixing transactions a so-called *backlink* is created, the anonymity that the outputs from mixing transactions had gained in mixing rounds is now gone. Moreover, if outputs from different mixing transactions are combined by a user, then a cluster-intersection attack can (sometimes) reveal a common denomination transaction (of these outputs) and as such the gained anonymity is undone.

More generally, during the study of Dash CoinJoin we realized that there is often a problem with change. Once a CoinJoin session is prepared, by creating a transaction that denominates a user's funds, there will usually be a remainder which is not denominated. This change, which is sometimes called '(radio-active) waste', can

leave traces, especially when it is recombined with CoinJoin outputs. The user behaviour becomes rather important when there is 'waste' like this, since depending on what the user decides to do with it, there may result avenues to decrease the user's anonymity.

5.6 Conclusion

In this chapter we discussed Dash's implementation of CoinJoin, elaborated how it works, and highlighted potential issues which will be further examined in upcoming chapters. We have explored the adoption of Dash's Coinjoin, and shown that it has seen increasing usage throughout the history of Dash, although absolute numbers of CoinJoin transactions per block are still low. Showing that there are ways to undermine the privacy-preservation that is offered by the Dash cryptocurrency may have impact on the view people have on Dash. The privacy-focused reputation of the Dash cryptocurrency all originated in the introduction of the CoinJoin implementation, back then called Darksend. Nowadays, Dash does not deem itself more privacy-focused than Bitcoin, and refers to their privacy-feature as just another CoinJoin implementation[61]. In that sense, showing that Dash's CoinJoin implementation does not provide fully privacy-preservation matches the perspective of the Dash Core Group. In this chapter we introduced a few issues with CoinJoin, which may also apply to Dash. We explore these issues in upcoming chapters.

6 Dash CoinJoin Denial-of-Service

As noted in the previous chapter, it is possible to execute a Denial-of-Service (DoS) attack on the CoinJoin protocol through malicious activity. In this chapter we explore this issue in Dash, and present an implementation and cost analysis.

To avoid that users behave maliciously in CoinJoin protocol runs, as described earlier, usually some functionality is implemented which punishes malicious behavior. This is also done in Dash, where the collateral transactions may be consumed to punish misbehaving CoinJoin participants. In practice, as noted in the documentation ([73]), the collateral transaction is consumed as a fee in two out of three (2/3) cases where a participant is uncooperative. The documentation also contains a reference to the code[1], where fee charging (collateral consumption) is implemented. This is done by the masternodes that facilitate CoinJoin sessions, their CoinJoin server functionality is implemented in *coinjoin-server.cpp*.

Although fee-charging functionality is there, when put to the test it appears that the cost of misbehavior in Dash CoinJoin is much lower than presented. This can, in part, be explained by a flawed implementation; the documentation and intention does not fully correspond with the code. As a result of the lower cost, it may be economical for some actors to execute a Denial-of-Service (DoS) attack in some cases. For example, a criminal investigation department that wants to temporarily avoid anonymization of Dash funds after a Dash exchange hack. Moreover, someone looking to devalue Dash or cause havoc may execute such a attack and publicize it. Although Dash's theory of protecting against malicious behavior is right, users can be malevolent without being punished most of the time.

Since we are actively looking at the Dash code to understand and elaborate the issues that enable attacks, it is necessary to specify what version of the code we consider. The analysis in this chapter is based on Dash release 0.17.0.0-rc5 [2], although it also

[1] `https://github.com/dashpay/dash/blob/master/src/coinjoin/coinjoin-server.cpp#L358-L374`, accessed November 11th, 2021

[2] `https://github.com/dashpay/dash/releases/tag/v0.17.0.0-rc5`

still applies to the latest release at the time of writing[3] since nothing changed in the code analyzed in this section.

The upcoming sections will discuss the CoinJoin DoS attack on Dash (Section 6.1 and Section 6.2), a (partial) fix (Section 6.3), and the cost and impact (Section 6.4). Moreover, limitations and further research ideas will be discussed (Section 6.5), leading to the conclusion of this chapter (Section 6.6).

6.1 Method

For an attacker to deny success to a single mixing session, they should participate in a mixing session. Once they are in, at some point in the protocol run they should refuse to cooperate, upon which the Dash masternode will have to abandon the mixing session and all other participants will have to find another session. Through that, the attacker has successfully denied one mixing session, and as such executed a DoS attack on the privacy-preserving mixing feature of Dash.

The attackers uncooperative behavior is implemented such that as soon as the transaction signing phase commences, the attacker stops cooperating and refuses to sign their inputs. To make sure that the attacker always refuses to sign during the signing phase of the mixing session, the attacker wallet source code is slightly changed. The wallet will always check for presence of their inputs and outputs in the final transaction. Normally, this check should evaluate to *true* when the masternode correctly constructed the final transaction. In the attacker wallet, this check is changed such that it always evaluates to *false*, such that the wallet refuses to sign[4].

The actual code change is implemented in *coinjoin-client.cpp*. The loop that checks whether the outputs provided for mixing by the client are in the final transaction is on lines 592-609. To make this check evaluate to false we set the variable that stores the result (of the check) to false. This is done by changing `!fFound` to `true` on line 600. The subsequent check on this variable results in the client forfeiting the CoinJoin session and accepting the risk of losing collateral.

Now to monitor the number of sessions that the attacker participated in, the debug log which is generated by the attacker client can be used. When the right

[3] `https://github.com/dashpay/dash/releases/tag/v0.17.0.3`

[4] There is also a way that does not require editing the wallet software. The attacker can decide to not let through the network packets that contain the signatures to their input; then the CoinJoin server will time-out, waiting for these signatures, and eventually abort the session. We have not tested this, but the idea is the same.

debug option is set, every time the attacker participates in a session, a message is printed right after the check described above. This message is *"CCoinJoinClientSession::SignFinalTransaction – an output is missing, refusing to sign!"*. Because this message appears only once per session, the number of occurrences of this message can be used to count the number of CoinJoin sessions that the attacker participated in. To exactly find the frequency of the message the linux tool *grep*[5] is used with the `-c` or `-count` option; which counts the number of lines in which a given string occurs. The exact command used to count the number of sessions is:

```
grep -c "an output is missing, refusing to sign!" debug.log
```

Next, to keep track of the costs, we retrieve the wallet balance (from the attacker wallet) before and after the attack. Moreover, all transactions that were made from the wallet during the attack are retrieved, such that the number of collateral payment transactions will be known. We aim to participate in at least 2000 CoinJoin sessions to gather data on attack costs.

6.2 Results

The initial balance of the mixing wallet was 19.98219981 tDash (testnet Dash). After taking part in exactly 2005 CoinJoin sessions, the wallet balance is 19.97603732 tDash. This can be used to compute the amount of tDash that was spent to take part in this many CoinJoin sessions, which is $19.98219981 - 19.97603732 = 0.00616249$ tDash. This amount can be used to estimate the monetary cost of executing a DoS attack. We compute a normalized cost (C by dividing the total amount spent (C_{total}) by the number of CoinJoin sessions (n) as follows:

$$C = \frac{C_{total}}{n} = \frac{0.00616249}{2005} \approx 307.356 \cdot 10^{-8}$$

The smallest unit of Dash (called a Duff) is equal to 0.00000001 ($1 \cdot 10^{-8}$) Dash; we take the conservative assumption that one Dash CoinJoin session will result in a cost of 308 Duffs (0.00000308 Dash). With the current exchange rate to United States Dollar (USD), which is 121.80 USD[6] this equates to a cost of 0.000375144 USD per session. To put this in perspective, the total cost over 2005 sessions in USD was about 0.75 USD. This is a very low cost, and will likely not hold back a motivated attacker.

[5] grep manual: `https://linux.die.net/man/1/grep`
[6] `https://coinmarketcap.com/currencies/dash/`, date: 14-07-2021

If the Dash implementation would follow its own specification, then we would have been punished 2/3 times, thus punished in 1336 sessions. Each punishment consumes a collateral transaction which should be 0.0001 Dash; which would result in a cost of 0.1336 Dash; which equates to about 16.28 USD. The intended cost is thus magnitudes higher than the actual cost.

During the 2005 CoinJoin sessions in which the attacker wallet participated, 55 collateral payment transactions and 13 make-collateral-input transactions were sent. Interestingly, not all the collateral payment transactions were equal to 0.0001 Dash; even though Dash's documentation mentions that collateral fees are 0.0001 Dash[72]. The collateral payment transactions account for a total of 0.0060169 tDash, 0.00014559 tDash was spent on fees for the make-collateral-input transactions, resulting in the total cost of 0.00616249. The attacker wallet took part in 2005 CoinJoin sessions and behaved uncooperative in all these sessions, thus we can compute the ratio of how often collateral was consumed. As mentioned before, a fee should have been charged in 1336 sessions (to reach the punishment ratio of 2/3 offenses charged). In fact it was consumed $\frac{55}{2005}$ times, which equals to about 1/36 offenses; roughly 2.7% rather than 66%.

Finally, we experimentally measured that our node processed about 3 to 4 CoinJoin sessions per minute.

6.3 Fixing the DoS cost

The Dash documentation clearly states that fees should be charged two out of three times when misbehavior is detected. To implement this, a slight code change should be done in the CoinJoin server. The function responsible for charging misbehavior fees (*ChargeFees*) should automatically return 1/3 times and run 2/3 times. Currently, it returns 2/3 times and runs 1/3 times. Line 374 in *coinjoin-server.cpp* which contains the following code, is responsible for this:

```
if (GetRandInt(100) > 33) return;
```

This should be changed to the code below; the if-statement above will evaluate to true 2/3 times and thus not charge fees 2/3 times, while the statement below evaluates to true 1/3 times and does does not charge fees 1/3 times.

```
if (GetRandInt(100) < 33) return;
```

However, this will only fix part of the problem. Currently, fees are charged for roughly 1 out of 37 detected misbehavior occurrences. This means that there is

another problem next to the implementation mistake in the *ChargeFees* function. In fact, in *init.cpp*, for masternodes a scheduler is set that calls *DoMaintenance* every second on line 2298. *DoMaintenance*, which is a *coinjoin-server.cpp* function, checks for time-outs and if these occur it will call *ChargeFees* to charge any offender that caused the timeout. In theory, this scheduled call should detect any offense. As such, we are not yet aware what causes this mechanism to fail, and this should be a topic of further research.

6.4 Impact

To determine the impact of the attack, we need to know how effective the attacker can be at participating in ongoing CoinJoin sessions on the Dash mainnet. For that, the attacker will need certain resources in terms of computing power, bandwidth, and presence in the Dash network (in terms of connections to nodes). Feasibility of this needs to be studied. From the research done in the previous chapter (Chapter 5), we know how many CoinJoin sessions occur every block. The attacker should be capable of joining and processing 4 to 9 sessions per block, although occasionally they would have to be able to process up to 20 sessions. Moreover the cost for joining and attacking all sessions in a block would be on average 6.5×0.00000308 Dash per block, which equates to about 0.002438436 USD per block. In terms of feasibility, we saw that our node, which has average computational capabilities, could process 3 to 4 CoinJoin sessions per minute, since the blocktime is about 2.6 minutes this means our node could process about 7.5 to 10 sessions per block. Therefore, on average one node like the one we used would suffice to process 4 to 9 sessions per block, and when multiple nodes are used it should be feasible to join all sessions.

In practice, the cost (and the number of CoinJoin sessions per block) is likely to increase: if no mixing sessions are getting through, and new nodes/wallets wishing to mix are coming in; then the total number of nodes/wallets trying to mix their funds will keep growing (until some limit, probably). As such, the attacker will face a rising cost when trying to attack all mixing sessions. We presume this rising cost will cause it to be infeasible to sustain the DoS for an extended period. How long the attack may reasonably be sustained, and what happens when a longer attack is attempted, could be a topic of further research.

On the other hand, when the attacker is somewhat smart and only blocks some of the mixing sessions, they may stay under the radar and disturb mixing only for some users. If an attacker can effectively participate in many (or most) of the CoinJoin

sessions that happen in Dash, the feasibility of a targeted DoS attack should be analyzed. Such an attack would use a blocklist of addresses who will be denied access to CoinJoins. Then, the attacker can decide to execute the DoS attack only on the CoinJoin sessions in which addresses from the blocklist take part. This gives the attacker quite some power, since they can, at will, decide to refuse some users access to Dash's privacy functionality. Still, it should be noted that they can only do this for a limited number of users at the same time, to avoid rising costs as described above. In fact, research can reveal how the cost correlates with the size of the blocklist (number of victims), which may be useful to know for an attacker and to gain insight in the potential impact of this attack. Also, such an attack would be detectable for the target if they consistently monitor their logs.

In short, it is rather hard to describe the impact of this attack. It seems that impact here much depends on the cost that the attacker is willing to accept: the larger the accepted cost, the larger the impact. Still, with very low monetary costs, the attacker is able to do targeted denial of service, which shows there is already significant impact with low cost. The potential impact of this attack also depends on its complexity, more complex attacks will be less feasible in general. To execute this attack, the attacker needs to be able to modify the Dash wallet implementation and recompile it, or they should be able to monitor and alter their network traffic. Moreover, they will need some computational capacity and network connectivity, although these are not really limiting requirements. The knowledge and skills required to execute this attack do limit the impact, since the attack can only be executed by someone who understands and can modify the Dash implementation and/or knows how to monitor and modify network traffic.

Depending on what variant of the attack is deployed, it may also be noticeable. A serious (non-targeted) DoS attack will most likely be noticed, which would likely result in patches from the Dash developers that disable or disincentivize it. This may greatly decrease the impact of the attack. On the other hand, if the attacker only rarely executes a targeted DoS, it may well go unnoticed, resulting in larger impact over time.

6.5 Discussion

The analysis of the DoS attack on CoinJoin, as presented in this section, has a few limitations. First, the attack was only performed on testnet. This choice was made to not disturb the mainnet functionality, nor hinder any Dash user in their experience. Moreover, we did not want our tests to be prematurely discovered, but first wanted

to analyze how and why the attack works like it does. Essentially, other than a few minor changes[7], the mainnet has the same CoinJoin/PrivateSend functionality and implementation as the testnet. Therefore, to our knowledge, performing the attack on the mainnet should not be any different.

The second limitation is that the cost estimation for the attack is only an estimation. It remains unclear why collateral fees are not consumed more often. Therefore, the cost was determined experimentally. Also, we only include monetary cost but the attacker also needs to pay for a machine, electricity and network connection (or hosting of such). Especially when the attack is sustained for increasing period of time that may result in extra costs.

In this study, only one node was used to execute the attack. To make sure that the attacker can participate in as many sessions as possible it would be better to utilize multiple nodes. In fact, when at least one node is used for each mixing denomination, the attacker can ensure that their nodes will not be mixing with each other. Moreover, the attacker then has a chance to participate in mixing sessions of any denomination; there will always be a match with one of the attacker's nodes. Using multiple nodes will also make the attack more efficient, the attacker can then process more CoinJoin sessions per minute. Furthermore, with more nodes the attacker can increase their presence in the Dash network such that they will be close (in terms of network hops) to any Dash node. This is advantageous because an attacker can then quickly discover newly created CoinJoin sessions and try to join.

As mentioned before, further research is required to gain more insight in the potential impact of the attack. First, the implementation of the attack should be further developed to include multiple attacker nodes (with each node using their own denomination), and to allow for targeted DoS attacks. This will provide a more sophisticated and realistic attacker scenario. Second, when a more sophisticated attacker implementation is available, the feasibility and cost of participating in many CoinJoin sessions (over a longer period of time) should be studied. After this, the impact of the attack can be more accurately estimated and described.

Throughout analyzing the Dash code, it was also noticed that in a CoinJoin session no fees will be charged if two participants of the same session present the same input. In fact, the inputs will both be declined, since the server cannot decide which one is legit. This could be used by an attacker to decrease the monetary cost slightly, at expense of potentially revealing to the victim that they are being targeted. There may be more ways to exploit this; where the attacker tries to join CoinJoin sessions (as many as possible) with (random) inputs that are not their own but that are likely

[7] `https://docs.dash.org/en/stable/developers/testnet.html`

to be used in CoinJoin transactions (e.g. outputs from denomination transactions). We leave further exploration of this attack vector and its potential impact to future research.

The attacker can exploit this to decrease the cost of a targeted DoS attack as follows. Given that an attacker wants to deny a user with input A access to CoinJoin. The attacker can try to join as many CoinJoin sessions as possible, where they provide input A as the input they want to mix. Now if the victim also joins that session, and also presents input A as their input, then both the attacker and the victim will get a notification from the server that their input cannot be included because of a conflict. In that way, the attacker blocks the victim's access to the session. However, if this repeatedly occurs, then the victim will become aware that they are being targeted by someone, who is maliciously using their input. This way of attacking saves some costs for the attacker, because they will never be charged fees after they have got the notification of conflicting inputs.

There may be another interesting way to decrease the cost of the attack by delaying CoinJoin sessions. If the attacker can join multiple CoinJoin sessions simultaneously[8], they can provide the same collateral transaction for multiple sessions. The attacker then tries to somewhat synchronize its activities in multiple CoinJoin sessions, for example by not immediately providing their inputs when they receive a *dsq ready* message. Obviously, this would require some care, to not prematurely time-out the session. Assuming the attacker succeeds in somewhat synchronizing n sessions, then it can only be charged collateral in one of these sessions (since it provided the same collateral to each). Further research should unveil whether this is a viable way to decrease the cost of this attack.

Another issue in Dash's implementation of CoinJoin is in the way that collateral is consumed. This is supposedly done by masternodes to punish abuse and randomly to pay miners. However, the implementation does not seem to punish a masternode that is malicious and always charges collateral. The feasibility and implications of this malicious behavior should be a topic of future studies.

Collateral payments do also leak information about a user's CoinJoin participation. CoinJoin collaterals are not (necessarily) created from anonymized funds. When collaterals are created and/or consumed, this is a sign that a user is participating in CoinJoin sessions. Exploring how much information an attacker can gain by observing collateral creation and consumption transactions is an interesting topic for further research.

[8]There is a 'multisession' option that can be enabled in the Dash wallet.

In Section 6.3 we provide a fix to increase the cost of this DoS attack. As we note there, this is only a partial fix. It may be necessary to apply further patches to sufficiently disincentivize this attack. As suggested in [76], (temporarily) banning non-cooperative IP-addresses could be used although this is not viable when anonymity networks (like TOR) are used. [76] advises (temporarily) banning non-cooperative UTXOs from further mixing sessions to be used for DoS protection. This may also be employed as a solution against the attack introduced in this chapter.

6.6 Conclusion

In this chapter, we studied an issue in Dash's CoinJoin implementation, which provides a cheap way for an attacker to execute a denial of service attack on the CoinJoin feature. Through that, the attacker can block mixing in general or execute a targeted attack. The costs of this attack are rather low, there are low monetary costs and no high-performance hardware is required, although there is some expertise required to implement it. There are more variants of this attack possible, we leave their implementation and analysis to further research. Moreover, the collateral that is used in CoinJoin sessions may provide additional information to attackers, this also requires additional analysis.

7 Dash queue gaming

In this chapter we address another vulnerability that was discovered while studying Dash's CoinJoin implementation. The general idea of this issue was introduced in [77] in 2014. At that time, the vulnerability was called DarkSend Queue Gaming, since DarkSend was a previous name of PrivateSend/CoinJoin. In this study we will address the attack as Dash CoinJoin Queue Gaming (DCQG), whereby the attack targets Dash CoinJoin queues.

Although the concept of the attack is briefly described in [77], no implementation details are presented. Moreover, this attack was discovered in DarkCoin version 0.10.12.17. We take a recent version of Dash (0.17.0.3), and study whether this vulnerability still exists. Moreover, we explore how an attack may be implemented, and aim for a proof-of-concept. Furthermore, we analyze how the issue may be fixed and what the costs and impact are, which are all not addressed in [77]. After our implementation attempt, we also discussed this issue with Dash developers, which provided new insights.

As a masternode you can announce CoinJoin Queues using *dsq messages*. However, there are some limitations to when a masternode can announce a queue. The *dsq message* will not be relayed by other network nodes if you recently announced a queue already; a certain amount of time must pass between two queue announcements by the same masternode. Still, even though other nodes will not relay the queue announcement, it seems they will join the queue if they have matching denominations. This implementation detail can be used by a masternode to gain information about someone trying to mix their funds. If the masternode is able to target a node that it (the masternode) does not want to be able to mix, then the masternode can send that node targeted queue messages. If that node subsequently joins the queues announced by the masternode, then the masternode knows the input-output pairs that are sent to it whilst mixing and it can trace the target's funds through mixing sessions (which it hosted).

In this chapter, we will further explore this vulnerability and how it may be exploited in the resulting DCQG attack (Section 7.1). We also briefly discuss our attempt at implementing it (Section 7.2), although we made a significant effort we did not manage to get it working due to time constraints and various issues we encoun-

tered. As noted, we also discuss a potential fix (Section 7.3) and the costs and impact of this attack (Section 7.4). Moreover, we provide some further research suggestions(Section 7.4), and conclude the chapter (Section 7.5).

7.1 Method

The attack utilizes an implementation detail of the Dash cryptocurrency. Dash nodes check if a masternode has recently hosted a CoinJoin queue when deciding whether they should relay incoming queue messages, and seemingly not when they decide whether to join a queue announced by an incoming queue message. This detail can be exploited by an attacker that owns a Dash masternode. They can make their target join their mixing sessions and believe it is successfully mixing, while the attacker controls the mixing session and knows input-output links.

The attacker can modify its own masternode code to allow itself to create CoinJoin queues more often than is normally allowed. Through a restriction that is implemented on the nodes, a masternode can not be host of a mixing session again until 20% of all other unique masternodes have hosted a mixing session. In this way, it is impossible for a few masternodes to dominate the queuing process. The attacker must modify their masternode implementation such that this restriction is removed locally. Other nodes will still use this restriction, however that does not prohibit the attack. After local modification, the attacker can send CoinJoin queue announcements (*dsq* messages) to their target, which is a Dash user who is trying to mix their funds. The target will, if the queue messages match the denominations that the target wants to mix, join the existing queue at the attacker's malicious masternode. When checking an incoming ready masternode CoinJoin queue, they will not consider whether the masternode may have hosted recent CoinJoin sessions, they only do this check when there are no matching denominations[1].

Since the regular peers of the malicious masternode will not forward the *dsq* messages, the attacker also needs to host some sybil Dash wallets (colluder nodes) which are modified such that they will join the CoinJoin queues of the malicious masternode. This is necessary because a CoinJoin session needs a minimum of three participants. Moreover, when their are sufficient participants the target will believe they are participating in a regular mixing session. The attacker can also send their

[1] In the version that we use for the attack, which is Dash Core release 0.17.0.3, the code that processes *dsq* messages if found on lines can be seen on lines 42-121 in *coinjoin-client.cpp*. On lines 88-94 we see that a node submits inputs without checking recent activity of a masternode.

queue announcements to other regular (non-colluder) peers, which may join as well. However, to avoid detection the attacker should use sybils.

As a result, the malicious masternode can host a CoinJoin session for the target while the target is not aware of the malicious masternode, nor of the fact that the other participants are sybils of the attacker. Thus, when the CoinJoin session succeeds, and the CoinJoin transaction is published, the target thinks they successfully completed a CoinJoin round which is supposed to increase their anonymity. However, the attacker knows the input-output pairs of that CoinJoin transaction, and thus from the perspective of the attacker the target did not gain any additional anonymity.

7.2 Implementation & results

In this study, we tried to implement the attack on a local Dash network, so as to not disturb functionality on the regular Dash networks, and to avoid potential ethical issues. Therefore, we set up a local network of Dash masternodes using *dashmate*[2]. This is a tool based on Docker[3] and NodeJS[4] developed by the Dash Core Group, to setup (local) masternodes and test networks for development. Then we used *docker compose*[5], to connect a small network of regular nodes to our *dashmate* masternode network. The next step is to activate mixing on the regular nodes, however, we were unable to get this working. Various issues arose throughout our research which prevented mixing on a local network. Some had to do with Dash's or *dashmate's* implementation, which did prevent mixing on small masternode networks, while others have unclear causes. The failure to execute mixing on a local Dash network could also be caused by undiscovered flaws in our testing environment or implementation. The steps we took to achieve mixing on a local network are elaborated in Appendix F.

When mixing on a local network is successful, the next step is to modify one of the masternodes to become an attacker masternode, and to also modify one of the regular nodes to become a colluder. In test networks, the number of required CoinJoin session participants is decreased to two, so one colluder node (sybil of the attacker) is sufficient. For a proof of concept of the attack, it would be enough to modify the masternode such that it will always create a new session when a client requests a mixing session, regardless of whether it hosted a mixing session recently. This means that the checks whether the masternode is eligible to host a

[2] https://github.com/dashevo/dashmate/
[3] https://www.docker.com/
[4] https://nodejs.org/
[5] https://docs.docker.com/compose/

session should be removed. Implementing targeted sending of *dsq* messages is not necessary for the proof of concept. The colluder node should be modified such that it will not pick a random masternode for hosting a new mixing session, but it should always pick the attacker masternode. Moreover, the colluder node should not check whether the masternode is eligible to host a session, but assume it always is.

With this setup, the concept of the attack can be proven. The attacker masternode should make sure they are connected to the target node, and the colluder node should start a mixing session at the masternode. Now as soon as the target node starts mixing (with the same denomination as the colluder node), and the target node receives a queue announcement of the attacker masternode, they should join the queue and mixing would commence. This would prove the possibility of the attack.

In this reseach we did implement the modifications that we presented, which are necessary for the attacker masternode and the colluder node. However, we were unable to test the attack, since we can not get mixing on a local Dash network to work. The steps we took to implement this attack, and the modifications we made to the masternode and colluder node, are elaborated in Appendix F. We did achieve several interesting results, resulting in bug fixes in the Dash core implementation[6].

7.3 Queue gaming fix

Dash Queue gaming is possible when the Dash nodes do not verify a masternode's eligibility to host a CoinJoin session, when they decide to join the session upon its announcement. Therefore, fixing the issue may be achieved by introducing a client-side check, to see whether a CoinJoin queue announcement is legitimate, before joining a CoinJoin session.

Still, it is impossible to fully fix this attack. Even if the client checks whether the queue announcement is legitimate, they will still not always catch a malicious masternode. This is because the malicious masternode will still be allowed to host a CoinJoin when they have the right to do so. Although, with introducing that check the Dash queue gaming attack is much less powerful since a targeted attack will have limited success. After one mixing session with the target, the target will move to another (random) masternode, and as such the attacker will only be able to undo 1 round of mixing.

[6] e.g. `https://github.com/dashpay/dash/pull/4394`

However, there is another issue then: if the attacker owns multiple masternodes, say n masternodes where n goes up to the maximum number of mixing rounds, then if they can do a coordinated attack on the target they may still be able to undo n rounds of mixing, while the target thinks they successfully anonymized their funds. This issue can only be fixed if the masternodes have no knowledge of what happens in mixing rounds. However, currently they construct the mixing transactions, and therefore they know the input-output pairs. Removing this centralization within CoinJoin, and adopting a solution like WabiSabi[78] or Zerolink[76] would fix this.

7.4 Discussion

There are several considerations to be made for the impact of this attack. First, we were thus far unable to implement the attack, which is something that should be done first, before considering this a threat. We did find various issues that hindered implementing this attack within a local test network, some of these issues were fixed as elaborated in Appendix F. Further research should aim to resolve the remaining problems, to get CoinJoin on a local network to work.

This attack allows for targeted 'demixing' of a user's funds, while the user is unaware. However, the attack does require significant expertise to be implemented and executed. Thorough understanding of the Dash core implementation and coding skills for the necessary modifications are required. Moreover, to execute the attack the attacker needs to have access to at least one masternode. Hosting a masternode will require computational, storage and networking resources, since to be a masternode multiple services must be provided to the Dash network. The attacker can not get away from providing these services, since the Dash network has a system that bans masternodes that do not provide them in a timely manner. To host a masternode, a collateral of 1000 Dash must be provided, although this is returned to the owner when the masternode is taken down. 1000 Dash equates to about $121800[7]. There are also some costs involved by hosting the CoinJoin sessions, since the attacker needs to also join these sessions with sybils, these sybils will be charged collateral.

There are several variants of this attack that may be explored. To make sure that the target joins the attacker masternode's session, the masternode could send *dsq* messages for each denomination. This ensures that the attacker and target have a matching denomination, and as such increases the chance that the target joins the attacker's session. It may not be possible to send all these announcements from the same masternode, however, to fix that, the attacker could use multiple

[7] https://coinmarketcap.com/currencies/dash/, date: 14-07-2021

masternodes. Further research should study the possibilities of announcing multiple queues simultaneously.

A masternode wins the masternode payment with a given frequency (depending on the amount of active masternodes), which is paid when a block is mined. The nodes that are trying to find a masternode to host their mixing session will skip the masternodes that will (soon) receive the next masternode payment. This is checked before a node joins a CoinJoin session, and therefore it is impossible to avoid this check. As such, the attacker will not always be able to make the target join their queue since when the attacker masternode is a soon-to-be masternode payment winner, then the target will refuse to join. This slightly reduces the impact of the attack, although there are many masternodes so this will not happen often on the main network.

It is also unsure how successful the attacker can be at making the target join the attacker's CoinJoin queues. For example, the target may also receive queue announcements from other (non-malicious) masternodes and decide to join their queue. Moreover, the target may not receive a queue announcement in time, and ask for a queue at another masternode. Future research should study how effective the attacker can be at hijacking the target's mixing rounds.

When disclosing this vulnerability and resulting attack to Dash developers, it became clear that it is unlikely that this attack has much impact. In this work we assumed that a client can join a CoinJoin queue after receiving only one *dsq* message with the ready bit set. However, we missed an aspect which prevents this. A client should only join a CoinJoin queue when it has first received a queue announcement (from the same masternode) without the ready bit set, and later (before a set timeout) received a queue announcement with the ready bit set. This prevents most of this attack, because when a client receives a *dsq* message without the ready bit set, then it will check whether the masternode is eligible for hosting the CoinJoin session. When it is not eligible, it will discard the *dsq* message and the corresponding CoinJoin queue. Therefore, the client would never join the session when subsequently receiving a *dsq* message from the same queue with the ready bit set. Testing and future research should uncover whether this actually behaves as expected from this discussion. It should also be noted that even when this is true, the scenario discussed in the last paragraph of Section 7.3 is still feasible. Since masternodes have a clear view of input-output pairs they will always be able to negate some anonymization. Although, requiring two *dsq* messages like this prevents that a masternode can host multiple subsequent CoinJoin sessions for a client.

7.5 Conclusion

In this chapter, we discussed and explored a vulnerability introduced in [77], in the way a Dash nodes processes incoming CoinJoin queue announcements. Using a (modified) masternode and some sybils, an attacker can exploit this issue to undo the anonymization that is achieved through a CoinJoin session. We provided implementation instructions for a proof-of-concept of this attack, but were unable to test it due to various issues and time constraints. We did manage to solve various issues along the line, which are elaborated in Appendix F. Future research should first focus on solving the remaining issues and then analyze the impact of the attack. Moreover, as noted in the discussion this chapter, upon discussing this attack with a Dash developer we found it may already be largely prevented in the implementation of Dash CoinJoin. This should be verified and tested, while feasibility of variants of the attack should be explored.

8 Conclusion and future research

This research started with posing three research questions, guiding the work that has been presented in previous chapters. In this conclusion, we come back to these research questions and summarize their answers on the basis of our results. Furthermore, we highlight the main avenues for future research. Many issues that have to be studied were already presented throughout the chapters. Therefore, for the details of future research suggestions we refer to the corresponding sections in the individual chapters.

8.1 Research questions

1. **How are privacy and anonymity defined in the context of cryptocurrencies? Can a comprehensive definition be developed that is useful across disciplines?**
   We gathered the currently used definitions in the literature on privacy and anonymity in distributed ledger technologies (DLTs). We summarized the definitions that we found in Table 2.1. In conclusion, on the basis of existing definitions we defined anonymity as “the state of being not identifiable within a set of subjects, the anonymity set", and privacy in the context of DLT as “the right of a DLT system user to keep secret any data stored or used within the DLT system that pertains to them". Through our research it became clear that privacy is regularly not interpreted as a right. Instead, privacy is often perceived to be equal to hiding things, which is a way of exercising that right. Therefore we decided to differentiate between privacy and privacy-preservation, where privacy-preservation is the protection of the right to privacy. As such a privacy-preserving method or technology does enable protecting the right of a user to keep data secret. We discussed many of the privacy-preserving technologies in Appendix A. The definitions that are introduced support communication across disciplines on privacy and anonymity in DLT systems.

2. **How can privacy of users in cryptocurrency systems be evaluated? How may a framework to guide this evaluation be developed and validated?**
   Similarly to definitions, current literature also does not agree on a unified framework or structure evaluating the privacy of users in cryptocurrency systems. The available approaches are discussed in Chapter 3, and a framework for privacy-preservation evaluation is proposed. This framework was subsequently assessed by experts, who provided their insights on the accuracy, completeness, comparability, and durability of the framework, and provided suggestions for improvements. Although the concept was received positively, the reviews uncovered various issues with the current version of the framework. We conclude a future iteration of development of this framework will be valuable. In this next iteration, introducing a more sophisticated adversary model and reviewing the statements to be used (and their weights), will be fruitful advancements.

3. **How (well) does the Dash cryptocurrency protect the privacy of its users? Can the privacy features it offers be exploited?**
   After application and evaluation of the framework, it became clear that some cryptocurrencies that have publicly been labeled 'privacy-coin' are close to cryptocurrencies without that label in terms of privacy-preservation performance (according to our framework). We specifically applied the framework to Dash, around which there has been some debate regarding whether it should be considered a 'privacy-coin'. We further investigated the privacy-preservation offered by Dash and found that there are various ways to diminish the privacy-preservation that Dash offers. We conclude that Dash does not protect the privacy of its users as well as may be expected from a 'privacy-coin'. Instead, Dash's privacy-preservation preservation seems to be comparable to (or arguably even less than) Bitcoin's. This indicates that some cryptocurrencies might unjustly be perceived as 'privacy-coin'. Generally, it shows that the introduced framework can be used to find interesting cases.

   In our study of Dash's privacy-preservation feature, we found multiple issues that could be exploited to decrease user privacy preservation. First, we discovered and implemented a denial-of-service attack, which can inhibit user's access to the privacy feature. Second, we discussed an attack that can decrease the anonymity that a user gains by using Dash's privacy feature.

8.2 Future research

As noted in Section 2.3, the privacy(-preservation) and anonymity of a user in a cryptocurrency system will be somewhere in a spectrum. Further research into defining these spectra could be useful. This should expand upon and specify the definitions introduced in this chapter.

The chapter on a privacy-preservation evaluation framework, which included evaluation, provides many avenues for future studies. In Section 3.5 we list multiple improvements that can be made to the framework that we introduced, which can be explored in further research.

Regarding the attacks that we implemented there are also still some gaps to fill. The CoinJoin DoS attack, discussed in Chapter 6 still has a partially unclear cause. Moreover, we were unable to test the Dash queue gaming attack because of various issues with running CoinJoin on a local Dash network. Studying these open ends will be valuable research contributions.

8.2.1 Other interesting topics

Throughout our research on privacy and anonymity in cryptocurrencies, and within our study of Dash's privacy feature implementation, we encountered various interesting topics that may be explored in future studies. We briefly note these in this section.

For this study, we gathered background information on technologies that can be used to improve user privacy and anonymity, as well as on attacks that aim to decrease user privacy and anonymity. A (regularly updated) complete overview of both technologies and attacks, structured in groups and classes, is a useful addition to current literature. Within the research on anonymity protecting mechanisms, we reviewed the plethora of mixing mechanisms that exist. There exist no comprehensive study of all these mixing mechanisms. A study on mixing including how different technologies compare and perform will be a valuable addition to current literature.

Through studying Dash's implementation we also encountered some interesting topics to study further. There may be more ways to exploit Dash's CoinJoin implementation. The potential issues we found are noted in Section 5.5, Section 6.5 and Section 7.4.

Bibliography

[1] Saqib Hakak et al. *Recent advances in Blockchain Technology: A survey on Applications and Challenges*. 2020. arXiv: 2009.05718.

[2] Marc Pilkington. "Blockchain technology: principles and applications". In: *Research Handbook on Digital Transformations*. Cheltenham, UK: Edward Elgar Publishing, 2016. ISBN: 9781784717759. URL: https://www.elgaronline.com/view/edcoll/9781784717759/9781784717759.00019.xml.

[3] Mehrdokht Pournader et al. "Blockchain applications in supply chains, transport and logistics: a systematic review of the literature". In: *International Journal of Production Research* 58.7 (2020), pp. 2063–2081. DOI: 10.1080/00207543.2019.1650976.

[4] J. Bao et al. "A Survey of Blockchain Applications in the Energy Sector". In: *IEEE Systems Journal* (2020), pp. 1–12. DOI: 10.1109/JSYST.2020.2998791.

[5] Qin Wang et al. "Blockchain for the IoT and industrial IoT: A review". In: *Internet of Things* 10 (2020). Special Issue of the Elsevier IoT Journal on Blockchain Applications in IoT Environments, p. 100081. ISSN: 2542-6605. DOI: 10.1016/j.iot.2019.100081.

[6] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[7] L. Herskind, P. Katsikouli, and N. Dragoni. "Privacy and Cryptocurrencies—A Systematic Literature Review". In: *IEEE Access* 8 (2020), pp. 54044–54059. DOI: 10.1109/ACCESS.2020.2980950.

[8] M. C. Kus Khalilov and A. Levi. "A Survey on Anonymity and Privacy in Bitcoin-Like Digital Cash Systems". In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 2543–2585. DOI: 10.1109/COMST.2018.2818623.

[9] Aisha Zahid Junejo, Manzoor Ahmed Hashmani, and Abdullah Abdulrehman Alabdulatif. "A Survey on Privacy Vulnerabilities in Permissionless Blockchains". In: *International Journal of Advanced Computer Science and Applications* 11.9 (2020). DOI: 10.14569/IJACSA.2020.0110915.

[10] Daniel Genkin, Dimitrios Papadopoulos, and Charalampos Papamanthou. "Privacy in Decentralized Cryptocurrencies". In: *Commun. ACM* 61.6 (May 2018), pp. 78–88. ISSN: 0001-0782. DOI: 10.1145/3132696.

[11] Niluka Amarasinghe, Xavier Boyen, and Matthew McKague. "The Cryptographic Complexity of Anonymous Coins: A Systematic Exploration". In: *Cryptography* 5.1 (2021). ISSN: 2410-387X. DOI: 10.3390/cryptography5010010.

[12]Harry Kalodner et al. “BlockSci: Design and applications of a blockchain analysis platform”. In: *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 2721–2738. ISBN: 978-1-939133-17-5. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/kalodner.

[13]Michel Rauchs et al. “Distributed Ledger Technology Systems: A Conceptual Framework”. In: *SSRN* (Aug. 2018). DOI: 10.2139/ssrn.3230013.

[14]Ali Sunyaev. “Distributed Ledger Technology”. In: *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*. Cham: Springer International Publishing, 2020, pp. 265–299. ISBN: 978-3-030-34957-8. DOI: 10.1007/978-3-030-34957-8_9.

[15]Serguei Popov. *The Tangle*. [White Paper]. Version 1.4.3. Apr. 18, 2018. URL: https://www.iota.org/foundation/research-papers.

[16]Leemon Baird, Mance Harmon, and Paul Madsen. *Hedera: A Public Hashgraph Network & Governing Council*. [White Paper]. Version 2.1. Aug. 15, 2020. URL: https://hedera.com/hh-whitepaper.

[17]Sarah Meiklejohn et al. “A Fistful of Bitcoins: Characterizing Payments among Men with No Names”. In: *Commun. ACM* 59.4 (Mar. 2016), pp. 86–93. ISSN: 0001-0782. DOI: 10.1145/2896384.

[18]Michele Spagnuolo, Federico Maggi, and Stefano Zanero. “BitIodine: Extracting Intelligence from the Bitcoin Network”. In: *Financial Cryptography and Data Security*. Ed. by Nicolas Christin and Reihaneh Safavi-Naini. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 457–468. ISBN: 978-3-662-45472-5.

[19]Gregory Maxwell. *CoinJoin: Bitcoin privacy for the real world*. Forum post. Mar. 22, 2021. URL: https://bitcointalk.org/index.php?topic=279249.

[20]E. Ben Sasson et al. “Zerocash: Decentralized Anonymous Payments from Bitcoin”. In: *2014 IEEE Symposium on Security and Privacy*. 2014, pp. 459–474. DOI: 10.1109/SP.2014.36.

[21]Aram Jivanyan. *Lelantus: Towards Confidentiality and Anonymity of Blockchain Transactions from Standard Assumptions*. 2019. URL: https://lelantus.io/lelantus.pdf (visited on 03/12/2021).

[22]Danny Bradbury. “Anonymity and privacy: a guide for the perplexed”. In: *Network Security* 2014.10 (2014), pp. 10–14. ISSN: 1353-4858. DOI: 10.1016/S1353-4858(14)70102-3.

[23]Qi Feng et al. “A survey on privacy protection in blockchain system”. In: *Journal of Network and Computer Applications* 126 (2019), pp. 45–58. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2018.10.020.

[24]D. Wang, J. Zhao, and Y. Wang. “A Survey on Privacy Protection of Blockchain: The Technology and Application”. In: *IEEE Access* 8 (2020), pp. 108766–108781. DOI: 10.1109/ACCESS.2020.2994294.

[25]J. Lee. “Rise of Anonymous Cryptocurrencies: Brief Introduction”. In: *IEEE Consumer Electronics Magazine* 8.5 (2019), pp. 20–25. DOI: 10.1109/MCE.2019.2923927.

[26]N. Alsalami and B. Zhang. “SoK: A Systematic Study of Anonymity in Cryptocurrencies”. In: *2019 IEEE Conference on Dependable and Secure Computing (DSC)*. 2019, pp. 165–174. DOI: 10.1109/DSC47296.2019.8937681.

[27]Andreas Pfitzmann and Marit Köhntopp. “Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology”. In: *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability Berkeley, CA, USA, July 25–26, 2000 Proceedings*. Ed. by Hannes Federrath. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 1–9. ISBN: 978-3-540-44702-3. DOI: 10.1007/3-540-44702-4_1.

[28]*Anonymity*. In: *Cambridge English Dictionary*. Cambridge University Press. URL: https://dictionary.cambridge.org/dictionary/english/anonymity (visited on 02/26/2021).

[29]*Privacy*. In: *Cambridge English Dictionary*. Cambridge University Press. URL: https://dictionary.cambridge.org/dictionary/english/privacy (visited on 02/24/2021).

[30]N. Khan and M. Nassar. “A Look into Privacy-Preserving Blockchains”. In: *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*. 2019, pp. 1–6. DOI: 10.1109/AICCSA47632.2019.9035235.

[31]Elli Androulaki et al. “Evaluating User Privacy in Bitcoin”. In: *Financial Cryptography and Data Security*. Ed. by Ahmad-Reza Sadeghi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 34–51. ISBN: 978-3-642-39884-1.

[32]Li Peng et al. “Privacy preservation in permissionless blockchain: A survey”. In: *Digital Communications and Networks* (2020). ISSN: 2352-8648. DOI: 10.1016/j.dcan.2020.05.008.

[33]Sarah Meiklejohn and Claudio Orlandi. “Privacy-Enhancing Overlays in Bitcoin”. In: *Financial Cryptography and Data Security*. Ed. by Michael Brenner et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 127–141. ISBN: 978-3-662-48051-9.

[34]Niluka Amarasinghe, Xavier Boyen, and Matthew McKague. “A Survey of Anonymity of Cryptocurrencies”. In: *Proceedings of the Australasian Computer Science Week Multiconference*. ACSW 2019. Sydney, NSW, Australia: Association for Computing Machinery, 2019. ISBN: 9781450366038. DOI: 10.1145/3290688.3290693.

[35]Mauro Conti et al. “A Survey on Security and Privacy Issues of Bitcoin”. In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3416–3452. ISSN: 2373-745X. DOI: 10.1109/comst.2018.2842460.

[36]Yuchong Cui, Bing Pan, and Yanbin Sun. “A Survey of Privacy-Preserving Techniques for Blockchain”. In: *Artificial Intelligence and Security*. Ed. by Xingming Sun, Zhaoqing Pan, and Elisa Bertino. Cham: Springer International Publishing, 2019, pp. 225–234. ISBN: 978-3-030-24268-8.

[37]Tianjiao yu and Chunjie Cao. “Privacy Protection in Blockchain Systems: A Review”. In: *Data Processing Techniques and Applications for Cyber-Physical Systems (DPTA 2019)*. Ed. by Chuanchao Huang, Yu-Wei Chan, and Neil Yen. Singapore: Springer Singapore, 2020, pp. 2045–2052. ISBN: 978-981-15-1468-5.

[38]Y. Li et al. “Toward Privacy and Regulation in Blockchain-Based Cryptocurrencies”. In: *IEEE Network* 33.5 (2019), pp. 111–117. DOI: 10.1109/MNET.2019.1800271.

[39]A. Averin, A. Samartsev, and N. Sachenko. “Review of Methods for Ensuring Anonymity and De-Anonymization in Blockchain”. In: *2020 International Conference Quality Management, Transport and Information Security, Information Technologies (IT QM IS)*. 2020, pp. 82–87. DOI: 10.1109/ITQMIS51053.2020.9322974.

[40]Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. “CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin”. In: *Computer Security - ESORICS 2014*. Ed. by Mirosław Kutyłowski and Jaideep Vaidya. Cham: Springer International Publishing, 2014, pp. 345–364. ISBN: 978-3-319-11212-1.

[41]I. Miers et al. “Zerocoin: Anonymous Distributed E-Cash from Bitcoin”. In: *2013 IEEE Symposium on Security and Privacy*. 2013, pp. 397–411. DOI: 10.1109/SP.2013.34.

[42]*About Monero | Monero - secure, private, untraceable*. Monero Core Team. URL: https://www.getmonero.org/resources/about/ (visited on 07/27/2021).

[43]Isabel Wagner and David Eckhoff. “Technical Privacy Metrics: A Systematic Survey”. In: *ACM Comput. Surv.* 51.3 (June 2018). ISSN: 0360-0300. DOI: 10.1145/3168389. URL: https://doi.org/10.1145/3168389.

[44]Michael Fleder, Michael S Kester, and Sudeep Pillai. “Bitcoin transaction graph analysis”. In: *arXiv preprint arXiv:1502.01657* (2015). arXiv: 1502.01657 [cs.CR].

[45]Steven Goldfeder et al. “When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies”. In: *Proceedings on Privacy Enhancing Technologies* 2018.4 (2018), pp. 179–199.

[46]Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. “Deanonymisation of Clients in Bitcoin P2P Network”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’14. Scottsdale, Arizona, USA: Association for Computing Machinery, 2014, pp. 15–29. ISBN: 9781450329576. DOI: 10.1145/2660267.2660379.

[47]Philip Koshy, Diana Koshy, and Patrick McDaniel. “An Analysis of Anonymity in Bitcoin Using P2P Network Traffic”. In: *Financial Cryptography and Data Security*. Ed. by Nicolas Christin and Reihaneh Safavi-Naini. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 469–485. ISBN: 978-3-662-45472-5.

[48]Jan Henrik Ziegeldorf et al. “CoinParty: Secure Multi-Party Mixing of Bitcoins”. In: CODASPY ’15. San Antonio, Texas, USA: Association for Computing Machinery, 2015, pp. 75–86. ISBN: 9781450331913. DOI: 10.1145/2699026.2699100.

[49]Joseph Poon and Thaddeus Dryja. *The bitcoin lightning network: Scalable off-chain instant payments*. 2016. URL: https://lightning.network/lightning-network-paper.pdf.

[50]George Kappos et al. “An Empirical Analysis of Anonymity in Zcash”. In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 463–477. ISBN: 978-1-939133-04-5. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/kappos.

[51]Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. “P2P Mixing and Unlinkable Bitcoin Transactions.” In: *NDSS Symposium*. 2017, pp. 1–15.

[52]Tom Elvis Jedusor. *MIMBLEWIMBLE*. 2016. URL: https://github.com/mimblewimble/docs/wiki/MimbleWimble-Origin.

[53]Evan Duffield and Kyle Hagan. *Darkcoin: Peertopeer cryptocurrency with anonymous blockchain transactions and an improved proofofwork system*. [White Paper]. Mar. 2014. URL: https://github.com/dashpay/docs/raw/master/binary/Dash%20Whitepaper%20-%20Darkcoin.pdf.

[54]Fernando Gutierezz. *Happy 5th Birthday Dash!* Jan. 18, 2019. URL: https://blog.dash.org/happy-5th-birthday-dash-6da05af9b5d2 (visited on 01/25/2021).

[55]Andy Greenberg. *Online Drug Dealers Are Now Accepting Darkcoin, Bitcoin's Stealthier Cousin*. Nov. 4, 2014. URL: https://www.wired.com/2014/11/darkcoin-and-online-drug-dealers/ (visited on 01/25/2021).

[56]Evan Duffield. *Rebranding and Scalability*. Forum post. Mar. 10, 2015. URL: https://www.dash.org/forum/threads/rebranding-and-scalability.4254/ (visited on 02/15/2021).

[57]Bitcoin Wednesday. *Evan Duffield Explains Dash Technology and Announces Evolution at Bitcoin Wednesday*. YouTube. Oct. 22, 2015. URL: https://www.youtube.com/watch?v=0Jw5Gk-iuy0.

[58]Jamie Redman. *An In-Depth Interview With Evan Duffield, Creator of Dash*. CoinGecko. URL: https://www.coingecko.com/buzz/interview-evan-duffield-dash (visited on 02/15/2021).

[59]Evan Duffield and Daniel Diaz. *Dash:A Payments-Focused Cryptocurrency*. [White Paper]. 2018. URL: https://github.com/dashpay/dash/wiki/Whitepaper (visited on 01/26/2021).

[60]Omar Hamwi. *Dash Complies with the Financial Action Task Force (FATF) Guidelines Including the 'Travel Rule'*. Oct. 23, 2019. URL: https://blog.dash.org/dash-complies-with-the-financial-action-task-force-fatf-guidelines-including-the-travel-rule-a4c658efc89d (visited on 02/15/2021).

[61]Andrew Thurman. *Following delisting, Dash pushes back against 'privacy coin' label*. Cointelegraph. Jan. 2, 2021. URL: https://cointelegraph.com/news/following-delisting-dash-pushes-back-against-privacy-coin-label (visited on 02/15/2021).

[62]Evan Duffield. *The Birth Of Darkcoin*. Forum post. URL: https://www.dash.org/forum/threads/the-birth-of-darkcoin.162/ (visited on 02/16/2021).

[63]Dashdot. *Was The Instamine A Positive Thing For Dash?* URL: https://dashdot.io/alpha/?page_id=118 (visited on 02/16/2021).

[64]Robert Wiecko et al. *Dash Instamine Issue Clarification*. Dash Core Group. URL: https://dashpay.atlassian.net/wiki/spaces/OC/pages/19759164/Dash+Instamine+Issue+Clarification (visited on 02/16/2021).

[65]TaoOfSatoshi. *Reality Check: The Truth about Dash's Launch!* Forum thread. URL: https://www.dash.org/forum/threads/reality-check-the-truth-about-dashs-launch.48625/ (visited on 02/16/2021).

[66]Andrew Asmakov. *Firo Gets Hit by 51% Attack: 300 Blocks Rolled Back*. Jan. 20, 2021. URL: https://decrypt.co/54751/firo-gets-hit-by-51-attack-300-blocks-rolled-back (visited on 12/10/2021).

[67]Dash. *Dash Evolution Initial Design Document*. 2017. URL: https://www.dash.org/wp-content/uploads/Dash-Evolution-Initial-Design-Document.pdf.

[68]Lei Wu et al. "Towards Understanding and Demystifying Bitcoin Mixing Services". In: *Proceedings of the Web Conference 2021*. WWW '21. Ljubljana, Slovenia: Association for Computing Machinery, 2021, pp. 33–44. ISBN: 9781450383127. DOI: 10.1145/3442381.3449880. URL: https://doi.org/10.1145/3442381.3449880.

[69]Mikerah Quintyne-Collins. *Short Paper: Towards Characterizing Sybil Attacks in Cryptocurrency Mixers*. Cryptology ePrint Archive, Report 2019/1111. https://eprint.iacr.org/2019/1111. 2019.

[70]John R. Douceur. "The Sybil Attack". In: *Peer-to-Peer Systems*. Ed. by Peter Druschel, Frans Kaashoek, and Antony Rowstron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260. ISBN: 978-3-540-45748-0.

[71]Dominic Deuber and Dominique Schröder. "CoinJoin in the Wild". In: *Computer Security – ESORICS 2021*. Ed. by Elisa Bertino, Haya Shulman, and Michael Waidner. Cham: Springer International Publishing, 2021, pp. 461–480. ISBN: 978-3-030-88428-4. DOI: 10.1007/978-3-030-88428-4_23.

[72]*CoinJoin*. Dash Core Group. URL: https://dashcore.readme.io/docs/core-guide-dash-features-privatesend (visited on 06/17/2021).

[73]*CoinJoin*. URL: https://dashcore.readme.io/docs/core-guide-dash-features-privatesend (visited on 07/13/2021).

[74]*Dash Core Source Documentation - Release notes*. URL: https://dash-docs.github.io/en/doxygen/html/ (visited on 10/20/2021).

[75]F. K. Maurer, T. Neudecker, and M. Florian. "Anonymous CoinJoin Transactions with Arbitrary Values". In: *2017 IEEE Trustcom/ BigDataSE/ICESS*. 2017, pp. 522–529. DOI: 10.1109/Trustcom/BigDataSE/ICESS.2017.280.

[76]TDevD nopara73. *ZeroLink: The Bitcoin Fungibility Framework*. URL: https://github.com/nopara73/ZeroLink (visited on 11/17/2021).

[77]Kristov Atlas. *An Analysis of Darkcoin's Blockchain Privacy via Darksend+ (v002)*. Sept. 19, 2014. URL: https://cdn.anonymousbitcoinbook.com/darkcoin/darksend-paper/ (visited on 09/09/2021).

[78]Ádám Ficsór et al. "WabiSabi: Centrally Coordinated CoinJoins with Variable Amounts." In: (2021). https://ia.cr/2021/206.

[79]F. Reid and M. Harrigan. “An Analysis of Anonymity in the Bitcoin System”. In: *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. 2011, pp. 1318–1326. DOI: 10.1109/PASSAT/SocialCom.2011.79.

[80]Ronald L. Rivest, Adi Shamir, and Yael Tauman. “How to Leak a Secret”. In: *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*. ASIACRYPT ’01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 552–565. ISBN: 3540429875.

[81]Eli Ben-Sasson et al. “Scalable, transparent, and post-quantum secure computational integrity.” In: *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 46. URL: https://eprint.iacr.org/2018/046.

[82]B. Bünz et al. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.

[83]Aram Jivanyan. “Lelantus: A New Design for Anonymous and Confidential Cryptocurrencies”. In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 373. URL: https://eprint.iacr.org/2019/373.

[84]Shaileshh Bojja Venkatakrishnan, Giulia Fanti, and Pramod Viswanath. “Dandelion: Redesigning the Bitcoin Network for Anonymity”. In: *Proc. ACM Meas. Anal. Comput. Syst.* 1.1 (June 2017). DOI: 10.1145/3084459.

[85]Giulia Fanti et al. “Dandelion++: Lightweight Cryptocurrency Networking with Formal Anonymity Guarantees”. In: *Proc. ACM Meas. Anal. Comput. Syst.* 2.2 (June 2018). DOI: 10.1145/3224424.

[86]Dorit Ron and Adi Shamir. “Quantitative Analysis of the Full Bitcoin Transaction Graph”. In: *Financial Cryptography and Data Security*. Ed. by Ahmad-Reza Sadeghi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 6–24. ISBN: 978-3-642-39884-1.

[87]Husam Al Jawaheri et al. “Deanonymizing Tor hidden service users through Bitcoin transactions analysis”. In: *Computers & Security* 89 (2020), p. 101684. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2019.101684.

[88]M. Möser and R. Böhme. “Anonymous Alone? Measuring Bitcoin’s Second-Generation Anonymization Techniques”. In: *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. 2017, pp. 32–41. DOI: 10.1109/EuroSPW.2017.48.

[89]Giulia Fanti and Pramod Viswanath. “Anonymity properties of the bitcoin p2p network”. In: *arXiv preprint arXiv:1703.08761* (2017). arXiv: 1703.08761 [cs.CR].

[90]Till Neudecker and Hannes Hartenstein. “Could Network Information Facilitate Address Clustering in Bitcoin?” In: *Financial Cryptography and Data Security*. Ed. by Michael Brenner et al. Cham: Springer International Publishing, 2017, pp. 155–169. ISBN: 978-3-319-70278-0.

[91] Haaroon Yousaf, George Kappos, and Sarah Meiklejohn. "Tracing Transactions Across Cryptocurrency Ledgers". In: *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 837–850. ISBN: 978-1-939133-06-9.

[92] Amrit Kumar et al. "A Traceability Analysis of Monero's Blockchain". In: *Computer Security – ESORICS 2017*. Ed. by Simon N. Foley, Dieter Gollmann, and Einar Snekkenes. Cham: Springer International Publishing, 2017, pp. 153–173. ISBN: 978-3-319-66399-9.

[93] Malte Möser et al. "An empirical analysis of traceability in the monero blockchain". In: *Proceedings on Privacy Enhancing Technologies* 2018.3 (2018), pp. 143–163. DOI: 10.1515/popets-2018-0025.

[94] D. A. Wijaya et al. "Monero Ring Attack: Recreating Zero Mixin Transaction Effect". In: *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. 2018, pp. 1196–1201. DOI: 10.1109/TrustCom/BigDataSE.2018.00165.

[95] Dimaz Ankaa Wijaya et al. "Anonymity Reduction Attacks to Monero". In: *Information Security and Cryptology*. Ed. by Fuchun Guo, Xinyi Huang, and Moti Yung. Cham: Springer International Publishing, 2019, pp. 86–100. ISBN: 978-3-030-14234-6.

[96] E. Daniel, E. Rohrer, and F. Tschorsch. "Map-Z: Exposing the Zcash Network in Times of Transition". In: *2019 IEEE 44th Conference on Local Computer Networks (LCN)*. 2019, pp. 84–92. DOI: 10.1109/LCN44214.2019.8990796.

[97] David L. Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms". In: *Commun. ACM* 24.2 (Feb. 1981), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/358549.358563. URL: https://doi.org/10.1145/358549.358563.

[98] David Chaum. "The dining cryptographers problem: Unconditional sender and recipient untraceability". In: *Journal of cryptology* 1.1 (1988), pp. 65–75. DOI: 10.1007/BF00206326.

[99] George Bissias et al. "Sybil-Resistant Mixing for Bitcoin". In: *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. WPES '14. Scottsdale, Arizona, USA: Association for Computing Machinery, 2014, pp. 149–158. ISBN: 9781450331487. DOI: 10.1145/2665943.2665955.

[100] Simon Barber et al. "Bitter to Better — How to Make Bitcoin a Better Currency". In: *Financial Cryptography and Data Security*. Ed. by Angelos D. Keromytis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 399–414. ISBN: 978-3-642-32946-3.

[101] Tim Ruffing and Pedro Moreno-Sanchez. "ValueShuffle: Mixing Confidential Transactions for Comprehensive Transaction Privacy in Bitcoin". In: *Financial Cryptography and Data Security*. Ed. by Michael Brenner et al. Cham: Springer International Publishing, 2017, pp. 133–154. ISBN: 978-3-319-70278-0.

[102] Pedro Moreno-Sanchez, Tim Ruffing, and Aniket Kate. "Pathshuffle: Credit mixing and anonymous payments for ripple". In: *Proceedings on Privacy Enhancing Technologies* 2017.3 (2017), pp. 110–129. DOI: 10.1515/popets-2017-0031.

[103] Joseph Bonneau et al. “Mixcoin: Anonymity for Bitcoin with Accountable Mixes”. In: *Financial Cryptography and Data Security*. Ed. by Nicolas Christin and Reihaneh Safavi-Naini. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 486–504. ISBN: 978-3-662-45472-5.

[104] Wafa F. Aldamegh and Laith A. Alsulaiman. “T-Mix: A Threshold Cryptography Mixing Service for Bitcoin”. In: *Sustainable Development and Social Responsibility—Volume 1*. Ed. by Miroslav Mateev and Jennifer Nightingale. Cham: Springer International Publishing, 2020, pp. 291–297.

[105] Luke Valenta and Brendan Rowan. “Blindcoin: Blinded, Accountable Mixes for Bitcoin”. In: *Financial Cryptography and Data Security*. Ed. by Michael Brenner et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 112–126. ISBN: 978-3-662-48051-9.

[106] QingChun ShenTu and Jianping Yu. “A Blind-Mixing Scheme for Bitcoin based on an Elliptic Curve Cryptography Blind Digital Signature Algorithm”. In: *CoRR* abs/1510.05833 (2015). arXiv: 1510.05833.

[107] Zijian Bao et al. “Lockmix: a secure and privacy-preserving mix service for Bitcoin anonymity”. In: *International Journal of Information Security* 19 (2020), pp. 311–321. DOI: 10.1007/s10207-019-00459-6.

[108] Muoi Tran et al. “Obscuro: A bitcoin mixer using trusted execution environments”. In: *Proceedings of the 34th Annual Computer Security Applications Conference*. 2018, pp. 692–701.

[109] Ethan Heilman et al. “Tumblebit: An untrusted bitcoin-compatible anonymous payment hub”. In: *Network and Distributed System Security Symposium*. 2017.

[110] N. Lu et al. “CoinLayering: An Efficient Coin Mixing Scheme for Large Scale Bitcoin Transactions”. In: *IEEE Transactions on Dependable and Secure Computing* (2020), pp. 1–1. DOI: 10.1109/TDSC.2020.3043366.

[111] Jan Henrik Ziegeldorf et al. “Secure and anonymous decentralized Bitcoin mixing”. In: *Future Generation Computer Systems* 80 (2018), pp. 448–466. ISSN: 0167-739X. DOI: 10.1016/j.future.2016.05.018.

[112] M. Xu et al. “CoinMingle: A Decentralized Coin Mixing Scheme with a Mutual Recognition Delegation Strategy”. In: *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*. 2018, pp. 160–166. DOI: 10.1109/HOTICN.2018.8605975.

[113] Sarah Meiklejohn and Rebekah Mercer. “Möbius: Trustless tumbling for transaction privacy”. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2018.2 (2018), pp. 105–121.

[114] barryWhiteHat. *Miximus*. URL: https://github.com/barryWhiteHat/miximus.

[115] István András Seres et al. “Mixeth: efficient, trustless coin mixing service for ethereum”. In: *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2019.

[116]Omer Shlomovits and István András Seres. “ShareLock: Mixing for Cryptocurrencies from Multiparty ECDSA.” In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 563. URL: https://eprint.iacr.org/2019/563.

[117]Xinyuan Zhang. *Mixing Strategies in Cryptocurrencies and An Alternative Implementation*. 2020. arXiv: 2010.01670 [cs.CR].

[118]Alexey Pertsev, Roman Semenov, and Roman Storm. *Tornado Cash Privacy Solution*. Dec. 17, 2019. URL: https://tornado.cash/Tornado.cash_whitepaper_v1.4.pdf.

[119]M. Möser, R. Böhme, and D. Breuker. “An inquiry into money laundering tools in the Bitcoin ecosystem”. In: *2013 APWG eCrime Researchers Summit*. 2013, pp. 1–14. DOI: 10.1109/eCRS.2013.6805780.

[120]Jaswant Pakki. “Everything You Ever Wanted to Know About Bitcoin Mixers (But Were Afraid to Ask)”. PhD thesis. Arizona State University, 2020.

[121]Svetlana Abramova, Pascal Schöttle, and Rainer Böhme. “Mixing Coins of Different Quality: A Game-Theoretic Approach”. In: *Financial Cryptography and Data Security*. Ed. by Michael Brenner et al. Cham: Springer International Publishing, 2017, pp. 280–297. ISBN: 978-3-319-70278-0.

List of Figures

List of Tables

A Privacy-preserving technologies and privacy attacks

This appendix consists of two main sections; first, Section A.1 discusses technologies that were designed to improve the user privacy-preservation in cryptocurrencies, wile Section A.2 considers attacks on user privacy presented in current literature.

A.1 Privacy-preserving and anonymity enhancing technologies

There are a number of DLT systems, and specifically cryptocurrencies, developed with a focus on privacy. Moreover, various technologies have been introduced to improve the privacy and anonymity of cryptocurrency users. The current section will discuss the lack of anonymity in Bitcoin and the vast amount of other technologies that have been presented to enhance privacy and anonymity. Moreover, cryptocurrencies in which these technologies are applied will be introduced.

A.1.1 Bitcoin

Historically, Bitcoin has often been perceived as an anonymous currency. However, in reality Bitcoin is rather a pseudonymous currency[17]. Transactions, and thus balances, are linked to addresses, which are all visible on the blockchain. Therefore, all the activity of an address can be traced and observed. Since presumably there is some entity (human, company, exchange, etc.) owning an address, the address really acts as a pseudonym for that entity. As such, if there is any way to link an address, a pseudonym, to a real-world identity, Bitcoin completely loses its privacy and anonymity features.

It is not unrealistic for pseudonyms to be coupled to real world identities. In fact, many exchanges gather data that allows coupling pseudonyms to real-world identities through their Know-Your-Customer (KYC) and Anti-Money-Laundering (AML)

policies. Moreover, many addresses are posted on social media and internet fora, which unveils additional links between Bitcoin pseudonyms and real-world identities. Additionally, techniques like internet scraping and clustering make deanonymization of Bitcoin pseudonyms much feasible. This has been shown in [17, 18, 79] among others. Since methods to deanonymize bitcoin users are publicly available, anyone with sufficient computing power can deanonymize such transactions and as such bitcoin users are pseudonymous at best.

Many cryptocurrencies have been based on the Bitcoin source-code, and developed themselves from there by adding features or changing parameters. This means that many deanonymization methods as well as anonymity improvements developed for Bitcoin can also be applied to cryptocurrencies. As such, the pseudonymity of Bitcoin implies many other cryptocurrencies are also pseudonymous at best, although, on the other hand, it likewise implies anonymity features developed for bitcoin can (often) be transferred to these currency systems as well. Pseudonymity, where users are identified by addresses acting as their pseudonym, can be seen as a weak form of anonymity, thus a weak form of privacy protection.

Bitcoin's lack of anonymity and privacy has stirred up many initiatives to improve these. In [7] and [8] elaborate discussions are presented of currently available features. This work and some additional research will be used to obtain an overview of the privacy-preserving and anonymity-enhancing technologies in the coming sections.

A.1.2 Coin mixing

One of the earliest methods to improve privacy and anonymity in Bitcoin was coin mixing or coin tumbling. This practice entails that coins of different users were mixed to confuse the 'trail of ownership' of the coins. The result of a mixing session would be that the mixed coins could be owned by anyone (who participated in the mixing), an observer should not be able to tell which mixing input corresponds to which mixing output. In the analogy of a 'trail of ownership' (where the trail is a path through the transaction graph that can be followed by a blockchain observer), mixing creates a junction with many branches, where many trails come together. However, for an observer it will be impossible to link an input and output trail because (at least) all output trails look alike such that an input trail could be linked to any output trail. In fact, usually mixing means that many transactions or funds are aggregated and then sent out again in a permuted fashion. Many mixing mechanisms have been proposed, starting on the BitcoinTalk forum, and later extending into the literature. A separate section of this report (Section B) will address various

mixing mechanisms to understand what features are offered and how they may affect anonymity and privacy.

A.1.3 Ring signatures

A technology to improve anonymity in blockchain based cryptocurrencies that works similarly to some mixing mechanisms is based on ring signatures. Ring signatures, introduced in [80], are a special type of group signature that will hide the signer of some message in a group of signers, the ring. Applied to cryptocurrencies this means that the sender(and thus the signer) of a transaction can hide among a set of probable senders, namely all members of the ring that was used to sign the transaction. As a result, the ring functions as an anonymity set for the sender of a transaction. An observer knows someone from the ring sent the transaction but they cannot determine who.

A.1.4 Stealth addresses

Another method to improve the anonymity of cryptocurrency users is using stealth addresses. These addresses are receiver addresses that are generated jointly by the sender and the receiver of a transaction. These addresses are meant to be one-time addresses, meaning that only one transaction will ever have that address as a destination. This protects the privacy of the receiver since the sender will not immediately learn about balances or other activities of the receiver through knowing this address. Still, if a users links their stealth address to other addresses they own, their privacy might be affected.

A.1.5 Zero-knowledge proofs

Zero-knowledge proofs have been used in various cryptocurrencies in different forms to support privacy. As explained in [7], a zero-knowledge proof "is a proof that convinces a verifier of some statement, without revealing any information other than that the statement is true". In cryptocurrencies this can, for example, be used by a transaction sender to prove they have control over some funds they want to send without showing to the public how many funds they own or want to send, or where these are stored. Examples of zero-knowledge proof schemes are, among others, zero knowledge succinct non-interactive arguments of knowledge (zk-SNARKS), zero-knowledge scalable transaparent arguments of knowledge (zk-STARKS)[81] and

Bulletproofs[82]. In practice, zero-knowledge proofs are used to hide transaction amounts as well as sender and receiver addresses.

A.1.6 Confidential transactions

Confidential transactions are an improvement focusing on transaction content, in confidential transactions the amounts that are transacted are hidden. To enable verification while amounts are hidden, cryptographic commitments to values are used. The used commitment schemes are homomorphic, which means that still some checks and verifications can be done on commitments of values without revealing the actual values. This can be used to check that total input and total output amounts of a transaction add up to zero; avoiding any double spending or creating coins out of nothing.

A.1.7 Mimblewimble

Mimblewimble is a technique that extends from confidential transactions to hide senders, receivers and transacted values. On a Mimblewimble blockchain, no addresses or amounts are visible, transactions consist of commitments which do not reveal anything about the sender or receiver to outsiders. Moreover, Mimblewimble allows for aggregation of multiple transactions into one transaction, saving space and acting as further confusion of transactions, just like what happens in mixers. Also, Mimblewimble provides a feature named *cut-through* to eliminate intermediary transactions, such that, when Alice sends a given amount, Bob receives that amount, and then Bob sends it again to Charlie, only the inputs from Alice and the outputs to Charlie have to be preserved on the blockchain, which allows for more space savings. Moreover, *cut-through* means some transactions will never leave any trace of data on the blockchain.

A.1.8 Lelantus

Lelantus[21, 83] builds on confidential transactions, zero-knowledge proofs and commitments schemes to develop a payment systems that provides anonymity of the sender and receiver, as well as confidential transaction values. In its privacy guarantees it is similar to the Zerocoin project[41], although Lelantus relies on more established cryptography, and does not require a trusted setup where Zerocoin does.

A.1.9 Network level mechanisms

The privacy and anonymity of cryptocurrency users may also be affected by their participation in the network that is part of the cryptocurrency system. Studies have been done that show deanonymization of users can be done based on their IP-addresses. Therefore, it has been suggested that cryptocurrency systems should employ mechanisms to allow users to anonymously participate in their network. For this, The Onion Router (TOR) and the Invisible Internet Project (I2P) are suggested as potential solutions. Moreover, Dandelion[84], and its improved variant Dandelion++[85], have been proposed, which are protocols to enhance propagation mechanisms such that deanonymization on the basis of propagation observations becomes less feasible. Other solutions that have been proposed to improve network-level anonymity are classic mix networks and dining-cryptographer's networks, although these face scalability issues.

A.1.10 Secure payment channels

Several payment channel protocols have been suggested to make off-chain transactions in a cryptocurrency network. In these protocols, users, e.g. Alice and Bob, use the blockchain to set up a channel between them, in which they (both) invest some funds. Then, while the channel is alive, which it stays until someone terminates it or until some set time, the funds are locked in the channel and can move back and forth between the users who set up the channel. When the channel is terminated, the current status of the balances in the channel is settled with a transaction on the blockchain. Various technologies, such as multi-signature addresses and temporary locks are used to secure these channels.

As pointed out in [24], these payment channels can be used to improve privacy. Transactions that happen in a channel do not occur on-chain and will thus not be publicly visible to everyone. Obviously, when extending this to a payment channel network by interconnecting payment channels between different users, there are cases in which some intermediaries between Alice and Bob may learn that either one or both of them were involved in a certain transaction.

Payment channels have been implemented in various ways; the Lightning Network[49] has been deployed in Bitcoin and Litecoin among others, whereas Beam introduced Laser Beams[1]. However, generally these solutions are not part of the core protocol of cryptocurrencies; still, they can be used to improve privacy.

[1] `https://documentation.beam.mw/en/latest/rtd_pages/laser.html`

	Coin mixing	Ring signatures	Stealth addresses	ZK-proofs	Confidential transactions	Mimble wimble	Lelantus	Payment Channels
Bitcoin								✓
Litecoin						✓		✓
Dash	✓							
PIVX				✓				
Decred	✓							✓
Monero		✓			✓			
Zcash(-forks)				✓				
Firo (Zcoin)							✓	
Beam						✓	✓	✓
Grin						✓		

Tab. A.1.: Privacy-preserving features per cryptocurrency.

A.1.11 Cryptocurrencies

There are cryptocurrency projects that implement features to improve the privacy and anonymity of their users. Some explicitly market themselves as 'privacy-coins', whereas others are more subtle. In table A.1 a summary is provided of cryptocurrencies that employ the features discussed above. The features that are enabled in these cryptocurrencies on a protocol level are checkmarked.

Bitcoin As the first cryptocurrency, Bitcoin was perceived as an anonymous cryptocurrency. However, as methods to deanonymize Bitcoins appeared, it should now be considered pseudonymous. It does not implement any privacy-preserving or anonymity enhancing features in its core protocol. Still, as second layer features coin mixing and using payment channels are available.

Litecoin Litecoin is very similar to Bitcoin in its features, although it recently introduced MimbleWimble, which is planned to be fully activated by the end of 2021. Moreover, payment channels are available in Litecoin.

Dash Dash is also a Bitcoin-like cryptocurrency, using a large part of Bitcoin's codebase. As further explained in Section 4.1, Dash shifted its focus from privacy to usability and speed. The privacy feature of Dash is called *PrivateSend* which is a CoinJoin[19] implementation. CoinJoin is a way to mix funds via transactions constructed together by multiple users. Da

PIVX PIVX was forked from the Dash code-base, but switched to PoS. Moreover, privacy protection was advanced by implementing zk-SNARKs based to support hiding transaction content, thereby mixing became obsolete.

Decred This cryptocurrency is partially PoW and partially PoS based, and has some built-in privacy features. Decred implemented CoinShuffle++ [51], which allows users to mix their coins within the system. Also, support for Lightning Network (payment channels) is available.

Monero Monero is one of the more widely known 'privacy-coins'. It combines ring signatures with confidential transactions to form their RingCT technology. Recently, Bulletproofs were also added to improve efficiency of RingCT. Monero transactions hide senders, receivers and amounts.

Zcash and its forks The other often mentioned 'privacy-coin' is Zcash, which uses zero-knowledge proofs (zk-SNARKS) to hide transaction contents. Zcash differentiates between private and transparent addresses. Normal transactions between transparent addresses are as public as Bitcoin transactions, whereas transaction between private addresses are shielded (hidden content).

Firo Zcoin recently rebranded to Firo and has Lelantus as its privacy-preserving feature. Simply put, users can join a huge anonymity set by burning the coins they want to send, after which the coins can be redeemed anonymously, potentially by a receiver. Firo also has support for network level anonymity through Dandelion++.

Beam Beam is another cryptocurrency in which all transactions are confidential. It is one of the two large MimbleWimble cryptocurrencies, although Beam managed to combine MimbleWimble and Lelantus. Furthermore, Beam also supports payment channels; called Laser Beam.

Grin The other large MimbleWimble implementation is Grin, which, through Mimblewimble, does not use sender or receiver addresses and hides transaction amounts.

A.2 Anonymity & privacy attacks on blockchain based cryptocurrencies

Several studies have been done that implement attacks to weaken the anonymity and privacy of cryptocurrency users, targeting different systems and employing various methods. This section will summarize the anonymity and privacy attacks aimed at blockchain-based cryptocurrencies. Again [8] and [7] will be taken as a starting point since excellent overviews are provided there. It should be noted that many anonymity studies have been done only on Bitcoin; however, they can also apply to other cryptocurrencies that are based on Bitcoin and/or function in a similar way. For example, these studies can often also be applied to currencies like Litecoin or Dash, where all addresses and amounts are public.

A.2.1 Heuristics and public address information

An early study on privacy of Bitcoin users was done in [31], in which heuristics are presented to recognize and link certain transactions. One of the heuristics states that input addresses that are used together in a transaction with multiple inputs are likely to belong to the same user and thus can be linked. The study combines this with other heuristics and behavior-based clustering to identify addresses belonging to the same user and profiling users, thereby decreasing their anonymity. The heuristic described above has been employed in various studies on privacy in Bitcoin (e.g. [79, 86]). It was also applied in [17], where it is combined with a heuristic to identify change addresses, and actively transacting with various entities to know which addresses they own. These studies aim to cluster Bitcoin addresses based on heuristics, and these heuristics are mostly based on so-called *idioms-of-use*, which may change over time. Behavior changes among Bitcoin users rendered one of the heuristics from [31] invalid, and as the use of multiple-input transactions with different users increases (such as through CoinJoin mixing), other heuristics will be affected as well. Another approach to deanonymizing transactions is presented in [44], where the authors scrape the BitcoinTalk forum to obtain addresses that are publicly posted, and are thus known to be most likely owned by the poster. This information is then used to link transactions to users. Moreover, a method is presented to track and deanonymize user activity with graph analysis. In [18] and [17] web scraping is applied as well, also including more possible sources (e.g. Twitter) of address-user links.

Interestingly, the lack of anonymity in Bitcoin can even be used to deanonymize other supposedly anonymous services. In [87], the authors were able to deanonymize

users of TOR hidden services by first deanonymizing their Bitcoin addresses (using web scraping) and then linking these addresses to the TOR hidden services.

The authors in [88] did an analysis on the adoption of several 'second-generation anonymization techniques', such as some decentralized mixing mechanisms and confidential transactions. They did this through analyzing the blockchain and recognizing specific types of transaction associated with these anonymization mechanisms. They found that the adoption of and support for these technologies was low. Moreover, they state further research into potential deanonymization avenues is necessary. They also note there is a lack of metrics for anonymity, which makes it hard to quantify the degree of anonymization of different techniques.

A.2.2 Network-level attacks

Other research to deanonymize transactions does not focus on the transaction graph but utilizes information from the P2P network that is used in a cryptocurrency. In [47], Bitcoin's P2P network is observed to obtain data on who relayed what transaction. On the basis of observed relay patterns some transactions (and thus addresses) can be linked to IP-addresses. It should be noted that most result were obtained from abnormal relay patterns, which suggests that regular transactions will be much harder to deanonymize with this method. Moreover, it is concluded that the heuristic of first relayer ownership, which says transactions are owned by their first (observed) relayer, is "ineffective at best and invalid at worst".

Another study building on the idea that one can deduce information from observing who relays what is [46], in which the authors manage to deanonymize Bitcoin clients using what they call *entry nodes*. These are the nodes that a Bitcoin clients connects to, and these will be the nodes that first receive a transaction issued by the client. *Entry nodes* also forward some information from a client to their peers when a client connects to them. This information can be used to fingerprint clients, and the resulting fingerprints can be used to deanonymize transactions that are observed in the network. The general idea is that clients are fingerprinted by the entry nodes they connect to, so when the attacker receives a transaction first from a specific set of nodes corresponding to a fingerprint, the attacker knows what client initiated the transaction. The methods provided in [46] also are able to circumvent clients hiding behind proxies or TOR. In [89] it is shown that even an update to the relaying protocol (from trickle spreading to diffusion spreading) did not fix these anonymity issues in the Bitcoin network.

Finally, in [90] the authors tried to combine transaction graph information with network level information, to see if network level information could improve address clustering. However, not much correlation between address clusters and network information was found. They conclude combining these methods does not seem to be fruitful thus far.

Again a very different approach to deanonymizing cryptocurrency users is presented in [45]; where cookies are used for deanonymization. The general idea of the proposed attack is that Bitcoin-accepting merchants place third-party web-trackers on their websites, which allows these web-trackers to obtain information about purchases via cookies. These cookies often contain information that allows the tracker to find blockchain transactions corresponding to certain purchases. Moreover, the cookies may also contain personally identifiable information (PII), which allows a web-tracker to link blockchain addresses and transactions to actual identities. This can be made even stronger by combining this with previously mentioned address clustering, which will allow the web-tracker to find other addresses from their target, from the addresses they obtained from cookies.

A.2.3 Non-Bitcoin attacks

Whereas most studies are focused on Bitcoin, also some research has been done on the anonymity and privacy provided across other cryptocurrencies. In [91] transactions were traced between different cryptocurrencies; a service that is offered by exchange services. This could be done using publicly available data from an exchange API, combined with blockchain data.

Other research has focused on Monero, in [92], a traceability analysis of Monero is presented. Three attacks are discussed, through which transactions can be linked. Some ways to address these attacks exist, although it is concluded perfect untraceability is not possible. Similar work was done in [93], in which Monero transactions are also successfully traced. Tracing transactions in Monero can be done because of the way mix-ins are selected for the ring signature, and because common user behavior could be exploited. The latter was based on users merging their funds, however merging has become obsolete since RingCT, and therefore tracing based on merging transactions will likely not be effective anymore. Still, because sometimes transactions use no mix-ins, or use only relatively old mix-ins, the actual transaction output can often be determined. Furthermore, in [94] and [95], attacks are presented that inflate the anonymity set. In these attacks, the attacker(s) try to obtain control of many of the available mix-ins, which they can then remove from transactions that include these mix-ins, thereby showing which

output is spent. Moreover, in [95] the Unencrypted Payment ID (UPID), which is an optional feature of a transaction, is utilized to decrease anonymity by linking transactions that have the same UPID.

Zcash has also been analyzed in a few studies. In [96] the topology of the underlying network is determined on the basis of timing analysis. Such information may be used to decrease the anonymity of nodes in the network. Moreover, the anonymity of Zcash is experimentally analyzed in [50]. It is shown that most Zcash transactions are not shielded, and thus privacy or anonymity of their owners is not protected, since outside of the shielded pool Zcash functions like Bitcoin. Furthermore, the authors are able to reduce the anonymity set of the shielded pool users by almost 70%. Also, they do a case study in which they exploit common user behavior to identify some potentially illicit transactions.

In [7] it is noted that although Mimblewimble provides anonymity from a blockchain-data perspective, cryptocurrencies employing this technology may still be vulnerable to transaction linking and deanonymization by a network observer. This would function like in Bitcoin, as discussed earlier this section [46, 89].

A.2.4 Linking attacks and privacy-preserving technologies

Previously, Section A.1 elaborated various methods to protect the privacy and anonymity of cryptocurrency users. On the other hand, Section A.2 thus far introduced various methods to attack the privacy and anonymity. Now, attacks will be linked to the technologies that aim to fix the vulnerabilities exploited in the attacks. Findings are summarized in A.2.

Attacks that employ the *'multiple inputs, same user'*-heuristic are mitigated by mixing technologies like CoinJoin; since these mixing technologies aim to invalidate this assumption. When also using confidential transactions, hiding transaction amounts, inputs and outputs in transactions can also not be linked based on amounts anymore; which was used in some tracing attacks. In other attacks, addresses and public information are combined to deanonymize users. This is solved by introducing methods to hide the addresses of the sender and receiver, which can be achieved by some forms of mixing, ring signatures, and stealth addresses. Zero-knowledge proofs, which can be used to hide amounts as well as addresses, can fix both heuristic based transaction deanonymization and privacy attacks that associate publicly announced addresses with transactions. Network-level attacks can be fixed using protocols such as Dandelion++, which changes propagation mechanisms in such a way that it becomes rather hard to find who the sender of a transaction is.

	Solutions		
Heuristics and address clustering	coin mixing	confidential transactions	zero-knowledge proofs
Utilizing public data	coin mixing	ring signatures	stealth addresses
Network-level attacks	Dandelion++	TOR/I2P	
Zcash topology attack	Dandelion++		
Mimblewimble observation	Dandelion++		

Tab. A.2.: Linking protection mechanisms to types of attacks.

None of the provided methods protects against cookie-based attacks, like presented in [45]. The attacks on Monero that were discussed are not solved by any of the introduced privacy-preserving technologies. The attack on Zcash presented in [96] may be prevented using a protocol like Dandelion++, which will hinder timing analysis, although this has not yet been implemented in practice. Using Dandelion++ means that knowledge of the topology of the network will be much less useful for deanonymization. Unshielded Zcash transactions face similar threats as Bitcoin transactions, and may be protected by measures that counter heuristics- and address-based clustering, as well as measures against attacks that utilize public data. Network observation attacks like suggested for Mimblewimble may similarly be made less feasible by introducing a unpredictable propagation mechanism like Dandelion++.

Mixing and mixers

B

This appendix elaborates our preliminary literature research on mixing and mixing services, as was done in preparation of the study on privacy-preservation in cryptocurrencies.

The idea of mixing messages to hide the sender of a particular message is quite old; well-known examples are mix-nets [97] and dc-nets[98]. This idea has also been applied to cryptocurrencies, in fact a plethora of mixing technologies have been introduced. Most of these are meant for use in Bitcoin and some for other cryptocurrencies. This section will first briefly introduce and categorize available mixing technologies, and second, discuss analyses of these technologies found in current literature. Mixing technologies have not often been surveyed or compared, although limited overviews do exist. This section contributes to filling this gap by gaining an overview of the state of the art of mixing in cryptocurrencies.

Mixing affects many of the transaction graph based deanonymization methods; it (theoretically) invalidates usually effective heuristics and assumptions about common behavior in transacting. For example, an often used heuristic is that input addresses from a multi-input transaction belong to the same user; however, many mixing protocols construct transactions where they combine transactions of multiple users, nullifying this heuristic.

A differentiation can be made between centralized and decentralized mixing technologies. Centralized mixers are services to which users connect and send their funds, after which the service sends back fresh coins with a fresh or unrelated history. In a decentralized protocol, users who want to mix somehow meet and participate in a protocol run where they together execute a mixing session, mixing the funds of the participants.

B.0.1 Mixing protocols

A plethora of protocols to do coin mixing has been introduced, most focus on Bitcoin, although some were developed for other cryptocurrencies. In this section, the protocols that have been discussed in literature will be introduced.

CoinJoin

One of the earliest transaction mixing ideas was introduced in [19]. In this forum post CoinJoin is introduced, which is a mixing mechanism that could be implemented in a decentralized as well as a slightly centralized version. The idea is that multiple cryptocurrency users meet via some (anonymous) platform (which is assumed to be in place) and then decide that they will construct a transaction together. The users choose on one common output amount and then all users will provide inputs, outputs of the decided amount, and potential change outputs. These are combined together in one transaction, which is subsequently signed by all participants of the transaction. As a result, a valid transaction has been constructed of which it will be hard to link the outputs to specific inputs. For the users who want to participate in a mixing session to find each other, as well as constructing the transaction, a central party could be used, although this is not strictly necessary. Commitments, blinding, and zero-knowledge systems could be used to also avoid that transaction participants (or the central party constructing the transaction) can link inputs to outputs.

This way of mixing transactions together invalidates the earlier discussed 'multiple-input same user' heuristic. It creates an anonymity set based on how many users participate in one mixing transaction, and how many times a user decides to mix their funds in a mixing transaction. Simply put, given some amount of coin (x) of a user (A), if the user mixes this with n unique other users in a total of l rounds then their anonymity set has size n^l.

CoinJoin is not a perfect solution, it is found to be vulnerable to DoS attacks, users can behave maliciously by not following the protocol. Moreover, in [75] it was shown that having CoinJoins with arbitrary amount, which did happen in practice, negates anonymity gains. It is concluded that output splitting could be used when arbitrary amounts are desired, although following CoinJoin's original design of common output values is more effective.

CoinJoin is applied in Bitcoin (e.g. Wasabi Wallet, Samourai Wallet, JoinMarket) and Dash (PrivateSend), but is not necessarily limited to these. It could be implemented in any currency that allows for aggregating and signing of multiple-user transactions.

Xim

Xim [99] is a decentralized two-party mixing protocol trying to improve existing solutions. In the study in which Xim is presented, several problems with CoinJoin and solutions alike are noted, mainly caused by the low costs of (Sybil-based) attacks.

Attackers can cheaply employ many Sybil identities, participate in mixing sessions without cost, and then disrupt the service by aborting early or compromising the anonymity of other participants. The Xim protocol provides resistance to Sybil attacks by requiring a fee for participation in a mixing session. This fee has to be paid before participating, such that attackers will need to pay for each mixing session they want to observe. To participate in a mixing session, a user has to make a transaction (which tips a set amount of coin to the miners) which either advertises a new mixing session or is confirming participation in some other user's session. Via such transactions, two users who want to mix will find each other and then mix their funds using the Fair Exchange protocol, which is introduced in [100]. Fair Exchange allows two mutually untrusting parties to exchange their coins in a fair way. Xim is compatible to Bitcoin as well as currencies alike, and it provides resistance against Sybil attacks, DoS attacks and some deanonymization attacks.

CoinShuffle

Another proposal which is based on CoinJoin, called CoinShuffle, was introduced in [40]. It works quite similar to CoinJoin, except it includes a method to construct the CoinJoin transaction in a decentralized manner while also holding non-cooperative or malicious participants accountable by excluding them. This solution is also Bitcoin compatible, although it can also be applied to other Bitcoin-like cryptocurrencies.

In [51] DiceMix is introduced, which is a Peer-To-Peer mixing protocol based on DC-nets, which is then also applied to improve CoinShuffle, resulting in CoinShuffle++. DiceMix makes CoinShuffle much more efficient and faster. Additionally, DiceMix can guarantee that participants who did not follow the protocol behaved maliciously, whereas in CoinShuffle malicious behavior could not be distinguished from accidentally not sticking to the protocol. CoinShuffle++ is Bitcoin-compatible and it is currently applied in Decred[1] and Bitcoin Cash[2].

ValueShuffle

ValueShuffle [101] is another improvement to the CoinShuffle protocol. By incorporating Confidential Transactions (CT) and Stealth Addresses (SA) privacy-preservation and usability of the CoinShuffle++ are improved. Because transaction values are hidden using CT, funds that are mixed do not anymore have to be sent to outputs of the same value, which was necessary for anonymity in other CoinJoin

[1] `https://github.com/decred/cspp`
[2] `https://cashshuffle.com/`

based protocols. Moreover, removing this restriction also allows mixing participants to directly send funds, via a mixing session, to another receiver and not first back to themselves, which also improves usability. Sending directly can be done with receiver-anonymity by using SA. ValueShuffle is compatible with Bitcoin only if Confidential Transactions and Stealth Addresses are implemented in Bitcoin, which thus far has not happened. Other cryptocurrencies that support these technologies and are otherwise UTXO-based may support ValueShuffle.

PathShuffle

In [102], a decentralized mixing mechanism is presented for credit networks, which have a different structure and goal compared to regular cryptocurrencies. Still, credit networks can be used to execute transactions between users, these transactions can be traced and deanonymized. In [102], CoinJoin, which is applied in cryptocurrencies, is adapted to PathJoin for credit networks, and it is combined with DiceMix to create PathShuffle. The protocol is compatible to Ripple, a large existing blockchain-based credit network, and can also be implemented in systems alike.

Mixcoin

Centralized mixes have been central to mixing in Bitcoin, especially in its early days. Users send some coins to such a service and will receive 'fresh' coins that (ideally) cannot be linked to the user's original coins. Of course, when sending funds to the service, a user faces risk of theft, as well as a risk of anonymity loss. A malicious service may keep the funds without sending anything back, or they might keep track of the links between inputs and outputs of the service. Even worse, if the service leaks these links, its users completely lose their anonymity. In [103], Mixcoin is proposed, which aims to hold centralized mixes accountable for their actions, thereby incentivizing honest behavior. Like in CoinJoin, Mixcoin mixes utilize set amounts, such that each transaction to and from the mixing service transfers (multiples of) that amount, which makes addresses to which the mix pays not distinguishable based on amount. However, mixing fees to pay the mixer for their service may hinder this, which is why randomized mixing fees are introduced which mean that a mix can incidentally keep the amount that is sent to them for mixing. This ability of the mixer is limited by employing randomness and accountability, such that it cannot serve as an excuse for the mixer to steal. Accountability of mixes in terms of mixing is guaranteed using signed warranties provided by the mixer. As a result, when a user can prove they received a warranty and they fulfilled their part of the warranty by sending funds to the mixer, the mixer can be held accountable if they

behave maliciously. The paper ([103]) also suggests sequential mixing with different mixes to avoid deanonymization by a single mix. Additionally, mixers and their users should always use fresh addresses for mixing which makes warranties easy to verify, and prevents an observer from distinguishing between different Mixcoin mixing services as well. Mixcoin is Bitcoin compatible, although similar services can be run on other cryptocurrencies

T-Mix

In [104], Mixcoin is improved in terms of availability and robustness against malicious mixers. In the proposed mixing protocol, T-Mix, threshold cryptography is used to for escrow addresses and warranties. As such, the funds of a user, and the warranty that is sent to a user, is never in control of a single mixer. This eliminates the single-point-of-failure issue in Mixcoin. T-Mix is very similar to Mixcoin, it is aimed at Bitcoin, but may also be implemented in similar cryptocurrencies.

Blindcoin

Blindcoin, introduced in [105], aims to improve Mixcoin by also hiding the input-output relations from the mixer. In Mixcoin, the central mixing service will still know the link between the address a user used to send them funds and the address to which they sent coins back. To break this link, Blindcoin uses blind signatures. When users take part in a mixing session, they give the mixer a blinded version of the output address at which they want to receive mixed funds, and this address is signed by the mixer. Later in the process the user then anonymously unblinds that output address, such that the mixer can verify that it indeed earlier signed this address and, if valid, transfer funds to it. A public log is used to enable external parties to verify honesty of both parties, and enable users to anonymously unblind output addresses.

Blind-Mixing

Another blind-signature based mixing scheme, which is very similar to Blindcoin, is suggested in [106]. This protocol improves on Blindcoin by not using a public log, whilst still avoiding linkage of input and output addresses by the mixing service. Moreover, they suggest that the mixing service should use CoinJoin for output transactions, which further improves users' anonymity.

Lockmix

Lockmix, introduced in [107], is another mixing protocol which improves on Blindcoin and Mixcoin. Where those mixing services do provide accountability, Lockmix also provides protection against theft by using multi-signature addresses. To mix funds, a deposit by the user is temporarily stored in a multi-sig address owned by the mixer and the user, and released to the mixer only after it has fulfilled its mixing duty. This does introduce a vulnerability to the mixer, who now may be victim to a malicious user who refuses to release the deposit in the multi-sig address, after it has received fresh funds from the victim. [107] states that this can be mitigated by setting the deposit appropriately. Lockmix requires multi-sig addresses which are usually available, therefore it is generally compatible with Bitcoin-like cryptocurrencies.

Obscuro

An efficient centralized mixing service that makes use of Trusted Execution Environments (TEE) is proposed in [108]. This solution aims to solve the theft threat of centralized mixers, as well as the limited scalability of decentralized services. The mixer's code is executed in a trusted environment, and remote attestation can be done to verify that the correct code is running. In fact, the TEE protects users from any deanonymization attacks by the mixing service owner, although DoS attacks are still possible. However, users are also protected from coin theft via a guaranteed refund. The mixer only receives users funds and then generates a large mixing transaction to all participants in a mixing session, and does not require other complicated cryptography or logic, which makes it easily implementable and compatible to many cryptocurrencies.

Tumblebit

Interestingly, coin mixing can also be done off-blockchain, as shown in [109]. TumbleBit is an anonymous payments hub that allows parties to make off-chain payments via payment channels in an anonymous fashion, even protecting their anonymity from a potentially malicious hub. In short, when a user Alice wants to make an anonymous payment via the tumbler T to Bob the process is as follows. Bob gets an RSA puzzle from T, to which only T knows the solution, and Bob blinds this puzzle and gives it to Alice. Alice solves this blinded puzzle through interaction with T, and obtains the solution from T for a set amount of coin. Then, Alice sends the solution of the blinded puzzle to Bob, who then unblinds it and can show the

solution at any point in the future to T. By showing that solution to T, Bob can claim a set a mount of coins from the Tumbler, in that way Alice paid Bob via the Tumbler. This system is Bitcoin compatible, and can also be implemented in other cryptocurrencies. The blockchain is, in this protocol, only used for setting up and closing payment channels.

CoinLayering

In [110], a somewhat centralized solution to coin mixing is proposed. The protocol, called CoinLayering, employs multiple mixes and a supervisor to implement mixing functionality. In short, a user who wants to mix coins makes a mixing request to the supervisor, who provides the user with a number of candidate mixes to use. The user then picks two mixes, sends funds to one mix and receives them back from the other mix, while the supervisor ensures mixes do not behave maliciously. Moreover, the supervisor makes sure that the mix who sent back funds to the user will also receive funds from the other mix. Several measures are taken to avoid theft or malicious behavior by the mixes and the supervisor. The system is compatible with Bitcoin and currencies alike.

CoinParty

CoinParty [48] is a mixing system that employs Secure Multi-party Computation (SMC) to combine advantages of centralized and decentralized mixes. The general idea is that a trusted third party is emulated by the participants through SMC. In four stages the participants mix their funds anonymously and securely. First, in the commitment phase the participants commit the funds that they want to mix to an escrow address. The escrow address is controlled by a threshold signature, where mixing participants hold shares of the address' private key. Second, the output addresses to which the funds will be transferred from the escrow address are shuffled in the address shuffling phase. Shuffling is done using decryption mixnets and explicitly verifying the correctness of the shuffling. Third, in the transaction phase transactions that transfer funds from the escrow addresses to the shuffled outputs are generated and need to be signed by a majority of the participants since threshold signatures are used. Finally, if anything goes wrong in the previous phases the fourth phase, error and reversion, is invoked in which the funds transferred to the escrow addresses are sent back to the input addresses. Moreover, malicious participants can be detected and held accountable. CoinParty is compatible with Bitcoin, and can also be implemented in cryptocurrencies alike. In [111], CoinParty

is reintroduced and improved; decentralization is increased, secure bootstrapping is included and DoS resistance is developed.

CoinMingle

In [112], all CoinJoin based mixing solutions, as well as existing ring signature and zero-knowledge mechanisms that are used to improve anonymity, are criticized and a new mixing scheme called CoinMingle is presented. The protocol employs ring signatures and Mutual Recognition Delegation Strategy (MRDS) to anonymously create mixing transactions. It is unclear whether this system can be readily implemented in existing cryptocurrencies.

Ethereum Mixing

Where most mixers discussed thus far have initially been designed for Bitcoin, some ideas for mixing on the Ethereum network have been presented as well.

A smart contract mixing solution called Möbius is presented in [113]. In this paper, an Ethereum variant of stealth addresses, namely stealth keys, are combined with ring signature to allow for coin mixing in a contract and subsequent anonymous withdrawing by a receiver.

Another solution is Miximus[114], which uses zk-SNARKS. This contract will accept payments, which can later be withdrawn anonymously. Withdrawal requires the withdrawer to prove (in zero-knowledge) that they payed earlier. This proof cannot be linked to the transaction that in which the withdrawer sent funds to the contract earlier.

MixEth[115] is a somewhat centralized mixing service proposal for Ethereum, which is based on a smart contract. Fist users deposit funds to the contract, after which the contract shuffles the public keys to which these funds should be sent. Shuffling can be verified by the participants, and after shuffling the owners of the shuffled public keys can each withdraw their funds.

Sharelock [116] is a mixing protocol is designed for Ethereum, but can also be ported to UTXO based cryptocurrencies like Bitcoin. The idea is that users send the funds they want to mix to a mixer, which is implemented in a smart contract. Then, after the mixer has received sufficient funds, it can be triggered to send out the funds to fresh participants' addresses.

Something similar is suggested in [117], which presents a contract-based mixing solution called Eth-Tumbler. This protocol uses layered encryption to hide output addresses to other participants of a mixing session. The protocol does require a common denomination and off-chain communication between the participants.

Another contract-based mixing solution is Tornado.Cash[118]. This contract employs zk-SNARKS, and users can deposit funds into the contract which they can then anonymously withdraw later. The protocol has recently also been deployed (in the form of a smart contract) on other blockchains.

B.0.2 Mixing service analyses

An early study of three Bitcoin anonymization services, which were mixing based, was done in [119]. They found input and output transactions of one of the services could be linked, whereas the other two successfully broke this link. The analyzed services were Bitcoin Fog, BitLaundry and SendShared, of which BitLaundry was found ineffective.

A nice overview of the available mixing services for Bitcoin is presented in [120]. Centralized mixing protocols that are discussed are Obscuro, Mixcoin, Blindcoin and TumbleBit; whereas the decentralized mixing protocols that are discussed are CoinJoin, CoinShuffle, CoinParty, and Xim. Moreover, the authors provide a list of 19 mixing services that they found using BitcoinTalk, and analyze some of their characteristics. Furthermore, in this research five of the mixing services (chosen based on reputation and popularity) are evaluated in terms of security and implementation. It was found that not many of academically proposed solutions are adopted in actual services, and many services perform poorly in terms of security.

[117] also discusses various mixing strategies and there problems. It is noted that peer-to-peer on-chain mixing (such as CoinJoin and CoinShuffle) is often vulnerable to DoS and Sybil attacks. Xim is mentioned as a potential solution to this, although Xim only allows mixing between two parties. Furthermore, decentralized on-chain mixing also usually requires several rounds of off-chain communication, causing large overhead in terms of time and complexity. Centralized solutions are an alternative but require a trusted intermediary, which also creates a single point of failure. The trust issue can be solved in solutions like TumbleBit.

A thorough study into existing mixing services is done in [68]. An effort is made to categorize and understand mixing service behavior, and four mixing services are chosen for further scrutiny. A method is developed to distinguish mixing transactions

generated by these services. The study also calculated profits of the discussed mixing services, and analyzed the flow of illicitly obtained coins through mixing services.

Finally, in [121], behavior of mixing service users is analyzed using game theory. It is concluded that depending on the information that users have, it may not be rational to take part in mixing, since there may also be money laundering happening in the mix. Further study of incentive to (not) use mixing services could be interesting, also for law enforcement to see how they can limit illicit money flows through mixers.

B.0.3 Conclusion

This section provided an overview of the state-of-the art of mixing technologies, although it still is not fully exhaustive. In general, it can be concluded that although many different protocols for mixing have been introduced, only few have been actually implemented. Moreover, many technologies are employed although it is often unclear what their (dis)advantagers are. Furthermore, many mixing mechanisms and services have not been thoroughly tested and scrutinized in literature, while the services that have been analyzed often show weaknesses. This suggests that further research into existing protocols and services is useful to understand which are the most robust proposals.

C Python code for visualization of Dash CoinJoin usage

C.1 Python-BlockSci script to gather CoinJoin data

Below is the Python scrip that utilizes BlockSci[12] to extract transaction data from the Dash blockchain and filter out the CoinJoin transactions. It requires installing the BlockSci library locally (setup documentation) and running a Dash core node to download the blockchain.

```
1  import blocksci
2  import collections
3  import csv
4
5  # This script requires installation of the BlockSci library.
   ↪ Moreover, it requires downloading the Dash blockchain, and
   ↪ generating a config to allow BlockSci to parse the blockchain.
6  # See the BlockSci documentation: https://citp.github.io/BlockSci/
7
8  # CoinJoin denominations are 10.0001, 1.0001, 0.100001, 0.0100001,
   ↪ 0.00100001
9
10 # Specify the blockchain to analyze by referring to the BlockSci
   ↪ config file generated for the Dash
11 chain = blocksci.Blockchain("dash.conf")
12
13 # Specify the block_interval. The blocks from the blockchain are
   ↪ analyzed per block_interval blocks, the transaction - and
   ↪ CoinJoin counts are aggregated per block_interval blocks.
14 block_interval=100
15
16 # Specify the starting block, if set to 0, the whole blockchain
   ↪ will be analyzed.
17 currentblock=0
```

```
18
19  # Retrieve the total number of blocks stored in the chain data.
20  totalblocks=chain.blocks.size
21
22
23  with open('cjdata_int100.csv', 'w', newline='') as file:  # Specify
    ↪  the CSV file to which the results should be written.
24          writer=csv.writer(file)
25          writer.writerow(["block_interval_start",
            ↪  "block_interval_end", "#transactions", "#coinjoins",
            ↪  "rate"])  # Specify column names.
26
27          # While block_interval blocks are still available, proceed
            ↪  to process the next batch of blocks.
28          while currentblock<=totalblocks-block_interval:
29
30          # Get all transactions in the blocks in the current block
            ↪  interval.
31                  transactions = chain.blocks.where(lambda b:
                    ↪  (b.height >= currentblock) & (b.height <
                    ↪  currentblock+block_interval)).txes
32
33          # Filter out all potential CoinJoin transactions in the
            ↪  blocks in the current block interval.
34                  coinjoincandidates = chain.blocks.where(lambda b:
                    ↪  (b.height >= currentblock) & (b.height <
                    ↪  currentblock+block_interval)).txes.where(
35                          lambda tx: (tx.fee == 0) & (tx.input_count
                            ↪  == tx.output_count) & (tx.input_count
                            ↪  >= 3) & (tx.inputs.all(
36                          lambda i: (i.value == 1000010000) |
                            ↪  (i.value == 100001000) | (i.value ==
                            ↪  10000100) | (
37                          i.value == 1000010) | (
38                          i.value == 100001))))
39
40          # Store the number of transactions and the number of
            ↪  CoinJoin transactions in the blocks in the current
            ↪  block interval.
```

```
        no_transactions=transactions.size
        no_coinjoincandidates=coinjoincandidates.size

        # Compute the rate of CoinJoin transactions over regular
        ↪  transactions.
        rate=(no_coinjoincandidates/no_transactions)

        # Write the results to a new line in the CSV file.
        writer.writerow([currentblock,currentblock+block_interval-1,
        ↪  no_transactions, no_coinjoincandidates,rate])

        # Print a progress message.
        if(currentblock % 10000 == 0):
        print("Progress: ", currentblock, " -- out of ",
        ↪  totalblocks, " blocks")

        # Increment the currentblock variable to move to the start
        ↪  of the next block interval.
        currentblock+=block_interval
```

C.2 Jupyter notebook code

The Jupyter notebook that was used for data processing and visualization is presented below. Some of the figures that are generated with this notebook have been presented in Section 5.4, and the others are presented in Appendix D.

[ ]:
```
# Jupyter notebook for visualizing and anlyzing Dash blockchain
↪  data, focussing on the adoption of CoinJoin.

# Import necessary dependencies
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statistics as sts
```

[ ]:
```
```

```
# Specify the columns in the data sheet and import the data.
columns = ["block_interval_start", "block_interval_end",
↪  "#transactions", "#coinjoins", "rate"]
cjc = pd.read_csv(r'C:\Users\arne_\Documents\CYBSEC 2018-2019
↪  Q4\TNO\CoinJoinData\cjdata_int100.csv', usecols=columns)
```

[ ]:
```
# Initialize seaborne, potentially with parameters, if desired.
# sns.set()
```

[ ]:
```
# Compress and process the data
# - Compute the number of transactions and CoinJoin transactions
↪  per block (the source data has aggregated transactions and
↪  CoinJoins per 100 blocks)
# - Aggregate 10 data points in 1 data point by computing the
↪  average of the number of transactions and average of number of
↪  CoinJoin transactions, and recomputing the rate.


# Create lists for intermediate storage during data processing.
int_start_c = []
int_end_c = []
no_trans_c = []
no_cjc_c = []
rate_c = []
trans_per_block = []
cj_per_block = []

cjc_trans_per_block = []
cjc_cj_per_block = []

# Initialize variables for storing aggregated values during
↪  compression.
int_start_acc = cjc.at[0, 'block_interval_start']
no_trans_acc=0
no_cjc_acc=0
rate_acc=0

# Iterate over all the rows.
for index, row in cjc.iterrows():
```

```
        # For each row, append average number of transactions and
        ↪  number of CoinJoins per block (instead of per 100).
        cjc_trans_per_block.append(row['#transactions']/100)
        cjc_cj_per_block.append(row['#coinjoins']/100)

        # For each row, add the current row's number
        no_trans_acc += row['#transactions']
        no_cjc_acc += row['#coinjoins']

        # Every 10 data rows, accumulate and store the aggregates
        ↪  (0-9, 10-19, etc.)
        if ( (index+1)%10 == 0):
                int_start_c.append(int_start_acc)

                int_end_c.append(row['block_interval_end'])

                no_trans_c.append(no_trans_acc)
                no_cjc_c.append(no_cjc_acc)
                rate_c.append(no_cjc_acc/no_trans_acc)

                trans_per_block.append(no_trans_acc/1000)
                cj_per_block.append(no_cjc_acc/1000)

                no_trans_acc=0
                no_cjc_acc=0
                rate_acc=0

        if ( (index!=0) and (index%10==0) ):
                int_start_acc = row['block_interval_start']

# Add the number of transactions per block and the number of
↪  CoinJoins per block to the original data frame.
cjc['#trans_per_block']=cjc_trans_per_block
cjc['#cj_per_block']=cjc_cj_per_block

# Specify the columns for the compressed data, and create a data
↪  object containing the data columns and their values.
```

```
columns_compr = ["block_interval_start", "block_interval_end",
↪  "#transactions", "#coinjoins", "rate","#trans_per_block",
↪  "#cj_per_block"]
data = {'block_interval_start':int_start_c,
↪  'block_interval_end':int_end_c, '#transactions':no_trans_c,
↪  '#coinjoins':no_cjc_c, 'rate':rate_c,
↪  '#trans_per_block':trans_per_block,
↪  '#cj_per_block':cj_per_block}

# Create a pandas dataframe from the data and columns.
cjc_compr = pd.DataFrame(data, columns=columns_compr)
```

[ ]:
```
# Set seaborne style
sns.set_style("ticks")

# Plot the number of CoinJoin transactions per block for all block
↪  intervals (0-15123000)
compr_p_b = sns.relplot(data=cjc_compr, x="block_interval_start",
↪  y="#cj_per_block", height=5, aspect=3)
compr_p_b.set(xlabel='Block height of block interval',
↪  ylabel='Number of CoinJoin transactions per block')
```

[ ]:
```
# Filter out only the recent transactions, from the regular
↪  dataframe (not compressed)
cjc_recent = cjc[cjc['block_interval_start'] >= 1006000]

# Plot number of CoinJoin transactions in recent blocks (height >
↪  1006000)
sns.set_style("ticks")
fig=plt.gcf()
fig.set_size_inches(20,8)
recent_cj_p_b = sns.regplot(data=cjc_recent,
↪  x="block_interval_start", y='#cj_per_block',order=2,
↪  line_kws={'color':'red'})
recent_cj_p_b.set(xlabel='Block height of block interval',
↪  ylabel='Number of CoinJoin transactions per block')
sns.despine()
```

```
# Compute and print the mean and standard devaiation of the data
↪  that is plotted.
print("Mean: ", sts.mean(cjc_recent['#cj_per_block']))
print("Standard deviation: ",
↪  sts.stdev(cjc_recent['#cj_per_block']))
print()
```

[ ]:
```
# Plot the ratio of CoinJoin transactions over regular transactions
↪  for each 1000 block interval
sns.set_style("ticks")
compr_rate_pb = sns.relplot(data=cjc_compr,
↪  x="block_interval_start", y="rate", height=5, aspect=3)
compr_rate_pb.set(xlabel='Block height of block interval',
↪  ylabel='CoinJoins to total transactions ratio')

# Plot the ratio of CoinJoin transactions over regular transactions
↪  for each 100 block interval (10x datapoints compared to
↪  previous graph)
rate_pb = sns.relplot(data=cjc, x="block_interval_start", y="rate",
↪  height=5, aspect=3)
rate_pb.set(xlabel='Block height of block interval',
↪  ylabel='CoinJoins to total transactions ratio')
```

[ ]:
```
# From here onward we create more visualizations which are not used
↪  or referred to in our research, but they may provide more
↪  insight in the adoption of CoinJoin in Dash.
# First we visualize the average number of transactions per block,
↪  to which we also apply outlier-filtering to improve the
↪  visualization.
# Moreover, we take a closer look at the recent developments both
↪  using the original dataset (values per block interval of 100
↪  blocks) and the compressed dataset (values
# aggregated per block interval of 1000 blocks).

# Plot the average number of transactions per block for each
↪  100-block interval.
notrans = sns.relplot(data=cjc, x="block_interval_start",
↪  y=cjc['#trans_per_block'], height=5, aspect=3)
```

```
notrans.set(xlabel='Block height of block interval',
↪  ylabel='Average number of transactions per block')

# Plot the average number of transactions per block for each
↪  1000-block interval.
notrans_comp = sns.relplot(data=cjc_compr,
↪  x="block_interval_start", y=cjc_compr['#trans_per_block'],
↪  height=5, aspect=3)
notrans_comp.set(xlabel='Block height of block interval',
↪  ylabel='Average number of transactions per block')
```

[ ]:
```
# As we can see in the graphs from the previous output, there are
↪  some outliers, where blocks have huge numbers of transactions.
↪  \
# To see the general development over time without losing too much
↪  information, we filter the outliers. Most blocks average below
↪  250 transactions per block.
# Therefore, we filter all datapoints where blocks have more than
↪  250 transactions on average. We plot the resulting data again.

trans_per_block_filtered = cjc[cjc['#trans_per_block'] < 250]
trans_per_block_filtered_compr =
↪  cjc_compr[cjc_compr['#trans_per_block'] < 250]

notrans_f = sns.relplot(data=cjc, x="block_interval_start",
↪  y=trans_per_block_filtered['#trans_per_block'], height=5,
↪  aspect=3)
notrans_f.set(xlabel='Block height of block interval',
↪  ylabel='Average number of transactions per block')

notrans_f_comp = sns.relplot(data=cjc_compr,
↪  x="block_interval_start",
↪  y=trans_per_block_filtered_compr['#trans_per_block'], height=5,
↪  aspect=3)
notrans_f_comp.set(xlabel='Block height of block interval',
↪  ylabel='Average number of transactions per block')
```

[ ]:
```
# We filter only the recent datapoints, where the block height is
↪  larger than 1000000, to get insight in the recent
↪  developments.
onlyrecent =  cjc[cjc['block_interval_start'] >= 1000000 ]
# We also create a dataset where the outliers (for number of
↪  transactions per block) are filtered.
onlyrecent_nooutliers = onlyrecent[onlyrecent['#trans_per_block'] <
↪  250]

# Subsequently we plot the recent developments in the number of
↪  transactions per block, the rate of CoinJoin transaction to
↪  total number of transactions,
# and the number of CoinJoin transactions per block.
notrans_f_recent = sns.relplot(data=cjc, x="block_interval_start",
↪  y=onlyrecent_nooutliers['#trans_per_block'], height=5,
↪  aspect=3)
notrans_f_recent.set(xlabel='Block height of block interval',
↪  ylabel='Average number of transactions per block')

rate_f_recent = sns.relplot(data=cjc, x="block_interval_start",
↪  y=onlyrecent_nooutliers['rate'], height=5, aspect=3)
rate_f_recent.set(xlabel='Block height of block interval',
↪  ylabel='CoinJoins to total transactions ratio')

nocjs_f_recent = sns.relplot(data=cjc, x="block_interval_start",
↪  y=onlyrecent_nooutliers['#cj_per_block'], height=5, aspect=3)
nocjs_f_recent.set(xlabel='Block height of block interval',
↪  ylabel='Numer of CoinJoin transactions per block')
```

[ ]:
```
# We do the same as in the previous section, only now we use
↪  compressed data (block intervals of 1000 blocks instead of 100
↪  blocks), thus 10x less datapoints.

# We filter only the recent datapoints, where the block height is
↪  larger than 1000000, to get insight in the recent
↪  developments.
cjc_compr_recent = cjc_compr[cjc_compr['block_interval_start'] >=
↪  1000000]
```

```
# We also create a dataset where the outliers (for number of
↪  transactions per block) are filtered.
cjc_compr_recent_nooutliers =
↪  cjc_compr_recent[cjc_compr_recent['#trans_per_block'] < 250]

# Subsequently we plot the recent developments in the number of
↪  transactions per block, the rate of CoinJoin transaction to
↪  total number of transactions, and the number of CoinJoin
↪  transactions per block.
notrans_comp_f_recent = sns.relplot(data=cjc,
↪  x="block_interval_start",
↪  y=cjc_compr_recent_nooutliers['#trans_per_block'], height=5,
↪  aspect=3)
notrans_comp_f_recent.set(xlabel='Block height of block interval',
↪  ylabel='Average number of transactions per block')

rate_comp_f_recent = sns.relplot(data=cjc,
↪  x="block_interval_start",
↪  y=cjc_compr_recent_nooutliers['rate'], height=5, aspect=3)
rate_comp_f_recent.set(xlabel='Block height of block interval',
↪  ylabel='CoinJoins to total transactions ratio')

nocjs_comp_f_recent = sns.relplot(data=cjc,
↪  x="block_interval_start",
↪  y=cjc_compr_recent_nooutliers['#cj_per_block'], height=5,
↪  aspect=3)
nocjs_comp_f_recent.set(xlabel='Block height of block interval',
↪  ylabel='Numer of CoinJoin transactions per block')
```

D

Dash CoinJoin analysis results

This appendix lists the other figures that are generated with the Jupyter notebook that is presented in Section C.2

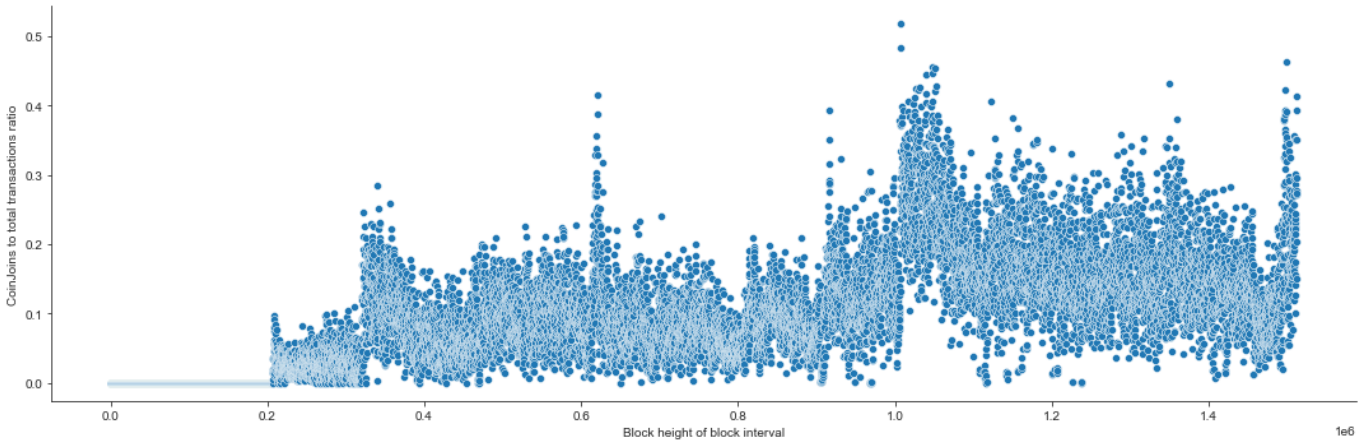

Fig. D.1.: The ratio of CoinJoin transactions to all transactions (non-compressed data, 100-block intervals).

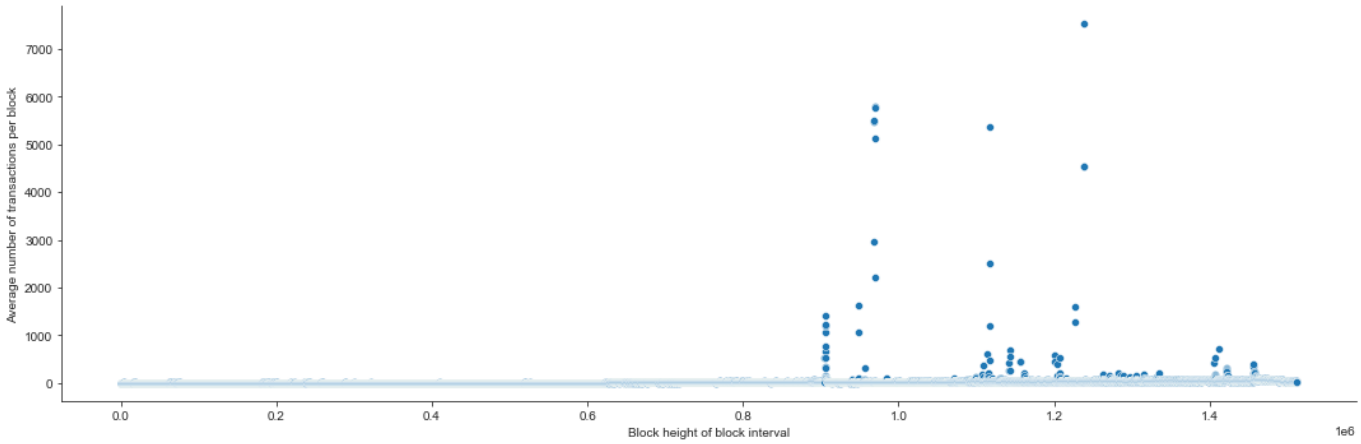

Fig. D.2.: The average total number of transactions per block for all block intervals (non-compressed data, 100-block intervals).

Other than these, we also took a look at the recent developments regarding the average number of transactions per block, the rate of CoinJoin transactions to the

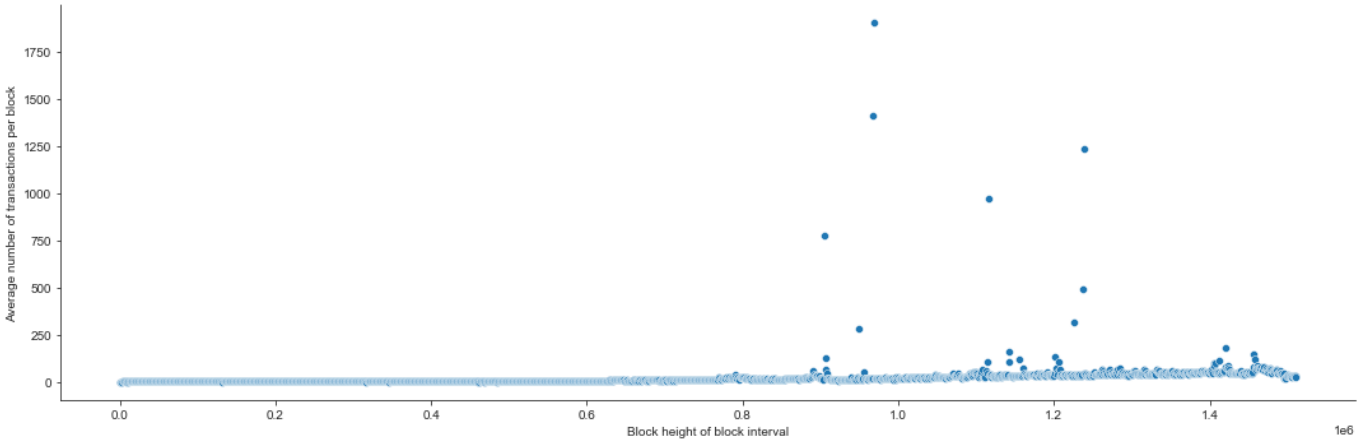

Fig. D.3.: The average total number of transactions per block for all block intervals (compressed data, 1000-block intervals).

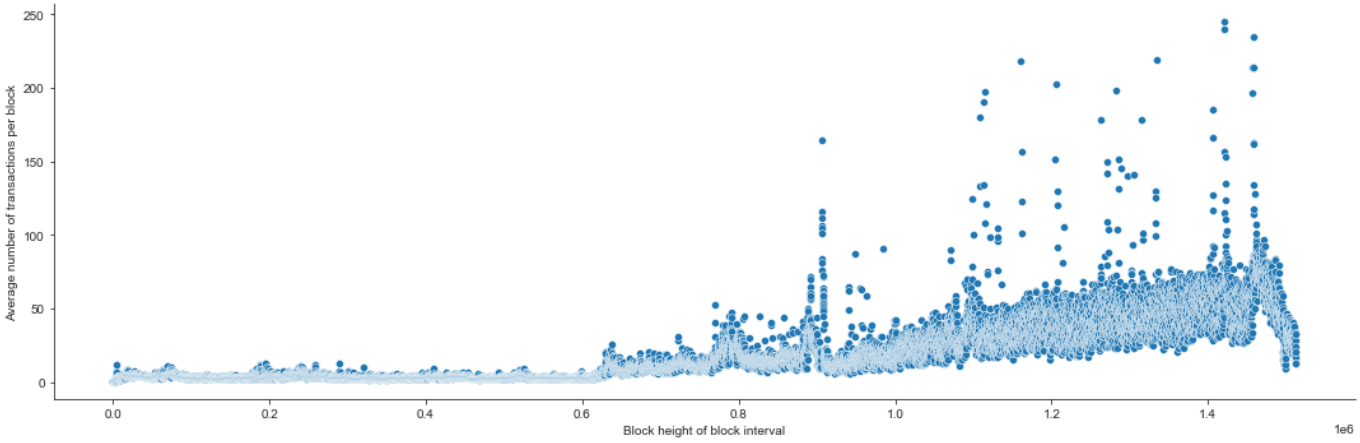

Fig. D.4.: The average total number of transactions per block for all block intervals (filtered outliers, non-compressed data, 100-block intervals).

total number of transactions, and the average number of CoinJoin transactions per block. The Jupyter notebook contains code for both the graphs with non-compressed data with 100-block intervals and graphs from compressed data with 1000-block intervals. We only show the figures from compressed data here, they are easier to read and except for more noise the non-compressed data figures do not provide much more information.

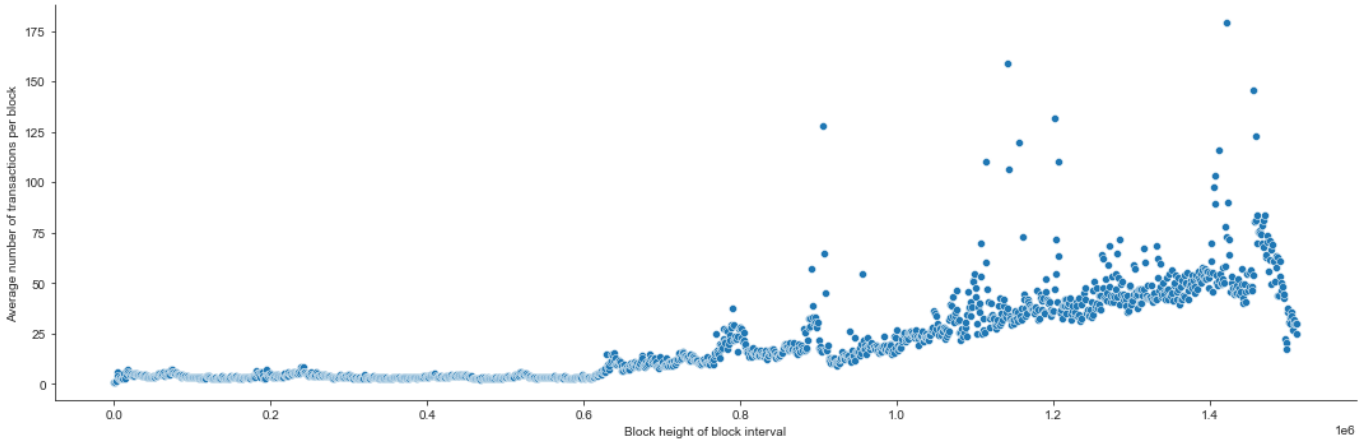

Fig. D.5.: The average total number of transactions per block for all block intervals (filtered outliers, compressed data, 1000-block intervals).

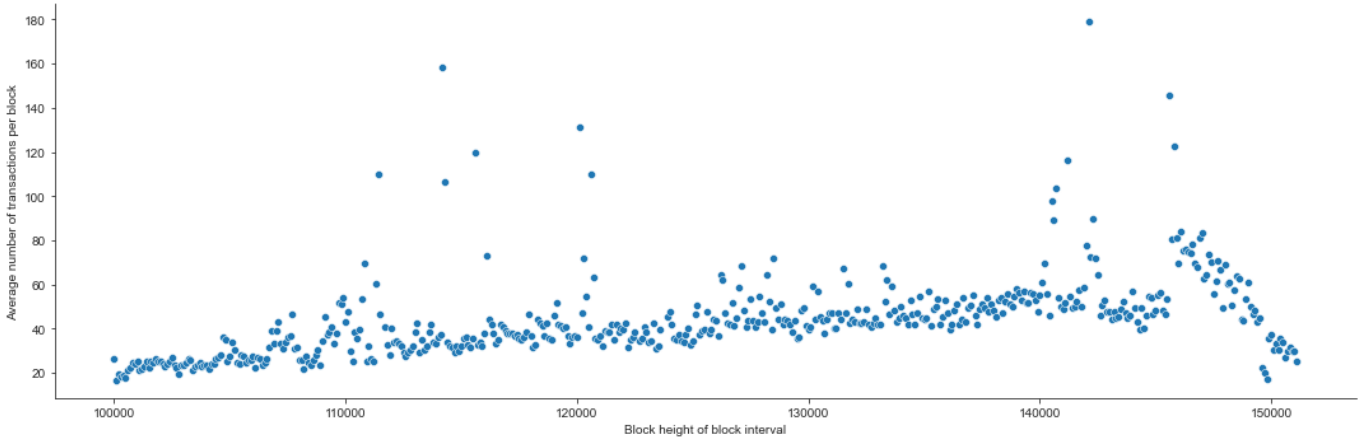

Fig. D.6.: The average total number of transactions per block for recent block intervals (filtered outliers, compressed data, 1000-block intervals).

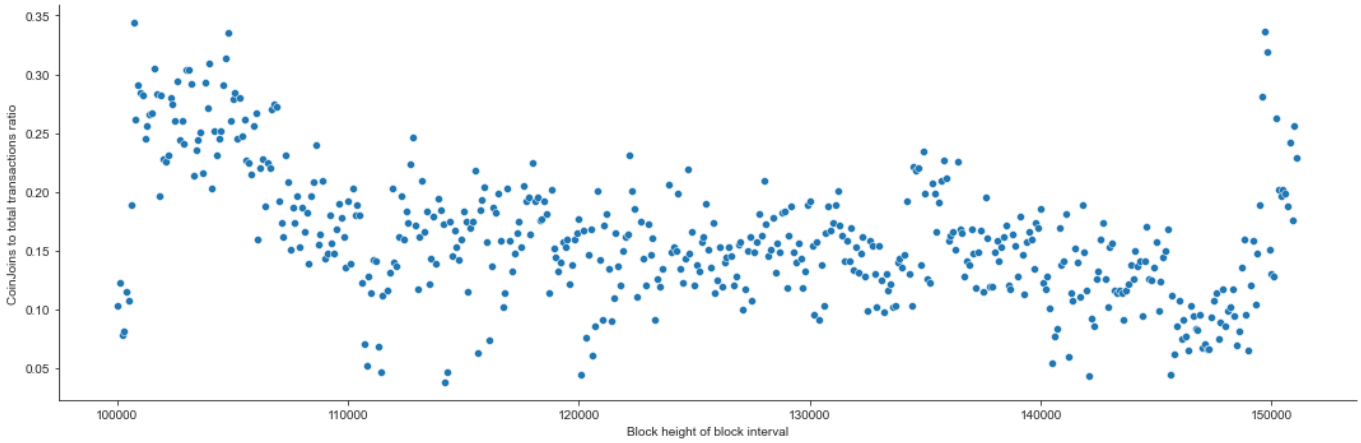

Fig. D.7.: The rate of CoinJoin transactions to the total number of transactions for recent block intervals (filtered outliers, compressed data, 1000-block intervals).

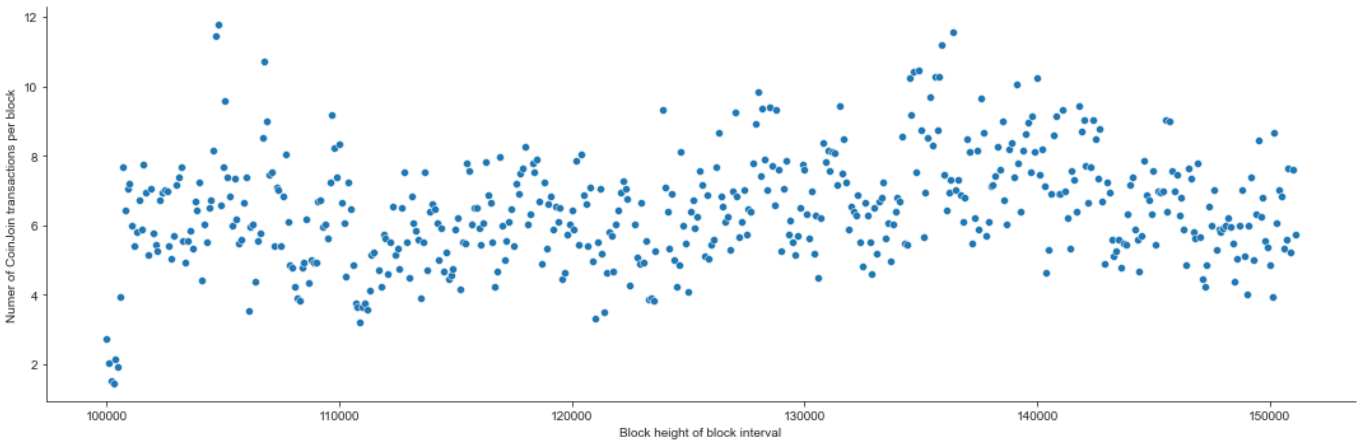

Fig. D.8.: The average total number of CoinJoin transactions per block for recent block intervals (filtered outliers, compressed data, 1000-block intervals).

Framework evaluation results

E

E.1 Thijmen Verburgh

1. *To what extent does applying this framework provide an accurate estimation of the privacy-preservation level of a cryptocurrency?*

This framework provides an accurate estimation of the privacy-preservation level of a cryptocurrency. The main components which enhance privacy (or reduce) privacy on the information level are described (sender, receiver and transaction 'information' that is being compromised or not).

Some considerations will be provided below.

2. *Are there any technical or non-technical aspects of privacy-preservation that are not covered by the statements used in this framework? If so, which aspects were missed?*

The focus of the current framework is mainly on the information from sender receiver and transaction. This allows many technical and non-technical components to be shared within the statements.

E.g. custodial mixers can be used within S13 to support whether transactions can be linked or not. All arguments whether this does or does not impact certain statements can be shared under this statement. However, scoring might become more complex if you also have to take into account the legality and availability of such services and options.

Maybe you can include the decentralization/globality/(management) network/open character aspects in the statements. Currently you have included this in the adversary model which is logical given the current currencies but might change in the future. Would a part-decentralised be an option for a currency?

Small sidestep to TOR. Anyone can contribute to the network by becoming nodes and even become HSDIR nodes. However, there are 9 directory nodes controlled by different organisations keeping track of the network state. So everyone can host a site, browse and be part of the network. However not all parts of the network is publicly available.

3. *How well can this framework provide meaningful comparability across cryptocurrencies in terms of privacy-preservation?*

The main benefit of the framework is that it does not provide a yes/no or good bad score. It provides an indication and relatively easy comparison of cryptocurrencies in a quick glance. It also does not limit itself on the technical methods used to achieve this in the framework but allows each statement to be supported with this.

The complexity of the 'worth/load' of each statement and its effects on each needs further research to provide a better comparability (also described in the chapter).

4. *How well will this framework be applicable in the future, with new developments in the field of privacy-preserving cryptocurrencies? Could this framework be used 20 years from now?*

The basis of the framework will be still applicable. Revisions and extensions are required. As mentioned in future, it is worthwhile to study the different 'load' of each statements and their interdependability.

5. *Other than the improvements discussed in the chapter; are there any improvements or approaches that were missed in this chapter but could provide additional value to privacy-preservation evaluation of cryptocurrency systems?*

I'm currently writing a piece with Ruggero and the others in which we classify/split the privacy features into the following: protocol level features (taproot), functional level features (e.g. mixers) and network features (e.g. vpn use ect).

In the piece we focus on the features instead of the coins. It might be interesting to try to map the underlying privacy features to the statements.

And: **6.** *Other than the points addressed by the previous questions, do you have any other issues or feedback you would like to share with us?*

How to deal with theoretical possibilities and the practical world. RingCT is obfuscation and therefore it will 'always' be possible to link transactions. However, would you still classify S13 as 1 in the current status? And what if triptych is applied? and what is triptych on steroids is implemented.

I would also include somewhere the dynamic world and that coin change over time. Same as adversarial methods. (might already be part of the chapter but then I missed this).

You claim that ciphertrace deanonimise but, disregarding the Osint additions they provide, I do not believe that what they do cannot be considered this. They iden-

tify entities which might be linked to identities by adding different sources (prior knowledge).

Can you elaborate/specify on the adversary used in your scoring? You refer to 2.2.1 as a starting point but do not specify aspects such as prior knowledge and resources.

Does this also entail the data requests to e.g. exchanges, ISP ect for data in the past, present and future. Did you consider legal and transnational complexity in your scoring?

(in know I'm a pain in the ass at this point but as you state in 2.2.1 the prior knowledge and resources need to be clearly defined I was looking for this)

You compared it with Fiat bank transfers. Another common comparison is to compare it with cash flows. How would cash be considered regarding the 15 statements?

It might be worthwhile to split the adversaries in different types. E.g. neighbor vs state. This might be relevant for policy take-aways part of the thesis.

7. *Can you briefly describe your past experience and expertise regarding privacy-preserving cryptocurrencies?*

As a social scientist I have studied the use of cryptocurrencies by criminals.

8. *We would like to use your review in our work; do you give permission to use your answers and mention your name?*

Yes, although as I typed this relatively quickly I would not use direct quotes ;).

E.2 Bart Marinissen

Sadly, I consider this a bad chapter. I feel the fundamental idea is flawed. I wish I had seen this work earlier, then I could have given this feedback earlier, and maybe this could have been prevented. Please know that I do not enjoy being this harsh. But if I were on the receiving side, I would want a frank assesement. So that is what I have given here.

My reasons why are largely reflected in the answers to your 8 questions. But I feel like I owe more of an explanation. Hence the wall of text.

My core issue is with two aspects:

- Reducing the results to a single score.

- Fixing the threat model rather than considering the range of attacker capabilities.

The first point is an issue because which cryptocurrency is more private depends on the scenario. For example, you can just as easily argue that Monero is more private that Zcash as you could argue the opposite. Which is correct depends on the specific scenario under consideration. The second point is an issue because the answers to the questions in the framework from a fixed threat model are rather uninteresting. What is interesting is how the answers vary as an attacker's capabilities change. This allows anyone evaluating a currency to identify what threats to privacy exist for a given currency, and how realistic those threats are.

The above two points are why I state the method is fundamentally flawed. If I were to suggest changes, they'd be rather significant. They might also be obvious based on my two core points above:

- Drop the single score. Instead score each individual question, and let someone who knows about the relevant scenario draw their conclusions from those scores.
- Consider, for each question, at what attacker capability the scoring would change. This is really useful for showing potential weaknesses and applicability of the coin in various scenarios.
- Improve the questions to focus less on implementation details, and cast a wider net. Including, developer disposition, best-practices and common practices, and the size of anonymity sets.

With those, rather significant changes, I think one could develop a framework that helps people actually evaluate how much privacy a coin gives in their context.

There are more textual notes in the annotated PDF attached. And more in-depth notes in the answers to the question. On a more positive note, the actual writing in the chapter is generally good. I only found 2 or 3 actual textual issues.

Answers to the questions:

1. This framework provides a very inaccurate estimation of the privacy-preservation level of a cryptocurrency. That is because there are many variables that matter that are fully left out of this system.
2. There are many aspects not covered by the statements in this framework. These include:

a. The size of anonymity sets

b. The ease of creating and maintaining multiple pseudonyms, and keeping those separate

c. The effect of behavior by the user base (consider that t-to-t Zcash transactions do not have mixers, so are worse than bitcoin)

d. The relation of guarantees to various attacker capabilities (I know the analysis is limited to a single attacker model, but that is a wrong choice)

e. The effect of available tools, guides, and best practices. Consider bitcoin without wasabi-wallet, with no qualms against address reuse, and without KYC exchanges

f. The responsiveness of the development team to newly found privacy issues / the degree to which the development team prioritizes privacy issues.

3. The framework, as presented with producing a single scoring system, and with only a single threat model, is not useful for comparing cryptocurrencies. That said, if the questions were expanded; the single score were dropped; and one would consider which attack capabilities change the answer to which questions, then the framework might give an interesting basis for comparing cryptocurrencies.

4. The questions, at this moment, are focused largely on the bitcoin model of transactions. I suspect there will be new classes of attacks, and classes of cryptocurrencies that are not covered by the current framework. These classes of attacks, could for example, use multiple input sources and limited extra information to draw strong statistical inferences.

5. I think a framework like this should drop the single-score approach. It is far to rigid, because technology A could be much more private than B under a given threat-model and common usage mode, whilst under a different threat model and common usage mode, technology B could be much more private. There is no way to distill down this complexity. Instead, you should aim for a framework that reveals under which conditions, which aspects of privacy a given cryptocurrency is better at. This could then be used on a case-by-case basis, or based on argued expert opinion, to come to conclusions.

6. I believe the approach taken here is fundamentally flawed. The two biggest flaws are trying to distill the scenario down to a single score, and having

a fixed thread model. Instead, each 'privacy requirement' should be scored separately. That score should not be a single score, but should be given over a range of attacker models.

Based on such a set of scores, people who would actually want to compare cryptocurrencies for a given purpose could make an informed decision. If that needs to be boiled down for decision-makers. An argued conclusion by an expert based on such a framework has a change to add value. A single score that originates from an unweighted average does not have such a chance, unless the expert tweaks the answers to give the desired answer, but that is not the intention.

In general, I wonder what the intended use of this framework is. What use-cases are there where a single score is need to evaluate privacy, when a more nuanced take is not also required? You reference [1] in the introduction, but never actually explain the value.

7. I have worked for 3 years, working a combined 1000 hours, on projects for the Public Prosecution researching privacy coins.

8. You have permission to use my answers, and mention my name. This permission is only if all of my answers are used because I do not want to come across as endorsing this scoring method.

Some further comments were provided in the document that I provided to the reviewer, these comments are addressed and processed in the discussion in Section 3.4.

E.3 Anonymous cryptocurrency developer

In further communication with this reviewer it was decided that their answers can be used but their name and background will be left out to respect their privacy.

I've read the chapter with great interest though, and will give you my personal impression now, hoping it's not too late. Overall I think that this framework provides a good starting point for the evaluation. Of course this is a very broad topic, and there would be many nuances to consider in order to make it a meaningful comparison tool.

> Q1. *To what extent does applying this framework provide an accurate estimation of the privacy-preservation level of a cryptocurrency?*

I think that we should have some sort of agreement on what the privacy-preservation metric ought to be first, in order to define a “level” (two technologies might achieve the same goal in different ways, making different engineering trade offs).

Intuitively we want to measure and compare the effectiveness of different technologies in achieving a specific goal (protecting user private data). Therefore a qualitative approach of this type (a statement is either not applicable, or partially/fully applicable) might often lack accuracy: two cryptocurrencies, for example, could both score 0 at S1, but there is no way to compare the security guarantees of the two systems (which would require, for this specific example, to evaluate *how strongly* the sender identity is kept hidden by each protocol, or, similarly, how each system performs with different adversary models, etc. . .) with the current tools.

Another limit comes, in my opinion, from the narrow "point of view" for the specific goal.

We are not considering how easily can the average user perform the steps required in order to keep his information private (thus, ultimately, how effective will be the particular privacy-preserving technology used).

If we look only from the point of view of the protocol design (and the Core reference client implementation), we will always get a incomplete answer to the question "how well does this cryptocurrency protect its users financial data?".

In order to get a more realistic answer, we must consider also an UX point of view (how the average user behaves), including the whole environment, with most used third-party services and tools (for example, a minimum percentage of Bitcoin users actually operates a Bitcoin Core wallet).

A cryptocurrency might, for example, offer a very strong protection "on paper", but a user could still easily use it in a wrong way, compromising its privacy. Let’s imagine a ZCash transaction t->z, with a single note created, and then spent few blocks later in a z->t with a single input. The two transactions could be, then, easily linked together, and the amounts, albeit hidden, can be easily deducted from the transparent sides.

This is mainly an implementation/UI issue. But, on the other hand, it could also be seen as an environmental issue: if all exchanges/fiat onramps allowed shielded addresses for deposits/withdrawals (and if all miners created blocks with shielded coinbases, which is possible since ZIP-213) ZCash could slowly deprecate the usage

of transparent addresses, reducing opportunities for the user to make privacy-compromising mistakes (meanwhile growing the anonymity pool, which provides better privacy preservation).

At the moment, though, this is in stark contrast with cryptocurrencies, such as Monero, where the privacy is "by-default", therefore it's easier for the average user to interact with the blockchain in a proper privacy-preserving way.

Focusing on the goal of the user, who tries to make a privacy-preserving transaction (possibly identifying his knowledge level), helps to reduce the ambiguity of some of the statements.

For example, when evaluating ZCash, it's not clear whether we should take the average of the four types of txes (t->t, t->z, z->t, z->z), or weigh the scores based on their actual usage on the network. But a ZCash user who wants to privately send a transaction follows a specific pattern (he doesn't randomly pick a transaction type). We might infer that a reduced anonymity pool contributes to a weaker overall privacy (but this is a separate aspect, more related to the first consideration).

> **Q2.** *Are there any technical or non-technical aspects of privacy-preservation that are not covered by the statements used in this framework? If so, which aspects were missed?*

Aside from the considerations above, regarding the usability, I think that the statements of this framework covers all the important aspects.

> **Q3.** *How well can this framework provide meaningful comparability across cryptocurrencies in terms of privacy-preservation?*

Due to what stated in reply to Q1, a comparison between cryptocurrencies might result distored in some case.

Let's pick, for example, the comparison of Bitcoin (37%) and Dash (40%).

Essentially, the only reason why Dash scores slightly better than Bitcoin is because it has (a form of) CoinJoin implemented directly in the RPC/GUI interfaces of the Core wallet, thus it performs better for the statement S15.

When we look at this from the user point of view, it offers very little advantage, and if consider the whole environment (its size, and maturity), Bitcoin offers far better privacy-preserving alternatives.

In Dash's implementation, a selected quorum of masternodes constructs CoinJoin transactions (and thus is trusted with the mixing privacy). A powerful attacker (e.g. law enforcement agency) could easily gain control of an high percentage of these

nodes (which are already controlled by a restricted number of individuals and are all hosted by a handful of VPS providers) and be able to link mixed transactions (even after multiple mixing rounds).

By contrast, with Bitcoin, there's wallets such as Samourai and Wasabi implementing a more secure coinjoin version (ZeroLink [1]) where the mixer is unable to link sender and receiver addresses (both these 3rd-party solutions are fully open-source and can be run in trustless way, alongside a Bitcoin full node).

So, for S15, given this example, the fact that the functionality is directly built in the Core wallet implementation (score 0), rather than a third party (score 1) does not necessarily translate into better privacy-preservation.

As a whole ecosystem, Bitcoin arguably performs better than Dash in this regard.

More generally it seems that the statement S15 itself leaves much to the interpretation (it is not clear how well this "engineering" of transactions contributes to the goal).

[1] https://github.com/nopara73/ZeroLink/

> Q4. *How well will this framework be applicable in the future, with new developments in the field of privacy-preserving cryptocurrencies? Could this framework be used 20 years from now?*

Given how much the space has evolved in its first 12 (or so) years of existence, it's very difficult to evaluate the durability of this framework.

While the statements are sufficiently general to be applied to a broad spectrum of present (and possibly future) solutions, what might likely change is how this "average user" will interact with blockchains: for example we might see a growth in the usage of L2 protocols, such as The Lightning Network for Bitcoin, which would require completely new statements to evaluate the privacy-preservation aspects (see https://arxiv.org/abs/2003.12470).

—-

For the last three questions I'd prefer to answer "no"...

I don't know if there is even still time for that, but in any case I'd prefer not to be mentioned. I just wanted to share my personal feedback, and some food for thought, hoping it can be useful to you in some way. Thanks for considering my opinion.

Bottom line, great job so far :) I would be glad to read the other chapters as well, if you'd like to share them. Best wishes for the continuation of your work and your studies.

F Dash CoinJoin queue gaming implementation

This section will outline the work that we did to implement the Dash Coinjoin queue gaming (DCQG) attack, which we introduced in Chapter 7. First we will specify the test environment that we set up. Second, we elaborate the steps we took to simulate the attack. Third, we discuss the remaining issues that need to be solved.

F.1 Local Dash test environment

To simulate and test the attack without disturbing the Dash networks (which might raise some ethical issues), we decided to set up a local Dash network, and implement a proof of concept of the attack within the local network. The Dash Core group, the developers behind Dash, have also implemented a tool called *dashmate*[1]. This tool can be used to set up (local) test networks of Dash masternodes. This tool utilizes Docker, each masternode and their components are hosted in separate containers, which are subsequently connected through a Docker network. We use *dashmate* to set up a network of three masternodes and a seed node/miner. We then connect a group of regular nodes, among which are sybils of the attacker, to the masternode network. The result is a local Dash network with masternodes, regular nodes and a miner; which we presume can function as a representative testing environment for the real Dash network. The setup is visualized in Figure F.1.

F.1.1 Machine & operating systems

We ran into various issues while trying to setup dashmate and while trying to modify and compile the Dash core implementation. Through communication with the developers it became clear that *dashmate* is developed on Linux Ubuntu, and should work there. However, on our Ubuntu 18.04 system we were initially unable to get *dashmate* running. However, we were able to run it on a Linux Manjaro (GNOME 21.1.6) system, so we decided to switch to Manjaro for running the local Dash

[1] `https://github.com/dashevo/dashmate/`

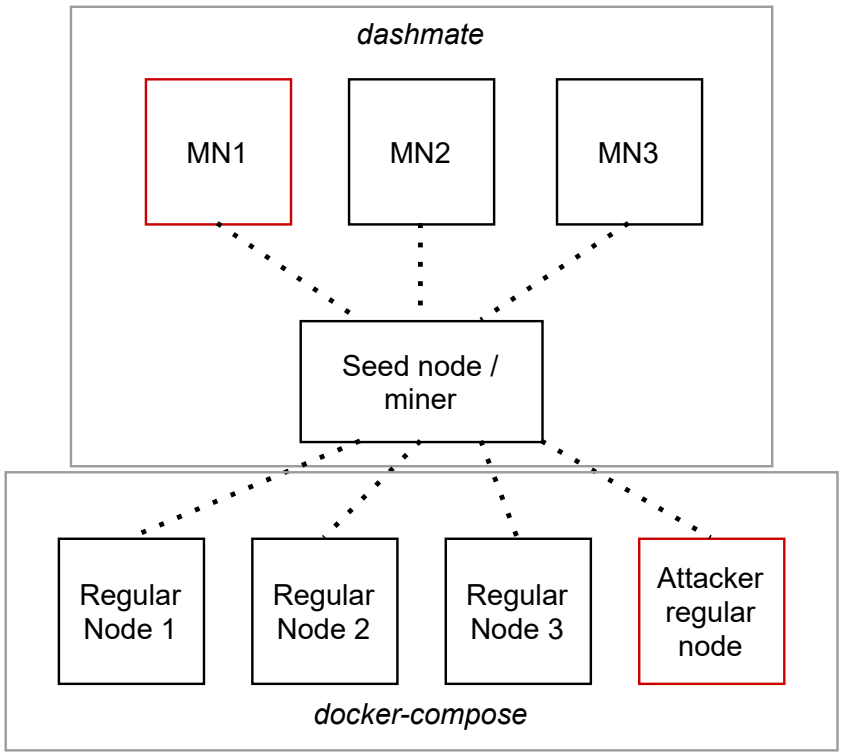

Fig. F.1.: Local Dash test network setup (dotted lines indicate connections, the red nodes are controlled and modified by the attacker).

network for most of the experiments. Building of Dash core was still done on our Ubuntu 18.04 system, since we encountered various issues while trying to build on Manjaro.

Later in this research project we also created a fresh setup on a different (and freshly installed) system with Ubuntu 20.04. Then we were able to run *dashmate* and compilation just fine, although we had to downgrade our NodeJS version (to v16.13.0 vs v17.1.0) and our Node package manager (npm) version (v8.1.0 vs v8.1.2) to fix several errors. Future studies into this attack should be able to deploy *dashmate* on a (fresh) Ubuntu 20.04 system, given that they have the right Node and NPM versions, downgrades as above may be necessary. It should be noted that *dashmate* is under active development. The issues that we faced may (and hopefully will) be fixed with future versions of *dashmate*. The fact that *dashmate* is still under development and not really used for the use case that we need it for are likely the main reasons for the issues that we faced.

The main system that we used in our tests is a machine with 8Gb of RAM and an Intel i7-471MQ CPU (2.5-3.5 GHz, 4 cores). The second system, on which we did

the fresh Ubuntu 20.04 install, is a machine with 12Gb of RAM and an Intel i7-920 CPU (2.66-2.93 Ghz, 4 cores).

The version of Dash core that we use is release 0.17.0.3, and the version of *dashmate* that was used is 0.21.0.

F.1.2 Setting up the test network

To set up the test network, first the dash masternode network must be deployed. To configure *dashmate* for such a network and to create it, *dashmate* is ran with the following command:

```
$ dashmate setup local
```

This command can be provided with a verbose flag (*-v*) to get somewhat more detailed output. Upon running this command *dashmate* gives a prompt and asks for the number of masternodes, whether logs should be generated and the desired blocktime. We leave the values to the suggested defaults, which are 3, yes, 2.5m respectively[2]. Once the setup command terminates without errors, the masternode network can be started with

```
$ dashmate group:start
```

Again, the verbose flag (*-v*) can be used to get more detailed output. Once *dashmate* is finished, the (containerized) masternodes and seed-node (miner) are up. Their logs can be monitored via the Docker logs command, and the nodes can be accessed/instructed via RPC using the Dash command line interface (dash-cli).

The second part of setting up the local Dash network, is to create some regular nodes and connect these to the masternode network that we created using *dashmate*. To setup the regular nodes we do not use the default Dash docker image, but we built our own image. We need a modified image to fix issues that are not yet fixed in the main Dash docker image. Moreover, we need to be able to make modifications to the colluder node's source code, and as such we need to create a modified docker image as well. The docker file that is used to create the image is shown in Appendix G under the Dockerfile section.

To setup multiple regular Dash nodes (among which the colluder node) and connect them in a network, we use docker-compose. docker-compose automates setting up a network between the docker containers, and allows to start multiple (connected and

[2]Setting a lower blocktime could be useful, blocks are mined faster then, which can decrease the time required for attack testing.

interdependent) containers simultaneously. docker-compose is instructed through the docker-compose file, which can also be found in Appendix G, under the docker-compose section. The regular nodes are started with the following command, which is to be executed in the folder where the docker-compose file is stored.

```
$ docker-compose up
```

Another method to connect regular nodes to the dash masternode network is by tricking *dashmate* such that it will start-up multiple regular nodes next to the masternodes and seed node (miner) which it starts normally. This can be done by modifying the configuration file that *dashmates* generates when the setup command is executed. This config file, which is normally stored under the .dashmate folder in the user's home directory, contains a section specifying the nodes in the local network. In this section, first the masternodes configurations are specified (sections local_1, local_2 and local_3) and then the seed node, which is also the miner, configuration is specified (section local_seed). Extra regular nodes can be created by adding multiple slightly modified copies of the seed node configuration (section local_seed) after the seed node configuration. The modifications that need to be made to make *dashmate* interpret these sections correctly and convert them to regular nodes are the following:

1. Provide each section with a unique section name (e.g. local_colluder and local_nodeX, where X is 1,2,3,.. depending on how many regular nodes are added).

2. Change the P2P and RPC ports. For the seed node these are 20301 and 20302 respectively, for additional nodes these should be changed to not have conflicting ports. The recommended way to do this is by increasing the port numbers by 100 for every additional node (e.g. colluder node (local_colluder) has 20401 and 20402, regular node 1 (local_node1) has 20501 and 20502, etc.).

3. Add the seed node in 'seeds' under the P2P section. The other regular nodes, which are not the seed node, must also have a connection to the seed node to become connected to the Dash network. In the masternode sections (local_1, local_2, etc.), the seed node's IP address and port are added under 'seeds'. This has to be done on the regular (non-seed) nodes as well. Thus

   ```
   "seeds": []
   ```

   becomes

```
"seeds":[
  {
          "host": "172.17.0.1",
          "port": 20301
  }
]
```

4. Disable miner functionality by changing ‘miner: true’ to ‘miner: false’.

Now when *dashmate* is used to start the network it will also start three regular nodes, based on the configs added to the configuration file, and add these nodes to the Dash network.

F.1.3 Testing CoinJoin

After setting up the testing environment, it is valuable to test whether regular CoinJoin is working on the local network. This is required to be able to test the attack, and this is the point that we repeatedly faced issues on. To do this, the following steps need to be taken:

1. Start up the local Dash network. (`$ dashmate group:start`)
2. (OPTIONAL) Start the regular nodes. (`$ docker-compose up`)
3. Get access to the seed node and at least two of the regular nodes (N1 and N2) by attaching a terminal to the containers in which these nodes are running. This is necessary to have easy access to the node via RPC calls. (`$ docker exec -it NODE_CONTAINER_NAME /bin/bash`)
4. On at least two of the regular nodes, generate a new address (A1 and A2) using dash-cli.

   ```
   $ dash-cli -dashuser=dashrpc -rpcpassword=rpcpassword getnewaddress
   ```

).
5. On the seed node, send 5 Dash to A1 and A2 by creating a new transaction through dash-cli.

   ```
   $ dash-cli -rpcuser=dashrpc -rpcpassword=rpcpassword sendtoaddress A1 5
   ```

   ```
   $ dash-cli -rpcuser=dashrpc -rpcpassword=rpcpassword sendtoaddress A2 5
   ```
6. Wait until the transactions are confirmed, this can be checked through executing the *getbalance* RPC on N1 and N2.

```
$ dash-cli -rpcuser=dashrpc -rpcpassword=rpcpassword getbalance
```

7. Set the number of CoinJoin rounds to 2 on N1 and N2.

```
$ dash-cli -rpcuser=dashrpc -rpcpassword=rpcpassword setcoinjoinrounds 2
```

8. Set the amount to be anonymized (CoinJoin amount) to 5 Dash on N1 and N2.

```
$ dash-cli -rpcuser=dashrpc -rpcpassword=rpcpassword setcoinjoinamount 5
```

9. Start CoinJoin on N1 and N2.

```
$ dash-cli -rpcuser=dashrpc -rpcpassword=rpcpassword coinjoin start
```

10. Wait while N1 and N2 are mixing, monitor the logs of N1 and N2 (and of the masternodes) to see whether mixing sessions are successful. After some time, some of the funds should show up as anonymized, this can be checked on N1 and N2 through the *getwalletinfo* RPC.

```
$ dash-cli -rpcuser=dashrpc -rpcpassword=rpcpassword getwalletinfo
```

F.2 Attack implementation

To implement the attack, several modifications must be made to the code. In our proof-of-concept, we aim to make only the necessary modifications that enable us to show the attack will be successful.

F.2.1 Attacker masternode

First, we modify the masternode such that the queue announcement restrictions, which we previously discussed, will be removed. These restrictions are implemented in *coinjoin-server.cpp*[3]. The modifications need to be made in the part where requests for new CoinJoin queues are processed, which is between line 33 and 104. Specifically on line 79-90, the checks that verify whether the most recent CoinJoin queue was not too recent are implemented. These checks must be disabled, so line 79-90 should be commented out or removed.

In this implementation we eliminate the need for the attacker masternode to target their queue messages at the victim. Instead, we make sure that the colluder node starts mixing before the target node, and through that ensure that the target will

[3]Soon there will be a refactorization that removes 'coinjoin' from this filename, however in the release that we used the filename is still coinjoin-server.cpp

have only one mixing session to join. Then if the target does indeed (repeatedly) join mixing queues announced by the attacker masternode, it is shown that this attack can work. This only works because we do this on a local network, and because we aim to get a proof-of-concept rather than a working live implementation. To get an implementation that works on the main Dash network, we would have to make sure that the attacker masternode is connected to the target and that it will send queue announcements to the target.

F.2.2 Attacker colluder node

Second, we modify the colluder node (the regular attacker node) such that it will ask for a new queue whenever it wants to mix some funds, so it will never join an existing queue. To do this we modify the DoAutomaticDenominating funtion. Instead of joining an existing queue by default (line 936-939) we start a new queue by default (line 941-942). To do this, line 936-939 must be removed or commented out.

The colluder node also does not care whether the masternode announced a queue recently, so that check is also disabled. This check is done on line 100-103 (when verifying a queue announcement) and on line 1136-1143 (when requesting the start of a new CoinJoin queue) in *coinjoin-client.cpp*. These pieces of code must therefore be removed or commented out. Moreover, the colluder node does not want to skip (soon to be) masternodes payment winners either, so this check is disabled in the StartNewQueue funtion, line 1128-1132 is commented out in *coinjoin-client.cpp*.

The colluder node has to make sure it creates a queue specifically at the masternode that is controlled by the attacker. This introduces a bit of complexity. It is possible to select a specific masternode using the hash of their masternode registration transaction. Within the StartNewQueue function in *coinjoin-client.cpp*, the a masternode is randomly picked using the GetRandomNotUsedMasternode function. This random selection, which happens on line 1117, is implemented on line 1000-1036. In this function, the masternode list is shuffled and then looped through until a a valid masternode is found. A masternode is valid if it is not in the set of used masternodes. We replace the functionality of the checking whether a masternode is valid. Instead of checking whether a masternode is not in the excludeset, the colluder node checks whether the registration transaction of the masternode is equal to the registration transaction of the attacker masternode. Thus, in the end the colluder node does not pick a random masternode but it picks the masternode that is operated by the attacker. In our implementation, line 1025 to 1032 is replaced by:

```
for (const auto& dmn : vpMasternodesShuffled) {
// Fill in the transaction hash of the attacker masternode registration transaction in hexadecimal notation below.
  // Example:
  // unsigned char  hexprotxhash[] = {0xf1, 0xc7, 0xdb, 0x1f, 0xfd, 0x85, 0xea, 0x5b, 0x90, 0x16, 0xf4, 0x1e, 0x80 ,0x5f, 0x56, 0x31, 0x64, 0x55, 0x73, 0x76, 0xf3, 0x74, 0xcb, 0xa6, 0x5d, 0x9c, 0x7a, 0xe1, 0x8a, 0x74, 0x10, 0xd3};
  unsigned char hexprotxhash[]={};

  const std::vector<unsigned char> protxvec(hexprotxhash, hexprotxhash+32);
  const uint256 comprmnhash = uint256(protxvec);

  if (dmn->proTxHash != comprmnhash) {
          continue;
  }

  LogPrint(BCLog::COINJOIN, "CCoinJoinClientManager::%s -- found, masternode=%s\n", __func__, dmn->collateralOutpoint.ToStringShort());
  return dmn;
}
```

This means that, before the modifications to the Dash core implementation can be made, and before the docker image for the attacker nodes can be created, the masternode network must have been set up already (according to instructions provided previously) so that the hash of the registration transaction of the attacker masternode can be known. This hash is printed during setup when *dashmate* is ran with the verbose flag. Otherwise, it can be retrieved via the 'masternodelist' RPC.

When these modifications are made, the Dash core implementation needs to be compiled again, according to the provided build instructions[4]. Moreover, docker images need to be generated for the attacker nodes. Since the modifications on the masternode side and the colluder node do not interfere or overlap with each other we can use the same docker image for the malicious masternode and the

[4] https://github.com/dashpay/dash/blob/master/doc/build-generic.md

colluder node[5]. To generate the docker image, we need to copy the bin folder, which containers Dash's built binaries, from the Dash core directory to the folder where the Dockerfile resides. The docker image can then be built with:

```
$ docker build -t dashd:compr .
```

After the modified docker image has been created, it must also be applied in the test setup. In the docker-compose file, one of the nodes, the colluder node, must be run with the modified image. Therefore, in the colluder node section of the docker compose file, the image must be changed to 'dashd:compr'. However, if the regular nodes are run via *dashmate* as well, then the docker-compose file is not used and the image for the colluder-node must be adjusted in the *dashmate* config file, which is found in the .dashmate folder in the users home directory. In the config file, the images of the attacker masternode (local_1) and the colluder node (local_colluder) must be changed from 'dashpay/dashd:0.17' to 'dashd:compr'.

F.3 Problems and fixes

Throughout the process of implementing a test setup and executing the attack we encountered various issues. In this section these are explained, and the applied fixes are explained. Some of these fixes must also be applied when implementing this attack using Dash Core release 0.17.03 and *dashmate* v0.21.0.

F.3.1 Skipping masternode payment winners

The first issue is with the Dash Core implementation, which hinders mixing on a local network with few masternodes. Masternode payments, which happen each block, are done in a deterministic fashion. By default, the upcoming 8 masternodes which are to get the masternode payment are skipped by a client who wants to join or create a CoinJoin queue. However, when there are only three masternodes in the Dash network, as in our local Dash network, this means there will never be an eligible masternode for hosting a mixing session. Therefore, this check had to be disabled, and upon discussion with the Dash developers this was also done[6]. However, it was only fixed in the JoinExistingQueue function, in *coinjoin-client.cpp*, while this is also an issue in the StartNewQueue function, where we had to manually

[5] In a real-world implementation of the attack it may be desirable to have separate images, since then the masternode does not have to be taken down to change it's image, after it has been started. This is necessary now, since the masternode must first be started to retrieve its registration transaction hash and then stopped to replace its image.

[6] `https://github.com/dashpay/dash/pull/4394`

fix this. We will hopefully be able to fix this with the Dash developers soon as well. Fixing this issue also required creating a new docker image for the local regular nodes, which we called 'dashd:mnskipfix', and replacing the regular node images in the docker-compose file and/or *dashmate* config with this image.

F.3.2 Wallet access

Through a bug in the *dashmate* software, it is impossible to access the wallet of the seed node. Access to that wallet is necessary to make transactions to other nodes. Making these transactions is required to fund the other nodes, which will subsequently have a role in the attack by participating in CoinJoin sessions. We initially used a workaround to counter this issue. We used the generatetoaddress RPC on the seed node to generate blocks and send the block reward to addresses of the other regular nodes. Although this works, it requires more effort since block rewards are not available initially. Therefore, another 100 blocks have to be generated after the blocks that have been generated for the other regular nodes. In theory this can be done quite fast, although we observed that it regularly resulted in syncing issues between the nodes, where some of the nodes would get out of sync with the blockchain and not function anymore. This may be caused by the limited computational capacities of the hardware that we used. It was acknowledged by the Dash developers that this was a bug, and we presented a fix[7], which was accepted by the developers.

F.3.3 Remaining issues

Thus far, we have still not been able to successfully simulate CoinJoin sessions on a local network. A CoinJoin session can be started just fine, although after the masternode accepts the started session and tells the nodes to submit; no response is created. Instead, after some time the nodes terminate the CoinJoin session because it times out, and the masternode later also terminates the session because of a timeout. Why the session does not continue regularly is thus far unclear. Moreover, the nodes are often unable to select masternodes to host their mixing sessions. This seems to be, in part, caused by the unability of a node to request another CoinJoin session at a masternode to which it is already connected (line 1076-1079 and 1145-1149). This would hinder futher CoinJoin sessions as well, on a network with few masternodes.

Further research should clarify why CoinJoin sessions do not continue as expected, and why nodes are often unable to find masternodes to host their CoinJoin sessions.

[7] `https://github.com/dashevo/dashmate/pull/466`

Because we were unable to do mixing on a local network, we also have not been able to verify that the changes we made to the Dash core implementation are correct. Therefore, once mixing on a local network does work, the correctness of the current implementation of the attack should be studied.

G

Configuration files and instructions for Dash queue gaming

G.1 Dockerfile for building Dash docker images

The dockerfile that is shown below can be used to build the docker images. This dockerfile requires compiled Dash binaries to be in the same directory as the dockerfile in a folder named *bin*. When Dash is compiled according to the build instructions[1], this *bin* folder is generated under the src/depends/<host> directory (where host is the host-platform-triplet).

Depending on what Dash image is built, the description label in the dockerfile may be changed to represent the container's purpose. The dockerfile can be used to build the image that fixes masternode skipping and the attacker compromised node image. The command to build the docker image is `docker build -t <tag> .`, where <tag> is replaced by either *dashd:mnskipfix* or *dashd:compr*, depending on the purpose of the image. Using the right tag is important, since the tag is used in the docker-compose file discussed in the next section.

Dockerfile

```
1  FROM ubuntu:bionic
2  LABEL description="Dockerised DashCore - fixed masternode skipping
   ↪  on local network"
3
4  COPY bin/* /usr/local/bin/
5
6  ARG USER_ID
7  ARG GROUP_ID
8  ARG TAG
9
```

[1] https://github.com/dashpay/dash/blob/master/doc/build-generic.md

```
10 ENV HOME /dash
11
12 # add user with specified (or default) user/group ids
13 ENV USER_ID ${USER_ID:-1000}
14 ENV GROUP_ID ${GROUP_ID:-1000}
15 RUN groupadd -g ${GROUP_ID} dash && \
16     useradd -u ${USER_ID} -g dash -s /bin/bash -m -d /dash dash  &&
       ↪  \
17     mkdir /dash/.dashcore && \
18     chown dash:dash -R /dash
19
20 RUN apt-get update && \
21     apt-get -y install --no-install-recommends \
22     wget \
23     ca-certificates \
24     && rm -rf /var/lib/apt/lists/*
25
26 RUN chmod a+x /usr/local/bin/*
27
28
29 USER dash
30
31 VOLUME ["/dash"]
32
33
34 EXPOSE 9998 9999 19998 19999
35
36 WORKDIR /dash
```

G.2 docker-compose file for local regular nodes

Below is the docker-compose file that is used to host the regular nodes when simulating the attack. It should be noted that when just trying to use the local nodes to execute CoinJoin (e.g. for testing the local setup), the image of the 'compr_node' (the attacker node) should be changed to 'dashd:mnskipfix' on line 3. Obviously, building 'dashd:mnskipfix' is required before this docker-compose file can be used. The docker-compose file places a Dash configuration file in the individual containers,

which is required for and used by the Dash nodes. This Dash configuration file is provided below (dash.conf).

docker-compose.yml

```
services:
  compr_node:
    image: dashd:compr
    container_name: colluder_node
    restart: unless-stopped
    ports:
      - 20011:20001 # P2P
      - 127.0.0.1:20012:20002 #RPC
    volumes:
      - ./dash.conf:/dash/.dashcore/dash.conf
      - dashd_datacompr:/dash
    command:
      - dashd
    environment:
      - NETWORK=local

  normal_node1:
    image: dashd:mnskipfix
    container_name: node1
    restart: unless-stopped
    ports:
      - 21001:20001 # P2P
      - 127.0.0.1:21002:20002 #RPC
    volumes:
      - ./dash.conf:/dash/.dashcore/dash.conf
      - dashd_data1:/dash
    command:
      - dashd
    environment:
      - NETWORK=local

  normal_node2:
    image: dashd:mnskipfix
    container_name: node2
```

```
    restart: unless-stopped
    ports:
      - 22001:20001 # P2P
      - 127.0.0.1:22002:20002 #RPC
    volumes:
      - ./dash.conf:/dash/.dashcore/dash.conf
      - dashd_data2:/dash
    command:
      - dashd
    environment:
      - NETWORK=local


  normal_node3:
    image: dashd:mnskipfix
    container_name: node3
    restart: unless-stopped
    ports:
      - 23001:20001 # P2P
      - 127.0.0.1:23002:20002 #RPC
    volumes:
      - ./dash.conf:/dash/.dashcore/dash.conf
      - dashd_data3:/dash
    command:
      - dashd
    environment:
      - NETWORK=local


volumes:
  dashd_data1:
  dashd_data2:
  dashd_data3:
  dashd_datacompr:
```

dash.conf

```
# general
daemon=0  # leave this set to 0 for Docker
logtimestamps=1
maxconnections=256
printtoconsole=1
debug=1

# JSONRPC
server=1
rpcuser=dashrpc
rpcpassword=rpcpassword

rpcallowip=127.0.0.1
rpcallowip=172.16.0.0/12
rpcallowip=192.168.0.0/16

rpcworkqueue=64
rpcthreads=16

# external network
listen=1
dnsseed=0
allowprivatenet=1

# Indices
txindex=1
addressindex=1
timestampindex=1
spentindex=1

regtest=1
[regtest]

# network
port=20001
bind=0.0.0.0
rpcbind=0.0.0.0
```

```
38 rpcport=20002
39
40 addnode=172.17.0.1:20301
```